

Adversarial Attack against Deep Reinforcement Learning with Static Reward Impact Map

Patrick P. K. Chan
School of Computer Science and
Engineering, South China University
of Technology
Guangzhou, China
patrickchan@ieee.org

Yaxuan Wang
School of Computer Science and
Engineering, South China University
of Technology
Guangzhou, China
yaxuan.wang1995@gmail.com

Daniel S. Yeung
Hong Kong
danyeung@ieee.org

ABSTRACT

Security problems of deep reinforcement learning draw much attention recently. Previous works on adversary attack mainly focus on preventing the targeted agent from choosing the most desirable action at each step, which may not reduce the cumulative reward effectively. In this paper, we first investigate how changing features affect the cumulative reward achieved by an agent. The static reward impact map is introduced to quantify the influence on the reward of each feature experimentally. By focusing on tasks with the static reward impact map, an adversarial attack method against deep reinforcement learning aiming to minimize the cumulative reward is proposed. Features with the large reward impact are perturbed in crafting an adversarial sample. Deep Q-network is selected to demonstrate the performance of our attack method in the experiments. The results indicate that our proposed method achieves better performance than the existing one-time attack method and the random attack in terms of the cumulative reward and the successful attack rate under both white-box and black-box settings.

CCS CONCEPTS

• Computing methodologies → Adversarial learning; • Security and privacy;

KEYWORDS

reinforcement learning, adversarial attack, static reward impact map

ACM Reference Format:

Patrick P. K. Chan, Yaxuan Wang, and Daniel S. Yeung. 2020. Adversarial Attack against Deep Reinforcement Learning with Static Reward Impact Map. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20)*, October 5–9, 2020, Taipei, Taiwan. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3320269.3384715>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASIA CCS '20, October 5–9, 2020, Taipei, Taiwan

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6750-9/20/10...\$15.00

<https://doi.org/10.1145/3320269.3384715>

1 INTRODUCTION

The great success of Deep Neural Networks (DNNs) has been demonstrated in reinforcement learning [13, 14] as well as supervised [3, 9] and unsupervised learning [12]. In Deep Reinforcement Learning (Deep RL), DNNs serve as the function approximator to handle various sequential decision-making tasks. Although DNNs based Deep RL models achieve excellent performance in various applications, *e.g.*, Go [19], Atari games [14], autonomous driving [24] and robot manipulation [11], they inherit the vulnerability of DNNs in the adversarial environments where data are perturbed by an adversary in order to mislead the outputs of DNNs [4, 21]. Reinforcement learning models are usually used as the control policies of the mechanical equipment (*e.g.*, vehicles and robots), and disrupting such control policies may cause serious damage, especially in safety-critical applications. Thus, security leakage of Deep RL is an important topic.

Current adversarial attacks against Deep RL [1, 7] mainly follow the idea of the ones in supervised learning, in which each sample is crafted independently in order to increase the generalization error of a classifier [6, 16, 21]. However, different from supervised learning, the ultimate goal of a RL agent is to maximize the cumulative reward, which is influenced by a sequence of decisions on actions. Current attacks against Deep RL only prevent a targeted agent from choosing the most desirable action at each step, which may not cause the lowest cumulative reward achieved by the agent in the long run. In addition, the feature correlation is not considered in crafting an adversarial sample. As a result, the current adversarial attacks may not attack Deep RL effectively.

In a decision-making process, a person usually focuses his attention on parts of the visual space [23], *e.g.*, regions near to the car are usually more important than ones far away from it while driving. This mechanism indicates that some features are inherently more important than the others, *i.e.*, these features provide more critical information in decision making. Any change on these features is more likely to lead to a bad decision which significantly reduces the cumulative reward of a targeted agent. By this observation, this study first proposes a measurement named the reward impact map to quantify the influence of changing a feature on the cumulative reward. A feature with a larger reward impact value indicates that the cumulative reward is reduced more significantly by perturbing this feature. A sliding-window method is applied to estimate the influence experimentally. The influence correlation of features in each attack window on the cumulative reward is considered. In this preliminary study, we focus on the tasks with a static reward impact map, *i.e.*, the importance of features is the same (or slightly

different) in any state. Our experimental results indicate that a static reward impact map does not depend on the agents but mainly relies on a task.

An adversarial attack based on the static reward impact map against Deep RL is then proposed. Features with the largest reward impact are perturbed according to the gradient function of the targeted agent. It should be noted that our method attacks the same features with different amounts of perturbation for all adversarial samples. It is different from other attack methods in which attack features are chosen differently and independently for each adversarial sample. One advantage of our method is the low time complexity of crafting adversarial samples in real time since the attack features are pre-selected. Deep Q-network (DQN) [14], which is a well-known Deep RL algorithm, is used as the targeted agent to demonstrate the performance of our model. The proposed method is evaluated, and compared with existing one-time attack methods and the random attack method experimentally by using Atari games, *i.e.*, Freeway, Pong, Breakout, FishingDerby, SpaceInvaders and BattleZone. White-box and black-box attack scenarios [7] indicating different knowledge levels of an adversary are investigated. Finally, the conclusion and the future work are discussed.

The main contributions of this work are summarized as follows:

- This study introduces an attack method aiming to reduce the cumulative reward achieved by the targeted Deep RL agent, which is different from the existing attack methods focusing on the short-term influence.
- The static reward impact map is introduced to quantify the influence of slight change to input features on the cumulative reward in the tasks in which the importance of features is similar in any state. Experimental evaluation indicates that the map mainly relies on a task instead of an agent.
- An adversarial attack with the static reward impact map aiming to reduce the cumulative reward against Deep RL agent is proposed. An adversarial sample is crafted by perturbing the features with the largest reward impact according to the gradient function.
- The proposed model is evaluated and compared with the existing one-time attack method and the random method by Atari games. The experimental results consistently suggest effectiveness and efficiency of our model in both white-box and black-box attack settings.

2 RELATED WORK

In this section, the related concepts of Deep RL is first described, and then the adversarial attack method is discussed.

