# Adversarial Deep Learning for Cognitive Radio Security: Jamming Attack and Defense Strategies

Yi Shi*, Yalin E. Sagduyu*, Tugba Erpek*, Kemal Davaslioglu*, Zhuo Lu†, and Jason H. Li*
*Intelligent Automation, Inc., Rockville, MD 20855, USA, †University of South Florida, Tampa, FL 33620, USA
Email: {yshi, ysagduyu, terpek, kdavaslioglu}@i-a-i.com, zhuolu@usf.edu, jli@i-a-i.com

***Abstract*—This paper presents an adversarial machine learning approach to launch jamming attacks on wireless communications and introduces a defense strategy. In a cognitive radio network, a transmitter senses channels, identifies spectrum opportunities, and transmits data to its receiver in idle channels. On the other hand, an attacker may also sense channels, identify busy channels and aim to jam transmissions of legitimate users. In a dynamic system with complex channel, traffic and interference characteristics, the transmitter applies some pre-trained machine learning algorithm to classify a channel as idle or busy. This classifier is unknown to the attacker that senses a channel, captures the transmitter's decisions by tracking the acknowledgments and applies deep learning (in form of an exploratory attack, i.e., inference attack) to build a classifier that is functionally equivalent to the one at the transmitter. This approach is shown to support the attacker to reliably predict successful transmissions based on the sensing results and effectively jam these transmissions. Then, a defense scheme is developed against adversarial deep learning by exploiting the sensitivity of deep learning to training errors. The transmitter deliberately takes a small number of wrong actions (in form of a causative attack, i.e., poisoning attack, launched against the attacker) when it accesses the spectrum. The objective is to prevent the attacker from building a reliable classifier. For that purpose, the attacker systematically selects when to take wrong actions to balance the conflicting effects of deceiving the attacker and making correct transmission decisions. This defense scheme successfully fools the attacker into making prediction errors and allows the transmitter to sustain its performance against intelligent jamming attacks.**

***Index Terms*—Adversarial machine learning; deep learning; cognitive radio; exploratory attack; jamming attack; defense.**

## I. Introduction

*Cognitive radios* perform various detection, classification and prediction tasks such as spectrum sensing and automatic modulation recognition. These tasks can be accomplished by *machine learning* that allows cognitive radios to perceive and learn the spectrum environment and adapt to spectrum dynamics [1]–[4]. Examples of applying machine learning to cognitive radio tasks include modulation classification (e.g., with convolutional neural network (CNN) [5]) and spectrum sensing (e.g., with CNN [6] or generative adversarial network (GAN) [7]).

The success of machine learning in information systems raises security concerns, as machine learning itself may become subject to various exploits and attacks that expose the underlying tasks to threats. *Adversarial machine learning* studies learning in the presence of an adversary and aims to enable safe adoption of machine learning to the emerging applications. There are three broad categories of adversarial machine learning:

- *exploratory attacks or inference attacks* [8]–[11] that aim to understand how the underlying machine learning works for an application (e.g., inferring sensitive and/or proprietary information);
- *evasion attacks* [12], [13], where the adversary attempts to fool the machine learning algorithm into making a wrong decision (e.g., fooling a security algorithm into accepting an adversary as legitimate); and
- *causative attacks or poisoning attacks* [14], [15], where the adversary provides incorrect information (e.g., training data in supervised learning) to a machine learning based application.

These attacks can be launched separately or combined, e.g., causative and evasion attacks can be launched building upon the inference results of an exploratory attack [16].

Due to the open broadcast nature, wireless medium is susceptible to adversaries such as jammers. Therefore, it is critical to understand the security implications of machine learning in wireless communications. However, the vulnerabilities of cognitive radio systems using machine learning are not well understood yet. In this paper, we apply *adversarial deep learning* to launch an *exploratory attack* on the cognitive radio as a preliminary step before jamming. We consider a canonical wireless communication scenario with a transmitter, a receiver, an attacker, and some other background traffic. The transmitter senses the channel and transmits data to a receiver if the channel is found idle. While traditional algorithms for predicting idle channels may be as simple as comparing sensing results with some threshold (i.e., energy detector), more advanced techniques may be needed in a dynamic wireless environment with complex channel and transmitter characteristics. To identify idle channels, the transmitter applies some pre-trained machine learning classifier, which takes a number of features (recent sensing results) as input and classifies the channel as "idle" or "busy".

