# Defeating the Non-stationary Opponent Using Deep Reinforcement Learning and Opponent Modeling

Qian Yao[1,2], Xinli Xiong[1,2], Peng Wang[1,2], and Yongjie Wang[1,2(✉)]

[1] College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China
{yaoqian21,xiongxinli_,wangpeng21e,wangyongjie17}@nudt.edu.cn
[2] Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation, Hefei 230037, China

**Abstract.** In the cyber attack and defense process, the opponent's strategy is often dynamic, random, and uncertain. Especially in an advanced persistent threat scenario, it is not easy to capture its behavior strategy when confronted with a long-term latent, highly dynamic and unpredictable opponent. FlipIt game can model the stealth interaction of advanced persistent threat. However, it is insufficient for traditional reinforcement learning approach to solve real-time and non-stationary game model. Therefore, how to model a non-stationary opponent implicitly and keep the defense agent's advantage continuously is essential. In this paper, we propose an extended FlipIt game model incorporating opponent modeling. And then we propose an approach that combines deep reinforcement learning, opponent modeling, and dropout technology to perceive the behavior of a non-stationary opponent and defeat it. Instead of explicitly identifying the opponent's intention, the defense agent observes the opponent's last move actions from the game environment, stores the information in its knowledge, then perceives the opponent's strategy and finally makes a decision to maximize its benefits. We show the excellent performance of our approach whether the opponent adopts traditional, random or composite strategies. The experimental results demonstrated that our approach can perceive the opponent quickly and maintain the superiority of suppressing the opponent.

**Keywords:** Deep reinforcement learning · Opponent modeling · FlipIt game · Non-stationary environment

## 1 Introduction

In recent years, cyberspace incidents have occurred frequently. One of the most serious attacks is advanced persistent threats (APT) [21]. Through long-term reconnaissance and investigation of the target, APT attackers usually adjust their attack policies correspondingly. The automated and intelligent cyber attack


H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 57–78, 2024.
https://doi.org/10.1007/978-3-031-54528-3_4

tools [1,11,22] constantly emerging. If attackers exploit these tools to launch APT attacks, it will bring disastrous consequences.

It is essential to model APT attacks and study defense strategies against intelligent attacks. FlipIt game [12,23] can model the continuous cyber attack and defense process, which can characterize the stealth interaction characteristics of APT attacks. In a FlipIt game, the attacker and defender compete for sensitive resources without knowing the opponent's behavior. Some researchers have improved the modeling of FlipIt game [10,27]. However, the modeling of highly dynamic and non-stationary opponents is insufficient. Therefore, We proposed an extended FlipIt game model combined with opponent modeling.

As the cyber modeling becomes more fine-grained, the complexity problem comes with it. When confronted with the real-time and complex cyber attacks, it is not easy for humans to make correct decisions immediately. Therefore, the solution method [5,15,28] of FlipIt game is also a research hotspot. At present, it's recommended to train a reinforcement learning agent to execute the cyber defense decision-making tasks. On the one hand, reinforcement learning agent can observe and interact with the game environment, and then make sequential decisions without prior knowledge. On the other hand, reinforcement learning agent has shown excellent performance in both human-agent confrontation [18] and agent-agent confrontation [20]. However, the classic reinforcement learning algorithms have a shared limitation of over-fitting and over-estimation, and cannot adapt to the non-stationary game environment. In this work, we proposed an improved Deep Q-Network (DQN) [14] approach to learn the opponent's strategy in a FlipIt game.

At the same time, the FlipIt game environment is non-stationary due to the influence of the opponent's behavior. Opponent modeling [6,7,26] are the main solutions to deal with the non-stationary problem. It mainly refers to modeling the behavior of the opponent by using interactive information and exploring the opponent's weakness.

Overall, due to intelligent attack tools, APT attacks may become more unpredictable and disastrous. FlipIt game can be used to model APT attacks. However, the exited FlipIt game models without modeling the non-stationary opponent well. Reinforcement learning is widely used to solve FlipIt game model, but the shortcomings such as over-fitting and over-estimation need to be further improved.

In this paper, we construct an extended FlipIt game model with opponent modeling. And we incorporate an perception vector into DQN, which stores the observation of the opponent's last move actions into the knowledge of the defense agent. And then defense agent analyzes the opponent's strategy to maintain a long-term advantage over the opponent. For adapting to the non-stationary game environment and preventing the over-fitting problem of the reinforcement learning, we adopt the dropout technology in the DQN. Simultaneously, we adopt Constrained Q-Learning to overcome the over-estimation problem. Meanwhile, the proposed method can be further extended to multi-agent collaborative decision-making problems. The contributions of this work are as follows:

- Given the cyber attack and defense environment is dynamic and random, we introduce a controllable random factor and propose random strategies for the FlipIt game. To adapt the non-stationary game environment, we construct an extended FlipIt game model with opponent modeling, called FlipIt-OM.
- We propose an improved deep reinforcement learning approach to solve the FlipIt-OM game, called ND3QN. ND3QN combines NoisyNet, Dueling Q-Network, Dropout technology and Constrained Q-Learning. It solves the problems of over-estimation, over-fitting and insufficient exploration.
- The experimental results show the superior performance of ND3QN to the baseline DQN, PER-DQN. Through stealth interaction with the environment, ND3QN agent can model the opponent's strategy well and maintain its advantage over its opponents. ND3QN is more suitable for the non-stationary game environment.

The rest of this paper is structured as follows. Section 2 introduced the related works about FlipIt game and opponent modeling. Section 3 presented the extended FlipIt game model. Section 4 elaborated on the basic architecture and technical implementation details. Section 5 analyzed the experimental results. Section 6 summarized the paper and pointed out the future research direction.

## 2 Related Work

FlipIt game aims at simulating an APT-like scenario. The defender and opponent fight for controlling a sensitive resource, such as a private key, a password, etc. Laszka et al. [10] introduced two control models (AND and OR models) and proposed the FlipThem model. The attacker tries to destroy one or all of the resources. The existing FlipIt game models without considering model a non-stationary opponent.

It is a research hotspot to solve the FlipIt game with reinforcement learning algorithms. Lisa et al. [15] introduced QFlip, an adaptive strategy based on Q-Learning for the FlipIt game. Deep reinforcement learning integrates the powerful perceiving ability of deep learning (DL) and decision-making ability of reinforcement learning (RL). Laura et al. [5] used DQN to maximize the defender's rewards in a FlipIt game, and extended to an n-player security game. However, a common limitation is that Q-Learning or DQN can't adapt to the non-stationary environment well. Zhu et al. [28] presented adaptive DQN to apply to a non-stationary FlipIt game. The approach can realize the rebalance of exploration and exploitation quickly by tracking the variance changes of Q-value distribution. However, this work only considers the non-stationary problem of the game environment, the opponent still adopts the fixed period and exponential strategy. Furthermore, our proposed FlipIt-OM models a non-stationary opponent.

