AutoPilot: Steering OFDM Pilots Away From Jamming Using Deep Learning

Daniel Uvaydov, *Member, IEEE*, Salvatore D'Oro, *Member, IEEE*, Tommaso Melodia, *Fellow, IEEE*

***Abstract*—Pilot jamming attacks are among the most cost-effective, yet successful, attacks against wireless networks designed around fixed pilot locations, e.g., IEEE 802.11a/g/p. In this paper, we aim at improving the security and robustness of those systems against this class of attacks. Specifically, we present AutoPilot, a data-driven and reconfigurable transceiver designed to combat narrowband jamming attacks, with specific focus on pilot jamming. AutoPilot embeds a customized RF front-end to effectively demodulate waveforms transmitted using non-standard compliant pilot configurations as well as two data-driven logic units that are designed to (i) adapt via pseudo-randomization pilot configurations according to the current channel conditions and attack profiles; and (ii) detect rapidly and with high accuracy pilots non conforming to standard-defined pilot sequences. We introduce and describe AutoPilot design and eventually present a prototype that is evaluated via over-the-air experiments on an actual wireless testbed. Our results show that AutoPilot can detect pilots with an accuracy as high as 99% on over-the-air unseen data and can improve network throughput under jamming attacks by 4x at high jammer powers. Finally, by randomizing pilots location, AutoPilot prevents eavesdroppers from demodulating eavesdropped packets, thus providing an increased layer of security to the network.**

***Index Terms*—Deep Learning, Jamming, Pilots, OFDM**

I. Introduction

Thanks to technological advancements in recent years, wireless devices have become more and more inexpensive and ubiquitous. For this reason, communication standards have proliferated over the last decades by specifying, among others, handshake procedures, retransmission policies, medium access control and packet formats designed to ensure reliable and efficient communications between multiple wireless nodes.

Communication standards offer a solid base for any communication systems as they specify operations and structures that each communication system (or device) must follow in order to be standard-compliant. On the one hand, these specifications aim at guaranteeing that each and every communication attempt is successful and no information is lost due to errors. On the other hand, however, they establish predictable system behavior and operations which can be leveraged by an adversaries.

A practical example that we can observe is that of *pilots*, which are used in the majority of wireless standards to evaluate the wireless channel and mitigate (or completely remove) its impact on ongoing wireless transmissions. In this context, *pilot jamming*—an attack where the jammer targets its destructive attack exclusively to the pilot symbols only—has been demonstrated to be among the most effective attacks toward wireless network, with an attack efficiency of 7.5dB greater than any other attacks by solely disrupting pilots in a OFDM network [1]. In Fig. 1, we provide an experimental evidence on a real over-the-air testbed of such effectiveness. This illustrative example shows how a pilot jammer can reduce the frame success rate (FSR) of a pair of WiFi nodes by 80% or more by generating four narrowband jamming signals that overlap with the four pilot symbols and thus occupy less then 8% of the bandwidth used for transmission by the WiFi pair.

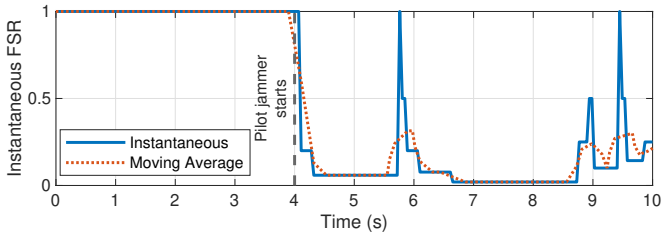

Fig. 1: An example of narrowband pilot jammer's effect on the frame success rate (FSR) of an over-the-air WiFi pair.

This shows that, although standards are necessary to functional and reliable wireless networks, they result in predictable behavior that exposes the network to a variety of effective attacks. This inevitably results in the need for innovative and smart transceiver designs that are capable of mitigating the above vulnerabilities especially against those attacks targeted at pilots, which are necessary tools to ensure reliable communications yet are extremely fragile and easily tampered.

A naive approach against pilot jamming would be to frequently change the transmission center frequencies via *frequency hopping*. While hopping can indeed be a viable solution to avoid such attacks, especially when the hopping sequence is pseudo-random and hard to predict, it might not be a viable solution for many wireless technologies that rely upon the availability of large spectrum chunks such as WiFi.