2.1 Reinforcement Learning

In reinforcement learning (RL), there usually contains an agent and an environment, where the agent interacts with the environment. The internal state of the environment ξ is represented by states s . The agent learns to take actions to maximize the cumulative reward [20]. The RL agent is usually defined as a deterministic policy function (*i.e.*, $\pi_\theta(s) = a$), which maps a state s to an action a , where $a \in \mathcal{A}$, $s \in \mathcal{S}$ and θ is the parameter of the agent. The agent learns from the environment ξ by considering the instant reward r after taking a in s . More specifically, at the timestep t , an

agent takes action a_t based on policy π_θ and current state s_t , the environment ξ will return the instant reward r_t and also the next state s_{t+1} . ξ can be modelled as a stochastic finite state machine, *i.e.*, $\xi(s_t, a_t) = (r_t, s_{t+1})$. The cumulative reward R is defined as the summation of r_t in an episode, *i.e.*, $R = \sum_{t \in \mathcal{P}} r_t$, where $\mathcal{P}$ is a set of episodes. That is, the agent aims to maximize R by taking reasonable actions under a series of states.

Q-Learning algorithm [22] is one of the classic reinforcement learning algorithms. It learns the policy iteratively according to samples generated by the interaction between an agent and the environment. $Q(s_t, a_t)$, which is the expected reward obtained by taking action a_t at current state s_t , is updated in each iteration. The agent usually selects the highest-weighted action in a_t as the most desirable action. Due to the satisfying performance of convolutional neural networks (CNNs) in many applications, especially for image-based applications, CNNs have been used as the approximator of Q [14], named Deep Q-learning Network (DQN). DQN achieves breakthroughs in many domains, *e.g.*, autonomous driving [8], and robot manipulation [10], and it becomes one of the most popular reinforcement learning algorithms nowadays.

Proximal Policy Optimization (PPO) [18] is an off-policy RL algorithm, in which training samples generated by both current and old policies. The samples are weighted according to the difference between their policies and the current one measured by KL divergence. PPO uses multiple epochs of stochastic gradient descent aiming to maximize the reward by selecting proper actions. The policy update and sample collection are alternately processed until convergence.

This paper mainly focus on image-based RL applications, *i.e.*, an observation from the environment is an image. For example, video game and autonomous driver. Given such a RL application, the screenshot is defined as the observation o with the size $o_s \in \mathcal{Z} \times \mathcal{Z}$, which is a matrix of raw pixel values representing the current screen. A state s_t consists of latest i observations [14], since the reinforcement learning problem is modeled as a finite Markov decision process (MDP). s_t is defined as $\phi(o_{t-(i-1)}, \dots, o_{t-1}, o_t)$, where $\phi()$ is a function which preprocesses the observations.

2.2 Adversarial Attack Method

Under adversarial attacks, the attacker (*i.e.*, the adversary) manipulates the testing data in a specific way (*e.g.*, by adding noise) in order to mislead the outputs of model. An adversarial sample is the well-designed version of the clean sample which is intentionally perturbed to mislead the model, while adversarial perturbation is the noise that is added to the clean sample (*i.e.*, $\tilde{s} = s + \eta$).

DNNs have been proved to be vulnerable to adversarial attacks [21], and DNNs based Deep reinforcement learning models inherit such vulnerability. Reinforcement learning models are mainly used as control policies of the mechanical equipment (*e.g.*, vehicles and robots), and attacks may cause serious damage (*e.g.*, traffic accidents).

Most of existing adversarial attacks against Deep RL mainly follow the idea of the ones against supervised learning, which aim to change the decision of a classifier on each sample with the minimum perturbation. The adversarial attack methods, such as Fast Gradient Sign Method (FGSM) [6] and Jacobian-based Saliency

Map Attack (JSMA) [16], proposed against DNNs-based supervised learning are applied to attack Deep RL [1, 7, 17].

Adversarial attacks are divided into two categories, one-time attacks and iterative attacks, according to their computation cost. One-time attacks craft an adversarial sample through a single step of computation, *e.g.*, computing the gradient of the model function once, while iterative attacks perform such computation iteratively for one adversarial sample. Usually, iterative attacks are more computational expensive than one-time attacks. In this paper, only one-time attacks considered due to the short response time requirement of crafting adversarial samples in RL.

FGSM is one of the one-time attacks and is famous for its low time complexity. It is also the most common adversarial crafting algorithm in computer vision and Deep RL. For a sample s , $J(\theta, s, a)$ denotes the cross-entropy loss between all actions a and the distribution that places all weight on the highest-weighted action. And $\nabla_f J(\theta, s, a)$ denotes the gradient of the loss function with respect to feature f , which measures the sensitivity to the change of the loss with respect to a change in features. Larger $\nabla_f J(\theta, s, a)$ indicates that changing of a feature has more influence on the output. In order to prevent the agent from choosing the most desirable action, FGSM perturbed features with the largest $|\nabla_f J(\theta, s, a)|$ by $\alpha * \text{sign}(\nabla_f J(\theta, s, a))$, where α is the maximum perturbation on each feature.

3 STATIC REWARD IMPACT MAP BASED ATTACK MODEL

Existing attack methods against Deep RL aim to prevent an agent from choosing the most desirable action in each step. However, as the cumulative reward relies on a sequence of decisions, not choosing the most desirable actions may not reduce the cumulative reward efficiently. Therefore, an adversarial sample crafting method aiming to minimize the cumulative reward of a Deep RL agent is proposed in this section.

Our method measures the importance of a feature in selecting suitable actions by the proposed reward impact map. It is calculated by the influence of slight change to input features on the cumulative reward in one episode. Our study focuses on the tasks in which the importance of features is static for all states. An adversarial attack based on the static reward impact map against Deep RL is proposed. An adversarial sample is crafted by perturbing features with the largest reward impact according to the gradient function of the targeted agent in our method. Our method is expected to reduce the cumulative reward effectively since the features with the largest reward impact value are perturbed.

In this section, the adversarial setting is first introduced. We will then describe the static nature, the proposed static reward impact map and attack method with the static reward impact map.

3.1 Adversarial Setting

In this section, we describe the adversarial setting of our attack model in terms of the attack framework defined in [2].

Adversary's Goal. Our attack model aims to minimize the cumulative reward of the targeted agent.

Adversary's Knowledge. We consider two different attack scenarios: the white-box and the black-box attack. All information of

the targeted agent (π_θ) is obtained in the white-box attack. On the other hand, only the information of the input and output spaces is obtained in the black-box attacker. The standard procedure of the black-box attack [15] is applied to our study. A surrogate agent is required to approximate. Therefore, a surrogate agent is trained to serve as the targeted agent in the attack. As no knowledge of the network structure and the parameters of the targeted agent is obtained, the surrogate agent is usually different from the targeted one.

Adversary's Capability. Similar to other attack strategies, we assume that the adversary is capable of changing the value of each feature and the attack budget of any feature are the same. This setting is reasonable for RL applications in which all features represent the same type of information, *i.e.*, a pixel of observation. The perturbation is limited by the maximum number of manipulated features and the maximum amount of change on all the features.