There are inevitable defects in the classic RL approach, such as over-fitting, over-estimation and insufficient exploration. Dropout [19] is proposed to overcome the over-fitting problem. Dropout drops some units from the neural network randomly during the training process, which breaks up the co-adaptations

among specific hidden units. Co-adaptation means that the units depend on each other. Conservative Q-Learning [9] and Double Q-Learning [24] are proposed to address the over-estimation problem. To enhance the exploration ability of the agent to induce stochasticity of its strategy, NoisyNet [4] adds Gaussian noise to the neural network. Dueling Q-Network [25] divides the last layer in two streams. One of them estimates the state value function for state $s$ and the other one estimates the advantage function for each action $a$. Finally, both parts are combined into a single stream to calculate the Q-values.

Given the non-stationary problem, the researchers pay attention to the method of opponent modeling, including explicit and implicit modeling categories. Explicit modeling identifies the opponent's behavior directly. Richard et al. [2] combined DRL and opponent modeling and proposed a Switching Agent Model. Jakob et al. [3] presents Learning with Opponent-Learning Awareness(LOLA). Yuxi et al. [13] proposed an opponent portrait approach using MADDPG in a competitive environment. Roberta et al. [16] proposed Self Other-Modeling. An agent can predict the other agent's actions. Zhang et al. [8] added a policy inferring module into the DQN, called DPIQN. DPIQN models the observation of opponents into the Q-value learning process. While implicit modeling attempts to take a strategy through implicit reasoning. He et al. [6] presented deep reinforcement opponent network and encoded the opponent's behavior into DQN, instead explicitly predicting the opponent's action. In this paper, ND3QN perceives the strategy of opponent and stores the opponent's behavior data, and then makes defensive decisions.

In summary, RL is the mainstream method to solve the FlipIt game. However, RL has the problem of over-fitting, which can't adapt to the non-stationary game environment. In addition, the existing FlipIt game models without considering model the non-stationary opponent. And we often can't accurately identify the opponent's behavior, especially in the APT-like scenario. How to implicitly perceive and reason the opponent's policy through stealth interaction with the game environment is worth to research.

## 3 FlipIt-OM Game Model

In this Section, we introduce the proposed FlipIt-OM game model. Section 3.1 makes the assumptions of the FlipIt-OM game model. Section 3.2 elaborates on the FlipIt-OM model. Section 3.3 introduces the strategies adopted by the opponent.

### 3.1 Assumptions

FlipIt-OM game model makes the following assumptions.

- Assuming that the defense agent can observe the last move actions of the opponent.
- Assuming that when the defense agent and the opponent compete for resources at the same time, the control power belongs to the defense agent.

### 3.2 Model

We adopt the Last Move (LM) [23] version of the FlipIt game. The players are in stealth interaction with the environment. Specifically, the defense agent and opponent can neither know who controls the sensitive resource currently, nor observe all the actions of the opponent, they can only capture the opponent's last flip action. The illustration of the FlipIt game (LM) is shown in Fig. 1.

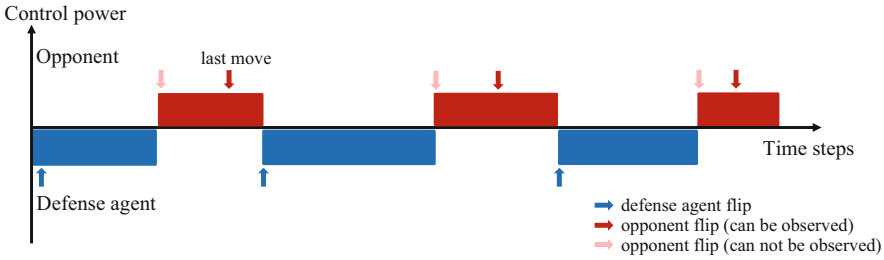



**Fig. 1.** illustration of FlipIt-OM game (LM)

Here, the red area indicates the time steps that the opponent controls, and the blue area indicates the defense agent controls. Based on the LM observation mechanism, the light red flip will never be perceived by the defense agent, while the dark red flip opportunity can be caught. For each step the agent decides to flip, it will control the resource and know the opponent's last flip opportunity. And it will spend flip cost at the same time.

In the FlipIt game (LM), $\mathcal{N}^d$ is a defense agent and $\mathcal{N}^o$ is the opponent. $T^i_{LM}$ is the time interval since the last flip of $\mathcal{N}^i$ at step $t$. $\mathcal{N}^d$ has full knowledge of $T^d_{LM}$, but only has last-move knowledge of $T^o_{LM}$. Then, $\mathcal{N}^d$ can perceive the opponent's strategy $\pi^{\rho^o}$ from observed opponent actions $a^o_t$. $\rho$ is a perceptual vector. Finally, $\mathcal{N}^d$ decide whether to flip the resources at each step $t$ based on the perceived knowledge $\pi^{\rho^o}$.

FlipIt-OM game models the opponent into FlipIt game. In a FlipIt-OM game, the observation space is influenced by the joint action of both sides. FlipIt-OM game model can be defined as a 8-tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \tau, \mathcal{R}, \gamma, \mathcal{K}, \pi)$.

- $\mathcal{N} = \{\mathcal{N}^d, \mathcal{N}^o\}$ represents the defense agent and opponent respectively.
- $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$ denotes the states observed by the defense agent $\mathcal{N}^d$, represented by $s_t = (T^d_{LM}, T^o_{LM} \mid \pi^{\rho^o}) \in \mathcal{S}$.
- $\mathcal{A} = \{\mathcal{A}^d, \mathcal{A}^o\}$, $\mathcal{A}^d$ and $\mathcal{A}^o$ represent the action space available to the defense agent and opponent, denoted as $a^d_t = \{flip, notflip\} \in \mathcal{A}^d$, $a^o_t = \{flip, notflip\} \in \mathcal{A}^o$.
- $\tau\{s, a^d, a^o, s'\}$ represents the transferring probability to a state $s'$ from the state $s$ when the defense agent selects $a^d$ and opponent selects $a^o$ .
- $\mathcal{R} = \{\mathcal{R}^d, \mathcal{R}^o\}$, $\mathcal{R}^d(s, a^d, a^o, s')$ and $\mathcal{R}^o(s, a^d, a^o, s')$ represent the reward of defense agent and opponent respectively. Three situations [28] are as follows. (1) If a player decides not to flip, the reward is empty. (2) If a player decides

to flip and $T^d_{LM} \leq T^o_{LM}$, the player will not obtain benefits $bef$ but need to spend the cost $cos$, for the reason that the control power originally belongs to it. (3) If a player decides to flip and $T^d_{LM} > T^o_{LM}$, the player regains control from opponent $\mathcal{N}^o$. Then the player will obtain the benefits $bef$ and spend the cost $cos$ at the same time. In conclusion, $r_t$ is represented as follows.

$$r_t = \begin{cases} 0 & if\ a_t\ =\ not\ flip \\ -cos & if\ a_t\ =\ flip\ and\ T^d_{LM} \leq T^o_{LM} \\ bef - cos & if\ a_t\ =\ flip\ and\ T^d_{LM} > T^o_{LM} \end{cases} \tag{1}$$

- $\gamma \in [0, 1)$ is the discount factor, which decides whether the defense agent pays attention to short-term interests or long-term interests.
- $\mathcal{K}$ is the knowledge about opponent. Defense agent stores the opponent's last flipping opportunity into knowledge $\mathcal{K}$. Thus the observation space of the defense agent is expanded.
- $\pi = \left\{\pi^d, \pi^o\right\}$ denotes the flipping strategy decided by defense agent and opponent respectively. The defense agent perceives the opponent's flip strategy $\pi^{\rho^o}$ through the knowledge $\mathcal{K}$. Then defense agent makes decisions $\pi^d$ correspondingly.