In this paper, we tackle this problem from a substantially unexplored angle and present an innovative and effective approach to (i) fight pilot jamming attacks while retaining pilots and their well-established effectiveness; and (ii) retaining as many standard procedures as possible so as to impact minimally the way standard-compliant transceivers operate. Specifically, the main contribution of this paper is twofold and can be summarized as follows. First, we present *AutoPilot* (Fig. 2), a data-driven wireless communication system that uses Deep Reinforcement Learning (DRL) and Deep Learning (DL) to escape pilot jamming attacks by randomizing the location of pilot symbols while using the same transmission channel. Apart from mitigating jamming attacks, randomizing pilots makes it

The authors are with the Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA. E-mail: {d.uvaydov, s.doro, melodia}@northeastern.edu.

This article is based upon work partially supported by the U.S. National Science Foundation under grants CNS-2112471, CNS-2312875 and CNS-1925601.

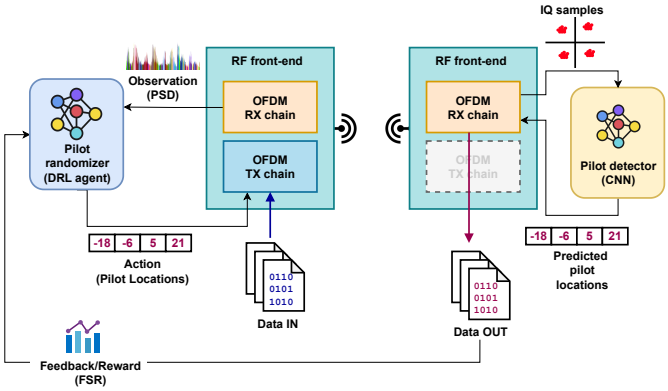

Fig. 2: High-level overview of *AutoPilot* transceiver design.

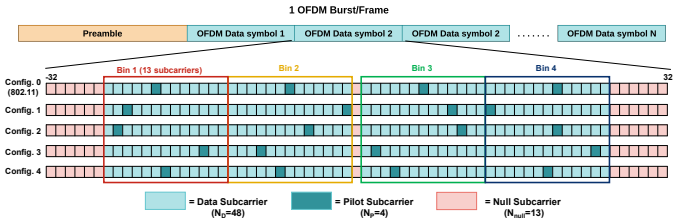

Fig. 3: (Top) OFDM burst frame structure; (Bottom) OFDM data symbol frame structure and varying pilot configurations that CNN based pilot detector is trained to identify.

difficult for eavesdroppers to equalize and demodulate eavesdropped communications, thus making our solution eavesdrop resistant. *AutoPilot* consists of a transmitter module that uses DRL to dynamically randomize pilot sequences to counteract ongoing attacks, and a receiving module that used DL to detect the location of the randomized pilot sequence *without* requiring any coordination with the transmitter node. Second, we follow an experimental approach where we train, prototype and evaluate *AutoPilot* over-the-air only, and without the need for offline training. Our results show that *AutoPilot* detects pilots with an accuracy as high as 99% on over-the-air unseen data, and improves network throughput under attacks by about 4×.

II. Related Work

Narrowband pilot jamming is known to be a power-effective approach to effectively disrupts communication by attacking pilots [1–4]. Randomizing pilot locations can mitigate such attacks [3]. However, using known pseudo-random sequences or look-up tables may not leverage information on channel conditions and attack profiles, leading to vulnerabilities. Another method assigns multiple pilot schemes to lawful users via randomization [5], but lacks of channel awareness. Frequency hopping has been considered as a solution to avoid jamming [6, 7]. Although effective, it requires complex transceiver designs and coordination between transmitters and receivers, which can be impractical and expensive. These strategies rely on handshakes and coordination, making them primarily proactive.

DL and DRL for wireless communications are rapidly growing fields and increasingly replacing traditional heuristic and deterministic approaches. DRL-enabled frequency hopping is proposed to mitigate jamming attacks through device coordination [8, 9]. Similarly, [10] develops a DL-enabled OFDM pilot sequence design using a data-driven approach to find optimal pilot locations and train a network on these locations. However, [10] does not consider attackers, leaving the computed pilot sequences vulnerable to jamming. Our work differentiates itself by computing pilot locations on the fly based on attack profiles, and using an experimental approach with online training and testing using over-the-air data.