3.2 Static Nature

A static nature of the reward impact map means that the importance of features is the same (or slightly different) in any state. Our proposed static reward map can be applied to applications in which influence on the reward of a feature is similar in different states, for example, the ones with first-person vision, *e.g.*, autonomous driving and shooting games, and the fixed moving path of the main character, *e.g.*, Atari game Freeway and the robotic grasping. Atari game BattleZone and Freeway shown in the left and middle subfigure in Figure 1 are the examples of the applications with the first-person vision and the fixed moving path. The screen of BattleZone is from the angle of the tank, and the yellow chick always walks across the street along the same path indicated by the red arrow.

In these applications, the information provided by the regions near the character generally affects the decisions more significantly in some applications. The importance of a feature mainly depends on its distance to the main character. This observation is consistent with the attention mechanism [23] which indicates that some features are inherently more important to a task than the others. Since some features provide more critical information in decision making in a RL task, changing these features is more likely to affect the decisions of an agent. Differently, the importance of features in some applications are not static. The right subfigure of Figure 1 shows the screenshot of Atari game Boxing. As the boxer is free to walk around inside the boxing ring, the importance of features changes dynamically.



Figure 1: Left: BattleZone with first-person vision; Middle: a fixed walking trajectory in Freeway; Right: two possible walking trajectories in Boxing

3.3 Static Reward Impact Map

The static reward impact map is introduced to quantify the influence of slight change to input features on the cumulative reward in an image-based RL task in which the importance of features is similar in any state. A feature with a larger reward impact value indicates that the changing on this feature will influence the cumulative reward more significantly, *i.e.*, the effective attacks will tend to perturb this feature more. A sliding-window method is applied to estimate the influence experimentally. The influence correlation of features on the cumulative reward is considered, since features in the same window are perturbed simultaneously.

Given an image-based application, an observation o is a screenshot. The latest i observations are concatenated to form a state at t , *i.e.*, $s_t = \phi(o_{t-(i-1)}, \dots, o_{t-1}, o_t)$, where o_k is the observation at the timestep k . Our study focuses on the image-based applications (*i.e.*, the first-person vision and the fixed movement path of the characters) in which the features at the same coordinate of different observations have the same influence on the cumulative reward. Let $f_{(x,y)}$ be a pixel at the coordinate (x, y) of an observation. Static reward impact is defined to quantify the importance of $f_{(x,y)}$ in the decision making of an agent. To be more specific, the influence of slight changing on all features in $f_{(x,y)}$ on cumulative reward achieved by a targeted agent is measured simultaneously. That is, a larger reward impact value indicates that the changing on features in $f_{(x,y)}$ will influence the cumulative reward more significantly.

An experimental approach is proposed to estimate the static reward impact map with sliding-window. The attack window W with the size of $w_s \in \mathcal{Z} \times \mathcal{Z}$ is defined as a region in an observation, where $w_s \leq o_s$. The reward impact of attack window $RI(W)$ is defined as follows:

$$RI(W) = R - R(W) \quad (1)$$

where R and $R(W)$ are defined as the cumulative reward without attack and under attack on W respectively. The calculation of $R(W)$ in one episode is illustrated in Figure 2. For one episode, W covers the same region in all the observations in a state, as shown by the blue rectangles in Figure 2. At the timestep t , any feature in the attack window of all observations of a state is attacked at the same time. s_t is attacked by

$$\tilde{f}_{(x,y)}^k = f_{(x,y)}^k + \alpha * \text{sign}(\nabla_{f_{(x,y)}^k} J(\theta, s_t, a_t)) \quad (2)$$

where α is the maximum perturbation on each feature, $\nabla_{f_{(x,y)}^k} J(\theta, s_t, a_t)$ is the gradient of the cross-entropy loss with respect to $f_{(x,y)}^k$, $(x, y) \in W$, $f_{(x,y)}^k$ is $f_{(x,y)}$ of o_k , and $k = t - (i + 1), \dots, t - 1, t$. Reward impact of all features in a sliding-window rather than a single feature is calculated simultaneously in order to consider the correlation among adjacent features.

W slides along x- and y-axis with d_x and d_y respectively until the end of an observation is reached, as shown in Figure 3. Especially, attack window at the beginning of the observation is defined as W_{init} , likewise, attack window at the end of the observation is defined as W_{fin} .

Finally, $RI(f_{(x,y)})$ is calculated as follows:

$$RI(f_{(x,y)}) = E([RI(W)] | (x, y) \in W, W \in \mathcal{W}) \quad (3)$$

where $\mathcal{W}$ is a set containing all sliding windows W . The detailed implementation is shown in Algorithm 1. Experiments can be run M times independently to obtain an average cumulative reward.

Algorithm 1 Algorithm on $RI(f_{(x,y)})$

Input: w_s : attack window size; d_x, d_y : the stride in x- and y-axis; α : upper bound on perturbation; π_θ : the targeted agent; ξ : RL environment; R : the cumulative reward without attack.

Output: $RI(f_{(x,y)})$: reward impact of $f_{(x,y)}$.

```

1: Set  $W = W_{init}$  with  $w_s$ ;
2: Set  $\mathcal{W} = \emptyset$ 
3: repeat
4:   Set  $\mathcal{W} = \mathcal{W} \cup W$ ;
5:   Set  $R(W) = 0$ ;
6:   for each  $t$  in the episode do
7:     Craft  $\tilde{s}_t$ ;
8:     Set  $a_t = \pi_\theta(\tilde{s}_t)$ 
9:     Set  $(r_t, s_{t+1}) = \xi(a_t, s_t)$ ;
10:    Set  $R(W) += r_t$ ;
11:   end for
12:   Set  $RI(W) = R - R(W)$ ;
13:   Slide  $W$  by  $(d_x, d_y)$ ;
14: until  $W = W_{fin}$ 
15: for each  $(x, y)$  in an observation do
16:   Set  $RI(f_{(x,y)}) = E([RI(W)] | (x, y) \in W, W \in \mathcal{W})$ ;
17: end for
```

The shape and the size of W affect which features are considered together in the reward map calculation. Generally, when the size is larger, the dependency of more features is considered, but the contribution of each feature to $RI(W)$ is smaller (*i.e.*, the estimation is less precise for a feature); and vice versa. Therefore, the size is a tradeoff between the precision and the feature dependency. On the other hand, the shape of W , including the square, horizontal and vertical rectangle, depends on the characteristics of the applications, *e.g.*, the path of the character and enemies. For example, a vertical rectangle may be suitable for Freeway since the player moves vertically. The setting of the shape and size of W depends on the application. One practical method to determine a suitable setting of W for an application is to consider a variety of settings. Each setting is evaluated according to its performance in a preliminary experiment, and the best one is chosen.

