Deep Reinforcement Learning Approaches Against Jammers with Unequal Sweeping Probability Attacks

Lan K. Nguyen[1], Duy H. N. Nguyen[2], Nghi H. Tran[3*], David Brunnenmeyer[1]

[1]KBR, Los Angeles, CA, USA
[2]Department of Electrical Engineering, San Diego State University, San Diego, CA, USA
[3]Department of Electrical and Computer Engineering, University of Akron, Akron, OH, USA

Abstract

This paper investigates deep reinforcement learning (DRL) approaches designed to counter jammers that maximize disruption by employing unequal sweeping probabilities. We first propose a model and defense action based on a Markov Decision Process (MDP) under non-uniform attacks. A key drawback of the standard MDP model, however, is its assumption that the defending agent can acquire sufficient information about the jamming patterns to determine the transition probability matrix. In a dynamic environment, the attacker's patterns and models are often unknown or difficult to obtain. To overcome this limitation, RL techniques such as Q-learning, deep Q-network (DQN), and double deep Q-network (DDQN) have been considered effective defense strategies that operate without an explicit jamming model. With Q-learning, defense strategies can still be computationally expensive and require long time to learn the optimal policy. This limitation arises because a large state space or a substantial number of actions causes the Q-table to grow exponentially. Leveraging the flexibility, adaptability, and scalability of RL, we first propose a DQN framework designed to handle large-scale action spaces across expanded channels and jammers. Furthermore, to overcome the inherent overestimation bias present in Q-learning and DQN algorithms, we investigate a DDQN framework. Assuming the estimation error of the action value in DQN follows a zero-mean Gaussian distribution, we then analytically derive the expected loss. Numerical examples are finally presented to characterize the performances of the proposed algorithms and the superiority of DDQN over DQN and Q-learning approaches.

Received on 03 October 2025; accepted on 30 October 2025; published on 04 November 2025

Keywords: Jamming Attacks; Data Rate Game; Markov Decision Process; Reinforcement Learning; Q-learning; Deep Q-Networks; Double Deep Q-Networks.

doi:10.4108/eetinis.v12i4.10461

1. Introduction

Over the last decade, there have been significant contributions on anti-jamming technologies, and the literature is vast. Among different innovative anti-jamming approaches, game theory has been widely considered as an effective tool to address the interaction between users and jammers for best defense actions against smart jammers in both wireless communications and SATCOM (please see [1–11] and references therein). For instance, reference [12] appears to be one of the first studies that used game theory to exploit the flexible access to multiple channels as an effective anti-jamming solution. Specifically, a channel hopping strategy based on Markov decision process (MDP) was proposed in [12] to deal with smart sweeping attacks in cognitive radio (CR) networks. As an extension of [12], the works in [13, 14] further considered optimal defense strategies against sweeping attacks under joint dynamic frequency hopping and rate adaptation. While the hopping strategies in [12–14] were developed under the assumption that an attacker can randomize the sweeping order, the jamming probabilities were chosen uniformly across multiple channels. In practice, an intelligent jammer can choose communication links to attack with non-equal probabilities for more damaging effects. Under this consideration, developing an optimal defense strategy requires a revised MDP-based hopping that relies on non-equal probability assignments, which is a challenging task.

While the above game theory-based approaches provide important insights into anti-jamming access strategies, they rely on a key assumption that the user is able to obtain certain information of the attacker's jamming patterns and models. In a more dynamic environment, acquiring such information

*Corresponding author: Email: nghi.tran@uakron.edu is a very difficult, if not impossible, task. To overcome this drawback, reinforcement learning (RL) such as Q-learning and multi-agent RL techniques have been considered as an attractive alternative in developing an efficient defense strategy without the need of explicit jamming models [2, 15–24]. Interestingly, it has been demonstrated that RL-based strategies can achieve near optimal access strategies in dynamic anti-jamming games [21, 22]. In [25], a novel Q-learning-based defense strategy for a multi-channel system under a smart non-equal probability sweeping attack was proposed. With this defense strategy, the user can learn the optimal countermeasure strategy without needing to know the attacking strategy nor the non-equal probability assignment from the attacker. The main idea was to initiate a Q-table to approximate an action-value function. As a result, we can achieve close to the performance of the MDP-based defense strategy without the explicit knowledge of the jammer's attacking model or probability distribution of attacks. This approach is further extended to a multi-agent system in which multi users compete for the same physical resources, i.e., channel and time slots, while evading jamming attacks. The proposed coordinated multi-agent Q-learning methods in [26] allow the agents to cooperate with each other through information exchange. While the agents execute the Q-learning method independently, an agent manager is designated to collect all the agents' intended actions, resolve potential conflicts among the actions, and assign the channels for the agents accordingly. It was shown in [26] that the multi-agent Q-learning coordinated approaches significantly improve the total system payoff compared to the uncoordinated Q-learning approach.

With Q-learning, when the decision space or the number of actions is large, the Q-table of the Q-learning becomes too large, and the update becomes a computational burden and costly to obtain the optimal policy. Subsequently, deep Q-network (DQN) was proposed to solve such high dimensional problems. DeepMind initially introduced DQN to play Atari games with several innovations were made to stabilize and improve the training [27]. DQN combines Q-learning and a deep neural network. Unlike Q-learning, the Q-value of DQN is not calculated directly from the state-action pair function but learned from a deep neural net-work. DQN expedites learning and enhances the efficacy of anti-jamming communications [28–30].