The descriptions of the symbols used in this paper are shown in Table 1.

**Table 1.** The descriptions of symbols used in this paper.

| Symbols | Descriptions |
|---|---|
| $S_t$ | state space |
| $A_t$ | action space |
| $R_t$ | reward function |
| $Q(s, t)$ | action-value function |
| $\tau$ | state transferring probability |
| $\gamma$ | discount factor |
| $\rho$ | perception vector |
| $\mathcal{K}$ | knowledge about opponent |
| $\pi^o$ | opponent strategy |
| $\pi^d$ | defense agent strategy |
| $a^o$ | opponent actions |
| $a^d$ | defense agent actions |
| $\delta$ | flipping with fixed $\delta$ steps (period strategy) |
| $\lambda$ | flipping with an exponential distribution (exponential strategy) |
| $\zeta$ | flipping with fixed $\delta + \zeta$ steps (random period strategy) |
| $c$ | a constant floating number $c \in [0, 1)$ (random strategy) |
| $q$ | a random floating number $q \in [0, 1)$ (random strategy) |

The FlipIt-OM game model perceives the opponent's actions and stores the perceived information in its knowledge, therefore, the state space of the RL agent is expanded. It means that the state space observed by the defense agent is non-stationary. The Q-function of the defense agent is dependent on the strategy of the opponent. It is necessary to adjust the definition of Q-function so that the actions of other agents are considered. We consider a 1 vs. 1 scenario in a FlipIt game. The defense agent needs to perceive the opponent's strategy $\pi^o$ by observing its actions $a^o$. The Q function can be represented as Eq. 2.

$$Q(s, a|\pi^o) = \sum_{a^o} \pi^o(a^o|s) \sum_{s'} \tau\left(s, a^d, a^o, s'\right) \left[R(s, a^d, a^o, s') + \gamma E_{a'}\left[Q(s', a' \mid \pi^o)\right]\right] \tag{2}$$

The FlipIt-OM game model is updated by batch gradient descent to minimize the Mean Squared Error loss.

### 3.3 Opponent Strategies

Traditional strategies of the FlipIt game include period and exponential strategies. However, the opponent's behavior is random and dynamic in the real world. Therefore, we introduce a random factor and propose two new random strategies, called random period strategy and random strategy. Opponents can also adopt a composite strategy that contains more than two mentioned strategies. Therefore, the opponent adopts the following five strategies in this work.

- **Period.** Period strategy represents the opponent flips the resources with fixed $\delta$ time steps, represented as

$$\pi^o_{pe} = \begin{cases} flip & if\ T^o_{LM} = \delta \\ notflip & otherwise \end{cases} \tag{3}$$

  where, $T^o_{LM}$ is the time interval since the last flip of $\mathcal{N}^o$ at step $t$.
- **Exponential.** Exponential strategy returns flipping opportunities with an exponential distribution. The probability density function is expressed as

$$p(x) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & otherwise \end{cases} \tag{4}$$

$$\pi^o_{exp} = \begin{cases} flip & if\ T^o_{LM} \sim E(\lambda) \\ notflip & otherwise \end{cases} \tag{5}$$

  where, $x$ is the value of random variables.
- **Random Period.** We introduce a random factor $\zeta(|\zeta| < \delta)$, which can be positive or negative. Then the flipping strategy becomes $\delta + \zeta$. $\zeta$ makes the flipping opportunity of period strategy no longer fixed, which aggravates the non-stationary of the FlipIt game environment.

$$\pi^{o}_{ran\_pe} = \begin{cases} flip & if\ T^{o}_{LM} = \delta + \zeta \\ notflip & otherwise \end{cases} \tag{6}$$

- **Random.** Random strategy returns a random floating number $q \in [0, 1)$ with uniform distribution. Then the opponent compares the values of $q$ and $c$ ($c \in [0, 1)$)to decide whether to flip the resources. Random strategy can be formally expressed as

$$\pi^{o}_{ran} = \begin{cases} flip & if\ q \leq c \\ notflip & otherwise \end{cases} \tag{7}$$

- **Composite.** Composite strategy refers to the opponent's random choice of the above strategies to form a new composite strategy. Composite strategy is more unstable. This strategy poses greater challenges and requirements for defense agents.

## 4 ND3QN Decision-Making Approach

RL can be used to solve the solution of the FlipIt-OM game. To solve the non-stationary FlipIt-OM game, we propose ND3QN. In this Section, we elaborate on the architecture and implementation details of ND3QN. Section 4.1 outlines the architecture of ND3QN. Section 4.2 introduces the Constrained Q-Learning. Section 4.3 presents the process of interacting with opponents and environment. Section 4.4 explains the training procedure of ND3QN in detail.

### 4.1 The Architecture of NLD3QN

Due to the influence of the opponent's strategy, the process of FlipIt-OM game is dynamic and non-stationary. Therefore, we propose an approach called ND3QN to adapt to the non-stationary FlipIt game environment. The main objective of ND3QN is to defend against the potential opponent agent in a non-stationary game environment, such as an APT scenario. The approach combines NoisyNet, Dropout, Dueling Q-Network and Constrained Q-Learning into DQN to obtain a strategy that adapts to a non-stationary environment. ND3QN can learn a better reward function in a non-stationary environment than the classic DQN algorithm. The architecture of ND3QN is shown in Fig. 2.

The classic problems faced by reinforcement learning are over-fitting, over-estimation and insufficient exploration. ND3QN addresses these classic problems.

- To overcome the over-fitting problem, dropout is adopted during the training process of ND3QN. With dropout training, ND3QN is more suitable for a non-stationary environment.
- To enhance the exploring ability of ND3QN and make optimal decisions, NoisyNet is added in front of the action output layer. The Gassian noise is defined as $\xi \equiv (\mu + \sigma) \odot \epsilon$, where $(\mu + \sigma)$ is a set of learnable parameter vectors, $\epsilon$ denotes the zero-mean noise.

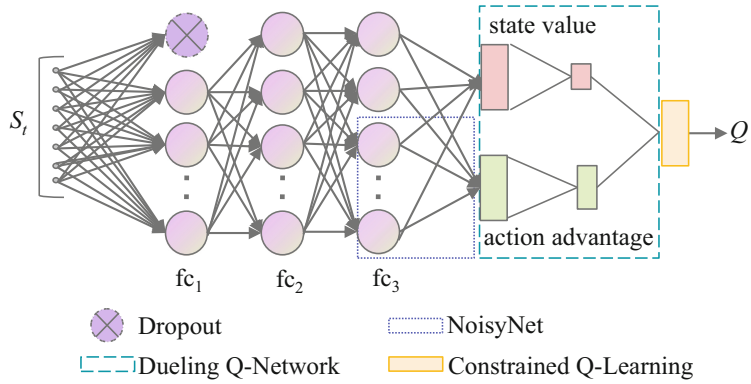



**Fig. 2.** The architecture of ND3QN.

- To improve the performance of ND3QN, Dueling Q-Network is incorporated to split the last layer of the same neural network in two streams: the state-value function $S$ and the advantage function $A$.