In the general setting of an *exploratory attack*, the attacker aims to build a classifier that is functionally equivalent to the target classifier under the attack, i.e., provides the same output as the target classifier for the same given input. The attack

This effort is supported by the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

considered in this paper shares this main idea but the input and the output in classifiers are different for the target (namely, the transmitter) and the attacker.

1) The input (sensing results) is different since the attacker and the transmitter have different sensing results at the same instance (i.e., on the same channel for the same time), since their locations and channel gains that they perceive are different.
2) An attacker does not need to distinguish idle channels or transmissions that are likely to fail. Instead, *the attacker should predict whether there will be a successful transmission* (i.e., whether the transmitter will decide to transmit and the signal-to-interference-plus-noise ratio (SINR) will exceed a threshold) so that it jams a transmission that would succeed without jamming.

The classifier built at the attacker is functionally equivalent to the one at the transmitter only in the sense that the attacker's classifier will decide to jam if and only if it predicts that there will be a successful transmission (in the absence of jamming) for the same instance. If the receiver successfully receives a packet, it sends an ACK as feedback to the transmitter, otherwise there is no feedback. During the learning period, the attacker senses the channel to distinguish whether there is an ACK or not, i.e., ACK signal plus noise vs. noise. The attacker needs to jam only if there is an ACK. Thus, the attacker builds a *deep learning* classifier (i.e., trains a deep neural network) with two labels ("ACK" or "no feedback") by using the most recent sensing results (received signal strengths) as the features. The attacker has two objectives: minimize the misdetection probability (for effective jamming) and minimize the false alarm probability (to save energy or avoid being caught). Thus, the attacker only jams if it predicts there will be an ACK and aims to minimize the maximum of misdetection and false alarm probabilities of its prediction.

We show that this adversarial deep learning approach is very effective, i.e., for the scenario studied in numerical results, it reduces the transmitter's throughput from $0.304$ packet/slot to $0.012$ packet/slot. However, random jamming is not as effective since the transmitter can still sustain throughput of $0.212$ packet/slot. We observe the same trend for success ratio.

Next, we design a *defense* scheme for the transmitter. The basic idea is to make the transmitter's behavior unpredictable, which can be done by the transmitter taking some deliberately wrong actions (i.e., transmitting on a busy channel or not transmitting on an idle channel) in some selected time slots. This corresponds to a *causative* attack launched by the transmitter back at the attacker. A very small number of wrong decisions cannot fool the attacker but a high number would prevent the transmitter from sensing the spectrum reliably and reduce the performance significantly even in the absence of the jammer. To maximize the impact of a small number of wrong actions, the transmitter uses the classification scores (an intermediate result) that are determined by the machine learning algorithm for spectrum sensing. Such a score is within $[0, 1]$ and compared with a threshold to classify channels. If this score is far away from the threshold (i.e., close to $0$ or $1$), the confidence of classification is high and the corresponding time instance should be selected to take the wrong action, since it can more successfully deceive the attacker that aims to mimic the transmitter's behavior. There is a balance on how many wrong actions to take. We show that by taking a small number of wrong actions on carefully selected time instances, the transmitter can fool the attacker into making a significant number of prediction errors. Thus the transmitter's performance can be improved significantly from $0.012$ packet/slot to $0.206$ packet/slot.

The rest of the paper is organized as follows. Section II describes the system model. Section III describes the transmitter's algorithm and shows the performance when there is no jamming. Section IV describes the attacker's algorithm and shows the performance under deep learning and random attacks. Section V presents a defense mechanism and shows how the performance improves. Section VI discusses the extension of network setting. Section VII concludes the paper.