In this work, we exploit deep Q-network to enhance the learning speed for more effective anti-jamming communications under non-uniform attacks. The new deep-learning schemes are efficient, and can be effectively used in a multi-channel system under the presence of multiple adversarial smart jammers. It can also address the scalability issue in the conventional Q-learning approach. Our results show that we can effectively learn the optimal countermeasure strategy without needing to know the attacking strategy. In particular, under the presence of a single jammer and when there is a small number of channels, the proposed deep Q-learning scheme achieves close to the performance of the Q-learning-based defense strategy. In the case of having a large number of channels with multiple jammers, the deep Q-learning-based strategy outperforms to the Q-learning counterpart. To further overcome the overestimation with upward biased errors in stochastic environments in DQN, which are resulted from either measurements or non-stationary, function approximators [31], we will also investigate double deep Q-network (DDQN) [32] as an overestimation solution in the considered framework. Through numerical examples, we show that the DDQN approach can remove the overestimation, and it outperforms both Q-learning and DQN algorithms. Furthermore, In addition, the expected loss due to the overestimation experienced in DQN is analyzed under the assumption that the estimate error of the action value follows a zero-mean Gaussian distribution.

The remaining of the paper is organized as follows. In Section 2, we introduce the system model under different jamming attacks, including non-uniform sweeping attacks. An anti-jamming strategy based on DQN is presented in Section 3. Following that, in Section 4, we introduce a DDQN-based anti-jamming approach and demonstrate its benefits. Numerical examples are provided in Section 5 to show the superiority of the proposed deep reinforcement learning methods. Finally, conclusions are drawn in Section 6.

2. System Model

We consider a communication system serving one user using N orthogonal communication channels, either in frequencies, beams, or transponders. At the same time, there exist multiple jammers trying to attack the communication. When the user accesses a channel n, $1 \le n \le N$, its signal-to-noise ratio (SNR) is given by $\gamma_{n,1}$, if there is no jammer on that very channel. In contrast, the user's SNR is $\gamma_{n,0}$ if there is a jammer on the channel. Clearly, $\gamma_{n,1} > \gamma_{n,0}$. The jammed channel is likely unusable when the jammer sends a high-power jamming signal. We assume that the user can access one channel, while a jammer can only attack one channel at any time. Using Shannon capacity, the achievable rate of the user in the absence of the jammer is given by

$$R_n = \log_2(1 + \gamma_{n,1}). \tag{1}$$

When the communication is jammed, the achievable rate is reduced to

$$L_n = \log_2(1 + \gamma_{n,0}). \tag{2}$$

For the system under consideration, at a given time slot t, the user receives a payoff defined as follows:

$$\begin{aligned} U_t \quad = \quad & R_n \cdot \mathbf{1}(\text{Successful transmission on channel } n) \\ & + L_n \cdot \mathbf{1}(\text{Jammed on channel } n) \\ & - C \cdot \mathbf{1}(\text{Choosing the action “hop”}), \end{aligned} \tag{3}$$

where R_n and L_n are defined by (1) and (2), respectively, and C is the cost associated with hopping. Furthermore, $\mathbf{1}$ is the indicator function that takes a value of 1 when the statement holds and a value of 0 otherwise.

2.1. Agent's Hopping Models

We consider two types of user: a smart agent or a random hopper, depending on the user's hopping strategies.

A ***smart agent*** (or ***smart hopper***) can learn to optimize its hopping policy. As in [25], we assume a smart hopper learns its hopping policy through Q-learning. Specifically, a smart hopper initiates a Q-table to approximate an action-value function. The smart hopper can observe its payoff function by accessing each possible channel and updating its Q-function accordingly.

A ***random hopper*** does not actively learn its hopping policy. Instead, it randomly hops around its accessible channels using a pre-determined hopping policy. In order to reduce the hopping cost, the random hopper must not hop at every time slot. We assume that the random hopper will stay at a particular channel with the probability of $P_a = \rho^k$, where k is the number of time slots that the random hopper has not been jammed on that channel and $\rho < 1$. Otherwise, the random hopper would choose to hop to another channel with the probability of $(1 - P_a)/(N - 1)$. Indeed, if the random hopper is jammed, resulting in $k = 0$, the random hopper will hop to another channel with a probability of 1.

2.2. Attacker's Jamming Models

Extending from the work of [25], we assume that there exist multiple types of smart jammers; each can perform one of the following jamming strategies:

- ***Sweeping attacker:*** A sweeping jammer uses a sweeping attack [12] to jam the user sequentially, one channel in each time slot, using a set of non-equal probabilities $\mathbf{q} = [q_1, \ldots, q_N]$ at the beginning of a sweeping sequence where q_m is the attacking probability at channel-m in the first round and $\sum_{m=1}^{N} q_m = 1$. The jammer will attack by sweeping all the channels until the user is jammed. It is assumed that the jammer knows the outcome of its attack. If the user were jammed on a particular channel, the jammer would continue jamming the same channel until the user hops to another channel. The sweeping jammer then initiates a new sweeping sequence. A sweeping attacker will force the agent to hop frequently instead of staying at the best available channel at all times [25].

  Suppose the jammer unsuccessfully attacks channel-m in the first round. In that case, the user does not use channel m, and the jammer will revise the attacking probability at channel $n \neq m$ in the second round. We proposed two different rules for updating the probability vector in [25]:

  – Rule R1:

  $$q_n^{(2)} = q_n + \frac{q_n q_m}{1 - q_m}. \tag{4}$$

  – Rule R2: The splitting of q_m is equal such that

  $$q_n^{(2)} = q_n + \frac{q_m}{N - 1}. \tag{5}$$

  Following these rules, we can generalize the channels' attacking probability in the $A + 1$ round. In particular, for a given set of attacked channels in the first A rounds $\mathcal{N}_A$, the next channel $n \notin \mathcal{N}_A$ attacked has the following probability:

  – Rule R1:

  $$q_n^{(A+1)} = q_n + \frac{q_n \sum_{m \in \mathcal{N}_A} q_m}{1 - \sum_{m \in \mathcal{N}_A} q_m}. \tag{6}$$

  – Rule R2:

  $$q_n^{(A+1)} = q_n + \frac{\sum_{m \in \mathcal{N}_A} q_m}{N - A}. \tag{7}$$

- **Random attacker:** The attacker jammer continues jamming a channel with a probability $P_a = \rho^k$, where k is the number of time slots the jammer has been jamming on that channel and $\rho < 1$. Otherwise, the jammer would choose to jam another channel with probability $(1 - P_a)/(N - 1)$.

