IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

On Assessing Vulnerabilities of the 5G Networks to Adversarial Examples

MIKHAIL ZOLOTUKHIN[1], PARSA MIRAGHAIE[2], DI ZHANG[3], AND TIMO HÄMÄLÄINEN[4].

[1]Magister Solutions Ltd., Jyväskylä, Finland (e-mail: mikhail.zolotukhin@magister.fi)
[2]Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland (e-mail: parsa.miraghaei@tuni.fi)
[3]Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland (e-mail: zhdizhan@jyu.fi)
[4]Magister Solutions Ltd., Jyväskylä, Finland (e-mail: timo.hamalainen@magister.fi)

Corresponding author: Mikhail Zolotukhin (e-mail: mikhail.zolotukhin@magister.fi).

ABSTRACT The use of artificial intelligence and machine learning is recognized as the key enabler for 5G mobile networks which would allow service providers to tackle the network complexity and ensure security, reliability and allocation of the necessary resources to their customers in a dynamic, robust and trustworthy way. Dependability of the future generation networks on accurate and timely performance of its artificial intelligence components means that disturbance in the functionality of these components may have negative impact on the entire network. As a result, there is an increasing concern about the vulnerability of intelligent machine learning driven frameworks to adversarial effects. In this study, we evaluate various adversarial example generation attacks against multiple artificial intelligence and machine learning models which can potentially be deployed in future 5G networks. First, we describe multiple use cases for which attacks on machine learning components are conceivable including the models employed and the data used for their training. After that, attack algorithms, their implementations and adjustments to the target models are summarised. Finally, the attacks implemented for the aforementioned use cases are evaluated based on deterioration of the objective functions optimised by the target models.

INDEX TERMS 5G networks, adversarial machine learning, artificial intelligence, deep learning

I. INTRODUCTION

As artificial intelligence (AI) and machine learning (ML) become a core part of almost every industry, including 5G mobile networks, there is an increasing concern about the vulnerability of AI/ML to adversarial effects. The problem of learning in the presence of adversaries is the subject of the study of adversarial machine learning that has received increasing attention in many research domains, e.g. computer vision and natural language processing [1]. As a rule, an adversarial machine learning attack may take place during the training or inference time. During the training time, the goal of the adversary is to manipulate the training process by either directly poisoning the training data or injecting perturbations to the training samples such that the target model is trained with erroneous features and thus it makes errors later in the inference time [2]. During the inference time, the goal of the adversary is most of the time to feed such features to the target model that it returns a certain wrong output [3]. Similar to the poisoning attacks, adversarial examples can be either generated from scratch or via adding carefully crafted adversarial perturbations to normal samples. As a rule, the adversary focuses on the samples with output labels that are closer to the decision region, allowing it to increase the probability of error at the target model [4]. The attack can be either non-targeted, when the adversary causes the classifier to predict any incorrect label, or targeted, in which case the adversary aims to increase the classifier's prediction probability to a particular output label. Depending on the information available to the adversary, the adversarial example attacks can be classified into either white-box and black-box category. The former includes cases when the adversary has perfect knowledge of either the machine learning model or the data used for its training or both of those. In the later scenario, the adversary's only capability is to observe labels assigned by the model for the inputs supplied. The black-box attacks are more practical for real-world adversaries with knowledge about neither the model nor the training data [5].

On the adversarial defence side, two different strategies can be considered: runtime detection of adversarial inputs and model hardening. The former includes explicit detection or rejection strategies for adversarial samples. Typically, these defences are designed to work under the so-called

manifold hypothesis which assumes that normal data samples lie in a low-dimensional manifold embedded in a high-dimensional space [6]. Such manifold-based defences work by identifying adversarial points from their distance to the manifold. If an input sample moves away from a class prototype, its support decreases. If the sample is not supported by any class, then it is rejected [7]. Among the model hardening methods, a widely explored approach is to augment the training data of the AI/ML model with adversarial examples [1]. Another approach is input data preprocessing, often using non-differentiable or randomised transformation [8], transformations reducing the dimensionality of the inputs [9], or transformations aiming to project inputs onto the normal data manifold [10]. Other model hardening approaches involve special types of regularisation during model training [11], or modifying elements of the classifier's architecture [12].

The key element to measure the vulnerability of a classifier with respect to particular attacks and to assess the effectiveness of adversarial defences is a robustness metric. Typically such metric quantifies the sensitivity of model outputs with respect to changes in their inputs, or more specifically, the minimal amount of perturbation that is required to cause a misclassification [3]. Unfortunately, finding the global minimum adversarial perturbation is close to impossible in any practical setting, and, therefore, heuristic attacks are often employed to find a suitable approximation [13]. However, employing such heuristics can fail, making one believe that a model is robust [14] whereas in fact it is not [15]. Thus, the best strategy is to employ as many attacks as possible, and to use the minimal perturbation found across all the attacks as an approximation to the true global minimum [16]. This strategy can be implemented with the help of one of the existing frameworks which allow one to evaluate vulnerabilities of an AI/ML model to various adversarial example generation algorithms. As a rule, these frameworks offer reference implementations of multiple state-of-the-art white-box and black-box attack algorithms, model hardening and manifold-based detection methods as well as robustness metrics and certifications [13], [16]–[20]. When crafting adversarial examples using such a framework, one can specify a target model that takes an input and makes a prediction, a criterion that defines what an adversarial perturbation is, a distance measure that measures the size of the perturbation and an attack algorithm that takes an input and its label as well as the model, the adversarial criterion and the distance measure to generate the adversarial perturbation. Thus, adversarial robustness of an AI/ML model can be tested against known adversarial example attacks before the model is deployed into production. Since the model is tested against specific attacks in specific settings, it does not guarantee full protection as this research topic has recently attracted significant attention and novel attack approaches are constantly emerging [21]. However, testing the target model against known attack approaches allows one to reduce the attack surface leaving the adversaries with fewer ways to perform attacks and therefore making it easier to implement a manifold-based detection and rejection system to further protect the target model.

In our research, we focus on adversarial example attacks that may take place in the inference stage in one of the AI/ML-based components of future 5G networks. These components may focus on channel estimation [22] and symbol detection [23], automatic modulation classification [24] and channel coding [25], beamforming [26] and power allocation [27], scheduling [28] and routing [29], as well as slicing [30] and caching [31]. Due to the shared and open nature of wireless medium, these components may be highly susceptible to adversaries that manipulate the inputs to the AI/ML models during the inference stage over the air. However, an adversary in the wireless domain as a rule cannot directly collect the same input features as the target AI/ML model, due to the different channel and interference conditions. Furthermore, the adversary most of the time cannot directly obtain the output label of the target model, since it is used internally by the model and it is not available to any other wireless node outside of the network. Finally, the adversary is not able to directly manipulate the input data to the target model, it can only add its own transmissions on top of existing transmissions over the air to change the input data indirectly [32].

The main goal of this study is to review potential applications of AI/ML in the next generation wireless networks and identify potential attack vectors against these applications via adversarial example generation. The rest of the document is organised as follows. Various AI/ML algorithms and adversarial example generation attacks against these algorithms are briefly summarised in Section II. Section III presents multiple examples of potential AI/ML deployment in different 5G frameworks found in recent scientific papers and overviews several adversarial example generation attacks proposed against these frameworks. Numerical simulation results for some of the use cases are presented in Section IV. Section V concludes the paper and outlines future work.

II. THEORETICAL BACKGROUND

A. AI/ML MODELS

The vast majority of the AI/ML applications proposed to be deployed in the next generation wireless networks is based on deep learning architectures trained in a supervised way. A deep neural network consists of multiple layers of non-linear processing units. The main idea behind deep learning is using the first layers to find compact low-dimensional representations of high-dimensional data whereas later layers are responsible for achievement of the task given, e.g. regression or categorical classification. All the neurons of the layers are activated through weighted connections. In order for the network to be capable of approximating a nonlinear transformation, a non-linear activation function is applied to the neuron output. The learning is conducted by calculating errors in the output layer and backpropagating gradients towards the input layer. In a hidden or output layer of a fully-connected neural network (FCNN), each neuron is connected to all neurons of the previous layer with the output

being calculated by applying the activation function to the weighted sum of the previous layer outputs. Such layers have few trainable parameters and therefore learn fast compared to more complicated architectures described below, however they may suffer when dealing with spatio-temporal data such as images and time-series.

Most of the time, convolutional neural networks (CNNs) are employed for automatic extraction of low-level features such as edges, colour, gradient orientation in image related problems [33]. The main building block of CNN is the convolutional layer which calculates an integral that expresses the amount of overlap of the layer's filter as it is shifted over the input data. Similarly to the previous case, the integral value is passed through an activation function to account for non-linearity in data. As a rule, multiple convolutions are performed on the input, each using a different filter. Resulting feature maps are then stuck together and become the final output of the convolution layer. CNNs can be employed to handle data of any dimension, since the result of the convolution operation is always a scalar. CNNs usually consist of several convolutional layers followed by standard fully-connected layers. Stacking multiple convolutional layers allows one to learn both basic features as well as higher level representations to recognize objects in different shapes and positions.

Temporal dependencies in the data can be extracted with the help of recurrent neural networks (RNNs). In distinction to a fully-connected and convolutional layers, a recurrent layer assumes that input data samples are time-series. To accommodate this fact, each recurrent layer has its own internal state the value of which is calculated based on the state value of the previous sample. The output of the recurrent layer is essentially an activation of the weighted sum of the previous layer outputs added to the weighted sum of the previous state values. During the learning process, derivatives are backpropagated through time, all the way to the beginning or to a certain point. All the derivatives multiply the same weight matrix which may result in either infinite or vanishing update values. While gradient exploding can be fixed by straight-forward clipping [34], dealing with gradient vanishing requires an intelligent control over the state via forget gates [35]. The most popular gate-based RNN layers are based on gated recurrent units (GRUs) [36] and long short-term memory (LSTM) [35].

Speaking of the supervised algorithms which are not based on deep learning, there are several ones that are still used by AI/ML researchers for classification and regression. For example, k-nearest neighbours (k-NN) algorithm is a classification algorithm, in which several closest training samples in a dataset are used to predict the class for a new sample. K-NN can also be used for regression tasks. In this case, the output is the average of the values of its nearest neighbours. Another simple regression method is linear regression which generates a sloped straight line describing the relationship within two continuous variables. From the algorithmic point of view, a linear regression model is simply a single-layer neural network with linear activation function. Closely related to linear regression and SVM algorithms is support vector regression (SVR). The aim of SVR is to find such a linear model that the norm of the weight vector is minimal under the constraint that the points are within the decision boundary line. It can be solved with the help of constrained optimization methods. The next one is a random forest algorithm which uses an ensemble of decision trees each of which aims to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from the training data. There are several popular algorithms used for training decision trees. For example, in the ID3 algorithm, at each iteration, for a set of data samples, an attribute which has the smallest entropy is selected and the set is split by the selected attribute to produce new subsets of the data samples. The random forest algorithm operates by constructing a multitude of decision trees at the training time and outputting the class that is the mean prediction of the individual trees. Finally, gradient boosting is a machine learning technique which builds a prediction model in the form of an ensemble of weak learners, e.g. decision trees.