## II. System Model

We consider a wireless communication scenario with one transmitter $T$, one receiver $R$, and one attacker $A$. This setting is instrumental in studying the fundamentals of jamming and defense strategies in wireless access [17]. The implications of extending the network setting are discussed in Section VI. The developed algorithms can be easily extended to multiple transmitters and receivers, while a single attacker alone can jam nodes within its transmission range. A general operation model for transmitter, receiver, and attacker is as follows.

- *Transmitter operation*: There may be transmissions from some unobserved transmitters (i.e., background traffic) and thus the channel status may be busy even when $T$ and $A$ do not transmit. The time is divided in slots. In each slot, the short initial period of time is allocated for $T$ to sense the channel, run its spectrum sensing algorithm and detect the channel (idle/busy) status. If the channel is detected as idle, $T$ can transmit data to $R$.
- *Receiver operation*: The transmission is successful if the SINR at $R$ is larger than some threshold $\beta$. The short ending period of a time slot is allocated for $R$ to send feedback (ACK) to $T$.
- *Attacker operation:* The attacker $A$ also senses the spectrum and predicts whether there will be a successful transmission (with feedback ACK), or not (without a feedback) in a time slot. If $A$ predicts that there will be a successful transmission, it jams this transmission in this time slot.

The general operation mode does not specify a particular algorithm to make transmission or jamming decisions. We consider the case that both $T$ and $A$ apply machine learning algorithms (unknown to each other) to make their decisions. Denote sensing results (noise power or noise plus interference power) at time $t$ as $s_T(t)$ and $s_A(t)$ for $T$ and $A$, respectively. Note that due to different locations of $T$ and $A$, their sensing results may be different, i.e., $s_T(t) \neq s_A(t)$ in general.

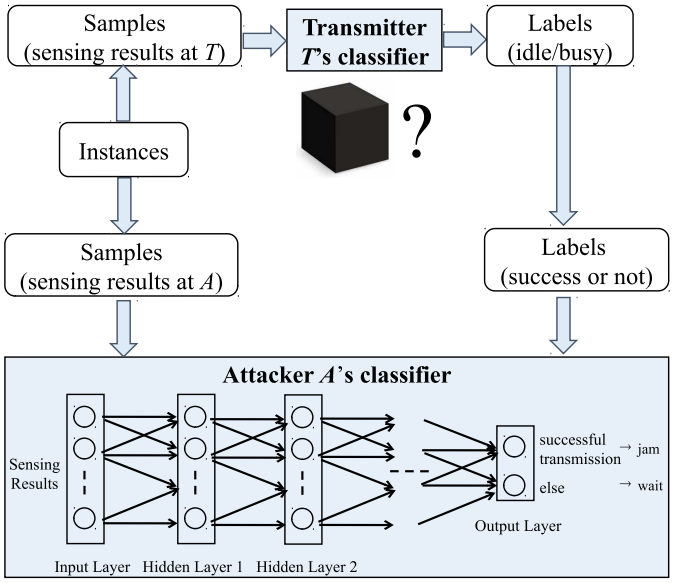


Fig. 1. The system model for attacker's learning.

- Transmitter $T$ has a classifier $C_T$ that is pre-trained by some machine learning algorithm, which identifies the current time slot $t$ as idle or busy based on recent $K$ sensing results $(s_T(t-K+1), \cdots, s_T(t-1), s_T(t))$.
- Attacker $A$ does not know classifier $C_T$ and needs to build a classifier $C_A$ by training a deep learning classifier, which predicts whether there will be a successful transmission, or not, in time slot $t$ based on recent $L$ sensing results $(s_A(t-L+1), \cdots, s_A(t-1), s_A(t))$.

The system model is shown in Fig. 1. Transmitter $T$ transmits with power $P_T$ if the sensed channel is determined as idle. The SINR at $R$ is $\frac{g_{TR}P_T}{N_0+I_R}$, if channel is busy, or $\frac{g_{TR}P_T}{N_0}$, if channel is idle, where $I_R$ is the interference from some unobserved transmitters to $R$ and $N_0$ is a Gaussian noise with its mean value normalized as unit power. A transmission is successful if this SINR is greater than some threshold $\beta$. Without loss of generality, we assume that only one packet is transmitted in a time slot. We measure $T$'s performance by throughput and success ratio.