3. Anti-Jaming with Deep Q-Network (DQN) Approach

3.1. Deep Reinforcement Learning Algorithm

When the jamming strategy is unknown to the user, it may not be feasible to implement an MDP-based learning algorithm because of the difficulty in calculating the transition model $T(\cdot|\cdot, a)$. On the other hand, when the number of channels N and the number of attackers increases, the evaluation of the Q-table in Q-learning, while being possible, requires prohibitive computational complexity. In practice, the update of action-value functions in [25] may not be feasible because the action-value function is estimated separately for each sequence at a new episode. Instead, it is possible to use a function approximator $Q(s, a; \theta)$ to estimate the action-value function $Q(s, a)$, where the weights in θ are trainable to parameterize the Q-values.

The idea behind the breakthrough work in deep reinforcement learning [27] is to use a neural network with weights θ a nonlinear function approximator. The neural network is trained to minimize a sequence of loss function $L_i(\theta_i)$ at iteration i where

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}\left[(y_i - Q(s, a; \theta_i))^2\right] \tag{8}$$

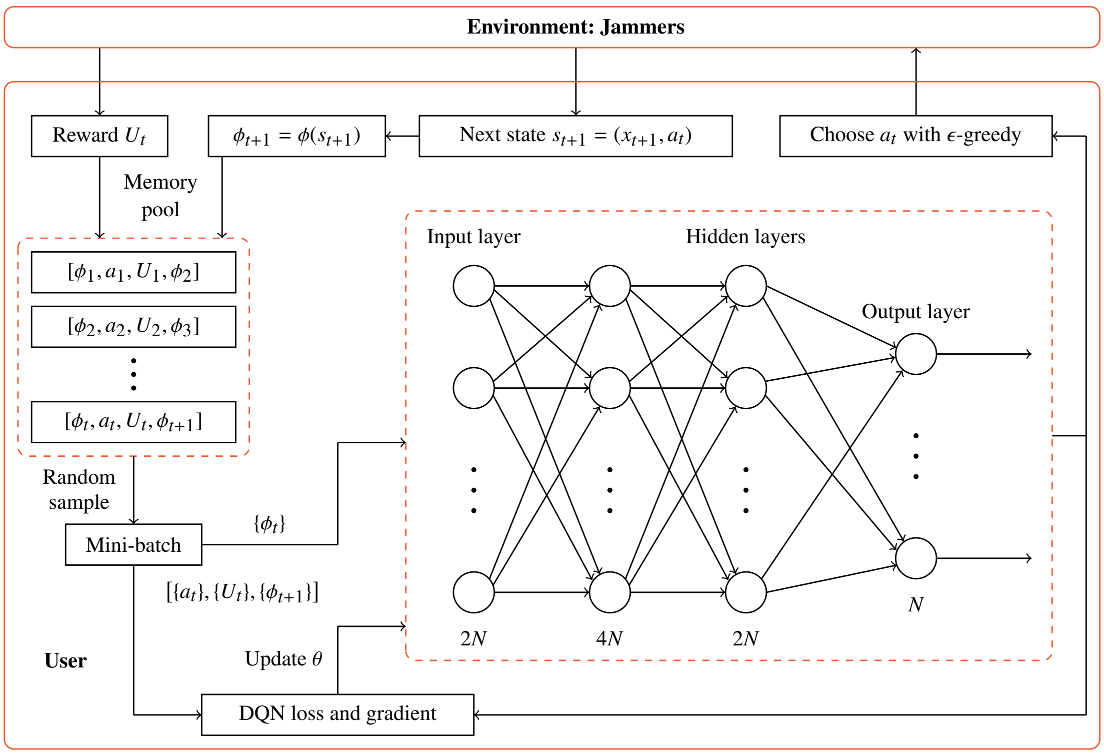
Environment: Jammers
Reward U_t
$\phi_{t+1} = \phi(s_{t+1})$
Next state $s_{t+1} = (x_{t+1}, a_t)$
Choose a_t with ϵ-greedy
Memory pool
$[\phi_1, a_1, U_1, \phi_2]$
$[\phi_2, a_2, U_2, \phi_3]$
$[\phi_t, a_t, U_t, \phi_{t+1}]$
Random sample
Mini-batch
$\{\phi_t\}$
$[\{a_t\}, \{U_t\}, \{\phi_{t+1}\}]$
User
Update θ
DQN loss and gradient
Input layer
Hidden layers
Output layer
$2N$
$4N$
$2N$
N

Figure 1. DQN-based anti-jamming strategy.

where

$$y_i = \mathbb{E}_s \left[U + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \right], \tag{9}$$

is the target for iteration i and $p(\cdot)$ is a probability distribution over states s and actions a. Furthermore, $0 < \gamma < 1$ is the discount factor and the parameters from the previous iteration θ_{i-1} are held fixed. The loss function can be optimized by stochastic gradient descent to update θ:

$$\theta_{i+1} = \theta_i + \mu_i \nabla_\theta \mathcal{L}_i(\theta_i), \tag{10}$$

where μ_i is the step size at iteration i.