Unsupervised learning is also proposed to be applied for enhancing various the next generation network components. The most popular deep learning architecture trained in an unsupervised way is called autoencoder. In general, an autoencoder consists of an input layer, several hidden layers, and an output layer. The objective of the network is for the output layer to be exactly the same as the input layer despite the information bottleneck caused by the hidden layers. The reconstruction error which is the difference between the input and the output is often used as the loss function. The process of going from the first layer to the hidden layer is called encoding. The process of going from the hidden layer to the output layer is called decoding. In this learning process, the autoencoder essentially learns the format rules of the input data. Another unsupervised deep learning approach that is employed in several AI/ML applications in 5G as well as in the attacks against those applications is based on generative adversarial networks (GANs) [37]. In GANs, the generator neural network takes a fixed-length random vector as input and generates a sample in the domain. Another neural network called discriminator generates an estimate of the probability that a given sample is real or generated. The discriminator is supplied with a set of samples which include both real and generated ones and it would generate an estimate for each of these inputs. The error between the discriminator output and the actual labels would then be measured by cross-entropy loss. The generator is updated based on how well, or not, the generated samples fooled the discriminator. Outside of the deep learning area, k-means, which is a partitioning technique which classifies a dataset of objects into a predefined number of clusters, is still broadly used in research.

Finally, multiple studies propose to deploy reinforcement learning (RL) algorithms in the future networking frameworks. Deep Q-Network (DQN) proposed in [38] presents

the most well-known deep RL model to learn control policies directly from high-dimensional sensory input. The value function of each action at each time step, Q-function, is evaluated using the Bellman equation [39] that is proven to converge to an optimal value [40]. DQN uses a deep neural network as the function approximation to estimate the value function. The network is trained by minimising the loss function, which is essentially the difference between the value of Q-function predicted in that particular time step and the target value function that is evaluated using the real reward value obtained from the environment. DQN is proven to be a powerful tool that could deal with problems involving low-dimensional, discrete observations and actions in the mobile networking domain [30], [41]. Other RL algorithms such as deterministic policy gradient (DDPG) [42], state-action-reward-state-action (SARSA) algorithm [43], trust region policy optimization (TRPO) [44] and proximal policy optimization (PPO) [45] are also proposed to be employed for various 5G network applications [29], [46]–[48].

B. ADVERSARIAL EXAMPLE ATTACKS

With the success deep learning has reached in recent years comes the price of the models that follow this approach to be the most popular target for adversarial example attacks. As a rule, deep neural networks have a differentiable loss function and use a gradient-based optimizer during the training which enables gradient-based adversarial example generation by modifying an input sample in the direction of the gradient of the loss function with respect to the input sample [1]. This allows one to craft an adversarial perturbation to carry out a non-targeted attack in the white-box setting. Another gradient-based attack called basic iterative method (BIM) is introduced in [49]. It extends the fast gradient sign method (FGSM) described above by applying it multiple times with small step size, clipping values of intermediate results after each step to ensure that the difference between them and the original input does not exceed a predefined threshold value. The size of this threshold value is the main parameter of the attack algorithm and it depends on the particular attack scenario. Further into the paper, we refer to this value as a perturbation budget or perturbation size interchangeably.

Another white-box approach for generating adversarial examples is introduced in [3]. In that study, adversarial examples are defined as inputs that look very similar to their real counterparts according to a distance metric, but one that causes the target classifier to misclassify it. In order to find such an input, one requires to solve a nonlinear optimization problem with the objective function being equal to the weighted sum of the adversarial perturbation norm and the target model loss calculated for the perturbed sample. Study [3] uses a non-linear gradient based numerical optimization algorithm called Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) to solve the resulting minimization problem. Studies [50] and [51] propose various improvements of this approach resulting in famous Carlini & Wagner (CW) and DeepFool attacks.

In the black-box settings, the target model is assumed to be unknown to the adversary. However, the adversary may have an ability to query an input sample to the target classifier in order to calculate the perturbation which results in the input being misclassified. The first fundamental study in this research area [52] proposes a transfer based attack approach which relies on information about training samples without the knowledge of the training model. The adversary's strategy is to learn a substitute for the target model using a synthetic dataset carefully generated by the adversary and labelled by observing the target model output. The adversary then uses the information acquired to train the substitute model. After that, a white-box attack is used to synthesise adversarial perturbations for the substitute model. The adversary expects the target model to misclassify the resulting perturbed samples due to transferability between AI/ML models. The approach can therefore be employed against various AI/ML models with different rates of success.

Several studies propose to search for the perturbation without training a substitute model. As a rule, the attack that follows this approach starts from a point that is already adversarial and then performs a random walk along the boundary between the adversarial and the non-adversarial region such that it stays in the former while the distance towards the target legitimate point is reduced. Both the length of the total perturbation of the adversarial sample and the length of the step towards the original input are adjusted dynamically similarly to a trust region method. Such boundary attack is introduced in [53] and further developed as HopSkipJump attack algorithm in [54].

III. ADVERSARIAL ML IN 5G

A. AI/ML IN 5G

Massive multiple-input and multiple-output (MIMO) has become an essential element of wireless communication standards including 5G. AI/ML models are frequently suggested to be applied to MIMO for channel estimation and symbol detection. The common goal of such models is to reduce feedback transmission overhead and delay required for the channel state information (CSI) estimation. For example, in [55], a deep neural network is used to map CSI at one set of antennas and one frequency band is mapped to the channels at another set of antennas and frequency band which allows for significantly reducing the pilot training and feedback overhead. Similarly, study [56] proposes a method based on deep neural networks that predicts the CSI in the downlink (DL) based on the past uplink (UL) measurements in an orthogonal frequency-division multiplexing (OFDM) system to eliminate the overhead caused by DL pilot transmissions. It basically assumes that since DL and UL channels share the same propagating environment, a data-driven approach can be employed to extract an environment information from the UL channel response to a latent domain and then transfer this information from the latent domain to the DL channel. In [23], a neural network is employed to estimate channels implicitly and recover the transmitted symbols directly. Study

[57] proposes a deep-learning-based CSI sensing and recovery mechanism for OFDM MIMO systems in which an UE uses an encoder network to transform channel matrices into codewords, and once these codewords are returned to the next generation NodeB (gNB), a decoder network at the gNB is used to recover the original channel matrices. This allows to reduce the dimensionality of the CSI matrices and which leads to the overhead decrease.

Another application of AI/ML in 5G is automatic modulation recognition which enables adaptive transceivers to automatically switch modulations based on the channel conditions without the need for a feedback channel between the transmitter and the receiver. For example, study [24] proposes a deep neural network enabled modulation recognition which can automatically learn to extract features from long symbol-rate signals at low signal-to-noise (SNR) levels. In [58], complex points representing the received signals are transformed into constellation diagrams by mapping signal samples into scatter points on the complex plane, and then an AlexNet model [59] deployed at the receiver is used to derive the modulation type employed at the transmitter.

Furthermore, recent studies devoted to application of AI/ML to the next generation wireless networks aim to develop deep learning-based channel coding schemes in order to overcome the existing problems of conventional decoding algorithms such as high decoding complexity and lack of robustness against channel variations. The first fundamental study in this area [25] proposes to interpret an end-to-end communication system as an autoencoder, where both the transmitter and receiver are implemented as deep neural networks. An end-to-end reconstruction optimization task using autoencoders allows one to jointly learn transmitter and receiver implementations as well as signal encodings without any prior knowledge. The autoencoder proposed seeks to learn representations of the messages that are robust with respect to the channel impairments mapping, i.e. noise, fading and distortion, so that the transmitted message can be recovered with small probability of error. Further studies [60], [61] focus on the decoding problem with the neural network being trained to minimise the error between its output and the original codeword sent.

Multiple studies employ AI/ML for beamforming and optimising MIMO antenna weights. For example, in [26], a deep learning solution for fast and accurate initial access (IA) in 5G mmWave networks is proposed. The IA time consists of two components: time for beam sweeping, i.e. measuring the received signal strengths (RSSs) for different beams, and time for beam prediction, i.e. identifying the beam for a given transmitter-receiver pair to communicate with. Since the beam sweep time dominates the overall IA time, it is essential to improve the IA time by utilising fewer beams. The study attempts to solve this problem with deep learning. In particular, it attempts to reduce the beam sweep time by measuring RSSs from only a subset of all available beams and mapping them to the best selection from the entire set of beams. The model is a neural network trained by feeding the RSS values from a subset of beams as the input. The output of the network consists of the probabilities of being the optimal beam calculated for each beam in the set. Another deep learning based beamforming approach is proposed in [62]. It uses uplink training pilot sequences for each beam coherence time sent from UEs to several neighbouring gNBs to predict the best beamforming vector with the highest achievable rate for each of these gNB s.

In multiple studies, various AI/ML models are employed for intelligent power allocation policies in order to minimise the interference and reduce the energy consumption. For example, study [63] considers the power allocation problem for downlink communications from the gNB using multiple different orthogonal subcarriers to communicate with several UEs. A neural network is employed at the gNB in order to deal with the complexity of the solution for the power allocation optimization problem when using traditional solving methods. In order to carry out the power allocation procedure, the gNB transmits pilot signals from each of its subcarriers one by one, each UE served by the gNB estimates the channel gains, and reports them back to the gNB. Based on these channel estimations, the gNB allocates power to its subcarriers to serve each of the UEs using the neural network trained to output an optimal power allocation vector calculated using a traditional method. Another example of using machine learning for optimal power allocation is described in [64]. The AI/ML framework proposed uses a neural network to learn the mapping between the positions of UEs and the optimal power allocation policies.

Network slicing allows operators to offer a diverse set of services over a shared physical infrastructure. Despite the advantages it brings to network operators, network slicing raises big challenges related to the optimal resource allocation. Several studies focus on deployment of AI/ML models for dynamic resource allocation to radio access network slices. There are two main approaches. The first approach relies on employing RNNs to predict usage of each radio access network (RAN) slice in order to adjust the resource distribution [65], [66]. The second approach formulates the resource allocation problem as a Markov decision process (MDP) and a deep RL algorithm is applied to solve it [30], [41], [48], [67]. The RL approach looks more promising as the resulting policy optimises RAN resource distribution directly and there is no need to predict each network slice usage. In the majority of the studies mentioned the resources available are assumed to be discrete physical resource blocks (PRBs), and a deep Q-network is employed in order to find an optimal policy. The policy decisions as a rule depend on the numbers of requests arriving at each slice and their priority, throughput, computational resources, and latency requirements.

Furthermore, AI/ML provides automated means to capture complex dynamics of wireless spectrum and support better understanding of spectrum resources and their efficient utilisation. In particular, cognitive radio capabilities empowered by machine learning allow for performing spectrum aware-

ness and spectrum sharing. For example, in study [32], an AI/ML model at an environmental sensing capability (ESC) station detects citizens broadband radio service (CBRS) as an incumbent user. If the incumbent user is not detected in a channel of interest, the ESC allows a gNB to communicate to UEs. Otherwise, the gNB cannot use this channel and it is reconfigured to vacate this particular channel to avoid interference with the incumbent signals.

Finally, AI/ML has been broadly applied for cyber security in wireless communications. For example, in studies [68], [69] systems for anomaly detection and cyber defence in the context of a 5G mobile network architecture are proposed. The systems are essentially network intrusion detection systems (IDSs) classifying network traffic flows as either normal or malicious. In the RAN domain, AI/ML models are also proposed to be employed for detection of jamming attacks based on values of the UE state features such as energy consumption, packets sent/received, distance to gNB and several others [70], [71].

B. ADVERSARIAL EXAMPLES IN 5G

In this subsection, we describe how an adversary may try to attack AI/ML-based network components described above. In the channel estimation cases [55] and [56], the adversary can aim crafting such a perturbation that would maximise the error between the real DL CSI matrix and the one predicted by the target model, whereas in the case of the framework described in [23], the adversary would try to maximise the difference between the transmitted symbols and the output of the target model. In all of these attack scenarios, the adversary would be required to have access to a dataset in order to train a substitute model [5]. The attack approaches in which it is required to query the target model with intelligently crafted samples would be hard to carry out since the outputs of the target models mentioned appear to be used internally by the gNB for efficient use of frequency bands and energy by performing various techniques, such as water-filling, appropriate precoding and beamforming [56]. Another option would be to craft a universal adversarial perturbation (UAP) [72].