- *Throughput*: the number of received packets at $R$ during a period divided by the number of time slots in this period.
- *Success ratio*: the percentage of successful transmissions by $T$ over all transmissions.

Attacker $A$ jams with power $P_A$ if it predicts that there will be a successful transmission in the absence of jamming. If a transmission is jammed, the SINR at $R$ is reduced to $\frac{g_{TR}P_T}{N_0+I_R+g_{AR}P_A}$ (if channel is busy) or $\frac{g_{TR}P_T}{N_0+g_{AR}P_A}$ (if channel is idle). Receiver $R$ still sends ACK to confirm that a transmission is successful under jamming by comparing its SINR with threshold $\beta$. To evaluate the accuracy of $A$'s classifier, we define two types of errors:

- *Misdetection*: $T$'s transmission is successful but $A$ does not decide to jam.
- *False alarm*: $T$ does not transmit or $T$'s transmission fails (even without jamming) but $A$ decides to jam.

TABLE I
SUMMARY OF NOTATION USED IN THE PAPER.

| Symbol | Description |
|---|---|
| $T$ | transmitter |
| $R$ | receiver |
| $A$ | attacker |
| $s_i(t)$ | sensing result by node $i$ at time $t$ |
| $C_T$ | transmitter's algorithm to detect channel status |
| $C_A$ | attacker's algorithm to predict transmission feedback |
| $g_{ij}$ | channel gain from node $i$ to node $j$ |
| $d_{ij}$ | distance from node $i$ to node $j$ |
| $N_0$ | noise |
| $I_i$ | interference at node $i$ |
| $P_A$ | transmitter's transmit power |
| $P_T$ | attacker's transmit power |
| $\beta$ | SINR threshold |

We set up the channel busy/idle status to define the background traffic as follows. There is an unobserved transmitter whose transmission behavior is not known by either $T$ and $A$. In particular, we assume random packet arrivals at the unobserved transmitter. If the unobserved transmitter is not transmitting, it becomes active with certain probability when its queue is not empty. Once activated, it will keep transmitting until its queue is empty. Such a transmission behavior is random and is also time correlated. Therefore, both $T$ and $A$ need to observe the recent channel status (over several time slots) to predict the current channel status. Table I summarizes the notation used in this paper.

## III. TRANSMITTER ALGORITHM

Transmitter $T$ applies a deep learning algorithm to determine the channel status. Note that $T$ could also use a simpler machine learning algorithm at the expense of potential performance loss, while attacker $A$'s algorithm is oblivious to $T$'s algorithm. $T$ senses the channel and records the most recent results (received signal strengths). Each result is either a Gaussian noise $N_0$ with normalized unit power (when the channel is idle) or noise plus the transmit power from the unobserved transmitter received at $T$, i.e., $N_0 + I_T$ (when the channel is busy), where $I_T$ is the interference received at $T$. $T$ uses the most recent 10 sensing results as features (i.e., $K = 10$) and uses the current channel busy/idle status as a label to build one sample. After observing a certain period of time, $T$ collects a number of samples as training data to build a deep learning classifier, where two labels are "idle" and "busy" (namely, the channel is idle or busy).

We implemented a *feedforward neural network* as the deep learning algorithm for $T$ by using Microsoft CNTK [18]. 1000 samples are collected by $T$ and split by half to build its training and test data. We optimize hyperparameters of the deep neural network to minimize the maximum of misdetection and false alarm errors. The optimized hyperparameters are given as follows. The deep learning network consists of two hidden

layers, each with 50 neurons. Backpropagation is used to train the neural network using the cross entropy loss function. The output layer uses softmax activation. Hidden layers are activated using the sigmoid function and all weights and biases are initialized to random values in $[-1.0, 1.0]$. In the first training pass, input values are unit normalized. The minibatch size is taken as 25, the dropout rate is taken as 0.9, and 10 epochs per time slot are considered.

