Adversarial Jamming Attacks on Deep Reinforcement Learning Based Dynamic Multichannel Access

Chen Zhong, Feng Wang, M. Cenk Gursoy, and Senem Velipasalar
Department of Electrical Engineering and Computer Science,
Syracuse University, Syracuse, NY 13244
Email: czhong03@syr.edu, fwang26@syr.edu, mcgursoy@syr.edu, svelipas@syr.edu

***Abstract*—Adversarial attack strategies have been widely studied in machine learning applications, and now are increasingly attracting interest in wireless communications as the application of machine learning methods to wireless systems grows along with security concerns. In this paper, we propose two adversarial policies, one based on feed-forward neural networks (FNNs) and the other based on deep reinforcement learning (DRL) policies. Both attack strategies aim at minimizing the accuracy of a DRL-based dynamic channel access agent. We first present the two frameworks and the dynamic attack procedures of the two adversarial policies. Then we demonstrate and compare their performances. Finally, the advantages and disadvantages of the two frameworks are identified.**

***Index Terms*—Adversarial policies, dynamic channel access, deep reinforcement learning, feed-forward neural networks.**

I. Introduction

There has recently been growing interest in employing machine learning to address certain problems in communication systems, such as modulation classification [1], and dynamic multichannel access [2]. However, this increasing interest brings forth potential security risks due to adversarial attacks. Since machine learning methods are highly data-driven algorithms, even a minor modification in the observation data can lead to dramatic changes in the learning-based decision policies. Therefore, adversarial machine learning has been intensively studied to better understand the vulnerabilities of machine learning methods. Motivated by this, we in this paper investigate the learning-based wireless jamming attacks on deep reinforcement learning policies on dynamic multichannel access.

In the literature, adversarial attacks have been considered and widely applied to deep learning-based classification problems, such as the classification of images [3], time series [4] and sound events [5]. In these cases, the victim models are trained and fixed, and the input data is accessible to the attacker, so that the attack can be realized by crafting adversarial examples to mislead the victim's decisions. This idea is also used in the attack on reinforcement learning-based tasks [6] and [7]. However, in certain control problems, the observations of the reinforcement learning agents are not available to the attacker, making it infeasible to craft any adversarial examples. To tackle this difficulty, in [8], the authors trained a reinforcement learning-based adversarial policy instead.

A deep learning-based wireless jamming attack has been studied in [9] and [10], in both of which, the system consists of a single transmitter, a receiver, one background traffic source and a deep learning-based jammer. Inspired by the framework presented in the literature, we address a more general channel model introduced in [11], and we propose two different jamming attackers, namely a feed-forward neural network (FNN) attacker and a deep reinforcement learning (DRL) attacker, to perform the jamming attacks on a user performing dynamic multichannel access using a DRL agent itself [11].

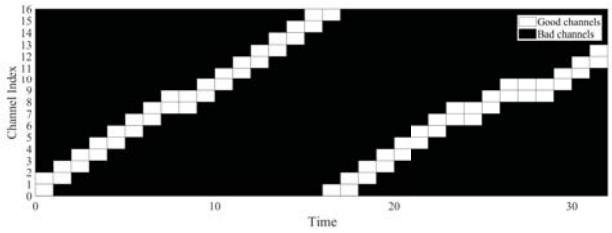

Fig. 1. Round-robin switching pattern when two of the 16 channels is in good condition and the switching probability is $\rho = 0.95$. The channel in good state at a given time is indicated by white squares.

II. Dynamic Channel Access Policies of the Victim User

In this section, we introduce the background on dynamic multichannel access. As noted above, we consider an actor-critic DRL agent proposed in [11] as the victim user to be attacked.

A. Channel Switching Pattern

In the considered dynamic multichannel access problem, the time is slotted and the user selects one channel to access at the beginning of each time slot. We assume that the state of each channel switches between good and bad in a certain probabilistic pattern. When the channel is in good condition, the user can transmit data successfully. Otherwise, a transmission failure will occur. We also assume that the channel switching pattern can be modeled as a Markov chain, and in each state of which, there are k out of the N channels in good condition. At the beginning of each time slot, the channel pattern can either switch to the next state with probability of ρ, or remain to be the same as the state in the last time slot with probability of $(1-\rho)$. In Fig. 1, we display a round-robin switching pattern with two out of 16 channels being good in each time slot and each channel has the same probability to be in good state.