Simultaneously, we have also made specific improvements for the non-stationary FlipIt-OM game environment appropriately.

- To solve the over-estimation problem of classic DQN, we propose Constrained Q-Learning(Cons-QL). The design of Cons-QL is introduced in Sect. 4.2.
- To adapt to the stealth interaction in an APT scenario, ND3QN is able to perceive the opponent's behavior. The implementation details of the perception process are presented in Sect. 4.3.

ND3QN can quickly achieve the balance between exploring and exploiting the environment, and continue to maintain its advantages over its opponent. Then we introduce the detail of Cons-QL and the perception process in the following subsections.

### 4.2 Constrained Q-Learning

The Bellman-style Q function of reinforcement learning is as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{8}$$

where $\alpha$ is the learning rate. However, due to the influence of $\max Q(s_{t+1}, a_{t+1})$, the value of $Q(s, a)$ in Eq. 8 may be overestimated. Cons-QL takes a constrained mean value as the lower bounds of Q value, as shown in Eq. 9.

$$Q(s_t, a_t) = Q(s_t, a_t) - \omega * (Q(s_t, a_t) - Q_{mean})^2 \tag{9}$$

where $\omega \in [0,1)$ is a balance factor, $Q_{mean} = \frac{(Q_{last}+Q(s_t,a_t))}{2}$, $Q_{last} = Q(s_{t-1},a_{t-1})$. After using Cons-QL, it can smooth out the Q-value spike caused by over-fitting. As a result, ND3QN will reduce wrong decisions caused by the over-estimation of Q value.

### 4.3 The Process of Interacting with Opponents and Environment

A diagram of the process of perceiving opponents and interacting with the environment using ND3QN is shown in Fig. 3. In the process of the FlipIt-OM real-time game, the observation space of the ND3QN agent is extended. Firstly, the ND3QN agent stores the information of the opponent's last move actions $a^o$ in its knowledge base $\mathcal{K}$. Perceptual vector $\rho^o$ analyzes a series of actions of the opponent from the knowledge base $\mathcal{K}$. Then ND3QN agent learns the opponent's strategy $\pi^o$ through perception vector $\rho^o$. According to the perceived opponent's strategy $\pi^{\rho^o}$, the ND3QN agent makes its strategy $\pi^d$ and then does defensive actions $a^d$.

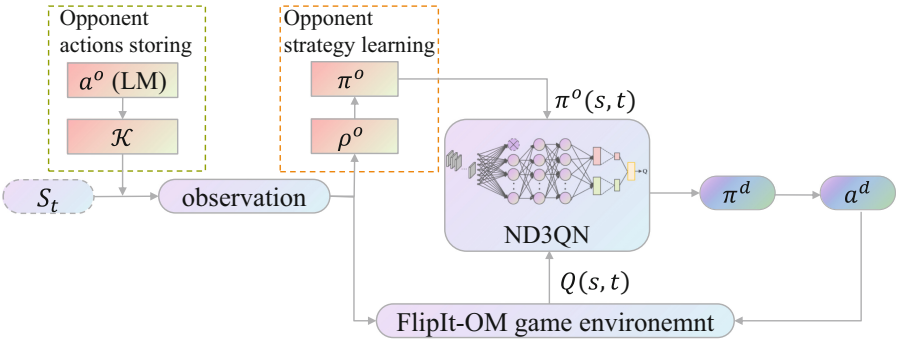



**Fig. 3.** Diagram of perception process of ND3QN.

### 4.4 The Training Procedure of ND3QN

Because of the concealment, latency, and unpredictability of APT attacks, it is essential to perceive opponents' behavior from the environment for learning agents. DRL can learn from the environment and make sequential decisions without given prior knowledge. DRL is accordance with the stealth interaction characteristics of APT attacks. A novel algorithm called ND3QN was proposed. The perception vector is set in the observation space of the ND3QN agent to capture the flipping opportunity of the opponent's last move actions and store it in the knowledge of the ND3QN agent. The training procedure of ND3QN is shown in Algorithm 1.

**Algorithm 1:** The training procedure of ND3QN.

Initialize replay memory $D$, environment *env*, and observation $s$
Initialize network with random weights $\theta$, dropout $p$ and noise $\xi$
Initialize target network with weights $\theta^-$
Initialize the advantage function parameter $\alpha$, the state-value function parameter $\beta$
**for** $t = 1$ *to* $T$ **do**
  Select an action $a_d \leftarrow Q(s, a, \xi; \theta, \alpha, \beta)$
  Execute $a_d$ in *env* and observe $a^o, r, s'$
  Store $a^o$ in knowledge $k^d$
  Perceive $\pi^o$ using perception vector $\rho^o$
  Store transition $\tau\left\{s, a^d, a^o, s'\right\}$ in $D$
  Sample random minibatch of transitions $\tau\left\{s_j, a_j^d, a_j^o, s_j'\right\}$ from $D$
  **for** $j$ = *1 to minibatch* **do**
    **if** $s_j'$ *is a terminal state* **then**
      $Q(s_j, a_j^d | \pi^o) \leftarrow R(s_j, a_j^d, a_j^o, s_j')$
    **end**
    $Q(s_j, a_j^d \mid \pi^o) \leftarrow \sum_{a^o} \pi^o(a^o \mid s_j) \sum_{s'} \tau(s_j, a_j^d, a_j^o, s_j')$ $\left[R(s_j, a_j^d, a_j^o, s_j') + \gamma E_{a'}\left[Q(s_j', a_j', \xi'; \theta^-, \alpha, \beta \mid \pi^o)\right]\right]$
    Calculate constrained Q value $Q(s_j, a_j^d \mid \pi^o)$
    Perform a gradient descent step with loss $\frac{1}{m}\sum_{i=1}^{m}\left(Q(s_j, a_j^d \mid \pi^o) - Q_E(s, a \mid \pi^o)\right)^2$
  **end**
  Every $C$ steps update the target network $\theta^- \leftarrow \theta$
**end**

## 5 Experiments

The experiments were conducted in a FlipIt game simulator. Section 5.1 describes the environmental settings. Section 5.2 introduces the baseline algorithms. Section 5.3 analyzes the experimental results.

### 5.1 Environmental Settings

Experiments were conducted in the FlipIt-OM (LM) game environment. Generally speaking, the resources are limited. Thus the cost of flipping the resources by the players is higher than the benefit. In the experiments, the flipping cost is 5, the flipping reward is 1. And the numbers of training episodes are 10000, the steps of per episodes are 200.

### 5.2 Baseline

In this section, we introduce two baseline algorithms (DQN and PER-DQN), and then we show the hyperparameters used in this experiment.

- **DQN.** Deep Q-Network stores the experience into the replay buffer, and then evenly samples a mini-batch of samples from the replay buffer for a Q-learning update in each time step $t$.
- **PER-DQN.** Prioritized Experience Replay (PER) [17] improves the sampling problem of DQN. PER samples a batch of experiences from memory based on the priority. The sampling probability of state transition $i$ can be calculated as follows.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{10}$$

where $p_i$ is the priority of $i$. $\alpha \in [0, 1]$ is a factor to adjust random sampling or prior sampling. PER adjusts the weight in importance sampling to correct the deviation of $i$ with $\beta \in [0, 1]$. $N$ is the size of the replay buffer.

$$w_i = NP(i)^{-\beta} \tag{11}$$

Table 2 shows the hyperparameters of the three algorithms. Where, $p$ denotes the proportion of dropped units in the fully connected layer to prevent overfitting.