III. Problem Overview

We consider the OFDM data transmissions scheme illustrated in Fig. 3 (top) where the entire packet frame (or burst of frames) consists of a preamble and a set of N data symbols. In practice, the former is used for frame detection, synchronization and channel estimation. The latter is instead used to transmit payload data then followed by N OFDM data symbols, where N depends on the amount of payload data being transmitted. Each OFDM symbol is split into a set of subcarriers, each serving multiple purposes. Specifically, a subset N_P of such subcarriers is allocated to pilot symbols which can be used for full channel equalization (e.g., to mitigate frequency selecting fading) and residual frequency offset compensation. N_{null} subcarriers are left unused and serve as guard bands to avoid interference with other signals being transmitted over neighboring bands, and the remaining N_D subcarriers are used to transmit both control and user data.

In Fig. 3 (bottom), we illustrate a practical instance of such a system by considering the well-established OFDM data frame structure (i.e., Config. 0) used in the IEEE 802.11 standard (commonly referred to as WiFi). This figure specifically portrays the frame format of IEEE 802.11a/g/p but the algorithms in this work can be generalized to any OFDM format with pilots within the OFDM data symbols. IEEE 802.11a/g/p uses a set of subcarriers numbered in increasing order from -32 to 32 allocated as follows. $N_{null} = 13$ subcarriers are left unused and serve as guard bands, $N_D = 48$ subcarriers are used for data transmission and $N_P = 4$ subcarriers are used to transmit BPSK-modulated pilots spread across the available subcarriers.

In IEEE 802.11a/g/p, pilots are uniformly distributed across the available subcarriers and occupy subcarriers with indexes $[-21, -7, 7, 21]$. Thanks to this allocation, the available subcarriers used to transmit pilots and data can be grouped into four equally sized *"bins"* each containing a total of 13 subcarriers (12 used to transmit data and 1 to transmit a pilot signal).

Although this allocation offers an approach to capture varying channel conditions affecting the four bins differently, it exposes the network to attacks that target pilots specifically. As we will describe in Section IV, we aim at relaxing the need for fixed-location pilots with the ultimate goal of designing a system that is resilient against pilot jamming attacks.

A. Narrowband Jamming: A Primer

Narrowband jamming attacks can turn the communication system into a random guesser with a bit error rate of 0.5 [1–3]. Pilot jamming attacks specifically target pilots. For this reason, apart from being extremely effective, pilot jamming is also more energy-efficient than barrage attacks [1], and are hard to detect in general [11, 12]. Narrowband pilot attacks

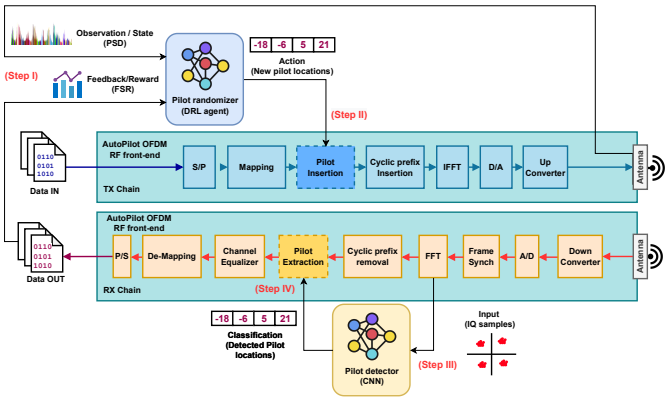

Fig. 4: Detailed overview of AutoPilot and its procedures.

are easy to achieve as pilot configurations are well-known and readily-available either through prior information (e.g., technology standards) or through observation (e.g., cyclostationary analysis). Despite some technologies allow the use of varying pilots in frequency and time, these configurations are either predictable (i.e., following well-defined re-configuration patterns) or based on pseudo-random sequences requiring hand-shakes and coordination procedures between transmitters and receivers [3, 6–9]. For this reason, it becomes essential to design and develop *reactive* countermeasures that can make pilot jamming ineffective by eliminating predictability and handshakes.

B. AutoPilot in a nutshell

In this paper, we present *AutoPilot*, an intelligent and novel OFDM transceiver design specifically tailored to combat narrowband jamming attacks in OFDM systems with specific focus on pilot jamming attacks. As shown in Figs. 2 and 4 (and as thoroughly described in Section IV), *AutoPilot* consists of an *OFDM RF front-end* and a set of *data-driven logic units*.