B. Actor-Critic Agent

It is assumed that the channel switching pattern is unknown to the user, and the user can only observe the channel selected in the current time slot. Hence, the multichannel access is a partially observable Markov decision process (POMDP). To help the user to access the good channels as frequently as possible under such conditions, we proposed in [11] an actor-critic deep reinforcement learning based agent to make the channel access decisions in each time slot.

The proposed agent is designed to learn the channel switching pattern through past decisions and the corresponding feedback from the channels. We assume that, at time t, the channel state can be denoted as $\mathtt{X}_t = \{\mathtt{x}_1, \mathtt{x}_2, ..., \mathtt{x}_N\}$, where

978-1-7281-3106-1/20/$31.00 ©2020 IEEE

N is the total number of channels, x_i stands for the state of the i^{th} channel. For each channel i, where $i = 1, 2, ..., N$, we have $\mathrm{x}_i = 1$ if the channel is in good state, or $\mathrm{x}_i = 0$ if the channel is in bad state. And each time the agent senses a channel, the state of the sensed channel is revealed to be either good or bad. Therefore, we define the reward (feedback) as follows: if a good channel is chosen, the reward r_t will be $+1$; otherwise, the reward r_t will be -1.

The agent's observation can be denoted as $O_t = \{o_1, o_2, ..., o_N\}$, where N is the total number of channels. If channel i, $i = 1, 2, ..., N$, is chosen, the agent senses it and learns its state, so we define $o_i = r_t$; otherwise, the agent will record $o_i = 0$. The agent will learn on the basis of its previous experience. We assume that the agent keeps an observation space $\mathcal{O}$ that consists of the most recent M observations. The observation space is initialized as an all-zero $N \times M$ matrix, and at each time t, the latest observation O_t will be added to the observation space, and the oldest observation O_{t-M} will be removed. The updated observation space $\mathcal{O}$ at time $t+1$ can be denoted as $\mathcal{O}_{t+1} = \{O_t; O_{t-1}; ...; O_{t-(M-1)}\}$.

Next, we consider a discrete action space denoted by $\mathcal{A} = \{1, 2, ..., N\}$, where N is the total number of channels. Each valid action in the action space describes the index of the channel that will be accessed. Hence, when an action is chosen, the agent will access the corresponding channel and receive the reward which reveals the condition of the chosen channel. The agent can only choose one channel to sense/learn in each iteration. The aim of the agent is to find a policy π, which maps the observation space $\mathcal{O}$ to the action space $\mathcal{A}$, that maximizes the long-term expected reward R of channel access decisions:

$$\pi^* = \arg\max_{\pi} R$$

where π^* denotes the optimal decision policy, and in a finite time duration T, we express R as

$$R = \frac{1}{T}\sum_{t=1}^{T} r_t.$$

And according to the definition of R, we have $R \in [-1, 1]$.

C. Performance in the Absence of Jamming Attacks

We consider the channel switching pattern shown in Fig. 1, and evaluate the accuracy of the good channel access by the user with $N = M = 16$. The evaluation is performed in the absence of any jamming attacks and after the DRL agent is well trained. In Fig. 2, we test the model in two cases. First, we consider the ϵ-greedy policy with $\epsilon = 0.1$, with which the user accesses a random channel with probability 0.1, and chooses the channel selected by the reinforcement learning policy with probability 0.9. Note that the ϵ-greedy policy allows the model to access bad channels by chance during exploration. In addition, we also consider the case in which ϵ is set to 0 to identify the performance of the pure DRL policy. We note that ϵ-greedy policies with $\epsilon > 0$ are generally employed to enhance the DRL agent's ability against changes in the channel patterns, as will be discussed in detail in Section V. We observe in the figure that high average accuracies (higher than 85% and around 95% with $\epsilon = 0.1$ and $\epsilon = 0$, respectively) are attained in the absence of jamming attacks.

III. FNN Jamming Attacker

In this section, we analyze the FNN method to perform jamming attacks on the actor-critic DRL dynamic multichannel access agent described in Section II-B. A presumptive attacker is able to choose and jam a single channel in each time slot to significantly reduce the selection accuracy of the actor-critic agent. We assume that the attacker employs a feed-forward neural network (FNN) to make the decision on which channel to attack.