After training the deep neural network with these hyperparameters, we run $T$'s classifier over 500 time slots to evaluate its performance. We assume that the channel gain $g_{ij}$ from $i$ to $j$ has the Gaussian distribution with mean value $d_{ij}^{-2}$, where $d_{ij}$ is the distance from $i$ to $j$ (note $d_{ij}$ is normalized by the free space reference distance $d_0$). More complicated channel models or real measurements from radios can be applied here to determine $g_{ij}$. We set $P = 1000N_0$, $\beta = 3$, $d_{TR} = 10$, $d_{AR} = 10$, and $d_{TA} = 10\sqrt{2}$ for numerical results. Then, we find that $T$ makes 206 transmissions and 152 transmissions of them are successful. Thus, the throughput is $152/500 = 0.304$ packet/slot and the success ratio is $152/206 = 73.79\%$.

## IV. Attacker Algorithm

In [11], we designed the mechanism to steal a machine learning (including deep learning) classifier via the *exploratory attack* applied to text classification. The basic idea was to poll the target classifier for labels of a number of samples and then train a functionally equivalent classifier using deep learning. Two classifiers are functionally equivalent if they provide the same labels for the same sample. However, this approach cannot be applied to the setting in this paper. Due to different locations of $T$ and $A$, random channel gain and random noise, the sensing results at $T$ and $A$ will be different. That is, when the channel is idle, both $T$ and $A$ sense a Gaussian noise $N_0$ but the value can be different due to different realizations. When the channel is busy, $T$ will sense $N_0 + I_T$ and $A$ will sense $N_0 + I_A$. Thus, in addition to different realization of $N_0$, the values of $I_T$ and $I_A$ are different due to different channel gains to $T$ and $A$, as well as their different realizations. Thus, even if $A$ has a functionally equivalent classifier (e.g., $T$'s algorithm), $A$ cannot use it to obtain the same channel status as the one predicted by $T$ due to different sensing results (or features computed for deep learning). Moreover, $A$ does not aim to predict whether the channel is idle or busy. Instead, its goal is to predict whether there will be a successful transmission of $T$, or not. There are four cases for the channel status and $T$'s behavior:

1) channel is idle and $T$ is transmitting,
2) channel is busy and $T$ is not transmitting,
3) channel is idle and $T$ is not transmitting, and
4) channel is busy and $T$ is transmitting.

Ideally, the last two cases should be rare cases, since they refer to wrong sensing decisions by $T$. We assume that $A$ can hear ACKs for $T$'s successful transmissions. Then, $A$ can use the most recent 10 sensing results as features (i.e., $L = 10$) and the current feedback (ACK or no confirmation) as a label to build one sample. $A$ aims to jam successful transmissions (with received ACK feedback) only. Thus, $A$ defines two labels as "a successful transmission" (ACK) and "no successful transmission" (no confirmation), i.e., the labels at $A$ are also different from $T$. In summary, for $T$'s classifier,

- the features for deep learning are $T$'s sensing results and
- the predicted labels are "idle" and "busy",

while for $A$'s classifier,

- the features are $A$'s sensing results and
- the predicted labels are "a successful transmission by $T$" and "no successful transmission by $T$".

After observing a certain period of time, $A$ collects a number of samples to be used as training data and trains a deep learning classifier. Once a classifier is built, $A$ uses it to predict whether there will be a successful transmission and if yes, $A$ transmits to jam the channel.

We use 1000 samples collected by $A$ and split them by half as training and test data to build a deep learning algorithm based on FNN for $A$. We optimize hyperparameters to minimize the maximum of misdetection and false alarm probabilities. We obtain a deep learning network with two hidden layers, each with 60 neurons. Backpropagation is used to train the neural network using cross entropy loss function. The output layer uses softmax activation. Hidden layers are activated using the sigmoid function and all weights and biases are initialized to random values in $[-4.0, 4.0]$. In the first training pass, input values are unit normalized. The minibatch size is taken as 25, the dropout rate is taken as 0.9, and 9 epochs per time slot are considered.