The effectiveness of the estimation on the static reward impact map will affect the attack ability of SRIMA. The effectiveness is proportional to M and inversely proportional to d_x and d_y . However, larger M , or smaller d_x or d_y , will also increase the time complexity of the static reward impact map estimation linearly. As a result, M , d_x and d_y should be determined accordingly to the cost of attack.

3.4 Static Reward Impact Map based Attack (SRIMA)

SRIMA aims to reduce the cumulative reward achieved by the targeted agent with the minimum perturbation on the input. The estimation of the static reward impact map is first carried out to measure the vulnerability of the agent in a task. It is expected that

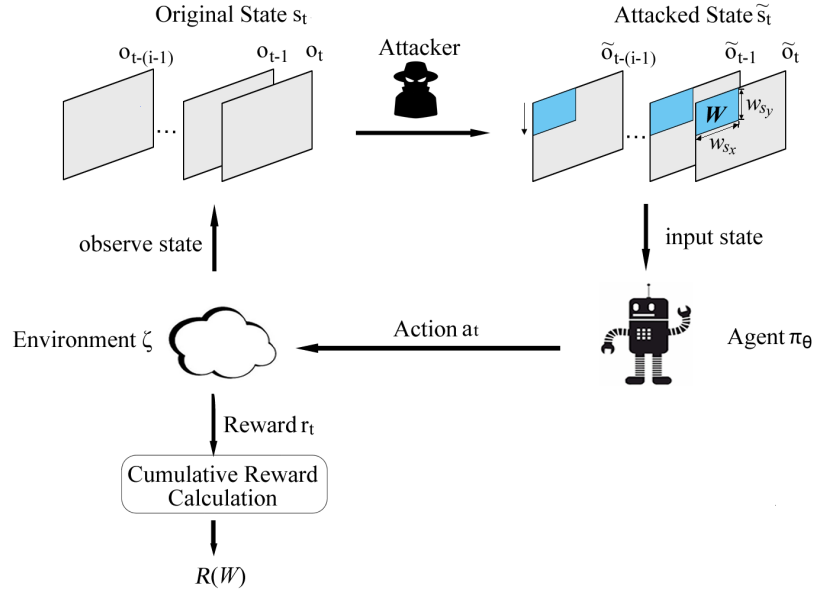


Figure 2: Flow chart of $R(W)$ calculation in one episode. Grey rectangle: the observation; blue rectangle: the attack window. $w_s = w_{s_x} \times w_{s_y}$

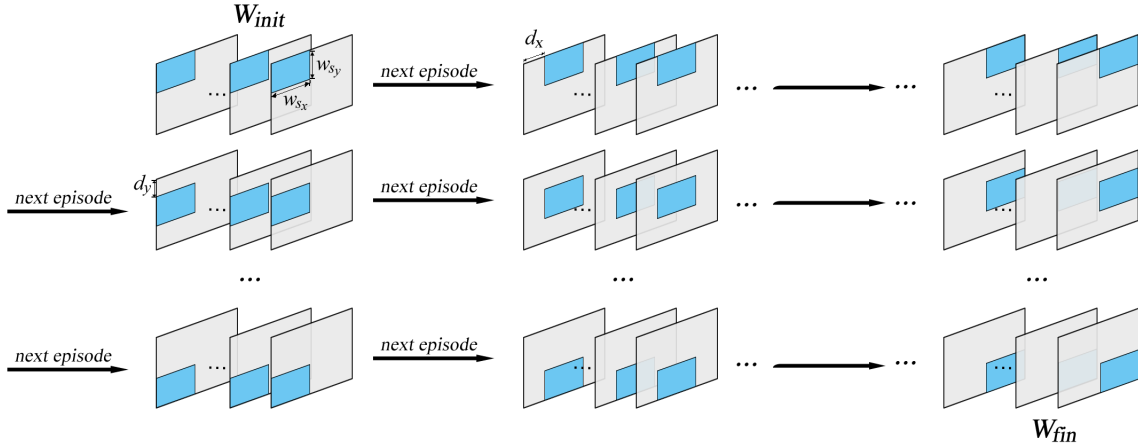


Figure 3: The sliding process of W . W_{init} : attack window at the beginning of the observation; W_{fin} : attack window at the end of the observation.

manipulating a feature with higher reward impact reduces the cumulative reward more significantly. As a result, the features with the largest reward impact $RI(f_{(x,y)})$ is selected to form a target feature set F_A in our attack method. The cardinality of F_A depends on the attack cost. This process can be considered as a feature selection method before the attack, which is different from other existing attack methods in which the attack features are selected differently in crafting a sample.

The selected feature $f \in F_A$ of each observation in a state is perturbed by a small value according to the maximum perturbation

(α) and the gradient function of the targeted agent, i.e., $\tilde{f}_{(x,y)}^t = f_{(x,y)}^t + \alpha * \text{sign}(\nabla_{f_{(x,y)}^t} J(\theta, s_t, a_t))$, where $\tilde{f}$ is the attacked f .

Since the attack features are selected according to the reward map, SRIMA only requires to calculate the gradient of features in F_A . This is different from the other existing attack methods which calculate the gradient for all features in each attack. The time complexity of crafting adversarial samples by SRIMA is lower than that by other existing methods in real-time. Although the computational complexity increases due to the static reward impact map estimation in SRIMA, it can be carried out offline, i.e., the calculation of

the reward impact map can be done before the attack. For adversarial attacks, the time complexity of crafting adversarial samples in real-time is more significant than the offline one. Therefore, the time complexity of our method may not be a critical factor.

4 EXPERIMENT

The proposed method is evaluated experimentally using DQN and PPO on six Atari games, *i.e.*, Freeway, Pong, Breakout, FishingDerby, SpaceInvaders and BattleZone. We first introduce the experimental settings. Experimental results of the static reward impact analysis, the performance of the methods in both white-box and black-box attack are then discussed.

4.1 Experimental Setting

ALL the experiments are performed on a server whose GPU is GTX 1080 with 8GB of memory and 1607 MHz base clock. The same pre-processing in [14] is used in all the Atari games. Each observation is converted from RGB to luminance and is resized to 84×84 . The last four observations are concatenated as an input state, *i.e.*, $s_t = \phi(o_{t-(i-1)}, \dots, o_{t-1}, o_t)$. All input values are linearly normalized to $[0, 1]$.