The OFDM RF front-end is a modified version of a standard-compliant OFDM transceiver. It embeds both transmission (TX) and receive (RX) chains to transmit and receive data over-the-air via an OFDM-based communication scheme. Among others, this includes adding pilots to mitigate the effect of time-varying and frequency-selective channels, performing channel equalization, and converting user data into IQ samples, and viceversa. The RF front-end has been extended to support non-standard compliant and unconventional pilot configurations.

The data-driven logic units are in charge of the following tasks: (i) when transmitting, the *Pilot Randomizer* adapts the pilot configuration to counteract ongoing jamming attacks; and (ii) when receiving, the *Pilot Detector* autonomously determines the new pilot configuration so that channel equalization procedures can account for the new locations of pilot symbols.

At the transmitter side, the *Pilot Randomizer* unit embeds a DRL agent that senses the spectrum to determine whether a pilot jamming attack or interference is ongoing (Step I). Based on the spectrum sensing outcome (e.g., channel state, attack state), the DRL agent computes a new pilot configuration that specifies the locations of the pilots to be used by the OFDM pilot insertion module for any subsequent data transmission

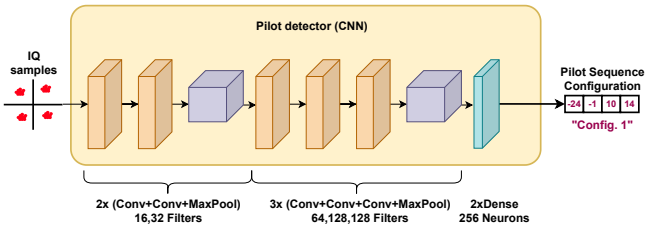

Fig. 5: Pilot Detector CNN

(Step II). At the receiver side, *AutoPilot's Pilot Detector* processes the received IQ samples to determine the pilot configuration being used by the *AutoPilot* transmitter (Step III). Upon detecting the pilots location (Step IV), this information is passed to the OFDM pilot extraction module to complete channel equalization and extract data.

As we will discuss in the following section, AutoPilot leverages feedback data already available to the system without the need to add any ad-hoc handshake procedure. *Indeed, our Pilot Detector is capable of detecting pilots in full autonomy without any information being exchanged with the transmitting node.*

IV. AutoPilot Transceiver Architecture

We now describe *AutoPilot's* design including its data-driven logic units for pilot detection (Section IV-A) and randomization (Section IV-B), as well as the customized OFDM RF front-end (Section IV-C) to support insertion and extraction of pilots.

A. Pilot Detector

AutoPilot is designed to go beyond pilot static policies by adapting pilot location according to ongoing attacks, their profile and effectiveness. This is achieved via an innovative transceiver that is able to autonomously detect with high accuracy pilots located anywhere within an OFDM frame, even when using unconventional pilot configurations. This feature is extremely important because it enables pilot randomization allows to escape jamming. Moreover, the transmitter can seamlessly transition from a pilot configuration to another one (possibly allocating pilots to unjammed portions of the OFDM spectrum) without requiring the exchange of any control information between the two parties. In this way, (i) the receiver is still be able to locate the pilots and use them to complete channel equalization and decoding, while (ii) an eavesdropper would still use the standard-complainant pilot configuration to demodulate eavesdropped waveforms, but will not be able to decode any data as pilots and data symbols are being transmitted at different locations than those specified by the standard.

To accomplish the above tasks, *AutoPilot* features a pilot detection data-driven logic unit that, as shown in Fig. 4, processes the IQ samples received on the *AutoPilot* RF front-end and extracted from the OFDM receive chain after the frame synchronization and Fast Fourier Transform (FFT) blocks. The extracted IQ samples sequence is then used to detect with high-accuracy the location of pilots. Pilot detection is performed via the Convolutional Neural Network (CNN) in Fig. 5, a reduced version of VGG16 [13–15]. One-dimensional (1D) operations are performed on a two channel input for the real (I) and imaginary (Q) parts of the signal. The input of the CNN consists

of 64 IQ samples (i.e., 1 OFDM data symbol) that are extracted from the receive OFDM chain after the FFT module. The output represents the estimated locations of the N_P pilots.

The estimated pilot locations are sent to the Pilot Extraction module (Section IV-C), which extracts them from the OFDM signal to perform channel equalization.

Complexity insights. Detecting randomized pilots via exhaustive search results in combinatorial complexity as the N_P pilots can be located anywhere. For example, we have 270,725 combinations in the case of $N_P = 4$ fully randomized pilots and $N_D = 48$ data subcarriers, where only 28,561 result in one random pilot per bin. In Fig. 3, we show four possible combinations where pilots are randomly located within the different subcarriers, but we retain one pilot per bin.