After training the deep neural network with these hyperparameters, we run classifiers of $A$ and $T$ over 500 time slots to evaluate the attack performance. In these time slots, if there is no attack, $T$ will have 152 successful transmissions. Under $A$'s attack, the number of misdetections is 6, i.e., misdetection probability is $6/152 = 3.95\%$ (almost all successful transmissions are jammed), and the number of false alarms is 63, i.e., false alarm probability is $63/(500 - 152) = 18.10\%$. The impact of this attack is significant. The throughput of $T$ is reduced from 0.304 packet/slot to $6/500 = 0.012$ packet/slot and the success ratio of $T$ is reduced from $73.79\%$ to $6/206 = 2.91\%$.

For comparison purposes, we consider an alternative attack scheme for $A$. One option for $A$ is to apply a sensing-based scheme, i.e., it jams if the received signal strength is greater than some threshold. However, this scheme does not work because $A$ cannot have the same sensing results as receiver $R$. Moreover, it is not possible for $A$ to learn the channel gain between $T$ and $R$ and thus a suitable threshold for sensing cannot be determined. Therefore, we consider a *random jamming* attack in which $A$ jams the channel in some randomly selected instances. Such an attack scheme does not require $A$ to learn the outcome of $T$'s transmissions. The misdetection probability is $69.60\%$ and the false alarm probability is $30.40\%$ for $A$. Since these error probabilities are not small, the impact of this attack is not significant. The throughput can be only reduced from 0.304 packet/slot

TABLE II
EFFECTS OF DIFFERENT ATTACK TYPES ON THE TRANSMITTER'S PERFORMANCE.

| Attack type | Throughput | Success ratio |
|---|---|---|
| No attack | 0.304 | 73.79% |
| Adversarial deep learning | 0.012 | 2.91% |
| Random attack | 0.212 | 51.36% |

to $0.212$ packet/slot and the success ratio can be only reduced from $73.79\%$ to $51.36\%$. Thus, to perform effective attacks, it is necessary to build a deep learning classifier and jam in the carefully selected time slots instead of launching random attacks. The results are summarized in Table II.

## V. DEFENSE STRATEGY

We present a defense strategy where the transmitter changes the labels for some samples such that the attacker cannot build a reliable classifier in an exploratory attack. This corresponds to a *causative* attack of $T$ back at $A$ as a defense mechanism, since $T$ poisons the training process of $A$ by providing wrong training data. Against the jamming attack, $T$ needs to change the labels for "a successful transmission" and "no successful transmission". This can be done by flipping labels, i.e., by

- not transmitting even if channel is detected as idle, and
- transmitting even if channel is detected as busy.

It is clear that $T$ wants to limit the extent of defense operations such that the overhead for defense (i.e., the increased classification error) can be minimized. Otherwise, $T$ would start making a large number of transmission errors and could not sustain a good throughput even without jammer in presence. For this purpose, $T$ needs to carefully select on which time slots to perform defense operations by examining the output of its deep learning algorithm. In fact, a deep learning based classifier provides not only labels, but also a score for classification. In particular, there is a classification score in [0,1], namely the likelihood of whether a channel is idle. If this score is less than a threshold, a time slot is classified as idle, otherwise it is classified as busy. Moreover, if this score is far away from the threshold, then such a classification has a high confidence, otherwise the confidence is low. Therefore, to maximize the impact on $A$, $T$ should perform defense operations in time slots when the scores close to $0$ or $1$ are obtained, since they correspond to time slots when $T$'s transmission decisions are more predictable. As a consequence of this defense, $A$ builds different classifiers with different hyperparameters (see Table III) compared to the previous case of no defense in Section IV. Note that when the number of layers is one, the deep learning network reduces to a standard neural network.