**Table 2.** Hyperparameters of three algorithms.

| Hyperparameters | ND3QN | DQN | PER-DQN |
|---|---|---|---|
| buffer_size | 2000 | 2000 | 2000 |
| batch_size | 32 | 32 | 32 |
| learning rate | 0.003 | 0.003 | 0.003 |
| $\tau$ | 0.003 | 0.003 | 0.003 |
| $\gamma$ | 0.99 | 0.99 | 0.99 |
| eps_start | 1 | 1 | 1 |
| eps_end | 0.01 | 0.01 | 0.01 |
| eps_decay | 0.995 | 0.995 | 0.995 |
| update_network | 5 | 5 | 5 |
| p | 0.05 | – | – |
| $\alpha$ | – | – | 0.2 |
| $\beta$ | – | – | 0.2 |

– indicates that there are no values.

### 5.3 Analysis of Experimental Results

To validate the effectiveness and efficiency of ND3QN, we adopted a series of experiments. Section 5.3.1 introduced the evaluation index. Section 5.3.2 compared the three methods against the opponent with traditional strategies, including period and exponential strategies. Section 5.3.3 compared the three methods against an opponent with non-stationary strategies, including random period and random strategies. Section 5.3.4 compared the three methods against an opponent with a composite strategy.

**Evaluation Index.** The average score, last score, and winning rate are selected as evaluation indexes. The criteria for judging the effectiveness of defense agents are as follows: (1) defeating the attack agent is the premise of the defense agent; (2) if (1) is satisfied, the higher last score of the defense agent is better.

- **Average score**. The average rewards obtained by the players from the FlipIt-OM game environment every 100 episodes.
- **Last score**. The average rewards of last 100 episodes obtained by the players in 10000 episodes of the FlipIt-OM game.
- **Winning rate**. The probability of the defense agent defeating the opponent in the total times.

**ND3QN Against an Opponent with Traditional Strategies.** In this section, we compare the performance of the three algorithms against an opponent with traditional strategies, including period and exponential strategies.

- **ND3QN against an opponent with period strategy.**

The average scores of the three algorithms against an opponent with a period strategy are shown in Fig. 4.

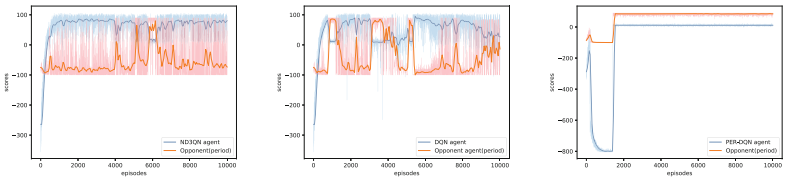
Due to the integration of Dueling Q-Network and Noisynet, the performance of the ND3QN agent is improved. And dropout solves the co-adaption between units. Especially, the ND3QN agent almost always maintains its advantage over the opponent owing to the perception vector. Combined with these advantages, the performance of the ND3QN agent is the best. However, the DQN agent can't adapt to the opponent's strategy, and the convergence process is unstable. The standard deviation of ND3QN is 16.58 after 1000 episodes, while DQN is 27.87 ($\delta$ = 10). This is because without opponent modeling, DQN needs to relearn from experience. The performance of PER-DQN is the worst, and it's not able to take effect. This is because PER-DQN adds priority experience replay to evaluate a priority of historical experience. PER may work in a static environment, but it is difficult to take effect in a continuous game process.

The last scores against a period opponent with different $\delta$ is shown in Fig. 5. The ND3QN agent always keeps its advantage even if the opponent's flipping opportunity changes. If the opponent flips the resource at regular intervals, the DQN agent also performs well, but the learning progress of the DQN agent is relatively slow and the convergence is unstable during the game process. PER-DQN agent neither adapts to the game environment nor learns the optimal strategy, and it always loses to the opponent.

- **ND3QN against an opponent with exponential strategy.**

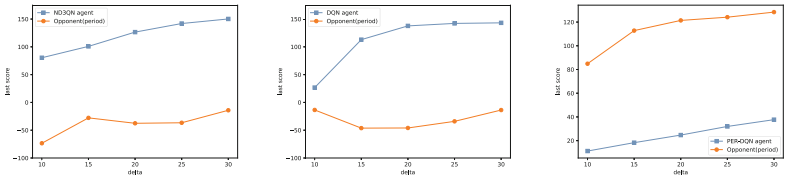
As shown in Fig. 6 and Fig. 7, exponential strategy is more difficult to capture its flipping opportunity compared with period strategy.

However, we still come to a similar conclusion. The ND3QN agent can always beat the opponent and obtain a higher reward. It can be seen that the ND3QN



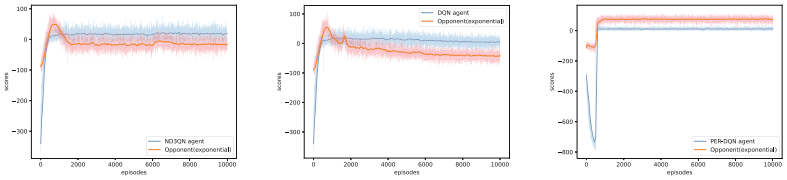

(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 4.** Average scores against an opponent with period strategies ($\delta = 10$).



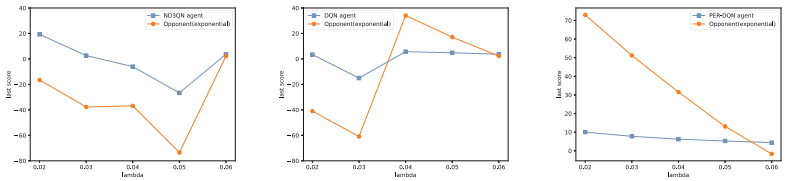

(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 5.** Last score against a period opponent. ($\delta = 10, 15, 20, 25, 30$.)




(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 6.** Average scores against an opponent with exponential strategies ($\lambda = 0.02$).




(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 7.** Last score against an exponential opponent. ($\lambda = 0.02, 0.03, 0.04, 0.05, 0.06$.)

agent has strong adaptability to the opponent's strategy, and can learn the optimal flipping strategy through opponent modeling. Overall, the DQN agent is unable to adapt to the opponent's different strategies. Sometimes DQN agent beats its opponent but sometimes loses to its opponent. Because the DQN agent has the defects of over-fitting and over-estimation, it can't quickly adjust its

strategy when the opponent's strategies change. PER-DQN agent still doesn't work, and it can't learn a strategy to beat its opponent.

The comparison of the last scores of the three algorithms against an opponent with traditional strategies is shown in Table 3. When the opponent adopts the period strategy, the efficiency of ND3QN and DQN is similar, because the period strategy is relatively stable. While when the opponent adopts exponential strategy, the advantage of ND3QN is more obvious, and DQN sometimes fails to analyze the law of exponential strategy and loses to the opponent.