Study [73] proposes an adversarial example generation attack against the autoencoder based framework for CSI feedback described in [57]. In particular, the attack is performed against the neural network which acts as the decoder. The adversary aims to maximise the error between the real CSI matrix and the one predicted by the decoder model. The attack is white-box i.e. the adversary is required to know the target decoder network. Moreover, in the attack scheme described, the adversary somehow has access to the input codeword which is essentially the DL CSI matrix encoded at the UE. The former problem can be mitigated via training a substitute autoencoder model [5]: since the target model is unsupervised, the adversary would only need to get access to a dataset of DL CSI matrices. The latter would require the adversary to be able to eavesdrop the signal that contains the CSI matrix encoded and then jam this signal in such an intelligent way that a necessary perturbation is added to the decoder input. Crafting a UAP would also be possible. As in the previous case, the attacks that rely on querying the target model's public API do not look applicable in this scenario, as the output of the decoder is not returned to the UE, but it is used internally by the gNB.

Attacks against AI/ML-based modulation recognition models is probably the most well-studied category of the attacks against wireless communication systems based on adversarial example generation [74]–[77]. In a real world scenario, the adversary would have access to neither the exact input of the receiver nor the modulation type selected by the target model. It would also be fair to assume that the adversary does not know the exact channel between the adversary and the receiver, but several realisations of that channel are available to the adversary [75], [77]. It can also be assumed that the information available to the adversary includes an arbitrary dataset of the received signals with their corresponding modulation types. In such settings, the adversary will be able to first train a substitute model [5] and craft adversarial perturbations using one of the white-box attack methods described in details in [75]–[77]. After that, a universal perturbation can be generated.

To attack intelligent channel decoding frameworks, the adversary can try to generate a perturbation that causes decoding errors at the receiver [74], [78]. In white-box settings, this perturbation can be carried out e.g. by employing FGSM and then projecting the resulting optimal perturbation on the ball with the centre at the original sample and the radius equal to the power budget available to the attacker [78]. In the black-box settings, the attack can be carried out by training a substitute model using the dataset generated for a similar task since again there is no possibility to query the target model with various test samples. Alternatively, a UAP can then be crafted as it is proposed in [78].

In the case of AI/ML-driven beamforming, the adversary may search for either a perturbation that causes any misclassification at the receiver's classifier or such a perturbation such that it not only causes a misclassification at the receiver's classifier, but also tries to change the beam to one of the worst beams. In [79], such adversarial perturbations are crafted in white-box settings with the FGSM algorithm. In black-box settings, the adversary would again most likely train a substitute model or search for a universal perturbation since querying the target model does not appear to be a feasible option due to the nature of the attack. An adversarial example generation attack which targets the deep learning model introduced in [62] is demonstrated in [80]. The adversary aims to maximise the error between the real achievable rate and the one predicted by the model. The attack implemented in [80] is white-box. Furthermore, the study assumes the adversary has perfect knowledge of the input feature vector for which the prediction is made. In other words, the adversary most likely needs to have access to the aforementioned cloud processing unit during the inference stage to be able to perform this attack. In black-box settings, a substitute model should be trained by the adversary and

then used during the inference to craft an input-agnostic adversarial perturbation.

In order to attack the power allocation model proposed in [63], the adversary can try to jam the channels between the gNB and the UEs in order to make the neural network at the gNB output a non-optimal power allocation vector. In [63], this attack is performed in white-box settings using FGSM algorithm. In a realistic use-case scenario, the adversary would first need to train a substitute model and then search for a universal perturbation during the attack. In order to attack the power allocation system described in [64], the adversary perturbs the input that is fed to the target model by employing one of the GNSS spoofing techniques [81]. The objective of the attacker is to compute the adversarial perturbation of UEs positions in the direction of the gradient to increase the loss function such that the power allocation system outputs non-optimal power allocation vector. In [82], such an attack is implemented in white-box settings using FGSM and PGD algorithms. In black-box settings, a substitute model is suggested to be trained. In this use-case, the UE positions are assumed to be known to the adversary. Otherwise, an input-agnostic adversarial perturbation can be crafted as it was done in the previous scenarios.

The adversary may also attempt to attack the network slicing models proposed in [30], [41], [48], [67] by generating such requests that would force the target RL model to make incorrect resource distribution decisions. Study [83] proposes such an attack against the RL-based resource allocation model presented in [30] in black-box settings. In particular, the adversary aims to determine resources to be specified by fake requests for the most efficient flooding attack. If these fake requests are selected and network resources are allocated to them, fewer resources will be left for real requests from legitimate users. The attack is based on Q-learning algorithm with each state being the number of available PRBs, the action being equal to the number of PRBs assigned for each fake request, and the reward being calculated as the number of served fake requests. It is assumed in the study that the adversary may sense the spectrum in order to detect available PRBs. It is also assumed that the adversary has information whether the request sent has been served or not. Other black-box algorithms also appear to be applicable in this use-case scenario: the adversary may query the target model by sending fake requests with various requirements in order to find an optimal solution.

In [32], the adversary aims to compromise the integrity of the target AI/ML model deployed for intelligent spectrum sharing during the sensing periods to force the ESC into making wrong transmit decisions. In particular, the adversary attempts to fool the ESC to allow the gNB to transmit when an incumbent user is present, and vice versa, to fool the ESC to stop the gNB transmissions even though there are no CBRS users. The attack proposed is black-box: the adversary trains a substitute model by monitoring both CBRS radar signals and whether the gNB transmits to its UEs. The former acts as the input to the substitute model, whereas the latter is used to provide ground truth labels for the model. However, the adversary is still required to know the input to the ESC and the channel between the adversary and the ESC for crafting correct perturbations. As previously, algorithms for crafting an input-agnostic perturbation can be employed in order to resolve the aforementioned issues.

An adversarial example generation based attack can also be carried out to craft adversarial network traffic flows that would deceive the detection models proposed in [68], [69]. For example, study [84] attempts to craft such a perturbation to a botnet related traffic flow that it is classified as a legitimate one. To achieve this goal in black-box settings, DQN algorithm for crafting adversarial perturbations is employed, as it allows the adversary to operate in the scenario when its feature space is different from the one employed by the target model, as the latter is assumed to be unknown to the attacker. In theory, however, any black-box algorithm that queries the target model with intelligently crafted input samples can be used in this use-case, since both the input sample and the model output can be derived by the attacker assuming any malicious flow will be blocked by the IDS. In the case of jamming attack detection, the adversary can try to manipulate the signal parameters in order to make the target model misclassify it as a legitimate UE. Under the same assumption that the target model output, i.e. whether the device is classified as normal or malicious, is known to the adversary, any of the black-box adversarial perturbation generation algorithms described previously in the study can be used.

IV. NUMERICAL SIMULATIONS

A. USE CASES

First, we briefly summarise several use cases attacks against which are evaluated in this study. The use case scenarios include modulation recognition based on raw in-phase / quadrature (I/Q) samples [85], channel state estimation based on the [55], optimal beam selection based on RSS measurements for a subset of beams [26] and sub-6GHz channels [86], decoding polar [60], convolutional and low-density parity-check (LPDC) [61] coding schemes as well as jamming detection [71]. We focus on these particular use cases for three main reasons: first, there is a clear benefit of AI/ML deployment in each of these use cases; second, the corresponding study provides detailed information how to implement and train the AI/ML model proposed; third, there is a room for a adversarial example generation attack against the resulting framework.

1. Modulation recognition. As it was mentioned in the previous section, study [85] proposes a deep neural network enabled modulation recognition which can automatically learn to extract features from complex base-band time series representation of the received signals at various SNR levels. Time series of the received signal in I/Q format acts as the input whereas the output is the modulation type.

2. Channel estimation. Study [55] introduces a concept of channel mapping in space and frequency, and it focuses

on the use case when the uplink channels at a subset of antennas are directly mapped to the downlink channels at all the antennas significantly reducing the training and feedback overhead. The channel mapping function is a deep neural network. Channel matrix for a subset of antennas in the uplink acts as the input to the network and the output is the entire channel matrix in the downlink.

3. Beam selection based on an RSS subset. In [26], a deep learning solution for fast and accurate initial access (IA) is proposed. The model is a neural network trained by feeding the RSS values from a subset of beams as the input. The output of the network consists of the probabilities of being the optimal beam calculated for each beam in the set.

4. Beam selection based on sub-6GHz channels. A similar problem is studied in [86]. However, instead of using RSS values received at a subset of beams, channel coefficients of a sub-6GHz network which is assumed to be deployed in the area act as the input to the AI/ML model. The model used in the aforementioned study is a fully-connected neural network.

5. Convolutional channel decoding. Study [61] aims to develop a deep learning based channel coding scheme for convolutional codes in order to overcome the existing problems of conventional decoding algorithms such as high decoding complexity and lack of robustness against channel variations. To generate training samples, a codeword is randomly picked from the codebook set given and then the received vector is obtained by performing channel encoding, the binary phase shift keying (BPSK) mapping and simulated channel noise. The received signal acts as an input to the neural network whereas the decoded signal plays the role of the output.

6. LDPC channel decoding. The same study [61] also applies this approach for decoding low-density parity-check (LPDC) channels. As previously, the received signal acts as an input to the AI/ML model and the decoded signal plays the role of the output.

7. Polar channel decoding. A similar approach is studied in [60] which involves training a neural network to guess the original word encoded with polar codes based on the signal received. Similar to the previous two cases, the output for each input codeword is obtained by performing channel encoding, BPSK mapping and adding simulated channel noise.

8. Jamming detection. Study [71] focuses on deploying an AI/ML based detection of jamming attacks on unmanned aerial vehicles (UAVs) that operate using OFDM communication. In particular, authors attempt to detect and classify multiple jamming attack types which include barrage, single tone, successive-pulse (SP) or P-aware (PA). Input features include average received signal and noise power.

B. DATA AND MODELS

In the modulation recognition use case, RadioML dataset [87] is employed for training the neural network. This is a synthetic dataset consisting of the received signals in I/Q format for several modulation types at varying SNR levels.

Use case	Input / Output	Model	Data
Modulation recognition	RSS time-series / Modulation type	CNN, 2 conv. layers: 256, 80, 1 dense layer: 256	RadioML dataset
Channel estimation	Channel matrix at the UL / Channel matrix at the DL	FCNN, 4 layers: 1024, 4096, 4096, 2048	DeepMIMO, scenario I1
Beam selection (subset)	RSS for a subset of beams / best beam index	FCNN, 5 layers: 2048, 2048, 2048, 2048, 2048	DeepMIMO, scenario O1
Beam selection (sub-6GHz)	Sub-6GHz channel RSSs / best beam index	FCNN, 5 layers: 2048, 2048, 2048, 2048, 2048	DeepMIMO, scenario O1
Channel decoding (convolutional)	Codeword received / original codeword	RNN, 1 LSTM layer: 256	Synthetic dataset
Channel decoding (LDPC)	Codeword received / original codeword	RNN, 1 LSTM layer: 256	Synthetic dataset
Channel decoding (polar)	Codeword received / original codeword	RNN, 1 LSTM layer: 256	Synthetic dataset
Jamming detection	Signal and noise power / signal type	FCNN, 2 layers: 512, 512	Real equipment data

TABLE 1: Data and models selected for evaluation of adversarial example generation attacks.

The models employed are a FCNN and two CNNs. The model that provides the best results in terms of the prediction accuracy is the CNN with two convolutional layers of 256 and 80 filters followed by one fully-connected layer of 256 neurons. We used this neural network as the target model in our experiments for the modulation recognition use case.