The step-by-step implementation of the deep reinforcement learning algorithm, as proposed in [27], is given in Algorithm 1. The state input s is first preprocessed by a function $\phi(\cdot)$. The detailed operation of $\phi(\cdot)$ is given in Section 3.2. At a given time slot t, the action a_t at state s_t is determined by the output of the function approximator $Q(\phi(s_t), a; \theta))$ in conjunction with an ϵ-greedy algorithm. This greedy algorithm ensures adequate exploration of the state space. The user then executes the action a_t, receives the reward U_t, and observe the next state s_{t+1}. The user then uses the function approximator $Q(\phi(s_{t+1}), a; \theta))$ to approximate the expected discounted reward from state s_{t+1}, which is combined with the immediate reward U_t to determine the target for iteration i. The tuple of user experience at time slot t, $e_t = (\phi_t, a_t, U_t, \phi_{t+1})$ is pooled over many episodes into a data set of replay memory $\mathcal{M}$. A mini-batch of random samples of experiences are drawn from the memory and then used for updating the neural network parameters θ to minimize the loss function for the whole mini-batch. The stochastic gradient descent update in (10) with random samples allows the algorithm to break the correlations between consecutive samples [27].

3.2. Neural Network Architecture for DQN-based Anti-jamming Strategy

In Fig. 1, we illustrate the overall framework of the DQN-based anti-jamming strategy. In the figure, the implementation of a neural network at the user as a nonlinear function approximator $Q(s, a; \theta)$ is given as follows:

- The input layer accepts a length-$2N$ for the preprocessed input of the current state $\phi_t = \phi(s_t)$. Since the current state $s_t = (x_t, n_t)$ is comprised of two parameters $x_t, n_t \in [1, N]$, we encode x_t and n_t into length-N one-hot vectors, $\mathbf{x}_t$ and $\mathbf{n}_t$, respectively. Here, the ith element of $\mathbf{x}_t$ is set to 1 and all other elements are set to 0 if $x_t = i$. The one-hot vector $\mathbf{n}_t$ is formatted similar using the value of n_t. Stacking $\mathbf{x}_t$ and $\mathbf{n}_t$ makes $\phi_t = [\mathbf{x}_t, \mathbf{n}_t]^T$ a length-$2N$ vector. Note that N is the number of channels described in Section 2.

- The network is comprised of two fully connected hidden layers: the first hidden layer has $4N$ nodes and the second one has $2N$ nodes. The configuration of the layers was chosen to maintain a good tradeoff between overfitting and underfitting in training the agent.

Algorithm 1 DQN with Experience Play

```
1:  Initiate the discount factor $\gamma$.
2:  Initiate replay memory $\mathcal{M}$ with length $L$.
3:  Initiate action-value function $Q$ with random weights.
4:  for Episode $i$ do
5:      for $t = 1, 2, \ldots, T$ do
6:          Initialize state $s_1$ and preprocessed state $\phi_1 = \phi(s_1)$.
7:          Use $\epsilon$-greedy algorithm to choose action $a_t$:
            $$a_t = \begin{cases} \arg\max_a \ Q(\phi(s_t), a; \theta), & \text{with probability } 1 - \epsilon \\ \text{random channel } \in [1, N], & \text{with probability } \epsilon. \end{cases} \quad (11)$$
8:          Execute action $a_t$ and observe reward $U_t$.
9:          Observe the next state $s_{t+1}$ and preprocessed state $\phi_{t+1} = \phi(s_{t+1})$.
10:         Store transition $(\phi_t, a_t, U_t, \phi_{t+1})$ in $\mathcal{M}$.
11:         Sample random minibatch of transition $(\phi_j, a_j, U_j, \phi_{j+1})$ from $\mathcal{M}$
12:         Set the target $Q$ value
            $$y_j = \begin{cases} U_j, & \text{for terminal } \phi_{j+1} \\ U_j + \gamma \max_{a'} \ Q(\phi_{j+1}, a'; \theta), & \text{for non-terminal } \phi_{j+1} \end{cases}$$
13:         Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ to update the parameters $\theta$.
14:     end for
15: end for
```

- The output layer comprised of N nodes provides the approximated action-value function $Q(s, a)$ for all N actions a.

Using the neural network output and the ϵ-greedy strategy, the user takes action a_t and interacts with environment. Depending on the jamming strategy by the jammers at time slot t and the user's own action, the user will receive the reward U_t and move to the next state s_{t+1}. After a mini-batch of random samples are drawn from the memory pool, the user evaluate the loss function and the gradient, which then enables the training of the neural networks parameters θ. To improve the efficiency in training the DQN, we set the parameters ϵ decaying over time. This setting allows the user to do more exploration at the beginning of the training process and more exploitation toward the end. The step size used in the stochastic gradient descent is also set to be decaying over time to improve the stability in training the DQN.

4. Anti-Jamming with Double Deep Q-Network (DDQN)

The DQN implementation of the target value shown in (9) can be expressed as:

$$y_t^{DQN} = U_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_{t-1}). \quad (12)$$

The maximum operator occurs in (12) and tends to be biased on overestimated values of the Q-function in the next state for both the action selection and evaluation. The idea behind DDQN is to reduce the overestimation by decoupling the maximum operation of the target into the action selection and action evaluation from two neural networks. The target update for DDQN is given as [32]

$$y_t^{DDQN} = U_{t+1} + \gamma Q\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t), \theta_{t-1}\right), \quad (13)$$

where θ_t denotes the weights of the action selection and θ_{t-1} denotes the weights for action evaluation. The above update was shown to reduce the overestimations present in DQN and improve the performance. There are two models: the primary model is for action selection using weights θ_t, and the target model for action evaluation with weights θ_{t-1}. The weights of the target model are periodically updated to match the primary model. Detailed implementation of the DDQN is summarized in Algorithm 2 below.