Table IV shows the results when $T$ performs different number of defense operations. We can see that even when $T$ makes deliberately wrong decisions only over $10\%$ of all time slots, $A$'s error probabilities increase significantly, i.e., misdirection probability increases from $3.95\%$ to $20.79\%$ and false alarm probability increases from $18.10\%$ to $25.16\%$. We also calculate the performance of $T$ when $A$ performs a jamming attack in any time slot when $T$ can have a successful transmission if not jammed. With more defense operations (i.e., more labels flipped), $T$ can increase its throughput and success ratio. However, if $T$ takes too many (e.g., $50\%$) defense operations, its performance starts dropping as its spectrum sensing decisions become more unreliable and its transmission becomes less likely to succeed even in the absence of jamming.

TABLE III
OPTIMIZED HYPERPARAMETER VALUES OF THE ATTACKER UNDER DIFFERENT LEVELS OF DEFENSE STRATEGY.

| Ratio of samples with defense operations | # hidden layers | # neurons per layer | activation function |
|---|---|---|---|
| 0% (no defense) | 2 | 60 | sigmoid |
| 10% | 2 | 90 | ReLU |
| 20% | 2 | 100 | ReLU |
| 30% | 1 | 70 | ReLU |
| 40% | 1 | 60 | ReLU |
| 50% | 2 | 40 | ReLU |

Table V shows the results under random jamming. We can see that jamming channels randomly is not effective (with larger error probabilities) compared to jamming based on adversarial deep learning. The defense actions do not have much impact on the success ratio but can increase throughput, since the original transmitter algorithm is not perfect (with misdetection of transmission opportunities). The defense actions, in fact, make the transmitter more aggressive to transmit and thus increase its throughput.

## VI. EXTENSION IN NETWORK SETTING

The network scenario has only one transmitter, one receiver, and one attacker. The developed solution can be extended for multiple transmitters and receivers, while interference from non-intended transmitters is sensed as additional interference term by receivers. A transmitter still aims to predict whether the signal strengths at its receiver (if it decides to transmit) will be good or not based on past signal strengths. That is, the only change in the transmitter algorithm is the training data, which includes interference from non-intended transmitters.

We would still consider one attacker. Since each attacker can jam a neighboring area, attackers can be deployed sparsely and each attacker can perform jamming independently. Note that an attacker only needs to predict whether there will be some successful transmissions, i.e., there is no need to figure out corresponding transmitters. Thus, an attacker does not need to build a classifier for each transmitter. Instead, an attacker aims to predict whether there will be successful transmissions. The only change in the attacker algorithm is the training data, which includes superimposed signals received from all transmitters and uses ACK from all receivers.

Moreover, both transmitter and attacker algorithms can be readily applied in mobile wireless networks. Although we focused on a static network instance, we do not customize our algorithm to explore the static topology. In the training data, features are derived from sensing results and labels are derived

TABLE IV
RESULTS FOR DEFENSE STRATEGY AGAINST JAMMING ATTACK BASED ON ADVERSARIAL DEEP LEARNING.

| # of defense operations /# of all samples | Attacker error probabilities | | Transmitter performance | |
|---|---|---|---|---|
| | Misdetection | False alarm | Throughput | Success ratio |
| 0% (no defense) | 3.95% | 18.10% | 0.012 | 2.91% |
| 10% | 20.79% | 25.16% | 0.074 | 17.13% |
| 20% | 33.88% | 40.69% | 0.124 | 28.84% |
| 30% | 40.09% | 44.79% | 0.170 | 35.42% |
| 40% | 45.18% | 43.75% | 0.206 | 41.53% |
| 50% | 41.63% | 45.10% | 0.204 | 39.23% |

TABLE V
RESULTS FOR DEFENSE STRATEGY AGAINST RANDOM JAMMING ATTACK.

| # of defense operations /# of all samples | Attacker error probabilities | | Transmitter performance | |
|---|---|---|---|---|
| | Misdetection | False alarm | Throughput | Success ratio |
| 0% (no defense) | 69.60% | 30.40% | 0.212 | 51.36% |
| 10% | 64.40% | 35.60% | 0.229 | 53.07% |
| 20% | 63.40% | 36.60% | 0.232 | 53.96% |
| 30% | 57.60% | 42.40% | 0.244 | 50.88% |
| 40% | 54.40% | 45.60% | 0.248 | 50.01% |
| 50% | 51.00% | 49.00% | 0.250 | 48.06% |

from ACKs, which do not depend on topology. As a result, the same algorithms can be applied if network is mobile.