The network structures of all agents used in this paper are shown in Table 1. DQN₃, with the same architecture and training process as [14], is used as the targeted agent due to its popularity. Five surrogate agents are considered in the black-box attack setting in different experiments. The learning model of three surrogate agents (sDQN_{*i*}, where $i = 2, 3, \text{and } 4$) is as the same as the one used by the targeted model, while other two agent is based on PPO [18], *i.e.*, sPPO_{*i*}, where $i = 3, 4$. i indicates the number of layers used in the model. One should be noted that although the structures of DQN₃ and sDQN₃ are the same, the agents will not be identical due to the random initialization. The detail of the settings of each model is shown in Table 1.

	conv_1	conv_2	conv_3	conv_4	full
sDQN ₂	32,8,4	64,4,1	/	/	512
sDQN ₃ , DQN ₃	32,8,4	64,4,2	64,3,1	/	512
sDQN ₄	64,16,4	64,8,4	128,4,2	512,3,1	1024
sPPO ₃	32,8,4	64,4,2	64,3,1	/	512
sPPO ₄	64,16,4	64,8,4	128,4,2	512,3,1	1024

Table 1: The network structure of different RL agents. “conv_*i*” and “full” denote the *i*-th convolutional layer and the fully-connect layer respectively. (l, m, n) in each cell represents the layer convolves l number of $m \times m$ filters with stride n . “/” denotes this layer does not exist.

In our method, a static reward impact map is estimated first on the targeted or the surrogate agent. The preliminary evaluation is carried out to tune the parameters, aiming to minimize the cumulative reward, for $(d_x, d_y) \in \{(1, 1), (3, 3), (5, 5)\}$, $M \in \{5, 10, 15\}$ and $w_s \in \{(34, 34), (16, 72), (72, 16), (30, 30), (14, 63), (63, 14), (24, 24), (12, 50), (50, 12)\}$, which represent different sizes and shapes. The preliminary results indicate that the performance of our method is more sensitive to w_s than (d_x, d_y) and M . A preliminary experiment

is carried out to evaluate the candidates of (d_x, d_y) , M and w_s . The candidates yields the best results is selected. According to the results of the preliminary experiments, $w_s = (63, 14), (72, 16), (34, 34), (30, 30), (34, 34)$, and $(24, 24)$ are set for Freeway, Pong, Breakout, FishingDerby, SpaceInvaders and BattleZone respectively. The vertical rectangles are chosen in Freeway and Pong. It may be because of the vertical movement of the players. Since the fishing line is swayed inside a square region in FishingDerby and the enemies are similar to squares in BattleZone, the squared windows are chosen. Although the paddle in Breakout and the rocket in SpaceInvaders moves horizontally, the windows with the square shape are selected in order to take rebound of the ball in Breakout and the forts in SpaceInvaders into consideration. The window sizes of Pong, Breakout and SpaceInvaders are larger than that of Freeway and FishingDerby. It may due to the size of the main character, *i.e.*, the characters in Pong, Breakout, and SpaceInvaders are larger than the ones in Freeway, FishingDerby and BattleZone. $(d_x, d_y) = (3, 3)$ and $M = 10$ are set for all tasks. Each attack result is carried out on ten episodes independently.

Only one-time attack (FGSM) and the random method (RND), are considered in our experiments since the short response time requirement of crafting samples in RL. The ℓ_1 -norm restricted FGSM is used since it has stronger attack ability than ℓ_2 -norm and ℓ_∞ -norm restricted ones [7]. For each sample s , $J(\theta, s, a)$ denotes the cross-entropy loss between all actions a and the distribution that places all weight on the highest-weighted action. ℓ_1 -norm restricted FGSM perturbs the features f with highest $|\nabla_f J(\theta, s, a)|$. On the other hand, the attack features are selected randomly by RND. The only difference between the attack methods in our setting is the set of the attack features. The attack features selected by all methods are all perturbed according to the gradient of the loss function by the value of α , *i.e.*, $\tilde{f} = f + \alpha * \text{sign}(\nabla_f J(\theta, s, a))$, where $\tilde{f}$ is the attacked f .

For an adversarial sample, ℓ_1 -norm [5] is used to measure the extent of the perturbation (*i.e.*, attack strength ϵ). This constraint accounts for the total variation of the adversarial noise. The attack strength ϵ is defined as follows:

$$\epsilon = \|s - \tilde{s}\|_1 = \alpha * \frac{c}{N} \quad (4)$$

where α is the maximum perturbation on each feature, c is the number of attack features and N is the total number of features.

Two evaluation metrics are applied to measure the influence of an attack method to an agent. The drop of the cumulative reward [7] is one metric which quantifies the quality of the decisions of an agent affected by the attack. The other one is the percentage of successfully misled actions [1], which quantifies the ability of the attack in misleading actions of the targeted agent.

4.2 Static Reward Impact Map Analysis

This section discusses the information provided by our proposed static reward impact map. Static reward impact maps of different targeted agents on each game are shown in Figure 4. The figures from left to right are the screenshot of the game, and the reward impact maps of sDQN₂, DQN₃, sDQN₄, sPPO₃, and sPPO₄. All values of reward impact are normalized to $[0, 1]$ linearly. The color of a pixel indicates its value of reward impact as shown by the color-bar.

Some regions provide more important information in decision making than others in games. For example, the features near to the route of the chicken in Freeway, the vertical region near to the movable paddles in Pong, the features near to the horizontal region close to the paddle in Breakout, the features near to the fisherman and his fishing line in FishingDerby, the vertical region of the rocket's moving path in SpaceInvaders, and the features near to the tank in BattleZone. This is because when a vehicle near to the chicken in Freeway, the ball near to the paddles in Pong and Breakout, the fish near to the fishing line in FishingDerby, the bullets near to the rocket in SpaceInvaders, and the enemies near to the tank in BattleZone, an agent may not have enough time for recovering from a wrong decision and causes a serious punishment. When an adversarial attack perturbs the features in these important regions, the cumulative reward of an agent can be reduced dramatically. It explains the reason why the regions with high static reward impact are consistent with the important regions.

Moreover, reward impact maps of different agents in a game are highly similar to each other as shown in Figure 4. Since a static reward impact map indicates how sensitive of changing features to the cumulative reward, the maps for a game are expected to be similar by assuming the agents make decisions reasonably. It suggests a static reward impact map does not depend on an agent but a game. This characteristic allows an adversary to effectively attack an agent without its knowledge.