In order to generate data for numerical evaluations at the channel estimation use case, DeepMIMO dataset is used [88]. This dataset is essentially a light-weight massive MIMO mmWave simulator. Given a scenario and several input parameters, it generates channel matrices for each BS-UE ray-tracing path. In the channel estimation use case, indoor scenario I1 is used. The scenario comprises a 10 square metre room with two tables and 64 antennas tiling up part of the ceiling. 2.5 GHz is used as the operating frequency, both the BS and UE antenna shapes are selected to be equal to $(1, 1, 1)$. Other input parameters such as the number of OFDM subcarriers and the bandwidth can be found in [88]. Once the channel matrices have been generated, a subset of antennas is randomly picked. This subset remains the same for all the data samples during the training, validation and inference stage. The selected channels act as the input data to the AI/ML model in order to calculate the entire channel matrix. Speaking of the size of the subset, the value of 8 is selected. According to the results obtained in the original

study [88], this is the minimal subset size which allows for accurate channel estimation. The channel mapping function is a FCNN which consists of 4 layers with 1024, 4096, 4096, and 2048 neurons for each layer respectively.

At the beam selection use cases, outdoor scenario O1 from DeepMIMO dataset is used. One base station (BS 3) is selected as the serving base station and UEs are assumed to be located between rows 700 and 1300. The operating frequency is 28 GHz whereas the BS and UE antenna shapes are equal to $(1, 64, 1)$ and $(1, 1, 1)$ respectively. Other input parameters can be found in [86]. For the beam subset use case, channel values for randomly picked 25% of the beams act as the input data. As previously, the subset is the same for all the data samples. In the sub-6GHz case, additional channel matrices for the same scenario but with the frequency equal to 3.5 GHz are generated and they act as the input for AI/ML beam selection. For each data point in both cases, the mmWave beam in a codebook F which provides the highest sum-rate for the channel matrix given is calculated. The corresponding one-hot vector that indicates the index of this optimal beam acts as the output data point. It is worth mentioning that a simple quantized beam steering codebook where the i-th beam for $i = 1, 2, \ldots, |F|$ is defined as $f_i = a(\frac{2\pi i}{|F|})$, with a representing the mmWave array response vector [86]. The model is a FCNN with 5 hidden layers of 2048 neutrons which is proposed to be used for the best beam prediction in [86].

To train AI/ML models for channel decoding, three datasets are generated, one for each type of codes: convolutional, LDPC and polar. A sample in each such dataset includes the received vector and the true codeword transmitted. Each codeword is randomly picked from the set of all possible binary vectors of the length given. In our experiments, the length values are 50 for convolutional and LDPC codes and 16 - for polar codes. The received vectors are then obtained by performing channel encoding, BPSK mapping and simulated channel noise [60]. Speaking of the noise, an SNR value is first sampled uniformly from the range given and the noise variance is set to be equal to one divided by this value. After that, the noise vector is obtained by independently sampling the noise distribution from the resulting Gaussian distribution [61]. In [60], three neural network architectures are tested: FCNN, CNN and RNN. Results of the numerical experiments carried out by the authors show that the RNN has the best decoding performance at the price of the highest computational time. The RNN tested consists of one LSTM cell of size 256.

Finally, in the case of the jamming detection problem, the dataset is obtained by conducting experiments with real equipment [71]. Input features in the dataset provided include subcarrier spacing, symbol time, subcarrier length, cyclic prefix length, average received power, threshold, average signal power, average noise power, and SNR. In our experiments, we only use the features that can be affected by an adversary over the air, i.e. average received power, average signal power, average noise power, and SNR. In the original study [71], several models are tested including neural networks, k-NN and random forests. Each model is implemented in two variants: two-class, which predicts whether jamming is launched or not, and five-class, which detects jamming presence and classifies its type. In our experiments, we evaluate attacks only against a small two-class fully-connected neural network. The datasets and models used for the use cases selected are summarised in Table 1.

Use case	Evaluation metric	Value
Modulation recognition	Prediction accuracy	55.17%
Channel estimation	Relative sum-rate	99.02%
Beam selection (subset)	Relative sum-rate	98.21%
Beam selection (sub-6GHz)	Relative sum-rate	87.04%
Channel decoding (convolutional)	Bit error rate	4.09%
Channel decoding (LDPC)	Bit error rate	7.81%
Channel decoding (polar)	Bit error rate	10.94%
Jamming detection	Prediction accuracy	100%

TABLE 2: Metrics for attack evaluation.

Once the datasets necessary have been obtained, each of them is divided into three parts: training (50%), validation (20%) and inference (30%). The training parts are used to train the corresponding AI/ML models, whereas the main function of the validation parts is to control the models' overfitting. The inference parts are then used to evaluate the models. Each trainable layer in the neural network models trained is followed by a dropout layer in order to reduce overfitting. In addition, early stopping is employed in order to stop training when the validation loss starts increasing. Speaking of the loss function, standard categorical cross-entropy is used for the classification models at the modulation recognition and beam selection use cases whereas the mean absolute error and mean squared error are used for training the channel estimation and the channel decoding models respectively. In all the cases, the training is carried out in batches of 512 with learning rate 0.0025.

C. ATTACKS AND METRICS

In our experiments, we test several white-box and black-box attacks from the adversarial example generation frameworks: Cleverhans [89], Adversarial Robustness Toolbox (ART) [17], Foolbox [16] and Advbox [20]. The list of the algorithms available in the frameworks mentioned in addition to already described FGSM [1], BIM [49], CW [50] and DeepFool [51] include implementations for such powerful white-box attack algorithms as projected gradient descent (PGD) [90], momentum iterative method (MIM) [91], NewtonFool [92], Jacobian-based saliency map attack (JSMA) [4], and ElasticNet attack [93]. Speaking of the black-box attacks other than aforementioned boundary and HopSkipJump, we employ the following attack algorithms against the target models selected: simultaneous perturbation stochastic approximation (SPSA) [94], the attack based on genetic algorithm (GenAttack) [95], ZOO [96], simple black-box adversarial (SimBA) [97] and Square attack [98]. We test the aforementioned attacks with three different perturbation

budget sizes that are relative to the strength of the input signal and equal to $||X||$, $0.1||X||$ and $0.01||X||$, where $||X||$ is the average second norm for the samples of the corresponding dataset.

The attack efficiency is evaluated based on the effect it produces upon the component that uses the target AI/ML model under attack. In the modulation recognition and jamming detection use cases, the attack efficiency is evaluated by comparing the prediction accuracy before and after the attack has been carried out. The less accurate the target model predictions once the attack has been carried out the more efficient the attack algorithm. In the channel estimation and beam selection use cases, the evaluation metric is the theoretical achievable data sum-rate. In the channel estimation use case, the sum-rate is calculated as $r = \log_2(1+H^{*T}b)$, where H is the true channel matrix and b is the weight calculated via conjugated beamforming based on the channel estimations $\hat{H}$ derived with the target AI/ML model as follows: $b = \frac{1}{diag(\hat{H}^{*T}\hat{H})}\hat{H}$ as described in the original study [55]. In the beam selection use case, the data rate is calculated as $r = \log_2(1 + H^{*T}f)$, where H is the channel matrix and f is a column vector from the codebook F described in the previous section. In both cases, the sum-rate is calculated relative to the perfect metric value, i.e. when the channel is estimated with 100 % accuracy and the true best beam selected for communication. The less the relative sum-rate for a perturbed sample the more efficient the corresponding attack algorithm. Finally, in the channel decoding use cases, the difference in the bit error rate (BER) acts as the attack efficiency metric as the adversary aims to increase the BER value. Table 2 summarises the metrics used for attack evaluation and their values obtained when applied the target models trained to the inference parts of the datasets at each of the use cases selected.

D. RESULTS AND DISCUSSION

All the attack evaluation results for the use cases selected can be respectively found in Tables 3-10 and Figures 1-16. In the tables, values of the metric selected are compared to each other for different attacks and adversarial perturbation budget values. Each table is divided into three parts, the first of which shows the effect of a perturbation generated randomly in the result of a jamming attack. The second and the third parts correspond to white-box and black-box adversarial example generation attacks respectively. All the values presented are calculated as percentages of the baseline metric value when there is no attack that can be found in Table 2. As one can see, adversarial example attacks impact on the target model performance is much more significant compared to the random perturbations. At the same time, white-box algorithms provide for noticeably better results on average, even though the black-box algorithms tested are not far behind for high values of the perturbation budget. Below we summarise the attack evaluation results for each use case.

As one can notice when looking at the results presented in Table 3 as well as Figure 1, in the modulation recognition use case, the most efficient white-box attack algorithms for high values of the perturbation budget are PGD and BIM, which are essentially two variations of the same algorithm in which a gradient step is taken in the direction of the greatest loss and then the resulting perturbation is projected into the ball with the centre at the original sample and the radius equal to the power budget available to the attacker [90]. For the lowest perturbation size, CW attack provides for the highest impact on the target metric. As mentioned in Section II, this is a minimization attack algorithm similar to the L-BFGS attack, i.e. it aims to find the minimal perturbation

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00176991	-0.02831858	-0.75398230
FGSM	-0.15044248	-0.79469027	-0.94690265
PGD	-0.20884956	-0.95398230	**-1.00000000**
BIM	-0.20707965	**-0.96283186**	**-1.00000000**
MIM	-0.04070796	-0.57876106	-0.88849558
CW	**-0.21415929**	-0.94513274	**-1.00000000**
JSMA	-0.01061947	-0.12035398	-0.85132743
DEEPFOOL	-0.20000000	-0.87079646	-0.98761062
NEWTONFOOL	-0.13982301	-0.66194690	-0.72566372
ELASTICNET	-0.07610619	-0.48849558	-0.99646018
SPSA	-0.00530973	-0.04955752	-0.75221239
GEN	-0.00884956	-0.07256637	-0.77345133
HOPSKIPJUMP	**-0.06902655**	**-0.64955752**	-0.91858407
BOUNDARY	-0.05663717	-0.55398230	**-0.92743363**
ZOO	-0.00884956	-0.06725664	-0.79469027
SIMBA	-0.00530973	-0.05132743	-0.74336283
SQUARE	-0.00530973	-0.04955752	-0.74690265

TABLE 3: The detrimental effect of the adversarial perturbation on the modulation recognition accuracy for different attack algorithms and perturbation budget values (the less the value the more efficient the attack).

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	-0.00000124	-0.00017702	-0.10716894
FGSM	-0.00070614	-0.01926888	**-0.84461592**
PGD	-0.00073591	-0.03383005	-0.21015514
BIM	**-0.00073630**	-0.03677270	-0.37127251
MIM	-0.00073607	**-0.04176789**	-0.81183656
CW	-0.00066852	-0.01888006	-0.02988577
JSMA	-0.00016670	-0.00450827	-0.03356486
DEEPFOOL	-0.00067968	-0.01785903	-0.83505510
NEWTONFOOL	-0.00065657	-0.01831213	-0.02263343
ELASTICNET	-0.00024949	-0.00607235	-0.05657986
SPSA	-0.00000345	-0.00016276	-0.26241590
GEN	-0.00001047	-0.00103772	**-0.64394562**
HOPSKIPJUMP	-0.00011782	-0.01295305	-0.02742713
BOUNDARY	**-0.00015525**	**-0.01413878**	-0.01629002
ZOO	-0.00008233	-0.00171015	-0.06725237
SIMBA	0.00000373	0.00003059	0.00008974
SQUARE	-0.00000016	-0.00038680	-0.21063616

TABLE 4: The detrimental effect of the adversarial perturbation on the average sum-rate with the beam calculated based on the channel estimations for different attack algorithms and perturbation budget values (the less the value the more efficient the attack).