## VII. CONCLUSION

We applied adversarial machine learning to design an intelligent jamming attack on cognitive radio transmissions and presented a defense strategy against this attack. We considered a wireless communication scenario with one transmitter, one receiver, one attacker, and some background traffic. We discussed the extension of our algorithms for multiple transmitters and receivers in a mobile network with complex channels. The transmitter senses the channel, applies a pre-trained machine learning algorithm to detect idle channel instances for transmission. The attacker does not have any knowledge of transmitter's algorithm. Instead, it senses the channel, detects the transmission feedback (if available), applies a deep learning algorithm to predict a successful transmission, and jams such a transmission. We showed that this attack is effective in reducing the transmitter's throughput and success ratio. Finally, we designed a defense mechanism for the transmitter that intentionally takes wrong actions in selected time slots to mislead the attacker. We showed that even a small percentage of wrong actions in systematically selected time slots can significantly increase the errors in attacker's decisions and prevent major losses in the performance of the transmitter.

## REFERENCES


[1] C. Clancy, H. J. Stuntebeck, and T. O'Shea, "Applications of machine learning to cognitive radio networks," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 47-52, 2007.

[2] K. Thilina, K. W. Choi, N. Saquib, and E. Hossain, "Machine Learning Techniques for Cooperative Spectrum Sensing in Cognitive Radio Networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 11, pp. 2209-2221, 2013.

[3] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine Learning for Wireless Networks with Artificial Intelligence: A Tutorial on Neural Networks," *arXiv preprint arXiv:1710.02913*, 2017.

[4] M. Alsheikh, S. Lin, D. Niyato, H. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, 16(4):1996-2018, Apr. 2014

[5] T. O'Shea, J. Corgan, and C. Clancy, "Convolutional radio modulation recognition networks," *International Conference on Engineering Applications of Neural Networks*, 2016.

[6] W. Lee, M. Kim, D. Cho, and R. Schober, "Deep Sensing: Cooperative Spectrum Sensing Based on Convolutional Neural Networks," *arXiv preprint arXiv:1705.08164*, 2017.

[7] K. Davaslioglu and Y. E. Sagduyu, "Generative Adversarial Learning for Spectrum Sensing," *IEEE International Conference on Communications (ICC)*, 2018.

[8] G. Ateniese, L. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers," *International Journal of Security and Networks*, 10(3):137-150, 2015.

[9] F. Tramer, F. Zhang, A. Juels, M. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," *USENIX Security*, 2016.

[10] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[11] Y. Shi, Y. E. Sagduyu, and A. Grushin, "How to Steal a Machine Learning Classifier with Deep Learning," *IEEE Symposium on Technologies for Homeland Security*, May 2017.

[12] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion Attacks Against Machine Learning at Test Time," *ECML PKDD*, 2013.

[13] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Examples in the Physical World," *arXiv preprint arXiv:1607.02533*, 2016.

[14] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," *IEEE European Symposium on Security and Privacy*, 2016.

[15] L. Pi, Z. Lu, Y. Sagduyu, and S. Chen, "Defending Active Learning against Adversarial Inputs in Automated Document Classification," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016.

[16] Y. Shi and Y. E Sagduyu, "Evasion and Causative Attacks with Adversarial Deep Learning," *IEEE Military Communications Conference*, 2017.

[17] Y. E. Sagduyu, R. Berry, and A. Ephremides, "Jamming Games in Wireless Networks with Incomplete Information," *IEEE Communications Magazine*, vol. 49, no. 8, Aug. 2011.

[18] Microsoft Cognitive Toolkit (CNTK), https://docs.microsoft.com/en-us/cognitive-toolkit