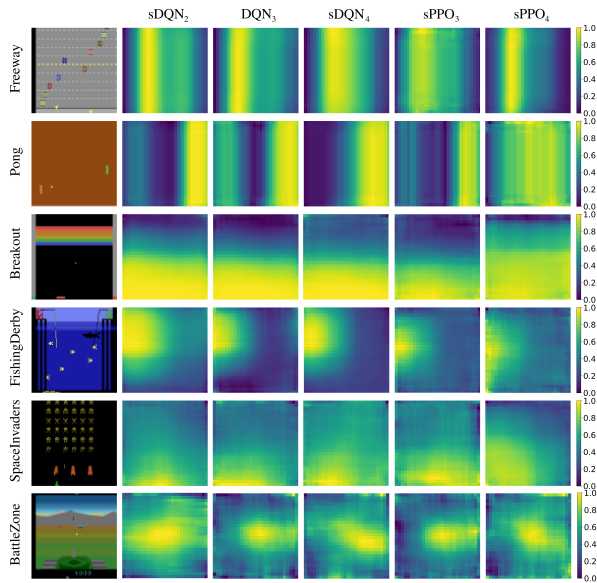


Figure 4: Static reward impact maps of different agents. From left to right: the screen shot of the game, the maps of sDQN₂, DQN₃, sDQN₄, sPPO₃ and sPPO₄.

4.3 Attack Ability

The attack ability of *SRIMA*, *FGSM* and *RND* with different attack strength (ϵ) in white and black-box attack are shown in Figures

5 and 6 in terms of the cumulative reward and the successful attack rate (SAR), *i.e.*, the percentage of actions misled in an episode. The level of attack strength (ϵ) is represented by the x-axis in the figures. The results in red, green and blue represent *SRIMA*, *FGSM* and *RND* respectively. The figures in a line from top to bottom are the results of the white-box attack, the black-box attack with sDQN₄ as surrogate, and the black-box attack with sPPO₃ as surrogate respectively

Figure 5 shows that the cumulative reward of the targeted agent drops significantly under our attack even with small attack strength under the white-box attack setting. The red line drops sharply from 29.4 to 0 at $\epsilon = 5 \times 10^{-4}$ in Freeway, from 21 to -21 at $\epsilon = 7 \times 10^{-4}$ in Pong, from 490 to 15 at $\epsilon = 1 \times 10^{-4}$ in Breakout, from 7.2 to -65 at $\epsilon = 5 \times 10^{-5}$ in FishingDerby, from 2132 to 260 at $\epsilon = 4 \times 10^{-5}$ in SpaceInvaders, and from 23200 to 5000 at $\epsilon = 1.3 \times 10^{-4}$ in BattleZone. After the drop, the reward remains constantly around a small value when the attack strength increases. These results suggest that *SRIMA* is able to choose the most effective attack features to downgrade the targeted agents with small attack strength. Differently, the rewards of the agents under *FGSM* and *RND* drop more slowly than *SRIMA*. In the black-box attack setting, the difference between the surrogate and the targeted agents reduces the influence of the attacks. The performance of all attack methods is worse in comparison with the ones under the white-box setting. However, our method still achieves the best performance, *i.e.*, the reward is the lowest, among all methods. The results indicate that our method is more effective than *FGSM* and *RND* in reducing the cumulative reward of the targeted agent under both white-box and black-box attack settings.

Generally, SAR of *SRIMA* is higher than *FGSM* and *RND* with the same level of attack strength under both white-box and black-box attack settings. It may be because that *SRIMA* considers the dependency of features in the attack windows when calculating the reward impact map, while *FGSM* and *RND* do not. SARs of all methods are reduced in the black-box attack setting in comparison to the ones in the white-box attack setting respectively. For example, *SRIMA* achieves to 100% at $\epsilon = 0.002$ in both Freeway and Pong, around 40% at $\epsilon = 4 \times 10^{-5}$ in FishingDerby, and around 90% at $\epsilon = 5 \times 10^{-5}$ in BattleZone with large attack strength under the white-box attack setting, but the rates drop to 40% in Freeway, to 80% in Pong, to 20% in FishingDerby, and to 60% in BattleZone under the black-box attack.

It should be noted that SARs of *SRIMA* and *FGSM* are similar in some cases, *e.g.*, Pong and SpaceInvaders, under the both white-box and black-box attack settings. However, the cumulative reward of the targeted agent under the attack of *SRIMA* is lower than the one under the attack of *FGSM* in these cases. One possible reason is that *SRIMA* misleads the agents in the critical states affecting the cumulative reward more frequently than *FGSM*. In addition, in some cases in Breakout, the reduction of the reward is similar under both *SRIMA* and *FGSM* while SAR under *SRIMA* is larger than that under *FGSM*. One possible reason is that the ball may rebound from the bricks or between the walls and bricks in Breakout. Any action in this scenario has less influence on the reward.

SRIMA achieves excellent performance in comparison with *FGSM* and *RND* in Freeway. It may be because tracks of objects in Freeway are relatively fixed comparing with the ones in other games, *e.g.*, the