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00000000	0.00031647	-0.01184631
FGSM	-0.00210116	-0.04235788	-0.51644550
PGD	-0.00236081	-0.06897709	**-0.59297667**
BIM	-0.00236081	**-0.06898766**	-0.52865036
MIM	-0.00028415	-0.05717071	-0.57654884
CW	-0.00210116	-0.04578858	-0.45842743
JSMA	0.00000000	0.00034313	-0.21801578
DEEPFOOL	**-0.00237815**	-0.04978454	-0.48192342
NEWTONFOOL	-0.00000786	-0.05183348	-0.42880927
ELASTICNET	0.00000000	-0.00956221	-0.43331060
SPSA	0.00000000	-0.00019772	-0.14912850
GEN	0.00000000	-0.00203163	-0.02978095
HOPSKIPJUMP	0.00000000	-0.04307449	**-0.84302100**
BOUNDARY	**-0.00026460**	**-0.04565825**	-0.82165714
ZOO	-0.00005614	0.00008188	-0.18433698
SIMBA	0.00000000	0.00002999	-0.14791272
SQUARE	0.00000000	-0.00019481	-0.07379259

TABLE 5: The detrimental effect of the adversarial perturbation on the average sum-rate with the beam selected using the beam subset for different attack algorithms and perturbation budget values (the less the value the more efficient the attack).

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00251647	-0.00814643	-0.35613440
FGSM	-0.05190932	-0.55312208	-0.86470818
PGD	-0.06856508	-0.59341468	-0.85622746
BIM	**-0.06893219**	**-0.63114235**	-0.82954778
MIM	-0.01820554	-0.57098431	**-0.86487392**
CW	-0.04356013	-0.47813089	-0.64286774
JSMA	-0.00355423	-0.02379815	-0.43157811
DEEPFOOL	-0.05637219	-0.57340684	-0.78358361
NEWTONFOOL	-0.02406948	-0.52673829	-0.70463417
ELASTICNET	-0.00670147	-0.16613535	-0.63283885
SPSA	-0.00309886	-0.02568962	-0.54937794
GEN	-0.00029334	-0.02238263	-0.27463939
HOPSKIPJUMP	-0.01514570	**-0.48588337**	-0.66091992
BOUNDARY	**-0.01617411**	-0.22238425	-0.60739140
ZOO	0.00074514	-0.11269332	**-0.75748890**
SIMBA	0.00353810	-0.01950014	-0.54099457
SQUARE	0.00005900	-0.00272400	-0.61858067

TABLE 6: The detrimental effect of the adversarial perturbation on the average sum-rate with the beam selected using the sub-6GHz channels for different attack algorithms and perturbation budget values (the less the value the more efficient the attack).

size which results in a misclassification. Therefore, when the perturbation budget is limited to a low value, it is not a surprise that the minimization algorithms outperform the ones that move straight towards the direction of the greatest loss. In the case of black-box attacks, the HopSkipJump algorithm provides the best results for lower perturbation budget values. This is an iterative algorithm similar to the boundary attack, it starts from a point that is already adversarial and then at each iteration the following three steps are carried out: estimation of the gradient direction, step-size

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00142928	0.05764655	7.01429267
FGSM	0.08480232	0.98189609	10.30633682
PGD	**0.20533585**	2.61362550	**13.24964313**
BIM	0.20485951	**2.65316823**	11.41162420
MIM	0.05574083	1.19485470	3.36493586
CW	0.01953310	0.36207720	10.86803255
JSMA	0.00000000	0.02953792	10.75035741
DEEPFOOL	0.13434968	1.28060979	10.36398291
NEWTONFOOL	0.03001434	0.53168184	10.71319692
ELASTICNET	0.03668416	0.56646029	12.05383530
SPSA	0.01000481	0.05955217	8.57122421
GEN	0.01238686	0.06765127	**14.24059063**
HOPSKIPJUMP	0.00952838	0.06574564	10.73034797
BOUNDARY	**0.05526440**	**1.19866613**	10.74940437
ZOO	0.00000000	0.02286801	10.36064837
SIMBA	0.00000000	0.02953792	10.76465008
SQUARE	0.00000000	0.03001434	10.75512188

TABLE 7: The detrimental effect of the adversarial perturbation on the convolutional codes decoding average bit error rate for different attack algorithms and perturbation budget values (the greater the value the more efficient the attack).

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00349921	0.04023996	2.36290949
FGSM	0.03299184	0.38815304	5.70182519
PGD	**0.11322176**	**1.07273191**	6.21894523
BIM	**0.11322176**	1.07173214	5.49212724
MIM	0.03174208	0.38715327	1.05823549
CW	0.01374664	0.17870537	5.51162234
JSMA	0.00824798	0.08023002	0.10897278
DEEPFOOL	0.07273182	0.85053747	6.45588628
NEWTONFOOL	0.04173962	0.51662092	**7.26568371**
ELASTICNET	0.02974263	0.33116722	0.61659593
SPSA	0.00499877	0.03999006	3.23444152
GEN	0.01349665	0.13046742	**5.55411202**
HOPSKIPJUMP	0.01099731	0.07473136	3.29017748
BOUNDARY	**0.05198705**	**0.75031246**	3.01474660
ZOO	0.00324921	0.02449396	3.46263460
SIMBA	0.00199955	0.02899275	4.29717599
SQUARE	0.00199955	0.02699330	4.27943026

TABLE 8: The detrimental effect of the adversarial perturbation on the LDPC codes decoding average bit error rate for different attack algorithms and perturbation budget values (the greater the value the more efficient the attack).

search via geometric progression, and boundary search via a binary search. This algorithm is computationally expensive and takes a long time to execute. For this reason, during experiments we had to adjust its default parameters in order to obtain results in a reasonable amount of time. In the case of the highest perturbation size, the boundary attack algorithm outperforms other black-box alternatives as one can see also from Figure 2.

In the channel estimation use case, the most straightforward algorithm FGSM provides the best results in terms of the sum-rate decrease for the highest perturbation size value available as one can see from Table 4 and Figure 3. MIM

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00000000	0.01115449	2.27607362
FGSM	0.03067485	0.24149470	3.37534858
PGD	**0.05298383**	0.49302844	**5.07863915**
BIM	0.05242610	**0.49581707**	4.46012270
MIM	0.00948132	0.12827663	2.29726715
CW	0.00000000	0.00780814	3.26938093
JSMA	0.00000000	0.00000000	0.00000000
DEEPFOOL	0.01784718	0.15337423	3.38036810
NEWTONFOOL	0.00446180	0.04238706	3.60736196
ELASTICNET	0.00000000	0.00780814	0.01450084
SPSA	0.00446180	0.02342443	2.61349693
GEN	**0.00725042**	0.02398215	5.52147239
HOPSKIPJUMP	0.00167317	0.01728946	4.13943112
BOUNDARY	0.00223090	**0.04573341**	**5.61461238**
ZOO	0.00000000	0.00000000	0.00000000
SIMBA	0.00446180	-0.00557724	3.42219743
SQUARE	0.00000000	-0.00055772	-0.00055772

TABLE 9: The detrimental effect of the adversarial perturbation on the polar codes decoding average bit error rate for different attack algorithms and perturbation budget values (the greater the value the more efficient the attack).

Attack algorithm	Perturbation budget (% of the signal strength)		
	1	10	100
Random	0.00000000	-0.07812500	-0.46484375
FGSM	0.00000000	-0.12402344	**-1.00000000**
PGD	0.00000000	-0.43359375	-0.50585938
BIM	0.00000000	**-0.59082031**	**-1.00000000**
MIM	0.00000000	-0.12402344	-0.36230469
CW	0.00000000	-0.05761719	-0.44335938
JSMA	0.00000000	-0.00292969	**-1.00000000**
DEEPFOOL	0.00000000	-0.33398438	**-1.00000000**
NEWTONFOOL	**-0.44238281**	-0.44238281	-0.44238281
ELASTICNET	0.00000000	-0.05761719	-0.44335938
SPSA	**0.00000000**	-0.08203125	-0.41015625
GEN	**0.00000000**	-0.44238281	-0.44238281
HOPSKIPJUMP	**0.00000000**	**-0.59082031**	**-1.00000000**
BOUNDARY	**0.00000000**	-0.27343750	-0.50878906
ZOO	**0.00000000**	0.00000000	0.00000000
SIMBA	**0.00000000**	-0.00097656	-0.00097656
SQUARE	**0.00000000**	0.00000000	0.00000000

TABLE 10: The detrimental effect of the adversarial perturbation on the jamming detection accuracy for different attack algorithms and perturbation budget values (the less the value the more efficient the attack).

outperforms alternatives for the lower perturbation size value. This algorithm is based on the momentum method which is a technique for accelerating gradient descent algorithms by accumulating a velocity vector in the gradient direction of the loss function across iterations. The memorization of previous gradients helps to barrel through narrow valleys, small humps and poor local minimums or maximums. For the lowest perturbation budget value, BIM remains the most efficient algorithm. Concerning black-box attacks, the boundary algorithm again provides good results for lower perturbation sizes as shown in 4. In addition, the attack based on a genetic algorithm is the most efficient method when the perturbation budget is the highest. It is worth noticing that this algorithm requires the target classifier to return scores, i.e. probabilities of belonging to each of the classes [95]. This poses additional requirements on the information available to the adversary.

In the beam selection use case, PGD, BIM and MIM algorithms remain among the most efficient ones for higher budget values as can be seen in Tables 5 6 as well as Figures 5 and 7. For the lowest budget size in the case when the best beam is selected based on a subset of RSSs, the perturbation can be generated with the DeepFool algorithm. Similarly to CW, it is a minimization attack method, at each iteration of which the target classifier is linearized around the current point and the minimal perturbation of the linearized classifier is computed. Speaking of the attacks in black-box settings for this use case, HopSkipJump is again the most efficient one for the highest perturbation budget value as one can also notice from Figure 6. In the case, when the best beam is selected based on the RSS values obtained from sub-6GHz channels, ZOO algorithm outperforms analogues for the highest perturbation budget value. This black-box attack algorithm is based on the CW attack. However, in distinction from the CW algorithm, ZOO computes an approximate gradient using a finite difference method instead of actual back propagation on the targeted model, and solves the resulting optimization problem via zeroth order optimization. In the case of lower budgets, the boundary attack algorithm shows promising results in both of the beam selection use cases analysed.

In the case of attacks against the channel decoding models, according to the results presented in Table 7, 8 and 9 as well as Figure 9, 11 and 13, PGD and BIM are among the most efficient white-box algorithms as they allow for the highest bit error rate increase. As one can notice, when the size of the perturbation allowed is high enough, the effect of these algorithms is noticeable at each of the SNR levels considered. In the case of LDPC codes decoding, NewtonFool algorithm provides for the biggest drop in the BER value. This algorithm tries to decrease the probability of the original class by performing gradient descent with the step of this descent being calculated in a certain way. Speaking of the black-box attacks, in convolutional and LDPC channel cases, the genetic algorithm provides the best results for the highest perturbation budget value and the boundary attack algorithm is the most efficient one for lower perturbation sizes as can be seen from Figures 10 and 12. In the case of polar codes, the situation is the opposite: the genetic algorithm outperforms other algorithms in case of the lowest perturbation budget value whereas the boundary attack provides for the best results in case of lower perturbation sizes as one can also see from Figure 14.

Finally, in the jamming detection use case, several attack algorithms would allow an adversary to achieve 100% accuracy reduction in the case of the highest perturbation budget value. As one can see from Table 10, these algorithms are already mentioned in previous use cases: FGSM, BIM and DeepFool. Another algorithm which provides for the perfect

result is JSMA. This white-box algorithm requires evaluating the network's forward derivative in order to construct an adversarial saliency map that identifies the set of input features relevant to the adversary's goal. The adversary can use this saliency map to either reduce the probability of the true class or increase the probability of other classes. NewtonFool is the only white-box attack algorithm that allows to reduce the jamming detection accuracy. As one can notice from Figure 15, this reduction comes from the normal signals being misclassified as the ones related to jamming. Speaking of the black-box attacks, the HopSkipJump algorithm outperforms analogues in terms of the evaluation metric selected for higher perturbation size values. When the perturbation size is the lowest, none of the black-box attack algorithms allows the adversary to reduce the jamming detection accuracy.