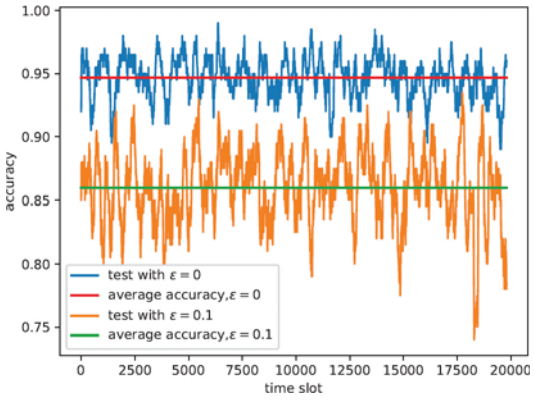

Fig. 2. Accuracy of the good channel access in the absence of jamming attacks.

A. Initial FNN Model

We build the FNN with TensorFlow as the attacker model. To collect initial training data for this FNN, we assume the attacker has another actor-critic agent which has a similar performance as the victim model does. These two models do not necessarily have the same parameters, as we need to retrain the initial FNN before attack. From this attacker actor-critic agent, we obtain the channel selection during 53 consecutive time slots as training data for FNN.

The FNN model feeds on 3 previous channels as input, and gives the probability among each channel in the next time slot as output. It has 2 hidden layers with 16 hidden neurons in each layer, with sigmoid activation function, RMSProp optimizer, and mean squared error loss function. With 50 sets of 3-previous-1-future data pairs from attacker's actor-critic agent, we set random weights in the FNN, and run 4000 iterations to train the FNN model (hereinafter referred to as the initial FNN), which has 88% accuracy on extra testing data. We intentionally limit the amount of training data and iterations to avoid overfitting to the initial policy of the victim actor-critic agent, which will greatly change under attack.

B. Channel Observation and Record

Before the attacker starts the jamming attack with FNN, it observes one channel and the reward of the victim user to determine if its attack is successful. The attacker also records the history of channel selection as input to the FNN to predict the next attacking choice. However, if the attacker simply records the attacked channel, once FNN misses to predict the victim user's channel selection, the attacker will lose track of the victim, and it will take some time to accurately predict the victim's chosen channel again. Thus, we suggest an alternative strategy, utilizing the initial FNN as a good channel detector. This initial FNN always keeps the initial parameter, and thus it is different from the adapting FNN which is affected by victim policy during the dynamic attack, as we will describe in Section III-C.

As depicted in Fig. 3, we use two sets of history records and FNNs. First, the initial FNN keeps its record of "ideal channel" and makes its own prediction. Since the initial FNN imitates the high-performance victim, the ideal channel generated by initial FNN is close to the good channels. Second, the adapting FNN keeps another set of "attacked channel" record and decides which channel to attack in the next time period. The observation of the channel and reward determines the next

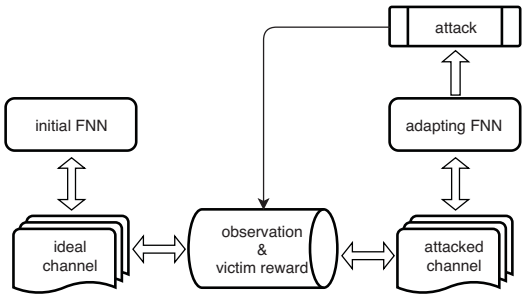

Fig. 3. The diagram of the history records and FNNs

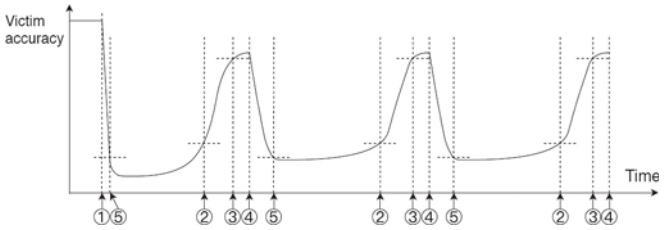

Fig. 4. Retrieve-retrain-attack-stop procedure of dynamic attack: ① initial attack ② stop attack ③ retrieve parameters and start retrain ④ stop retrain and start attack ⑤ stop updating the model (only for the DRL attacker)

channel to observe, and which channel to enter in both channel records. This is explicitly explained in Algorithm 1 below.