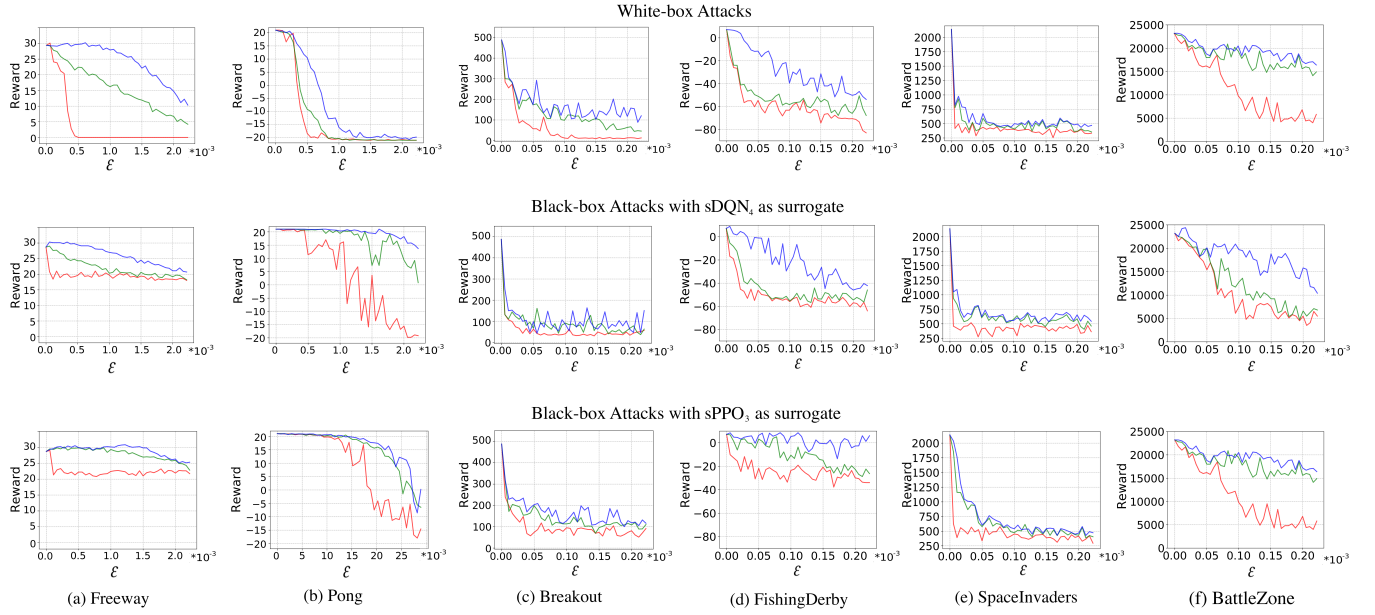


Figure 5: Cumulative reward attacked by *SRIMA* (red), *FGSM* (green), *RND* (blue) with different attack strength under the white-box (top) and the black-box attack with sDQN₄ as surrogate (middle) and black-box attack with sPPO₃ as surrogate (bottom).

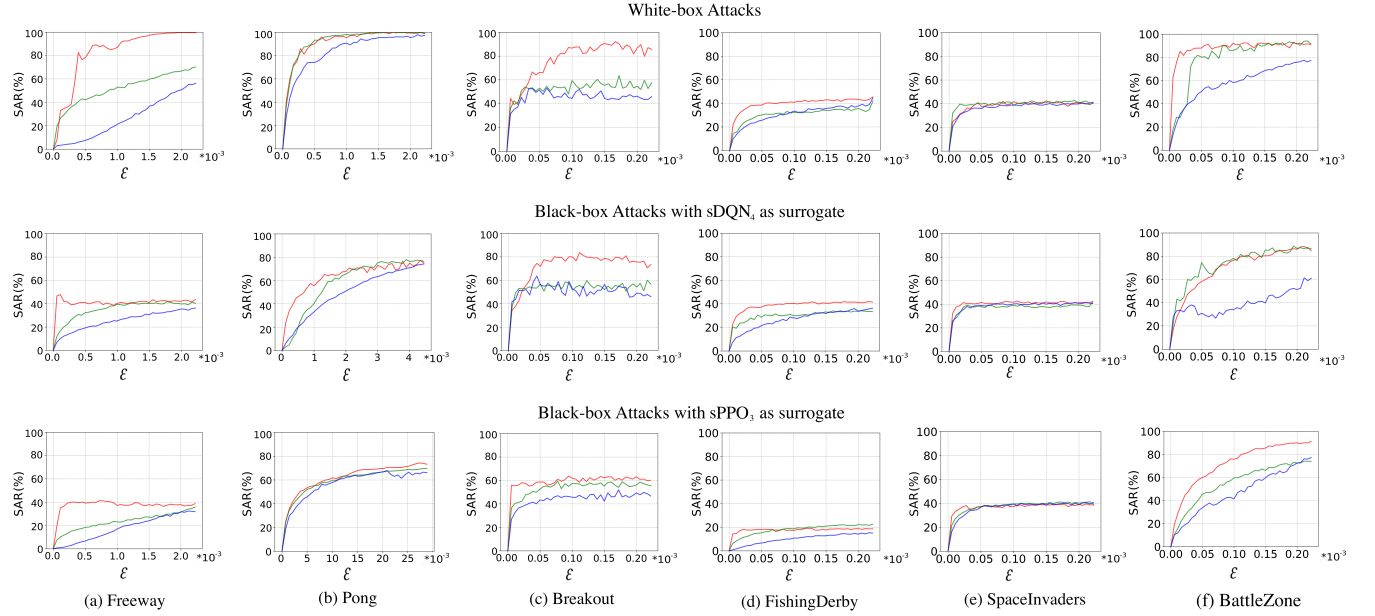


Figure 6: SARs attacked by *SRIMA* (red), *FGSM* (green), *RND* (blue) with different attack strength under the white-box (top) and the black-box attack with sDQN₄ as surrogate (middle) and black-box attack with sPPO₃ as surrogate (bottom).

cars move horizontally in lanes in Freeway but the ball can be moved in any direction in Pong and Breakout. The similar situations to Pong and Breakout can be observed in FishingDerby, SpaceInvaders and BattleZone. As the attack features are fixed in *SRIMA*, it may

have better performance on the games containing objects with less complicated movement.

To demonstrate the influence of a surrogate agent on the attack in the black-box attack setting, five surrogate agents including sDQN₂, sDQN₃, sDQN₄, sPPO₃, and sPPO₄ are trained to indicate different

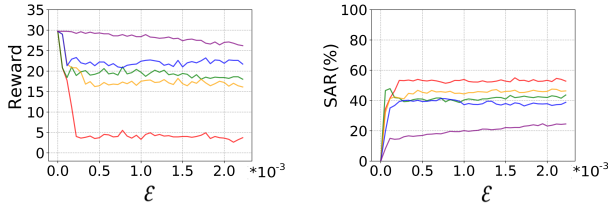


Figure 7: Cumulative reward (left) and SARs (right) attacked by *SRIMA* with different attack strength under the black-box attack with sDQN₂ (yellow), sDQN₃ (red), sDQN₄ (green), sPPO₃ (blue), and sPPO₄ (purple) as surrogate respectively, in Atari game Freeway.

levels of knowledge obtained by the adversary. The most knowledge is obtained in sDQN₃ as the surrogate agent has the same structure and learning algorithm as the targeted agent, while only the learning algorithm is the same in sDQN₂ and sDQN₄. Both sPPO₃ and sPPO₄ use different learning algorithm from the targeted agent. The structure of sPPO₃ is as the same as the targeted one. One should be noted that although the structures of sDQN₃ and the targeted agent are the same, these two model are still different caused by random initialization. sDQN₂, sDQN₃ and sDQN₄ are trained for 50 million steps as [14], while sPPO₃ and sPPO₄ are trained for 40 million steps as [18]. All these surrogate agents achieve similar performance as the targeted agent, such as, the cumulative rewards of the targeted agent, sDQN₂, sDQN₃, sDQN₄, sPPO₃, and sPPO₄ are 29.7, 29.3, 29.4, 30.7, 29.0 and 30.1 respectively.