Robust pilot detection against attacks. To improve jamming detection and increase robustness, we have developed a data augmentation procedure to simulate jamming attacks during training. Specifically, we have designed an augmentation layer that is attached to the first layer of the Pilot Detector's CNN during training only. The augmentation layer acts as an additive noise generation layer which produces narrowband signals aimed at emulating narrowband jamming attacks affecting different subcarriers (both pilots and data subcarriers). This is achieved by injecting four narrowband additive white gaussian noise (AWGN) tones into the unjammed training data that we have collected over-the-air. During training, and for each batch of data fed to the CNN, the augmentation layer generates narrowband jamming signals whose location is fully randomized and whose gain is varied at random to acheive a Signal-to-Jammer Ratio (SJR) in the range -10dB to 20dB.

Resilience against eavesdropping. We point out that by randomizing pilots, attackers would not be aware of the pilot locations, and would not be able to jam them without physically getting access to an *AutoPilot* node. Furthermore, they would not be able to eavesdrop. Indeed, they can use the standard compliant preamble to estimate the lawful user's channel, but they would not be able to correct for residual frequency offset. We have verified this with our own experiment where we attempted to equalize the channel using least squares, least mean squares, spectral temporal averaging, and comb-type equalization with the preamble but different pilot locations and were unable to successfully decode achieving an FSR of 0%.

B. Pilot Randomizer

The Pilot Randomizer data-driven logic unit periodically updates the pilots location to escape jamming attacks based on IQs sampled from the channel and feedback from the receiver.

In *AutoPilot*, this is achieved via a DRL agent that takes as observation a sample of the channel frequency power spectral density (PSD) and determines the best pilot sequence to utilize in transmission given the channel state as well as the presence of interference and jammers. The reward for the DRL agent is based on the instantaneous FSR at the receiver which is communicated in training via a wired feedback to the transmitter as shown in Fig. 4. Specifically, the reward is $r = 1/3 \sum_{t=1}^{3} FSR(t) * 100 = (1 - \frac{n_{lost}(t)}{n_{lost}(t)+1}) * 100$, with $n_{lost}(t)$ and $FSR(t)$ being the number of lost packets and the FSR after the t-th transmission, respectively. We consider 3 consecutive transmission slots to reduce the stochastic nature of the channel. During online performance we expect this reward feedback to becomes wireless (part of the ACK), and if no feedback is received before a timeout (the jammer is affecting the link in both directions) the worst possible attainable reward is assumed. Although randomizing pilots via a known pseudo random sequence (without DRL) can help the system in escaping (or at least mitigating) jamming attacks, the randomized sequence can still allocate pilots to locations with jamming or interference making this solution sub-optimal [3]. Therefore, the most important requirement to enable and perform automated pilot reconfiguration to avoid pilot jamming attacks is to design a system that is able to intelligently assign and rapidly detect pilots being used at various locations.

For our DRL agent we choose a Deep-Q Learning (DQL) agent with a target network with a decaying ϵ-greedy strategy. DQL uses a replay buffer to store observations from past experiences. The feature extractor for our DRL agent is the same network we use in the Pilot Detector shown in Fig. 5 which takes as input the PSD of 256 raw IQs sampled from the channel and outputs the Q-value for the available actions. The hyperparameters are as following: learning rate $lr = 0.0005$; discount factor $\gamma = 0.85$; the rate at which our ϵ decays $\epsilon_{decay} = 0.999$; minimum ϵ value $\epsilon_{min} = 0.01$; soft-update parameter $\tau = 0.125$ used to update the weights of the target network. Our γ has been chosen to emphasize short term reward and long term reward semi-equally as jammers changing locations will require immediate action.

As we have discussed in Section IV-A, designing a transmitter that can randomly select among any possible combinations of pilot locations might be unpractical (if not infeasible) due to the corresponding large action space. For this reason, we have assumed that the transmitter can only select among a finite subset of all of the possible pilot location combinations. In this way, both the randomization and detection problems can be carried out successfully with low complexity and high accuracy as the possible action space is discrete and small.