Algorithm 1 The loop of predicting, recording and observing during each time slot:

```
predict using adapting FNN, and attack the channel
record "attacked channel" as adapting FNN prediction
predict using initial FNN
record "ideal channel" as initial FNN prediction
if last time "attacker success" then
    observe the last different attacked channel
else if last time "observed success" then
    observe ideal channel
else if last time "failed" then
    observe the last observed channel
end if
if victim is observed and reward is positive then
    record observation as ideal channel and attacked channel
    mark as "observed success"
else if reward is negative then
    record observation as attacked channel
    mark as "attacker success"
else
    record attacked channel as ideal channel
    mark as "failed"
end if
```

C. Dynamic Attack

Based on the attacked channel record, we can use FNN to perform real-time jamming attacks against the victim. Although the initial FNN works well with the original policy of the victim actor-critic agent, it is not as accurate when the victim adapts to the attack with a new policy. As a control problem, there are two major considerations.

On the one hand, attacker FNN needs to retrain. When the attacker jams one of the good channels that the victim tends to choose, the victim will have low accuracy for the first few thousand time slots. After that, as the victim's actor-critic agent adapts to the attack, it learns a new policy to find the good channel and at the same time, mislead the attacker's FNN attack. Thus, the attacker should retrain the initial FNN (instead of starting with random weights), and attack with this retrained FNN to adapt the new victim policy.

On the other hand, attacker FNN needs to stop the attack, and retrieve the initial FNN parameters before retraining occasionally. If the attacker keeps retraining FNN, the victim accuracy would still recover gradually. There are two reasons for this. First, the parameters of FNN deviates from the initial FNN during long-term retraining, and lose the basic features (for example, taking the difference between channel values). This means that the attacker sets the FNN parameters to their initial values, and retrains to fit to the current victim policy. Second, if the attacker keeps on attacking, the data for retraining would reflect the setting in which the victim is under attack and is operating with low accuracy of good channel access. This prevents the attacker FNN from learning the desired victim pattern. One way to solve this problem is to stop attacking when the victim accuracy begins to recover, so the victim will converge fast to a stable policy with high accuracy. Then the attacker can retrieve, retrain using observations from this converged policy and perform better and more accurate attacks. Another benefit is that the attacker will not stay long in the recovering stage, where the victim average accuracy is up to 50% (which is much higher than the desired accuracy), so that the attacker can significantly reduce the overall average accuracy.

Therefore, we develop a retrieve-retrain-attack-stop (RRAS) procedure as depicted in Fig. 4 to perform dynamic attacks. At time ① shown in Fig. 4, we start the initial attack with the initial FNN, which is guaranteed to perform well at first. Then, the attacker will gradually lose control of the victim as it adapts to the initial attack. At time ②, the victim accuracy grows up to a lower threshold, so the attacker gives up attacking, and lets the victim recover fast from the initial attack, to reduce the time span between ② and ③ and reach a higher accuracy threshold at time ③. At time ③, the attacker retrieves the initial FNN parameters and collects the retraining data until time ④. Finally, the attacker initiates another attack at time ④, and the entire procedure is repeated as depicted in Fig. 4.

IV. DRL Jamming Attacker

In this section, we introduce an actor-critic deep reinforcement learning (DRL) based agent to perform the jamming attack on the aforementioned victim user without having any prior information about the channel switching pattern or the victim's action policy. The DRL attacker is also assumed to observe only one channel in each iteration. Different from the FNN attacker, however, we assume that the DRL attacker is able to observe the victim's interaction the environment for a period of time that is sufficiently long for the DRL attacker to learn the activity pattern.