V. CONCLUSION

In this study, we evaluated various adversarial example generation attacks against machine learning models which can be deployed in future 5G networks for intelligent modulation recognition, channel estimation, beam selection and channel decoding. First, we summarised each of the problems formulated and discussed the data generation process. After that, the AI/ML model training and evaluation procedures were overviewed. Finally, multiple white-box and black-box attacks using various adversarial perturbation budget values were employed against the target models and evaluated using the metrics selected.

Despite the significant negative impact the attacks tested may achieve when employed against the target AI/ML-based 5G network components selected, unless there is a serious flaw in the component security, the adversary should be able to neither have access to the exact inputs of the target model, due to the different channel and interference conditions, nor obtain the output label, since it is most of the time used internally by the model and it is not available to any other wireless node outside of the network. For these reasons, the adversary has the best chance to fool the target model by crafting an input-agnostic adversarial perturbation. For this reason, in our future work, we are planning to focus on algorithms for crafting universal input-agnostic adversarial perturbations that can be employed when the information about neither the user inputs to the model nor the resulting outputs is available to the attacker.

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.
[2] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," 2017.
[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.
[4] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," 2015.
[5] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," 2016.
[6] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017.
[7] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, and F. Roli, "Deep neural rejection against adversarial examples," 2020.
[8] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," 2018.
[9] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Proceedings 2018 Network and Distributed System Security Symposium*, 2018. [Online]. Available: http://dx.doi.org/10.14722/ndss.2018.23198
[10] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," 2017.
[11] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," 2017.
[12] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, *Efficient Defenses Against Adversarial Attacks*. New York, NY, USA: Association for Computing Machinery, 2017, p. 39–49. [Online]. Available: https://doi.org/10.1145/3128572.3140449
[13] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, R. Long, and P. McDaniel, "Technical report on the cleverhans v2.1.0 adversarial examples library," 2018.
[14] A. Nayebi and S. Ganguli, "Biologically inspired protection of deep networks from adversarial attacks," 2017.
[15] W. Brendel and M. Bethge, "Comment on "biologically inspired protection of deep networks from adversarial attacks"," 2017.
[16] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," 2018.
[17] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.0.0," 2019.
[18] G. W. Ding, L. Wang, and X. Jin, "advertorch v0.1: An adversarial robustness toolbox based on pytorch," 2019.
[19] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang, "Deepsec: A uniform platform for security analysis of deep learning model," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 673–690.
[20] D. Goodman, H. Xin, W. Yang, W. Yuesheng, X. Junfeng, and Z. Huan, "Advbox: a toolbox to generate adversarial examples that fool neural networks," *arXiv preprint arXiv:2001.05574*, 2020.
[21] C. Zhang, P. Benz, C. Lin, A. Karjauv, J. Wu, and I. S. Kweon, "A survey on universal adversarial attack," 2022.
[22] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," 2019.
[23] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in ofdm systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, 2018.
[24] F. Meng, P. Chen, L. Wu, and X. Wang, "Automatic modulation classification: A deep learning enabled approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 760–10 772, 2018.
[25] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.
[26] T. S. Cousik, V. K. Shah, T. Erpek, Y. E. Sagduyu, and J. H. Reed, "Deep learning for fast and reliable initial access in ai-driven 6g mmwave networks," 2021.
[27] Z. Lu and M. C. Gursoy, "Dynamic channel access and power control via deep reinforcement learning," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.
[28] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti, "Cellular network traffic scheduling with deep reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11339
[29] J. Suarez-Varela, A. Mestres, J. Yu, L. Kuang, H. Feng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Routing in optical transport networks with deep reinforcement learning," *Journal of Optical Communications and Networking*, vol. 11, no. 11, pp. 547–558, 2019.
[30] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5g radio access network slicing," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6.
[31] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach," in *2018*

IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). IEEE, 2018, pp. 1–6.

[32] Y. E. Sagduyu, T. Erpek, and Y. Shi, "Adversarial machine learning for 5g communications security," 2021.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[34] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014.

[37] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[39] R. Bellman, *Dynamic Programming*. Dover Publications, 1957.

[40] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[41] A. Nassar and Y. Yilmaz, "Deep reinforcement learning for adaptive network slicing in 5g for intelligent vehicular systems and smart cities," 2020.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[43] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. Citeseer, 1994, vol. 37.

[44] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2017.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[46] J. Ye and Y.-J. A. Zhang, "Drag: Deep reinforcement learning based base station activation in heterogeneous networks," 2018.

[47] M. Kozlowski, R. McConville, R. Santos-Rodriguez, and R. Piechocki, "Energy efficiency in reinforcement learning for wireless sensor networks," 2018.

[48] Y. Liu, J. Ding, and X. Liu, "Resource allocation method for network slicing using constrained reinforcement learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–3.

[49] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2017.

[50] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017.

[51] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016.

[52] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," 2017.

[53] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[54] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 ieee symposium on security and privacy (sp)*. IEEE, 2020, pp. 1277–1294.

[55] M. Alrabeiah and A. Alkhateeb, "Deep learning for tdd and fdd massive mimo: Mapping channels in space and frequency," in *2019 53rd asilomar conference on signals, systems, and computers*. IEEE, 2019, pp. 1465–1470.

[56] M. S. Safari, V. Pourahmadi, and S. Sodagari, "Deep ul2dl: Data-driven channel knowledge transfer from uplink to downlink," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 29–44, 2020.

[57] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive mimo csi feedback," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, 2018.

[58] S. Peng, H. Jiang, H. Wang, H. Alwageed, and Y.-D. Yao, "Modulation classification using convolutional neural network based deep learning model," in *2017 26th Wireless and Optical Communication Conference (WOCC)*, 2017, pp. 1–5.

[59] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014. [Online]. Available: http://arxiv.org/abs/1404.5997

[60] W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang, "Performance evaluation of channel decoding with deep neural networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[61] K. Yashashwi, D. Anand, S. R. B. Pillai, P. Chaporkar, and K. Ganesh, "Mist: A novel training strategy for low-latency scalable neural net decoders," 2019.

[62] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tujkovic, "Deep learning coordinated beamforming for highly-mobile millimeter wave systems," *IEEE Access*, vol. 6, pp. 37 328–37 348, 2018.

[63] B. Kim, Y. Shi, Y. E. Sagduyu, T. Erpek, and S. Ulukus, "Adversarial attacks against deep learning based power control in wireless communications," 2021.

[64] L. Sanguinetti, A. Zappone, and M. Debbah, "Deep learning power allocation in massive mimo," 2019.

[65] C. Gutterman, E. Grinshpun, S. Sharma, and G. Zussman, "Ran resource usage prediction for a 5g slice broker," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 231–240. [Online]. Available: https://doi.org/10.1145/3323679.3326521

[66] Q. Guo, R. Gu, Z. Wang, T. Zhao, Y. Ji, J. Kong, R. Gour, and J. P. Jue, "Proactive dynamic network slicing with deep learning based short-term traffic prediction for 5g transport network," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, 2019, pp. 1–3.

[67] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[68] L. Fernández Maimó, A. L. Perales Gómez, F. J. Garcia Clemente, M. Gil Pérez, and G. Martínez Pérez, "A self-adaptive deep learning-based system for anomaly detection in 5g networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018.

[69] J. Lam and R. Abbas, "Machine learning based anomaly detection for 5g networks," 2020.

[70] M. Hachimi, G. Kaddoum, G. Gagnon, and P. Illy, "Multi-stage jamming attacks detection using deep learning combined with kernelized support vector machine in 5g cloud radio access networks," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–5.

[71] J. Pawlak, Y. Li, J. Price, M. Wright, K. Al Shamaileh, Q. Niyaz, and V. Devabhaktuni, "A machine learning approach for detecting and classifying jamming attacks against ofdm-based uavs," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, 2021, pp. 1–6.

[72] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," 2017.

[73] Q. Liu, J. Guo, C.-K. Wen, and S. Jin, "Adversarial attack on dl-based massive mimo csi feedback," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 230–235, 2020.

[74] M. Usama, R. N. Mitra, I. Ilahi, J. Qadir, and M. K. Marina, "Examining machine learning for 5g and beyond through an adversarial lens," 2020.

[75] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Over-the-air adversarial attacks on deep learning based modulation classifier over wireless channels," 2020.

[76] B. Kim, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, and S. Ulukus, "Adversarial attacks with multiple antennas against deep learning-based modulation classifiers," 2020.

[77] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," 2021.

[78] M. Sadeghi and E. G. Larsson, "Physical adversarial attacks against end-to-end autoencoder communication systems," 2019.

[79] B. Kim, Y. E. Sagduyu, T. Erpek, and S. Ulukus, "Adversarial attacks on deep learning based mmwave beam prediction in 5g and beyond," 2021.

[80] E. Catak, F. O. Catak, and A. Moldsvor, "Adversarial machine learning security problems for 6g: mmwave beam prediction use-case," 2021.

[81] M. L. Psiaki and T. E. Humphreys, "Gnss spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[82] B. R. Manoj, M. Sadeghi, and E. G. Larsson, "Adversarial attacks on deep learning based power allocation in a massive mimo network," 2021.

[83] Y. Shi and Y. E. Sagduyu, "Adversarial machine learning for flooding attacks on 5g radio access network slicing," 2021.

[84] D. Wu, B. Fang, J. Wang, Q. Liu, and X. Cui, "Evading machine learning botnet detection models via deep reinforcement learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.
[85] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 2016, pp. 213–226.
[86] M. Alrabeiah and A. Alkhateeb, "Deep learning for mmwave beam and blockage prediction using sub-6 ghz channels," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5504–5518, 2020.
[87] T. O'Shea and N. West, "Radio machine learning dataset generation with gnu radio," *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/11
[88] A. Alkhateeb, "Deepmimo: A generic deep learning dataset for millimeter wave and massive mimo applications," 2019.
[89] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy *et al.*, "Technical report on the cleverhans v2. 1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2016.
[90] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.
[91] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9185–9193.
[92] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 262–277.
[93] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," 2018.
[94] J. Uesato, B. O'Donoghue, A. van den Oord, and P. Kohli, "Adversarial risk and the dangers of evaluating against weak attacks," 2018.
[95] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1111–1119.
[96] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.
[97] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2484–2493.
[98] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," in *European Conference on Computer Vision*. Springer, 2020, pp. 484–501.

MIKHAIL ZOLOTUKHIN received the M.S. degrees in applied mathematics and computer science from the St. Petersburg State University, Russia in 2009 and the Ph.D. degree in computer science from the University of Jyväskylä, Finland, in 2014.

Since 2014, he has been a project researcher with the Faculty of Information Technology in the University of Jyväskylä. His research interests include application of reinforcement learning to network security, software-defined networking, network function virtualization, multi-hop mobile networks, and game theory. He is the author of more than 35 publications devoted to these research problems which include journal and conference articles as well as several book chapters.

Dr. Zolotukhin is also currently working as a researcher in Magister Solutions Ltd. His research there focuses on adversarial machine learning algorithms and transferability of these attack algorithms to the domain of the next-generation wireless networks. He also participates in implementation of a prototype of the fuzzing framework for generating adversarial examples against machine learning components of future 5G networks.

PARSA MIRAGHAIE received a B.S degree in electrical engineering with a minor in the control field from K. N. Toosi University of Technology, Tehran, Iran, in 2019.

From 2019 to 2021, he was Research Assistance at the Institute of Computational Vision Research Laboratory at Shahid Rajaee Teacher Training University, Tehran, Iran. He is currently pursuing his Master's degree in Human-Technology Interaction at the Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland.