In our case, *AutoPilot's* Pilot Randomizer can select the specific pilot sequence to be used in transmission among a set of $M = 5$ possible configurations. Four of these pilot sequences are uniformly generated within equally sized bins of the OFDM symbol while the remaining one is the traditional pilot sequence used for 802.11a as seen in Figure 3. Uniformly generated in bins means the first pilot location is randomly picked between subcarrier indices -26 and -14 (or Bin 1), the second between -13 and -1 (or Bin 2), the third between 1 and 13 (or Bin 3), and the fourth between 14 and 26 (or Bin 4). This is done to preserve the efficacy of pilot tone based intra-symbol channel estimation techniques such as interpolation as they are more effective when pilot tones are equally spaced and symmetrical (hence why traditional 802.11a uses pilots in subcarrier indices of -21, -7, 7, 21). The pilots we randomly generate may be sub-optimal for interpolation but previous works have shown they are still very effective against pilot jamming [3].

C. RF Front-End

To train an autonomous detector we first implement the capability to channel estimate, equalize, and decode non-standard compliant pilot locations. Our RF front-end is based off of the 802.11a/g/p transceiver implementation in GNU Radio by [16]. This implementation has been modified to change and detect the pilot locations on a packet basis (per OFDM burst).

Pilot extraction. Pilot detection uses a modified channel equalization block that takes meta information from our pilot extraction block, with the detected pilot locations. To extract the pilot locations via our CNN, we modify the GNU Radio Out of Tree (OOT) module for deep learning inference using ONNX [17]. This module allows for real-time inference of deep learning modules that have been converted to the ONNX format, optimized for hardware acceleration and interoperability.

Our first modification to this module is to add stream tags or meta data capabilities that can allow for the passing of configuration information to different blocks of the receiver chain i.e. pilot locations. The module watches for the start of the OFDM data symbols before beginning its classification, skipping the OFDM preambles. Once the module detects the start of the OFDM data symbols using previous stream tags from the frame synchronization block, it begins to classify on a batch of the first L OFDM data symbols within the OFDM burst resulting in L pilot sequence classifications. Our second modification is to add a majority vote on these L classifications to make a final classification on the pilot sequence configuration of the entire OFDM burst. This allows for increased classification robustness under adversarial attacks and assumes that a pilot sequence is consistent for the length of a single OFDM burst, but may still vary between bursts. Attacks may vary in the middle of an OFDM burst but they would still not know the location of pilots.

Once a final classification is obtained, the classification meta data is attached to the first OFDM data symbol of the OFDM burst and sent to the channel equalizer to be processed.

Pilot insertion. The pilot randomizer is based on the GNU Radio DRL framework in [18] to train the DRL agent online. This framework allows integration of DRL agents with GNU Radio flow graphs via an OpenAI Gym environment. We extend this framework to allow AutoPilot to change pilot locations based on the decision of the pilot randomizer. Specifically, once the pilot randomizer makes an observation and a decision, we insert the desired pilot sequence during OFDM carrier allocation in the 802.11a/g/p transmitter.

V. Training and Results

We train each data-driven logic unit separately as to not propagate any classification errors from the Pilot Detector onto the reward of the Pilot Randomizer. Once these units are fully trained, we put them together for testing. For the following testing results we used the trained Pilot Randomizer without feedback, solely making decision from the channel observations. However, note that the Pilot Randomizer can continuously learn and be trained even during implementation.

Training is done for about 530 episodes where each episode is 150 steps long for a total 79,500 training steps. A step is completed once a reward has been received which takes at the most 30 seconds. During training we vary the location and power of 4 narrowband jammers each confined to one of the 4 bins in an OFDM symbol mentioned earlier. This will allow our system to adapt against a variety of narrowband jammer profiles and assumes that the jammer is aware of how we generate pilots but not their specific locations. To measure training progress, at the end of each training episode, the DRL-based Pilot Randomizer agent is tested for 15 steps against the baseline pilot sequence in 802.11a or Config. 0 in Figure 3.