A. Actor-Critic Model

In Fig. 5, we show the diagram of the actor-critic structure and the DRL attacker-environment interactions. The actor-critic structure consists of two neural networks, namely the actor network and critic network. The channels and victim's channel selection model form the environment to be observed by the attacker. Each time, after the DRL attacker observes the environment, an action will be selected based on this observation by the actor neural network. Then, the reward and

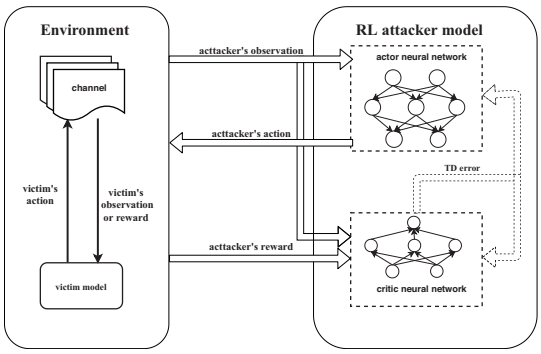

Fig. 5. Diagram of actor-critic structure and DRL attacker-environmental interactions.

the new state of the environment after executing the chosen action will be sent to the critic neural network to calculate the temporal difference (TD) error. This TD error will be used to update both critic and actor neural networks. When the update of network is completed, the DRL attacker model is ready to make the next decision.

At the beginning of each time slot t, the DRL attacker can select one channel based on its own action policy learned by the actor-critic neural networks. The action of the DRL attacker and the victim at time t are denoted as a_t^A and a_t^V respectively. Since both the DRL attacker and the victim select one out of the N channels, the sizes of their action spaces are the same. We assume that there are proper mechanisms and measurements (such as SINR levels, ACK signals) through which the attacker learns if the victim has selected the same channel as the attacker itself, i.e., $a_t^A = a_t^V$, and if the victim has transmitted successfully. The goal of the attacking DRL agent is to learn the victim's activity pattern so that it can jam the channels selected by the victim as much as possible. Based on this objective, we define the reward of the DRL attacker at time t as

$$r_t = \begin{cases} +1 & \text{if } a_t^A = a_t^V \text{ and victim selects a good channel,} \\ +0.5 & \text{if } a_t^A = a_t^V \text{ and victim selects a bad channel,} \\ -0.5 & \text{if } a_t^A \neq a_t^V \text{ and victim selects a bad channel,} \\ -1 & \text{if } a_t^A \neq a_t^V \text{ and victim selects a good channel.} \end{cases} \tag{1}$$

Within this setting, the DRL agent is encouraged to select the same good channels as the victim as its first priority. We also consider the case in which the attacker and victim select the same bad channel as partial success in terms of jamming.

As mentioned before, the DRL agent has no knowledge about the channels and the victim user. Hence, from the perspective of the DRL agent, the channels and victim form an unknown environment. We assume that in each time slot t, the observation of the DRL attacker is denoted as $\mathbf{S}_t = \{s_{t,1}, s_{t,2}, \ldots, s_{t,N}\}$. Then each element $s_{t,i}$, for $i = 1, 2, \ldots, N$, stands for the observation on the i^{th} channel at time t. As assumed before, the DRL attacker can only choose one channel at a time, so we have

$$s_{i,t} = \begin{cases} r_t & \text{if the } i^{\text{th}} \text{ channel is selected in time slot } t, \\ 0 & \text{if the } i^{\text{th}} \text{ channel is not selected in time slot } t. \end{cases} \tag{2}$$

Above, 0 indicates that the corresponding channels are not selected and therefore there is no information on these channels.

B. Operational Modes

Once the DRL agent is initialized, it switches between two different modes: listening mode and attacking mode.

- *Listening mode:* In this mode, the DRL agent only observes the environment and updates its own policy based on the reward, but does not jam the selected channels so that the victim is not influenced and updates to a new policy.
- *Attacking phase:* In this mode, the DRL agent jams the selected channels and decides whether to update its neural networks based on the victim's performance. When the victim performs well, the DRL agent should evolve its policy as the victim gradually adapts to the attacker's influence. However, when the victim performs poorly, the DRL agent should stop learning from the reward. Because in this situation the victim frequently chooses the bad channels, and the reward may misguide the attacker.

We assume that the victim's model is pre-trained so that the victim's activity pattern is stable when the attacker starts to train its own neural networks. In this training phase, the DRL agent works in the listening mode. And when the DRL agent is well trained, it can start the dynamic attack which we describe in detail in the following subsection.

C. Dynamic Attack

Similar to the FNN attacker, the DRL attacker also uses the RRAS procedure shown in Fig. 4. We note that the DRL agent requires less prior information about the victim' activity pattern than the FNN attacker. However, due to the differences in the learning method, the DRL attacker needs to observe the victim over a longer period to train a reliable policy. DRL attacker also aims at avoiding the situation in which the victim learns a totally new action policy once the model is well trained. For this purpose, the duration of each cycle of the DRL attacker is fixed at a certain value that prevents the victim to update to a new policy.