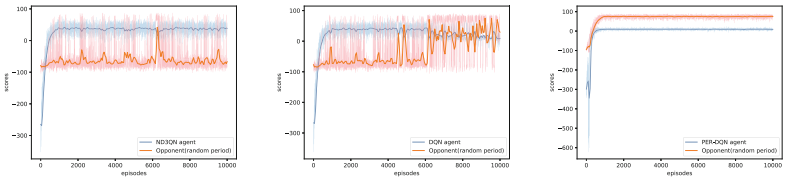
**Table 3.** Comparison of the last scores of the three algorithms against an opponent with traditional strategies.

| Opponent strategy | Last score of the defense agent vs. opponent | | |
|---|---|---|---|
| | ND3QN | DQN | PER-DQN |
| Period ($\delta = 10$) | **80.38/−73.48** | 26.70/−13.55 | 11.22/84.88 |
| Period ($\delta = 15$) | 100.84/−27.79 | **112.95/−46.25** | 18.29/112.81 |
| Period ($\delta = 20$) | 126.50/−37.55 | **137.68/−45.98** | 24.70/121.40 |
| Period ($\delta = 25$) | 141.96/−36.61 | **142.16/−34.06** | 31.97/124.13 |
| Period ($\delta = 30$) | **150.19/−14.14** | 143.18/−13.73 | 37.69/128.41 |
| Exponential ($\lambda = 0.02$) | **19.30/−16.55** | 3.44/−40.99 | 10.03/72.97 |
| Exponential ($\lambda = 0.03$) | **2.64/−37.74** | −15.01/−60.94 | 7.80/51.20 |
| Exponential ($\lambda = 0.04$) | **−6.04/−36.86** | 5.70/34.10 | 6.25/31.55 |
| Exponential ($\lambda = 0.05$) | **−26.66/−73.49** | 4.81/17.14 | 5.31/13.09 |
| Exponential ($\lambda = 0.06$) | **3.70/2.25** | 3.70/2.25 | 4.38/−1.63 |

**ND3QN Against an Opponent with Non-stationary Strategies.** In this section, we compare the performance of the three algorithms against an opponent with non-stationary strategies, including random period and random strategies.

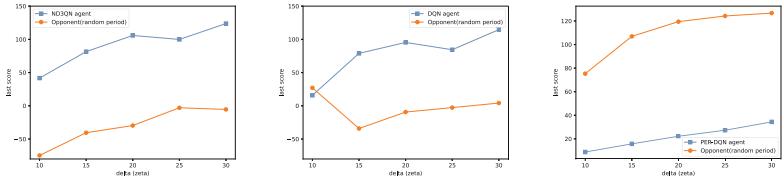
- **ND3QN against an opponent with a random period strategy.**

As described in Sect. 3.3, random period is represented as $T^o_{LM} = \delta + \zeta$. As shown in Fig. 8 and Fig. 9, the advantages of ND3QN are more obvious confronted with a non-stationary opponent. ND3QN has the highest average scores and keeps a stable convergence. However, the last score of DQN is lower than ND3QN. In addition, DQN has an unstable convergence, and in some cases it can't learn a decision-making strategy and eventually loses to its opponent. In contrast, PER-DQN can't learn the opponent's behavior.



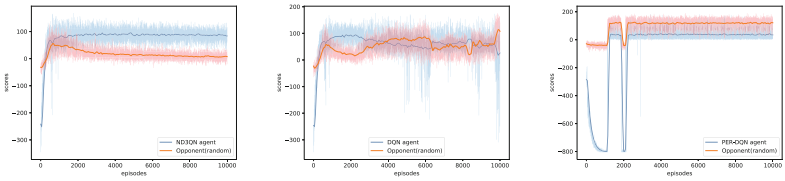

(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 8.** Average scores against an opponent with random period strategies ($\delta = 10$, $\zeta = 3$).



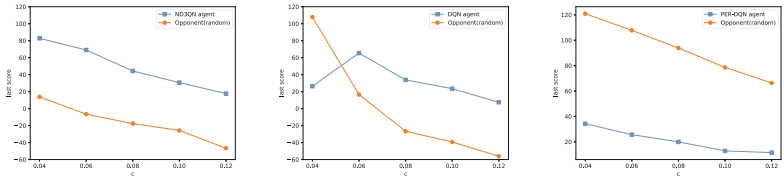

(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 9.** Last score against a random period opponent. ($\delta = 10, 15, 20, 25, 30$; $\zeta = 3, 4, 5, 7, 8$.)




(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 10.** Average scores against an opponent with random strategies ($c = 0.04$).




(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 11.** Last score against a random opponent. ($c = 0.04, 0.06, 0.08, 0.10, 0.12$.)

- **ND3QN against an opponent with a random strategy.**

Random strategy returns a random floating number $q \in [0, 1)$ with uniform distribution. Then the opponent compares the values of $q$ and $c$ to decide whether to flip the resources. As shown in Fig. 10 and Fig. 11, similarly, ND3QN continues to maintain an advantage over its opponent and has the highest average

scores. DQN can learn an optimal strategy in most cases. However, the unstable convergence of DQN is more severe, and sometimes it even loses to its opponent. PER-DQN utilizes the acquired historical experience for priority sampling, it still can't defeat the opponent who adopts a random strategy.

The comparison of the last scores of the three algorithms against an opponent with non-stationary strategies is shown in Table 4. ND3QN has an extraordinary performance when the opponent adopts random strategy. ND3QN has the highest last score and good convergence, and it can continue to beat opponents. Because ND3QN can perceive and analyze the opponent's strategy, it can quickly find the optimal strategy to defeat the opponent. However, DQN agent is unstable in convergence and sometimes loses to its opponents.

**Table 4.** Comparison of the last scores of the three algorithms against an opponent with non-stationary strategies.

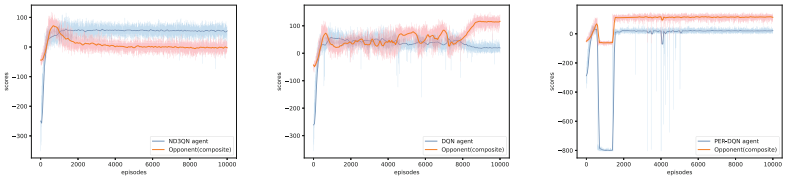
| Opponent strategy | Last score of the defense agent vs. opponent | | |
|---|---|---|---|
| | ND3QN | DQN | PER-DQN |
| Random Period ($\delta = 10$, $\zeta = 3$) | **41.72/−74.72** | 15.75/27.10 | 8.83/75.32 |
| Random Period ($\delta = 15$, $\zeta = 4$) | **81.54/−40.49** | 78.91/−34.16 | 15.75/106.95 |
| Random Period ($\delta = 20$, $\zeta = 5$) | **105.88/−29.68** | 95.38/−9.38 | 22.21/119.44 |
| Random Period ($\delta = 25$, $\zeta = 7$) | **99.89/−2.94** | 84.41/−2.71 | 27.31/124.24 |
| Random Period ($\delta = 30$, $\zeta = 8$) | **123.76/−5.46** | 114.61/4.14 | 34.40/126.70 |
| Random ($c = 0.04$) | **82.78/13.82** | 26.34/107.71 | 34.31/120.94 |
| Random ($c = 0.06$) | **69.05/−6.60** | 65.40/16.60 | 25.72/107.78 |
| Random ($c = 0.08$) | **44.31/−17.56** | 33.81/−26.66 | 19.95/93.90 |
| Random ($c = 0.10$) | **30.61/−25.76** | 23.49/−39.34 | 12.89/78.61 |
| Random ($c = 0.12$) | **17.74/−46.69** | 7.46/−56.11 | 11.54/66.36 |

**ND3QN Against an Opponent with a Composite Strategy.** Assuming that the opponent adopts a composite strategy of random ($c = 0.05$) and random period strategy ($\delta = 20, \zeta = 3$). It's more difficult to catch the flipping opportunities of composite strategy. Average scores against the opponent are shown in Fig. 12.