Algorithm 2 DDQN with Experience Play

```
1:  Initiate the discount factor $\gamma$.
2:  Initiate replay memory $\mathcal{M}$ with length $L$.
3:  Initiate primary network $Q_{\theta_t}$ and target network $Q_{\theta_{t-1}}$ with random weights.
4:  for Episode $i$ do
5:      for $t = 1, 2, \ldots, T$ do
6:          Observe state $s_t$.
7:          Use $\epsilon$-greedy algorithm to choose action $a_t$:
            $$a_t = \begin{cases} \arg\max_a \ Q(\phi(s_t), a; \theta), & \text{with probability } 1 - \epsilon \\ \text{random channel } \in [1, N], & \text{with probability } \epsilon. \end{cases} \quad (14)$$
8:          Execute action $a_t$ and observe reward $U_t$.
9:          Observe the next state $s_{t+1}$ and preprocessed state $\phi_{t+1} = \phi(s_{t+1})$.
10:         Store transition $(\phi_t, a_t, U_t, \phi_{t+1})$ in $\mathcal{M}$.
11:         for each update step do
12:             Sample random minibatch of transition $(\phi_j, a_j, U_j, \phi_{j+1})$ from $\mathcal{M}$
13:             Compute target $Q$ value
                $$Q_j^* = \begin{cases} U_j, \text{ for terminal } \phi_{j+1} \\ U_j + \gamma Q_{\theta_{t-1}}\left(\phi_{j+1}, \arg\max_{a'} Q_{\theta_t}\left(\phi_{j+1}, a'; \theta_t\right), \theta_{t-1}\right), \\ \text{for other terminals} \end{cases}$$
14:             Perform a gradient descent step on $\left(Q_j^* - Q_{\theta_t}(\phi_j, a_j; \theta_t)\right)^2$ to update premary weight $\theta_t$.
15:         end for
16:         Update target network weights $\theta_{t-1} \leftarrow \theta_t$.
17:     end for
18: end for
```

Given the above DDQN approach, it is also of interest to evaluate the performance difference of DDQN and DQN at convergence. Toward this end, we have the following theorem:

Theorem 1. Consider a state s in which a true optimal action value $Q^*(s,a) = V^*(s)$. Suppose the estimation error $\epsilon_a = Q(s,a) - Q^*(s,a)$ is a zero mean Gaussian random variable with variance σ^2, i.e., $\epsilon_a \sim \mathcal{N}(0,\sigma^2)$. We then have:

$$E\left[\max_a Q(s,a) - V^*(s) \,|\, \max_a Q(s,a) - V^*(s) \geq b\right] = \left(\frac{\sigma\phi\left(\frac{b}{\sigma}\right)}{1-\Phi\left(\frac{b}{\sigma}\right)}\right)^N, \quad b \geq 0, \tag{15}$$

where N is the number of channels, $\phi(x) = \frac{1}{\sqrt{2\pi}}\exp(\frac{-x^2}{2})$ is the standard normal probability density function (PDF) and $\Phi(x) = \int_\infty^t \phi(t)dt$ is the standard normal cumulative distribution function (CDF).

Proof. Define the error as:

$$\epsilon_a = Q(s,a) - V^*(s). \tag{16}$$

Since ϵ_a is a zero mean Gaussian random variable, the probability that $(\max_a Q(s,a) - V^*(s)) \geq x$ for some x is equal to the probability that $\epsilon_a \geq x$ for all a simultaneously. That is:

$$P\left[\max_a Q(s,a) - V^*(s) \geq x\right] = P(X_1 \geq x \cap X_2 \geq x \cdots X_N \geq x). \tag{17}$$

For a single error, the PDF of the distribution $(X \geq b)$ is given by:

$$f_T(x) = \frac{\frac{1}{\sigma}\phi\left(\frac{x}{\sigma}\right)}{1-\Phi\left(\frac{b}{\sigma}\right)}, \quad x \geq b, \quad \text{and} \quad \int_b^\infty f_T(x)\,dx = 1. \tag{18}$$

The expected value is given by:

$$E[X|X \geq b] = \frac{\int_b^\infty x f_T(x)\,dx}{\int_b^\infty f_T(x)\,dx} = \frac{\sigma\phi\left(\frac{b}{\sigma}\right)}{1-\Phi\left(\frac{b}{\sigma}\right)}. \tag{19}$$

Since all errors are independent, the joint PDF of the distribution is given by:

$$f_T(X_1, X_2, \cdots, X_N) = \prod_{i=1}^N f_T(X_i) = \prod_{i=1}^N \frac{\frac{1}{\sigma}\phi\left(\frac{X_i}{\sigma}\right)}{1-\Phi\left(\frac{b}{\sigma}\right)}. \tag{20}$$

Using (19) and (20), the conditional expected value of the join distribution is:

$$\begin{aligned} &E[X_1X_2\cdots X_N | X_1 \geq b, X_2 \geq b, \cdots, X_N \geq b] \\ &= \int_b^\infty \cdots \int_b^\infty x_1 \cdots x_N f_T(x_1, \cdots, x_N)\,dx_1 \cdots dx_N \\ &= \int_b^\infty x_1 f_T(x_1)\,dx_1 \cdots \int_b^\infty x_N f_T(x_N)\,dx_N \\ &= \left(\frac{\sigma\phi\left(\frac{b}{\sigma}\right)}{1-\Phi\left(\frac{b}{\sigma}\right)}\right)^N. \end{aligned} \tag{21}$$

□

Finally, the expected loss can be established in the following lemma.