As shown in Fig. 4, the DRL attacker also starts its first attack at time ① when the victim model has been working in a stable fashion and working well. Before this point, the DRL attacker works in listening phase to learn the victim's activity pattern, and we assume that at time ①, the DRL attacker can also function well with high stability. Once the attack is initiated, the performance of the victim drops rapidly. In this process, the victim keeps updating its model to overcome the influence of the attacks, and at the same time, the attacker also keeps updating its model to adapt to the victim's changing policy. However, we should note that the attacker is always encouraged to choose the same channel as the victim does. Hence, when the victim is forced to explore other channels which are not attacked in order to find a new policy to counteract against the attacks, it cannot avoid but try bad channels in order to find the good ones. From the perspective of DRL attacker, there is no need to follow the victim's selection because the victim's model updates dramatically and the policy may perform worse initially. On the one hand, it is difficult for the attacker to learn an unstable policy. On the other hand, copying the bad policy may give victim the chance to recover its performance. Based on this idea, the DRL attacker stops updating when the performance of the victim is lower than a threshold and we mark this time instant as time ⑤. Though the DRL attacker model stops learning, it still works in attacking mode, so the performance of the victim continues to decrease. As mentioned before, the DRL attacker should stop jamming the channels before the victim adapts to its attacks, because the victim is also a reinforcement learning agent that has the ability to act against attacks naturally. At time ②, the victim's performance starts to recover, meaning that a new policy is being formed in the victim model. To avoid pushing the victim to the new policy further, the DRL attacker needs to switch to the listening mode at time ② to encourage the victim to return to its old policy as quickly as possible. And at time ③, the victim is able to perform as well

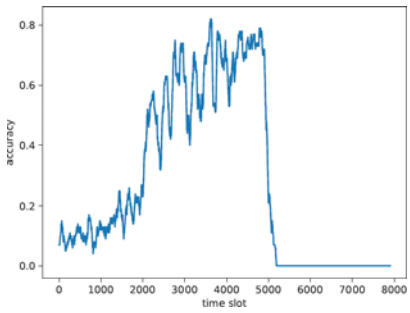

Fig. 6. Victim's accuracy under FNN attacker's RRAS procedure. The victim works without a ϵ-greedy policy.

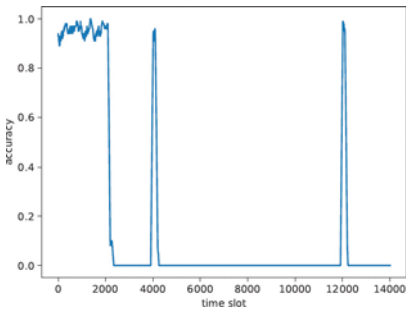

Fig. 7. Victim's accuracy under DRL attacker's RRAS procedure. The victim works without a ϵ-greedy policy.

as that before the attack, and the DRL attacker will retrieve the initial model and keep working in listening mode to adjust its policy based on the victim's activity until time ④ when the DRL attacker switches to attacking mode and starts a new cycle. In our implementation, the duration of each cycle is fixed to 2000 time slots, and the gap between time ③ and ④ is fixed at 200 time slots. Also, in the experiments, the duration between time ② and ③ is very small.

V. Experiments

In this section, we test the proposed FNN attacker and DRL attacker with a well-trained victim model and channel pattern introduced in Section II. In the following experiment, the FNN attacker starts attacking at time slot $t = 0$, and the DRL attacker starts attacking at time slot $t = 2000$.

First, we test both the FNN attacker and DRL attacker under the condition that the victim model works with $\epsilon = 0$ to show its full power. In Figs. 6 and 7, we plot the victim's accuracy over time to show the attackers' performance. For the FNN attacker, the victim's model crashes after about 5000 time slots and never recovers which means the victim's DRL agent has failed to adapt to the FNN attacker when it is not trying any random channels (due to the fact that $\epsilon = 0$). And for the DRL attacker, the victim's policy crashes immediately after the DRL attacker starts jamming the channels at time slot $t = 2000$. However, the victim's policy can recover for a short period of time after a few thousands of time slots. We should note that as a reinforcement learning-based agent, the DRL attacker always works with an ϵ-greedy policy with $\epsilon = 0.1$. The randomness in the DRL attacker's policy leads to a small chance for the victim to recover its performance from time to time. Overall, it is not challenging for the proposed to attackers to jam the channels selected by the victim most of the time, and considering this, we test the victim model with $\epsilon = 0.1$ in the following experiments to show the performance of the proposed attackers facing with a stronger victim user.