As can be seen from Fig. 12(a) and 12(b), the opponent defeats all the defense agent after approximately 1000 epsisodes. ND3QN can perceive the opponent's behavior and learn the strategy of defeating its opponent, while DQN cannot learn the strategy and eventually loses to the opponent. Opponent modeling becoming more important when faced with opponents of composite strategies. The standard deviation of ND3QN scores is 2.49 after 1000 episodes, while the value of DQN is 10.45. It indicates that the convergence of ND3QN is better.

For another example, when the opponent adopts a random ($c = 0.01$) and random period strategy ($\delta = 10, \zeta = 4$). The last score of the ND3QN vs. its

opponent is 3.62/−31.82, the value of DQN is 65.05/15.60, and the value of PER-DQN is 24.68/73.58. Although the performance of DQN is the best in this case, ND3QN can still learn the strategy of defeating the opponent quickly. When the opponent adopts random ($c = 0.04$) and exponential strategy ($\lambda = 0.03$). The last score of ND3QN vs. its opponent is 17.53/−21.23, the value of DQN is 15.85/−37.35, and the value of PER-DQN is 11.56/79.39. The standard deviation of ND3QN scores is 2.30 after 1000 episodes, the value of DQN is 3.49. ND3QN is slightly better than DQN in this case.




(a) ND3QN (b) DQN (c) PER-DQN

**Fig. 12.** Average scores against an opponent with composite strategies ($c = 0.05$, $\delta = 20$, $\zeta = 3$.)

The comparison of the last scores against an opponent with non-stationary strategies is shown in Table 5.

**Table 5.** Comparison of the last scores of the three algorithms against an opponent with composite strategies.

| Opponent strategy | Last score of the defense agent vs. opponent | | |
|---|---|---|---|
| | ND3QN | DQN | PER-DQN |
| Composite ($c = 0.05$, $\delta = 20$, $\zeta = 3$) | **52.60/−0.20** | 19.41/116.19 | 21.69/113.11 |
| Composite ($c = 0.01$, $\delta = 10$, $\zeta = 4$) | 3.62/−31.82 | **65.05/15.60** | 24.68/73.58 |
| Composite ($c = 0.04$, $\lambda = 0.03$) | **17.53/−21.23** | 15.85/−37.35 | 11.56/79.39 |

The performance of ND3QN is more stable and effective in the face of opponents with composite strategies, and it can maintain the advantage of defeating opponents. However, the effect of DQN is unstable, which is caused by its overfitting and insufficient exploration.

From the above experiments, the winning rates of the three algorithms are shown in the Table 6. The winning rate of ND3QN is over 95%, the winning rate of DQN is 78.26%, and PER-DQN is only 4.35%. Note that the winning rate is calculated from the above opponent strategies, which does not mean that it can be applied to all situations. Genarally, when the opponent's strategy is more random, it is more difficult to beat it. ND3QN has a high winning rate, regardless of the opponent adopting the traditional, random or composite strategies.

**Table 6.** Comparison of the winning rates against the opponent.

| Opponent strategy | ND3QN | DQN | PER-DQN |
|---|---|---|---|
| Period ($\delta = 10$) | ✓ | ✓ | ✗ |
| Period ($\delta = 15$) | ✓ | ✓ | ✗ |
| Period ($\delta = 20$) | ✓ | ✓ | ✗ |
| Period ($\delta = 25$) | ✓ | ✓ | ✗ |
| Period ($\delta = 30$) | ✓ | ✓ | ✗ |
| Exponential ($\lambda = 0.02$) | ✓ | ✓ | ✗ |
| Exponential ($\lambda = 0.03$) | ✓ | ✓ | ✗ |
| Exponential ($\lambda = 0.04$) | ✓ | ✗ | ✗ |
| Exponential ($\lambda = 0.05$) | ✓ | ✗ | ✗ |
| Exponential ($\lambda = 0.06$) | ✓ | ✓ | ✓ |
| Random Period ($\delta = 10$, $\zeta = 3$) | ✓ | ✗ | ✗ |
| Random Period ($\delta = 15$, $\zeta = 4$) | ✓ | ✓ | ✗ |
| Random Period ($\delta = 20$, $\zeta = 5$) | ✓ | ✓ | ✗ |
| Random Period ($\delta = 25$, $\zeta = 7$) | ✓ | ✓ | ✗ |
| Random Period ($\delta = 30$, $\zeta = 8$) | ✓ | ✓ | ✗ |
| Random ($c = 0.04$) | ✓ | ✗ | ✗ |
| Random ($c = 0.06$) | ✓ | ✓ | ✗ |
| Random ($c = 0.08$) | ✓ | ✓ | ✗ |
| Random ($c = 0.10$) | ✓ | ✓ | ✗ |
| Random ($c = 0.12$) | ✓ | ✓ | ✗ |
| Composite($c = 0.05$, $\delta = 20$, $\zeta = 3$) | ✓ | ✗ | ✗ |
| Composite($c = 0.01$, $\delta = 10$, $\zeta = 4$) | ✓ | ✓ | ✗ |
| Composite($c = 0.04$, $\lambda = 0.03$) | ✓ | ✓ | ✗ |

✓ represents winning; ✗ represents losing.

**Ablation Experiments of ND3QN.** The ablation experimental results of ND3QN agent vs. opponent is shown in Table 7.

It can be seen that each component has a certain positive impact. The random strategy of the opponent makes the observation space of ND3QN non-stationary. The optimization effect of dropout is relatively obvious for the reason that it can alleviate the over-fitting problem. NoisyNet and Dueling Q-Network are both improve the performance of ND3QN. When the opponent adopts the traditional strategy, Cons-QL doesn't work. While when the opponent adopts the random strategy, Cons-QL takes effect. This is because the traditional attack strategy is relatively fixed and has certain rules, and the estimation of Q value is relatively accurate. Instead, when the opponent's strategy is random, the over-estimation of Q value is more obvious.