Mr. Miraghaie is currently also working in Magister Solutions Ltd. as a researcher. His research interests include computer vision, cognitive machine learning, deep learning in telecommunication, and security vulnerabilities of artificial intelligence and machine learning systems.

DI ZHANG received the B.S. degree in 2011 and M.S. degree in 2013 from Northeastern University, Shenyang, China and Ph.D. degree in wireless communication from University of Jyväskylä, Finland in 2018.

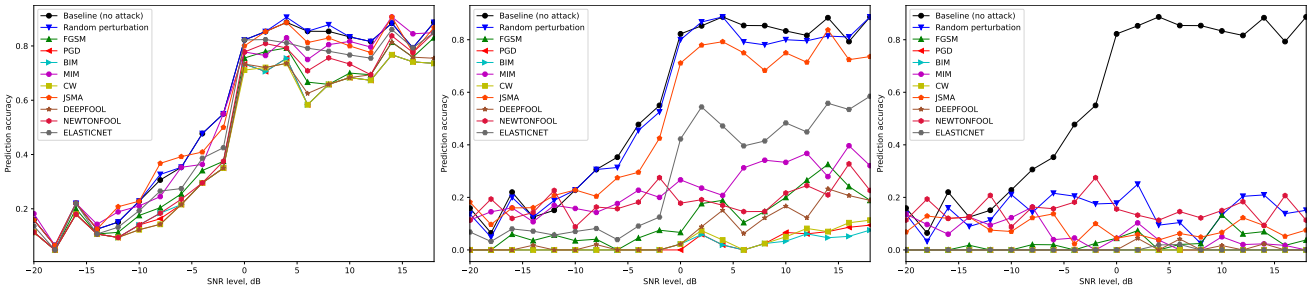
From 2018 to 2020, she was with Huawei Technologies Ltd, Shanghai, China as a standardisation researcher. Since 2020, she has been a postdoctoral researcher in University of Jyväskylä, Finland. Her research interests include machine learning, network security, beyond 5G communication, satellite communication.

Dr. Zhang is currently also working as a researcher in Magister Solutions Ltd. where her research focuses on applications of artificial intelligence and machine learning in the next generation mobile networks.

TIMO HÄMÄLÄINEN received the B.S. degree in automation from JAMK University of Applied Sciences, Jyväskylä, Finland, in 1991, the M.S. degree in telecommunication from Technical University of Tampere, Tampere, Finland, in 2006, and the Ph.D. degree in telecommunication from the University of Jyväskylä, Jyväskylä, Finland, in 2002.

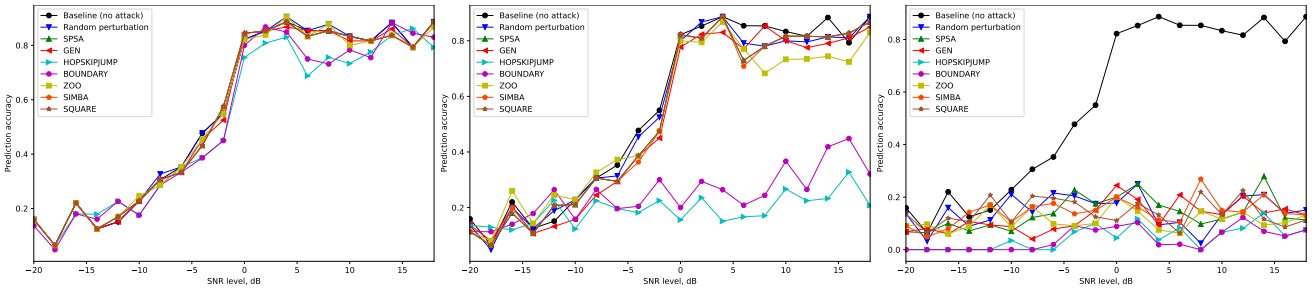
Since 2005, he has been a Professor with the Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland. He has more than 200 internationally peer reviewed publications and he has supervised more than 35 Ph.D theses. His research interests include performance evaluation and management of telecommunication networks, and in particular resource allocation, quality-of-service, anomaly detection and network security.

Prof. Hämäläinen is leading a research group in the area of network resource management and anomaly detection at the Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland. He is a board member of multiple journals and IEEE conferences in this research area.



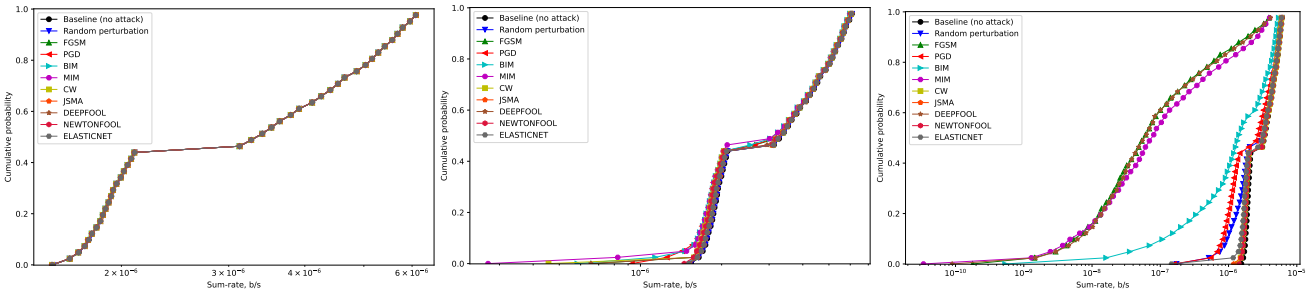
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 1: Dependence of the modulation recognition accuracy on the SNR level when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



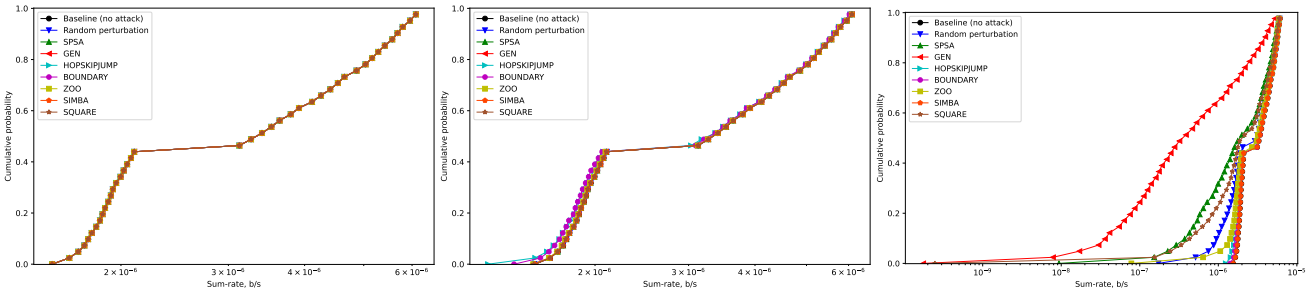
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 2: Dependence of the modulation recognition accuracy on the SNR level when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



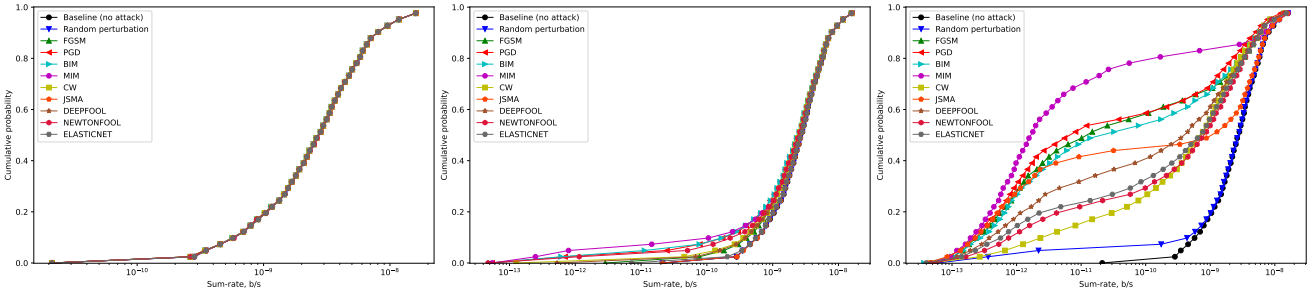
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 3: Dependence of the sum-rate CDF on the beam calculated based on the channel estimations when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



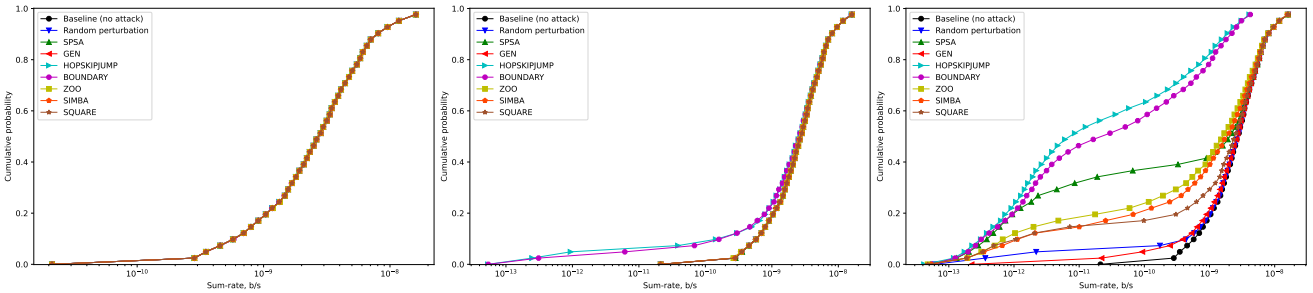
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 4: Dependence of the sum-rate CDF on the beam calculated based on the channel estimations when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



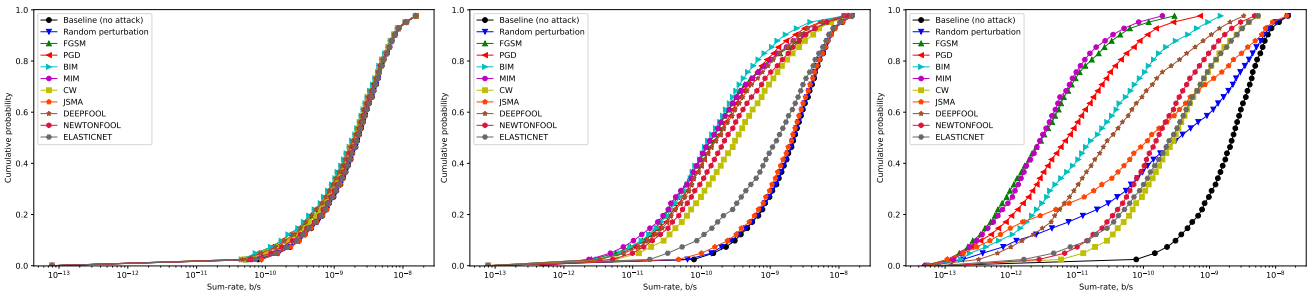
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 5: Dependence of the sum-rate CDF on the beam selected based on the beam subset when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



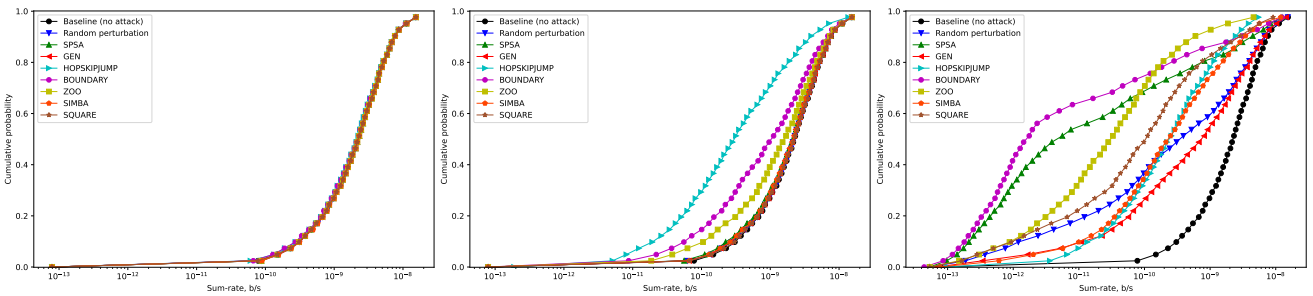
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 6: Dependence of the sum-rate CDF on the beam selected based on the beam subset when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