Lemma 1. Given the error is positively biased, the expected loss is given by:

$$E_L = E\left[\max_a Q(s,a) - V^*(s) \,|\, \max_a Q(s,a) - V^*(s) \geq 0\right] = \left(\sqrt{\frac{2}{\pi}}\sigma\right)^N. \tag{22}$$

Proof. Let $b = 0$ then, $\phi(0) = \frac{1}{\sqrt{2\pi}}$ and $\Phi(0) = \frac{1}{2}$ and substitute these parameters into (21), we have:

$$E_L = E\left[\max_a Q(s,a) - V^*(s) \,|\, \max_a Q(s,a) - V^*(s) \geq 0\right] = \left(\sqrt{\frac{2}{\pi}}\sigma\right)^N. \tag{23}$$

□

The result in from Lemma 1 indicates that E_L is the performance difference of DDQN and DQN at convergence. Using empirical data, we can estimate the standard deviation of the estimate error, which is

$$\hat{\sigma} = \sqrt{\frac{\pi}{2}} \cdot E_L^{\frac{1}{N}}. \tag{24}$$

As N goes to infinity, we have:

$$\hat{\sigma} \to \sqrt{\frac{\pi}{2}} = 1.253. \tag{25}$$

5. Numerical Examples

In this section, numerical results are provided to illustrate the effectiveness of the proposed deep Q-learning solutions. The DQN is implemented and trained using the Tensorflow package [33]. For each N-channel system, SNRs are set within the interval [10 dB, 30 dB] with a uniform spacing between each SNR and the step size of 20 dB/N. Furthermore, if a channel is hit by jamming, the SNR is reduced by 20 dB. Unless otherwise stated, the sweeping attack is assumed to be non-uniform, with the attacking probabilities being proportional to the SNRs. For the Q-learning algorithm, the initial learning rate is always set at 10^{-2}. In addition, when Q-learning is implemented, the discount rate is $\gamma = 0.95$. The performance of each scheme shall be compared using the achievable rate achieved over time as the reward.

Figure 2 first compares the results achieved by Q-learning in [25] and the proposed deep Q-network (Algorithm 1) as well as double deep Q-network (Algorithm 2) for a scenario consisting of 32 channels under jamming attack from a random jammer and a sweeping jammer. The cost associated with hopping is set at $C = 2$. The DQN and DDQN methods are implemented and trained utilizing the

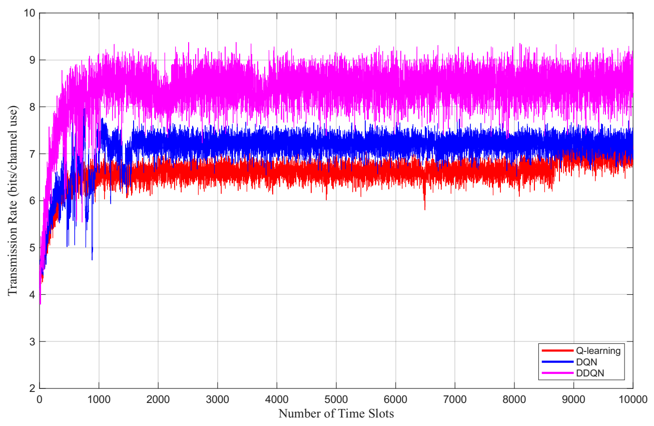

Figure 2. Achievable rates of a 32-channel communication system with hopping cost $C = 2$ achieved by Q-learning, DQN, and DDQN under jamming attacks by a random jammer and a non-uniform sweeping jammer.

TensorFlow package [24]. The step size μ in updating the policy of the DQN and DDQN is initialized at 0.01 and set to decay after 150 iterations with a decaying factor of 0.9. As shown in Fig. 2, all three algorithms converge at about the same rate. However, the DQN performs better than the Q-learning, and the DDQN removes the positive bias inherently in the Q-learning and DQN and performs the best. Specifically, as shown in Fig. 2, Q-learning converges at 7 bits/channel use, DQN converges at 7.2 bits/channel use, and DDQN converges at 8.5 bits/channel use. The results also indicate that DDQN removes the overestimations in the DQN and Q-learning algorithms.

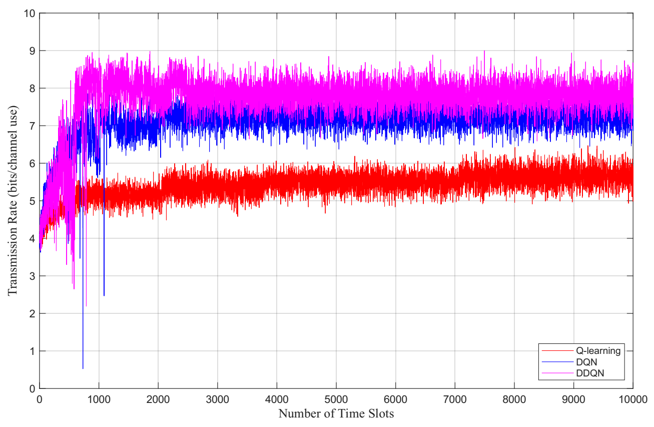

Figure 3. Achievable rates of a 64-channel communication system with hopping cost $C = 2$ achieved by Q-learning, DQN, and DDQN under jamming attacks by four random jammers and four non-uniform sweeping jammer.

Fig. 3 displays the Q-learning, DQN, and DDQN results for an agent with access to 64 channels. The agent communicates in an environment with four random jammers and four sweeping jammers. As shown in Fig. 3, Q-learning converges at 5.6 bits/channel use, DQN converges at 7.3 bits/channel use, and DDQN converges at 7.8 bits/channel use. The performance gain of DDQN over DQN is about 0.5 bits/channel use.