**Table 7.** Ablation results of ND3QN agent vs. opponent.

| Ablation components | Opponent strategy | Last score of ND3QN vs. opponent | |
|---|---|---|---|
| | | Original results | Ablation results |
| NoisyNet | Period ($\delta = 10$) | 80.38/−73.48 | 89.58/−23.36↑ |
| | Exponential ($\lambda = 0.02$) | 19.30/−16.55 | **15.79/−25.94**↓ |
| | Random ($c = 0.04$) | 82.78/13.82 | 83.34/14.71↑ |
| | Ran_Period ($\delta = 10$, $\zeta = 3$) | 41.72/−74.72 | **36.86/−65.36**↓ |
| Dropout | Period ($\delta = 10$) | 80.38/−73.48 | 83.59/−72.19↑ |
| | Exponential ($\lambda = 0.02$) | 19.30/−16.55 | **16.46/−23.36**↓ |
| | Random ($c = 0.04$) | 82.78/13.82 | **80.49/0.71**↓ |
| | Ran_Period ($\delta = 10$, $\zeta = 3$) | 41.72/−74.72 | **37.05/−78.15**↓ |
| Dueling Q-Network | Period ($\delta = 10$) | 80.38/−73.48 | 86.98/−82.43↑ |
| | Exponential ($\lambda = 0.02$) | 19.30/−16.55 | **11.62/73.83**↓ |
| | Random ($c = 0.04$) | 82.78/13.82 | 83.34/14.71↑ |
| | Ran_Period ($\delta = 10$, $\zeta = 3$) | 41.72/−74.72 | **40.10/−75.25**↓ |
| Cons-QL | Period ($\delta = 10$) | 80.38/−73.48 | 80.38/−73.48 |
| | Exponential ($\lambda = 0.02$) | 19.30/−16.55 | 19.30/−16.55 |
| | Random ($c = 0.04$) | 82.78/13.82 | 88.17/10.33↑ |
| | Ran_Period ($\delta = 10$, $\zeta = 3$) | 41.72/−74.72 | **37.24/−77.90**↓ |

## 6 Conclusion and Future Work

In this work, we incorporate random factors and propose random strategies of FlipIt game. We construct a FlipIt-OM model using opponent modeling and deep reinforcement learning. And we propose ND3QN to perceive the non-stationary opponent in a FlipIt-OM game to simulate an APT-like scenario. ND3QN solves the over-estimation, over-fitting and insufficient exploration problems by combining NoisyNet, Dueling Q-Network, Dropout and Cons-QL. Due to the stealth interaction characteristics of the APT attack, ND3QN can only observe the opponent's last move actions, then stores it into its knowledge. ND3QN perceives the opponent's strategy accurately, and eventually defeating its opponent. We demonstrate the efficiency and effectiveness of ND3QN in a FlipIt-OM game environment against opponents with traditional, non-stationary, and composite strategies. From the above experiments, the winning rate of ND3QN is over 95%, and the winning rate of DQN is 78.26%. ND3QN can adapt to more opponent strategies. And the convergence of ND3QN is more stable.

The FlipIt-OM model and the ND3QN approach proposed in this paper can be widely used in non-stationary cyberspace game scenarios. In addition, a real attacker can cheat or launch a feint to hide its real attack ability. Future work will further examine a deceptive opponent modeling and infer the opponent's real strategy to suppress the opponent immediately.

## References


1. Baillie, C., Standen, M., Schwartz, J., Docking, M., Bowman, D., Kim, J.: Cyborg: an autonomous cyber operations research gym. arXiv preprint arXiv:2002.10667 (2020)
2. Everett, R., Roberts, S.J.: Learning against non-stationary agents with opponent modelling and deep reinforcement learning. In: AAAI Spring Symposia (2018)
3. Foerster, J.N., Chen, R.Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., Mordatch, I.: Learning with opponent-learning awareness. arXiv preprint arXiv:1709.04326 (2017)
4. Fortunato, M., et al.: Noisy networks for exploration. arXiv preprint arXiv:1706.10295 (2017)
5. Greige, L., Chin, P.: Deep reinforcement learning for flipit security game. In: Benito, R.M., et al. (eds.) COMPLEX NETWORKS 2021, pp. 831–843. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-93409-5_68
6. He, H., Boyd-Graber, J., Kwok, K., Daumé III, H.: Opponent modeling in deep reinforcement learning. In: International Conference on Machine Learning, pp. 1804–1813. PMLR (2016)
7. Hernandez-Leal, P., Zhan, Y., Taylor, M.E., Sucar, L.E., Munoz de Cote, E.: An exploration strategy for non-stationary opponents. Auton. Agent. Multi-Agent Syst. **31**, 971–1002 (2017)
8. Hong, Z.W., Su, S.Y., Shann, T.Y., Chang, Y.H., Lee, C.Y.: A deep policy inference q-network for multi-agent systems. arXiv preprint arXiv:1712.07893 (2017)
9. Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative q-learning for offline reinforcement learning. Adv. Neural. Inf. Process. Syst. **33**, 1179–1191 (2020)
10. Laszka, A., Horvath, G., Felegyhazi, M., Buttyán, L.: Flipthem: modeling targeted attacks with flipt, for multiple resources. In: Poovendran, R., Saad, W. (eds.) GameSec 2014. LNCS, vol. 8840, pp. 175–194. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12601-2_10
11. Li, L., Fayad, R., Taylor, A.: Cygil: a cyber gym for training autonomous agents over emulated network systems. arXiv preprint arXiv:2109.03331 (2021)
12. Liu, Z., Wang, L.: Flipit game model-based defense strategy against cyberattacks on SCADA systems considering insider assistance. IEEE Trans. Inf. Forensics Secur. **16**, 2791–2804 (2021)
13. Ma, Y., et al.: Opponent portrait for multiagent reinforcement learning in competitive environment. Int. J. Intell. Syst. **36**(12), 7461–7474 (2021)
14. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
15. Oakley, L., Oprea, A.: QFlip: an adaptive reinforcement learning strategy for the FlipIt security game. In: Alpcan, T., Vorobeychik, Y., Baras, J.S., Dán, G. (eds.) GameSec 2019. LNCS, vol. 11836, pp. 364–384. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32430-8_22
16. Raileanu, R., Denton, E., Szlam, A., Fergus, R.: Modeling others using oneself in multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 4257–4266. PMLR (2018)
17. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
18. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)

19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
20. Tang, Z., Zhu, Y., Zhao, D., Lucas, S.M.: Enhanced rolling horizon evolution algorithm with opponent model learning. IEEE Transactions on Games (2020)
21. Tankard, C.: Advanced persistent threats and how to monitor and deter them. Netw. Secur. **2011**(8), 16–19 (2011)
22. Team, M.D.: CyberBattleSim (2021). https://github.com/microsoft/cyberbattlesim
23. Van Dijk, M., Juels, A., Oprea, A., Rivest, R.L.: Flipit: the game of "stealthy takeover.". J. Cryptol. **26**, 655–713 (2013)
24. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
25. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1995–2003. PMLR (2016)
26. Wu, Z., Li, K., Xu, H., Zang, Y., An, B., Xing, J.: L2e: learning to exploit your opponent. In: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2022)
27. Zhang, R., Zhu, Q.: Flipin: a game-theoretic cyber insurance framework for incentive-compatible cyber risk management of internet of things. IEEE Trans. Inf. Forensics Secur. **15**, 2026–2041 (2019)
28. Zhu, J., Wei, Y., Kang, Y., Jiang, X., Dullerud, G.E.: Adaptive deep reinforcement learning for non-stationary environments. Sci. Chin. Inf. Sci. **65**(10), 202204 (2022)