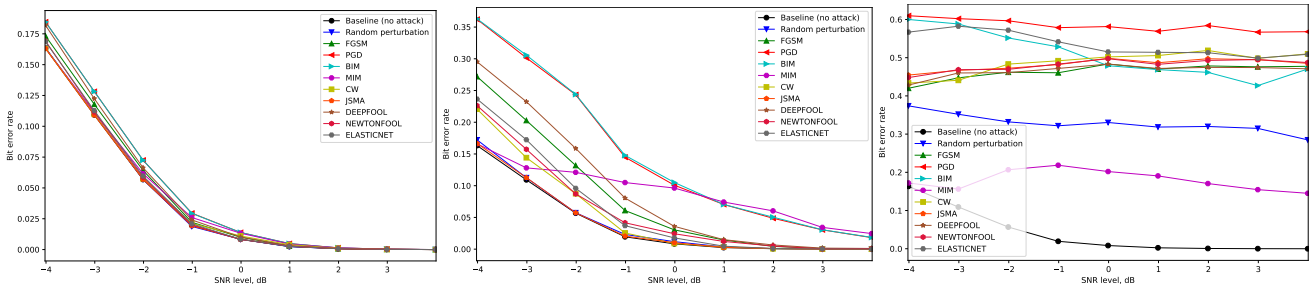
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 7: Dependence of the sum-rate CDF on the beam selected based on the sub-6GHz channels when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



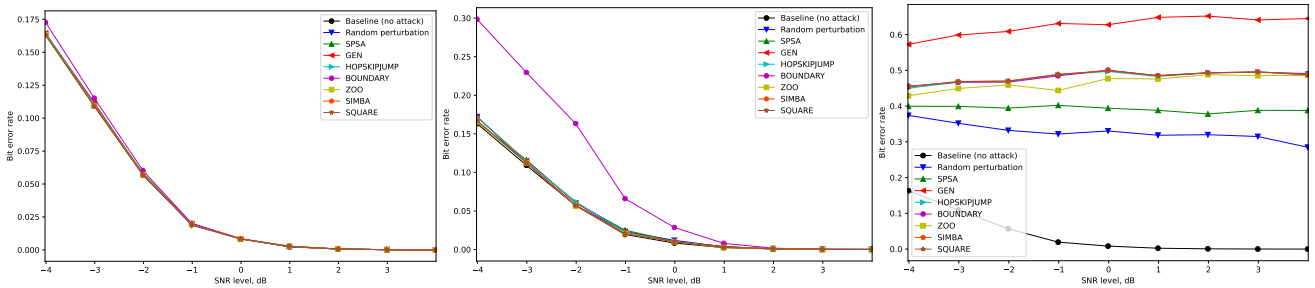
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 8: Dependence of the sum-rate CDF on the beam selected based on the sub-6GHz channels when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



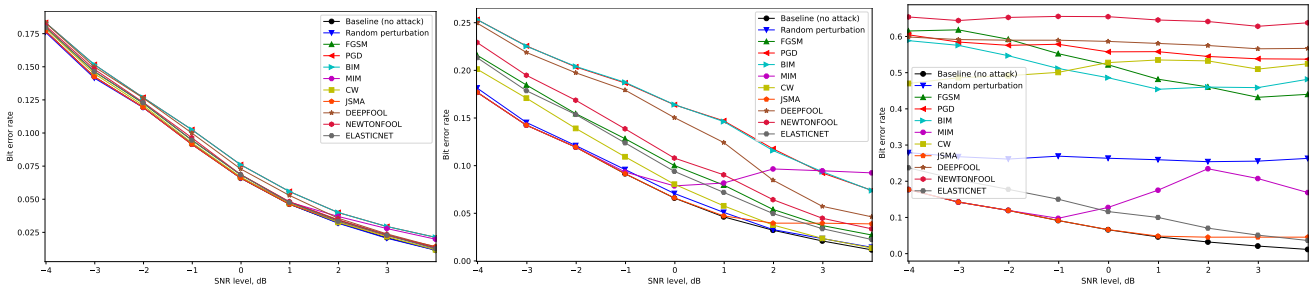
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 9: Dependence of the convolutional codes decoding BER value on the SNR level when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



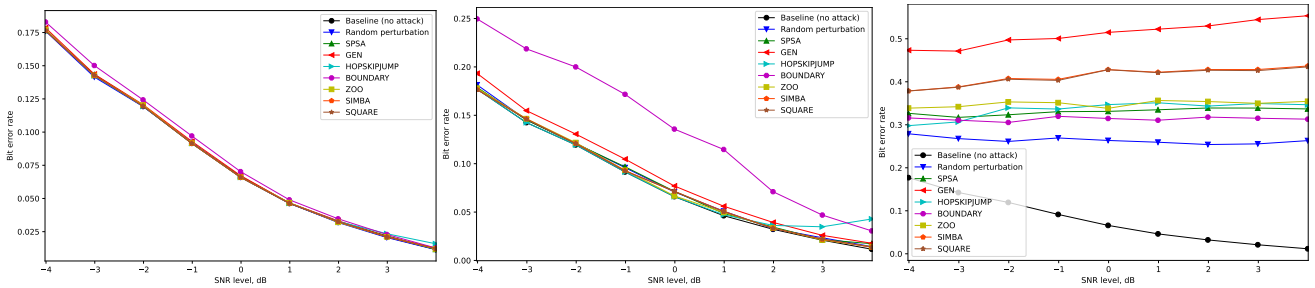
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 10: Dependence of the convolutional codes decoding BER value on the SNR level when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



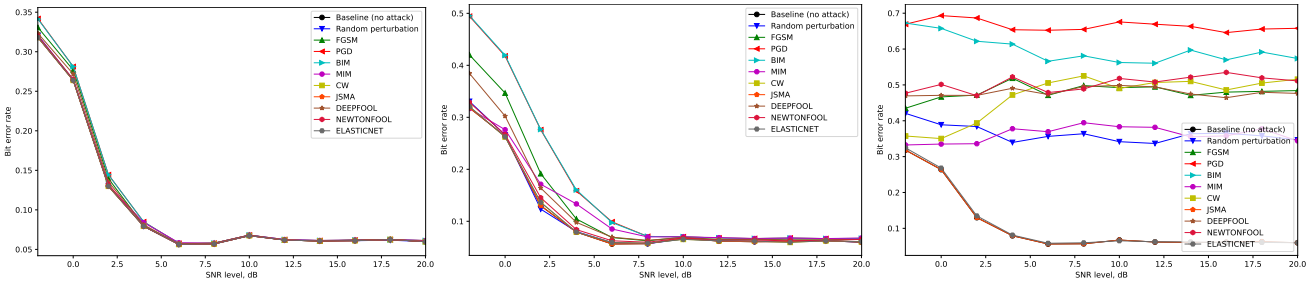
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 11: Dependence of the LDPC codes decoding BER value on the SNR level when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



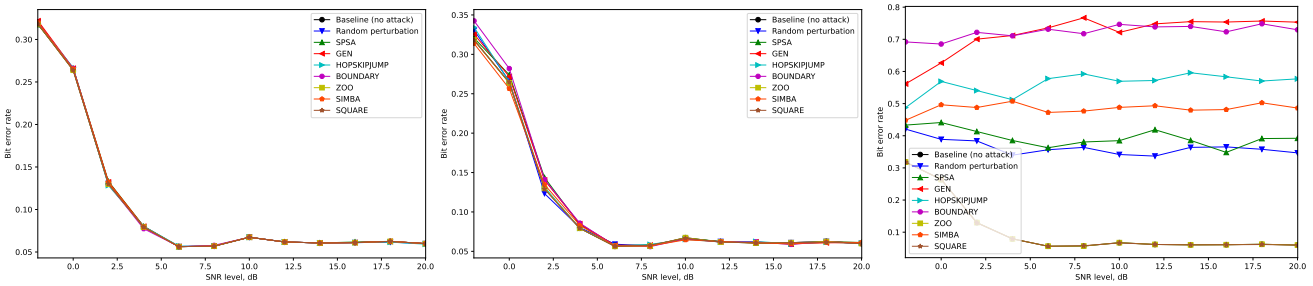
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 12: Dependence of the LDPC codes decoding BER value on the SNR level when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



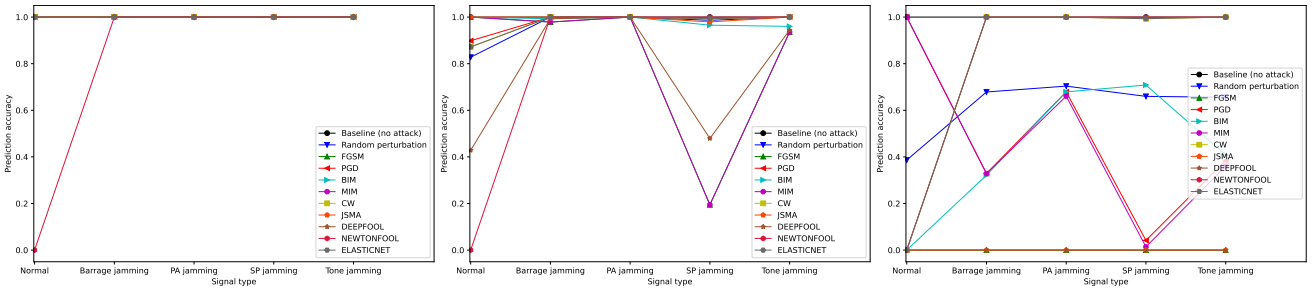
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 13: Dependence of the polar codes decoding BER value on the SNR level when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



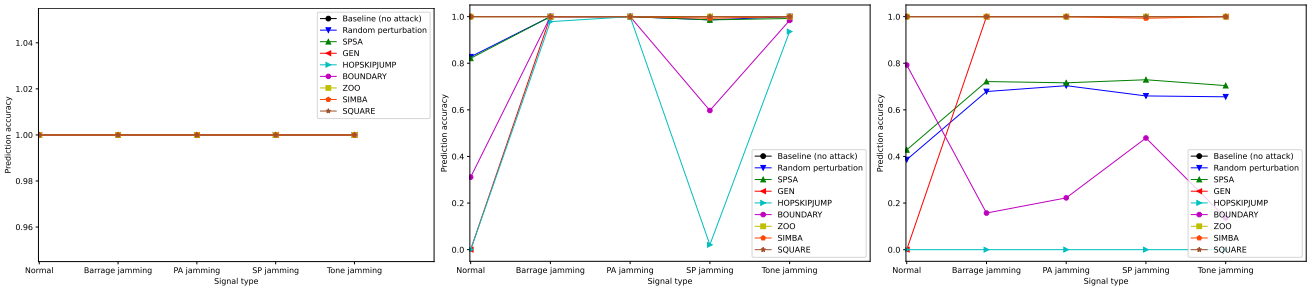
(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 14: Dependence of the polar codes decoding BER value on the SNR level when employing various black-box attack algorithms with different adversarial perturbation budget limit values.



(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 15: Dependence of the jamming detection accuracy on the signal type when employing various white-box attack algorithms with different adversarial perturbation budget limit values.



(a) 1% of the input signal strength. (b) 10% of the input signal strength. (c) 100% of the input signal strength.

FIGURE 16: Dependence of the jamming detection accuracy on the signal type when employing various black-box attack algorithms with different adversarial perturbation budget limit values.