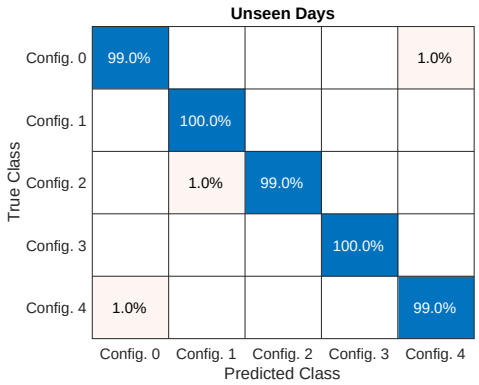

Fig. 6: Confusion matrix for Pilot Detector tested on two unseen days

We collect data over 6 days using 10 software-defined radios (SDRs) at varying distances on the Arena testbed [19]. The data is modulated using BPSK and QPSK while the pilots are always BPSK as per the standard. The location of the pilots vary between the 5 sequence configurations shown in Fig. 3 and the polarity of the 4 pilots within an OFDM data symbol alternates between (1,1,1,-1) and (-1,-1,-1,1) also as per the 802.11a standard. The varying polarity of the pilots makes the learning task even more difficult as they do not have the same constellation values between OFDM symbols. We train the CNN on 4 days out of 6, leaving the remaining 2 for testing. The narrowband noise augmentation layer at the beginning of the network is set to generate 4 narrowband tones at random locations with SJRs varying between -10dB and 20 dB. Figure 6 shows the confusion matrix for the CNN on the 2 unseen testing days. The CNN achieves 98% classification accuracy, showing exceptional generalization performance and demonstrating the effectiveness of AutoPilot in detecting randomized pilots.

To further test our detector, we test its performance in the presence of 4 randomly moving narrowband jammers at varying Signal-to-Jammer Ratios (SJRs) in real-time. Fig. 7 shows the accuracy of the pilot detector under the varying jammer powers. We can see that even in very low SJRs the detector performs better than random, 20%, by double and improves to perfect accuracy as jammer power is reduced. We will see later that even though the jammer reduces the accuracy of the pilot detector, *AutoPilot* is still able to outperform in throughput against the baseline 802.11a/g/p pilot configuration.

We report the end-to-end performance of *Autopilot* when compared to the 802.11a/g/p baseline pilot configuration in the presence of a jammer at the known pilot locations of the baseline pilot configuration. Figure 8, shows the FSR curve of

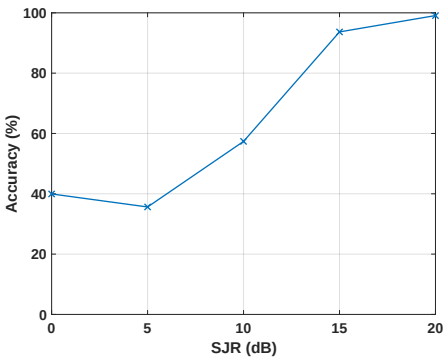

Fig. 7: Accuracy of Pilot Detector with varying SJR

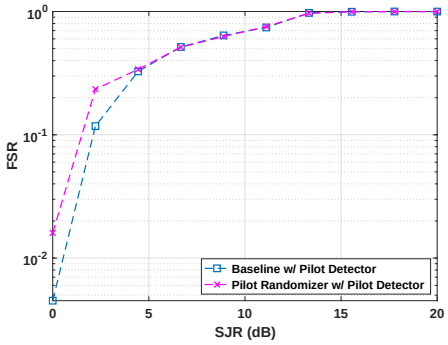

Fig. 8: FSR at varying SJRs when four narrowband jammers are on the 802.11a baseline pilot subcarriers

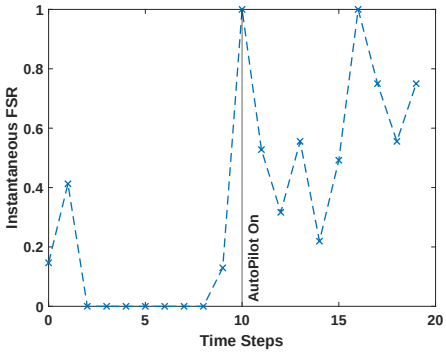

Fig. 9: Instantaneous FSR in the presence of a pilot jammer on the baseline pilot configuration before and after AutoPilot is turned on

AutoPilot using the Pilot Randomizer for the pilot configuration decisions compared with the baseline 802.11a pilot configuration. For this figure we run both methods when a static pilot jammer is present on the baseline pilot configuration (Config. 0) for 30 minutes for each SJR point. We can see that *AutoPilot* increases FSR from 0.004 to 0.016 at 0dB, a 4x increase, and at 2dB an increase from 0.11 to 0.23, a 2x increase. Higher than 2dB *AutoPilot* performs equally well and can act as a harmless alternative to the baseline pilot configuration. The reason for this, being, that after 2dB the jammer effect is very minimal so the choice of pilot configuration matters less. However in more severe cases, *AutoPilot* is clearly superior.