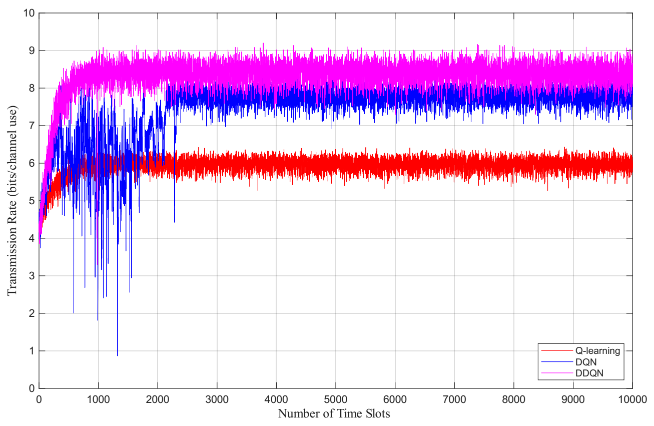

Figure 4. Achievable rates of a 128-channel communication system with hopping cost $C = 2$ achieved by Q-learning, DQN, and DDQN under jamming attacks by four random jammers and four non-uniform sweeping jammer.

The advantages of deep learning-based approaches can also be achieved in a larger action/state space. In particular, Fig. 4 presents the results when we have $N = 128$ channels; we keep all other parameters the same as before, using four random jammers and four non-uniform sweeping jammers. As expected, when N increases from 64 to 128, we see improvement in the transmission rate. With $N = 128$, Q-learning converges at 5.7 bits/channel use, DQN converges at 7.8 bits/channel use, and DDQN converges at 8.4 bits/channel use. Again, Q-learning is under-performing, and DDQN consistently outperforms DQN.

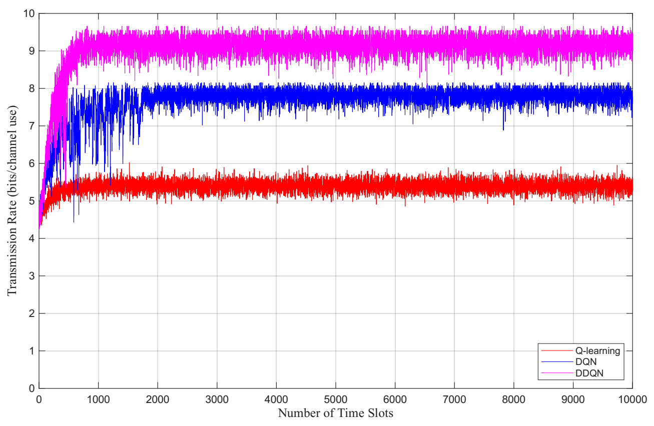

Figure 5. Achievable rates of a 256-channel communication system with hopping cost $C = 2$ achieved by Q-learning, DQN, and DDQN under jamming attacks by four random jammers and four non-uniform sweeping jammer.

Finally, Fig. 5 shows the achievable rates achieved by three learning methods in a very large space with $N = 256$ channels under jamming attacks from four random jammers and four non-uniform sweeping jammers. While the performance of DQN is almost identical to the case of $N = 128$ channels, DDQN provides further gains, achieving a rate of 9.3 bits/channel use at convergence. The advantage of DDQN over both DQN and Q-learning is clearly observed under this very large-action space scenario.

6. Conclusion

In this paper, we have proposed two deep reinforcement learning defense strategies against smart jammers that can attack channels with unequal probabilities. These algorithms offer effective solutions against the smart jammers without knowing their attacking models. The proposed DQN and DDQN approaches can overcome the main drawback of Q-learning-based methods and work well in a large decision or action space. Furthermore, it was shown that the DDQN method not only performs better than the Q-learning and DQN ones but also can eliminate the overestimation bias in scenarios with many channels under random and sweeping attacks. The expected loss of DQN was also derived under the assumption that the estimate error of the action value follows a zero-mean Gaussian distribution. Through numerical examples, the estimate of the standard deviation for the Gaussian distribution was provided, and in the limit, as N increases, the standard deviation approaches $\sqrt{\frac{\pi}{2}}$. The DDQN method appears to be the best anti-jamming defense strategy to counter smart jammers when the decision space is large without the need of explicitly knowing the jammers' attacking models.

7. Acknowledgment

This work was funded by the U.S. Space Force Space Systems Command under Contract No. FA8808-19-C-0006.

References

[1] L. Jia, N. Qi, Z. Su, F. Chu, S. Fang, K.-K. Wong, and C.-B. Chae, "Game theory and reinforcement learning for anti-jamming defense in wireless communications: Current research, challenges, and solutions," *IEEE Communications Surveys & Tutorials*, 2024.

[2] Y. Liu, B. Zhang, D. Guo, H. Wang, G. Ding, N. Yang, and J. Gu, "A game theoretical anti-jamming beamforming approach for integrated sensing and communications systems," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 10, pp. 15780–15785, 2024.

[3] X. Guan, Y. Hu, and K. Peng, "Bayesian-Stackelberg-game-based finite-time sliding mode fault-tolerant secure control for cyber–physical systems under jamming attacks and multiple physical faults," *IEEE Transactions on Cybernetics*, 2025.

[4] Y. Li, W. Miao, Z. Gao, and G. Lv, "Intelligent jamming strategy for wireless communications based on game theory," *IEEE Access*, 2024.

[5] D. Yang, G. Xue, J. Zhang, A. Richa, and X. Fang, "Coping with a smart jammer in wireless networks: A Stackelberg game approach," *IEEE Transactions on Wireless Communications*, vol. 12, pp. 4038–4047, Aug. 2013.