Figure 7 shows the performance of *SRIMA* with different attack strength against different surrogate agents in terms of the cumulative reward (left) and SARs (right) in Freeway. Unsurprisingly, a surrogate agent more similar to the targeted one achieves higher attack ability. The cumulative reward drops significantly to 4.5 under black-box attack with sDQN₃ at $\epsilon = 3 \times 10^{-4}$, while the rewards of sDQN₂, sDQN₄, sPPO₃ and sPPO₄ only drop to 17, 20, 22 and 29.5 respectively. Moreover, the attack performance of sPPO₃ and sPPO₄ is worse than others. This suggests that the knowledge of the learning algorithm is more important than the network architecture in terms of the influence to attack. The worst performance is achieved by sPPO₄ as it has the least knowledge about the targeted agent.

4.4 Time Complexity

The time complexity of an attack can be separated into two parts including the preparation and the crafting phases. As an adversarial sample is crafted in real time, the time complexity of the crafting phase is more critical than the one of the preparation phase. The average running time of calculating static reward impact maps of our method for Freeway, Pong, Breakout, FishingDerby, SpaceInvaders and BattleZone is 16.70 hour, while no preparation is needed by *FGSM* and *RND*.

The average running times (in ms) of crafting an adversarial sample with different numbers of attack feature are showed in Table 2. The running time of each perturbation relies on the attack feature selection and the feature value perturbation. When more

features are attacked, more features should be selected and perturbed. Therefore, the running time is proportional to the number of attack features. As all attack features are selected according to the static reward impact map, the running time of *SRIMA* is the smallest among all methods. The time required by *RND* is slightly longer than the one of *SRIMA* due to the random feature selection process. However, *FGSM* needs the longest time on crafting an adversarial sample since the attack features are selected in real time. It nearly takes ten times of *SRIMA* and *RND*.

	1%	5%	10%	20%	50%
<i>SRIMA</i>	0.37	1.29	2.51	4.91	12.70
<i>FGSM</i>	6.18	14.43	25.32	45.49	105.80
<i>RND</i>	0.46	2.01	4.06	8.01	19.38

Table 2: The average running times (ms) of crafting an adversarial sample according to different methods with different numbers of attack feature in the white-box attack setting.

5 CONCLUSION AND FUTURE WORK

Current attack methods against Deep RL are not designed specifically to prevent an agent from achieving its ultimate goal, *i.e.*, maximizing its cumulative reward. This study proposes an attack method aiming to reduce the cumulative reward achieved by the targeted agent directly. We first investigate the influence of feature change on the cumulative reward in order to understand which regions of the input space are more important in the decision-making. By focusing on the tasks in which the importance of features is similar in any state, the static reward impact map is then introduced to quantify the influence on the reward for each feature using the sliding-window method experimentally. The influence correlation of features in each window on the reward is considered. The features with the largest reward impact are perturbed according to the gradient function of the targeted agent when crafting an adversarial sample. The experimental results confirm that the proposed attack model outperforms *FGSM* and the random attack method in Freeway, Pong, Breakout, FishingDerby, SpaceInvaders and BattleZone in terms of effectiveness and efficiency. The results of this study provide a foundation for devising a more advanced attack strategy against RL in various applications, for understanding the potential vulnerabilities of RL, and for developing more secure RL algorithms.

One limitation of our method is the time complexity of the static reward impact map estimation, although it, as the same as other attack methods, can be carried out offline. We will try to propose a more effective algorithm to go through the features rather than sliding-windows in order to reduce the time complexity. Another future work may relax the static assumption on changing features to the reward since this assumption may not be valid, especially for complicated tasks. A possible solution is the dynamic reward impact map which measures the influence on the reward of feature change for different states. However, time complexity is a concern. A robust Deep RL method by reducing the variance of reward impact values on all features in order to increase the attack difficulty can also be further investigated.

6 ACKNOWLEDGMENTS

This paper is supported by the Natural Science Foundation of Guangdong Province, China (No. 2018A030313203), and the Fundamental Research Funds for the Central Universities (No. 2018ZD32).

REFERENCES

- [1] Vahid Behzadan and Arslan Munir. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. In *Proceedings of the Thirteenth International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, Berlin, German, 262–275.
- [2] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering* 26, 4 (2014), 984–996.
- [3] Patrick P. K. Chan, Xian Hu, Lili Zhao, Daniel S Yeung, Dapeng Liu, and Lei Xiao. 2018. Convolutional Neural Networks based Click-Through Rate Prediction with Multiple Feature Sequences. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, 2007–2013.
- [4] Patrick P. K. Chan, Zhe Lin, Xian Hu, Eric C. C. Tsang, and Daniel S. Yeung. 2017. Sensitivity based Robust Learning for Stacked Autoencoder against Evasion Attack. *Neurocomputing* 267 (2017), 572–580.
- [5] Pin Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho Jui Hsieh, Pin Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho Jui Hsieh. 2017. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. (2017).
- [6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the Third International Conference on Learning Representations*.
- [7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. In *ICLR Workshop*.
- [8] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. 2017. Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning. *arXiv preprint arXiv:1705.01196* (2017).
- [9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Twenty-Sixth International Conference on Neural Information Processing Systems*. Curran Associates., New York, NY, 1097–1105.
- [10] Tai Lei and Liu Ming. 2016. A robot exploration strategy based on q-learning network. In *Proceedings of the 2016 IEEE International Conference on Real-time Computing and Robotics*. IEEE, Piscataway, NJ, 57–62.
- [11] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.
- [12] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. 2017. Predicting deeper into the future of semantic segmentation. In *Proceedings of the 2017 IEEE International Conference on Computer Vision*, Vol. 1. IEEE Computer Society, Los Alamitos, CA.
- [13] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the Thirty-Third International Conference on Machine Learning*. JMLR.org, Cambridge, MA, 1928–1937.
- [14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
- [15] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, New York, NY, 506–519.
- [16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Proceedings of the 2016 IEEE European Symposium on Security and Privacy*. IEEE, Piscataway, NJ, 372–387.
- [17] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommanan, and Girish Chowdhary. 2017. Robust Deep Reinforcement Learning with Adversarial Attacks. (12 2017).
- [18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (2017).
- [19] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.
- [20] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.
- [21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the Second International Conference on Learning Representations*.
- [22] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College, Cambridge.
- [23] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the Thirty-Second International Conference on Machine Learning*. JMLR.org, Cambridge, MA, 2048–2057.
- [24] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation*. IEEE, Piscataway, NJ, 3357–3364.