To show the effectiveness of *AutoPilot*, we show it in action in Fig. 9. Here the same jammer was placed on the locations of pilot Config. 0 at 2dB SJR with AutoPilot turned off (pilots are at Config. 0) for the first 10 time steps. At time step 10, we turn on *AutoPilot* which immediately samples the channel and changes the pilot locations to Config. 1 where we immediately see an improvement in the instantaneous FSR. The Pilot Detector runs on a CPU and takes 4.5ms to provide a classification. The Pilot Randomizer takes 9ms using a GPU to decide on a pilot configuration and feed it into the transmitter chain.

VI. Conclusion

In this paper, we have presented *AutoPilot*, a data-driven and reconfigurable transceiver designed to combat narrowband jamming. *AutoPilot* embeds a customized RF front-end to effectively demodulate waveforms transmitted using non-standard compliant pilot configurations as well as two data-driven logic units that are designed to (i) adapt via pseudo-randomization pilot configurations according to the current channel conditions and attack profiles; and (ii) detect rapidly and with high accuracy pilots non conforming to standard-defined pilot sequences. We have developed *AutoPilot* and presented results obtained by prototyping *AutoPilot* on a over-the-air wireless testbed. Our results show that *AutoPilot* can detect pilots with an accuracy as high as 99% on over-the-air unseen data and can improve network throughput under jamming attacks by as much as 4x.

References

[1] T. C. Clancy, "Efficient ofdm denial: Pilot jamming and pilot nulling," in *2011 IEEE International Conference on Communications (ICC)*, 2011.

[2] C. Shahriar, S. Sodagari, and T. C. Clancy, "Performance of pilot jamming on mimo channels with imperfect synchronization," in *2012 IEEE International Conference on Communications (ICC)*, 2012.

[3] C. Shahriar, R. McGwier, and T. C. Clancy, "Performance impact of pilot tone randomization to mitigate ofdm jamming attacks," in *IEEE Consumer Communications and Networking Conference (CCNC)*, 2013.

[4] M. Lichtman, J. D. Poston, S. Amuru, C. Shahriar, T. C. Clancy, R. M. Buehrer, and J. H. Reed, "A communications jamming taxonomy," *IEEE Security & Privacy*, vol. 14, no. 1, 2016.

[5] H.-M. Wang, K.-W. Huang, and T. A. Tsiftsis, "Multiple antennas secure transmission under pilot spoofing and jamming attack," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 4, 2018.

[6] 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, 2015.

[7] M. Strasser, C. Popper, S. Capkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *IEEE Symposium on Security and Privacy*, 2008.

[8] L. Kang, J. Bo, L. Hongwei, and L. Siyuan, "Reinforcement learning based anti-jamming frequency hopping strategies design for cognitive radar," in *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2018.

[9] G. Han, L. Xiao, and H. V. Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2017.

[10] M. Soltani, V. Pourahmadi, and H. Sheikhzadeh, "Pilot pattern design for deep learning-based channel estimation in ofdm systems," *IEEE Wireless Communications Letters*, vol. 9, no. 12, 2020.

[11] C. P. Robinson, D. Uvaydov, S. D'Oro, and T. Melodia, "Narrowband Interference detection via Deep Learning," in *IEEE International Conference on Communications (ICC)*, 2023.

[12] ——, "DeepSweep: Parallel and Scalable Spectrum Sensing via Convolutional Neural Networks," in *Proc. of IEEE Intl. Conf. on Machine Learning for Communication and Networking (ICMLCN)*, Stockholm, Sweden, September 2024.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] Y. Lin, Y. Tu, Z. Dou, and Z. Wu, "The application of deep learning in communication signal modulation recognition," in *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, 2017.

[15] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, 2018.

[16] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, May 2018.

[17] O. Rodriguez and A. Dassatti, "Deep learning inference in gnu radio with onnx," in *Proceedings of the GNU Radio Conference*, vol. 5, no. 1, 2020.

[18] A. Zubow, S. Rösler, P. Gawłowicz, and F. Dressler, "Grgym: When gnu radio goes to (ai) gym," in *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, 2021.

[19] L. Bertizzolo, L. Bonati, E. Demirors, A. Al-shawabka, S. D'Oro, F. Restuccia, and T. Melodia, "Arena: A 64-antenna sdr-based ceiling grid testing platform for sub-6 ghz 5g-and-beyond radio spectrum research," *Computer Networks*, vol. 181, 2020.