[6] A. Garnaev, Y. Liu, and W. Trappe, "Anti-jamming strategy versus a low-power jamming attack when intelligence of adversary's attack type is unknown," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, pp. 49–56, Mar. 2016.

[7] L. Jia, F. Yao, Y. Sun, Y. Niu, and Y. Zhu, "Bayesian Stackelberg game for antijamming transmission with incomplete information," *IEEE Communications Letters*, vol. 20, pp. 1991–1994, Oct. 2016.

[8] L. Jia, Y. Xu, Y. Sun, S. Feng, and A. Anpalagan, "Stackelberg game approaches for anti-jamming defence in wireless networks," *IEEE Wireless Communications*, vol. 25, pp. 120–128, Dec. 2018.

[9] L. Jia, Y. Xu, Y. Sun, S. Feng, L. Yu, and A. Anpalagan, "A game-theoretic learning approach for anti-jamming dynamic spectrum access in dense wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 1646–1656, Feb. 2019.

[10] C. Han, A. Liu, H. Wang, L. Huo, and X. Liang, "Dynamic anti-jamming coalition for satellite-enabled Army IoT: A distributed game approach," to appear in *IEEE Internet of Things Journal*, 2020.

[11] Q. Wang, T. Nguyen, K. Pham, and H. Kwon, "Mitigating jamming attack: A game-theoretic perspective," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 6063–6074, July 2018.

[12] Y. Wu, B. Wang, K. J. R. Liu, and T. C. Clancy, "Anti-jamming games in multi-channel cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, pp. 4–15, Jan. 2012.

[13] M. K. Hanawal, M. J. Abdel-Rahman, and M. Krunz, "Joint adaptation of frequency hopping and transmission rate for anti-jamming wireless systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2247–2259, 2016.

[14] M. K. Hanawal, M. J. Abdel-Rahman, and M. Krunz, "Game theoretic anti-jamming dynamic frequency hopping and rate adaptation in wireless systems," in *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 247–254, 2014.

[15] Z. Yin, J. Li, Z. Wang, Y. Qian, Y. Lin, F. Shu, and W. Chen, "UAV communication against intelligent jamming: A Stackelberg game approach with federated reinforcement learning," *IEEE Transactions on Green Communications and Networking*, vol. 8, no. 4, pp. 1796–1808, 2024.

[16] Y. Qin, J. Tang, F. Tang, M. Zhao, and N. Kato, "Multi-agent reinforcement learning in adversarial game environments: Personalized anti-interference strategies for heterogeneous uav communication," *IEEE Transactions on Mobile Computing*, 2025.

[17] Z. Lin, L. Xiao, H. Chen, and Z. Lv, "Reinforcement learning based environment-aware v2i anti-jamming communications," *IEEE Transactions on Vehicular Technology*, 2024.

[18] B. He, N. Yang, X. Zhang, and W. Wang, "Game theory and reinforcement learning in cognitive radar game modeling and algorithm research: A review," *IEEE Sensors Journal*, 2024.

[19] M. Chen, F. Shu, M. Zhu, D. Wu, Y. Yao, and Q. Zhang, "Reinforcement-learning-based uav 3-d target tracking and digital-twin-assisted collision avoidance with integrated sensing and communication," *IEEE Internet of Things Journal*, 2025.

[20] Y. Ma, K. Liu, Y. Liu, X. Wang, and Z. Zhao, "An intelligent game-based anti-jamming solution using adversarial populations for aerial communication networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 3, pp. 1981–1995, 2025.

[21] Y. Gwon, S. Dastangoo, C. Fossa, and H. T. Kung, "Competing mobile network game: Embracing antijamming and jamming strategies with reinforcement learning," in *IEEE Conference on Communications and Network Security (CNS)*, pp. 28–36, 2013.

[22] N. Adem and B. Hamdaoui, "Jamming resiliency and mobility management in cognitive communication networks," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2017.

[23] M. A. Aref, S. K. Jayaweera, and S. Machuzak, "Multi-agent reinforcement learning based cognitive anti-jamming," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2017.

[24] X. He, H. Dai, and P. Ning, "Faster learning and adaptation in security games by exploiting information asymmetry," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3429–3443, 2016.

[25] L. K. Nguyen, D. H. N. Nguyen, N. H. Tran, C. Bosler, and D. Brunnenmeyer, "SATCOM jamming resiliency under non-uniform probability of attacks," in *IEEE Military Communications Conference (MILCOM)*, pp. 85–90, 2021.

[26] L. K. Nguyen, D. H. N. Nguyen, N. H. Tran, C. Bosler, and D. Brunnenmeyer, "Coordinated multi-agent q-learning for resilient SATCOM against smart jammers," in *IEEE Military Communications Conference (MILCOM) (Restricted Access)*, pp. 1–6, 2022.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[28] L. Xiao, D. Jiang, D. Xu, H. Zhu, Y. Zhang, and H. V. Poor, "Two-dimensional antijamming mobile communication based on reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 9499–9512, Oct. 2018.

[29] X. Liu, Y. Xu, L. Jia, Q. Wu, and A. Anpalagan, "Anti-jamming communications using spectrum waterfall: A deep reinforcement learning approach," *IEEE Communications Letters*, vol. 22, pp. 998–1001, May 2018.

[30] N. Gao, Z. Qin, X. Jing, Q. Ni, and S. Jin, "Anti-intelligent UAV jamming strategy via deep Q-networks," *IEEE Transactions on Communications*, vol. 68, pp. 569–581, Jan. 2020.

[31] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," 10 1993.

[32] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094––2100, AAAI Press, 2016.

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.