In Figs. 8 and 9, we plot the accuracy of the victim under FNN attacker's RRAS procedure and DRL attacker's RRAS procedure respectively. The FNN attacker retrieves and retrains when the victim's accuracy reaches 80% , and stops attacking when the victim's accuracy goes below 40%. Each retraining takes 100 samples, and runs over 850 iterations. Under the FNN attacker's RRAS procedure, the victim's accuracy drops rapidly after the attack begins. For the FNN attacker's initial attack cycle, the victim's performance recovers slowly. After that, the victim's performance can recover quickly when the attack is stopped and drops sharply once the attack resumes after retraining in each of the following RRAS cycles. This means that the FNN can make the victim perform poorly as much as possible in each RRAS cycle. The DRL attacker stops updating the policy when the victim's accuracy is lower than

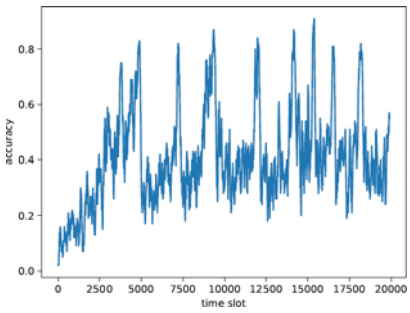

Fig. 8. Victim's accuracy under FNN attacker's RRAS procedure. The victim works with $\epsilon = 0.1$.

30% and switches to the listening mode when the victim's accuracy recovers to higher than 30% or if the duration of the current cycle is longer that 2000 time slots. In the listening mode, the DRL attacker reloads its initial policy and retrains for 200 time slots before the next attacking mode begins. In Fig. 9, the DRL attacker is able to have the victim's performance drop substantially and the recovery occurs over a short period of time but the performance drops again significantly, which means that the victim operates with very low accuracy most of the time. We note that under the DRL attacker's RRAS procedure, the victim's accuracy is more effectively constrained at a lower level.

To further compare the FNN and DRL attackers, we plot the corresponding probability density function (PDF) and cumulative distribution function (CDF) of the moving average of victim's accuracy in Figs. 10-13 based on the accuracy

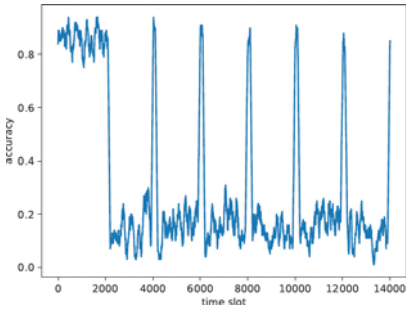

Fig. 9. Victim's accuracy under DRL attacker's RRAS procedure. The victim works with $\epsilon = 0.1$.

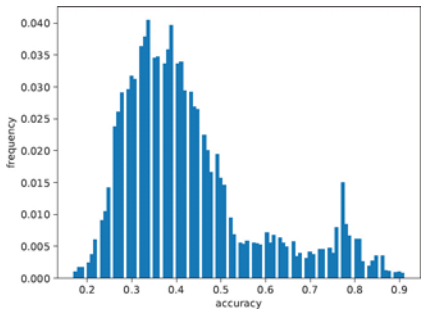

Fig. 10. PDF of victim's accuracy under FNN attacker's RRAS procedure.

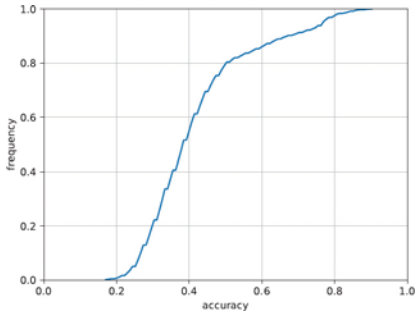

Fig. 11. CDF of victim's accuracy under FNN attacker's RRAS procedure.

curves shown in Figs. 8 and 9. Note that, the PDF and CDF under DRL agent's attacks starts collecting the accuracy data starting from the initial attacking phase at time $t = 2000$.

In Fig. 10, we observe that with the FNN attacker's jamming attacks, the victim's accuracy is more concentrated in the range of (0.3, 0.4). Correspondingly, in Fig. 11, the CDF increases at the fastest rate and approximates to 80% when the accuracy is 50%. For the DRL attacker, the victim's accuracy is highly concentrated at the level of 0.1 as shown in Fig. 12, and the corresponding CDF in Fig. 13 exceeds 80% when the accuracy is 20%. Since both proposed attackers stop attacking under specific conditions, the victim is able to recover its accuracy periodically. Hence, we can observe the increased distribution of the victim's accuracy at about 80% under both types of attacks.

As analyzed above, the DRL attacker can perform more effectively in the experimental environment. Additionally, the DRL attacker does not require any other auxiliary neural network as the FNN attacker does. However, if we consider the difference in the information regarding the victim-environment

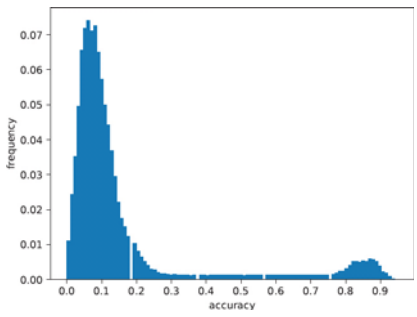

Fig. 12. PDF of victim's accuracy under DRL attacker's RRAS procedure.

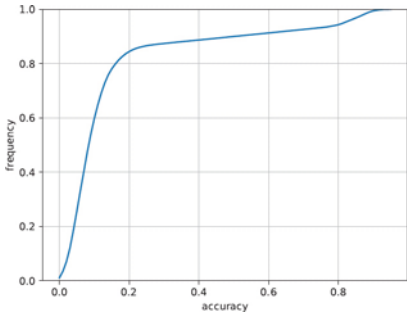

Fig. 13. CDF of victim's accuracy under DRL attacker's RRAS procedure.

interactions required by these two attacker, we note the advantage of the FNN attackers. The FNN attacker only needs to obtain the victim's activity records for a short period of time (50 or 100 time slots) and repeat the learning of the records over thousands of iterations to train and retrain its policy. However, the DRL attacker has to observe the victim-environment interactions for about 10000 time slots to train a stable policy after initialization. Therefore, if the channels patterns vary suddenly, the FNN is more promising in terms of adapting to a new policy quicker than the DRL attacker.

VI. Conclusion

In this paper, we have proposed two adversarial wireless jamming attackers aimed at minimizing the accuracy of the dynamic multichannel access performed by a DRL agent. We have introduced the frameworks of the proposed FNN and DRL attackers, and then presented their corresponding RRAS working procedures. Via simulations and numerical results, we have evaluated the performances of the two adversarial policies in terms of the victim's accuracy. In this analysis, we have specifically conducted experiments with a stronger victim that applies the ϵ-greedy policy. Finally, we have identified the advantages and disadvantages of the two frameworks.

References

[1] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4074–4077, April 2019.

[2] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 257–265, June 2018.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller, "Adversarial attacks on deep neural networks for time series classification," *arXiv preprint arXiv:1903.07054*, 2019.

[5] V. Subramanian, E. Benetos, N. Xu, S. McDonald, and M. Sandler, "Adversarial attacks in sound event classification," *arXiv preprint arXiv:1907.02477*, 2019.

[6] Y. Zhao, I. Shumailov, H. Cui, X. Gao, R. Mullins, and R. Anderson, "Blackbox attacks on reinforcement learning agents using approximated temporal information," *arXiv preprint arXiv:1909.02918*, 2019.

[7] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

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

[9] Y. Shi, T. Erpek, Y. E. Sagduyu, and J. H. Li, "Spectrum data poisoning with adversarial deep learning," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 407–412.

[10] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 1, pp. 2–14, 2018.

[11] C. Zhong, Z. Lu, M. C. Gursoy, and S. Velipasalar, "Actor-critic deep reinforcement learning for dynamic multichannel access," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 599–603.