Defending Adversarial Attacks on Deep Learning-Based Power Allocation in Massive MIMO Using Denoising Autoencoders

Rajeev Sahay, Graduate Student Member, IEEE, Minjun Zhang, David J. Love, Fellow, IEEE, and Christopher G. Brinton, Senior Member, IEEE

Abstract—Recent work has advocated for the use of deep learning to perform power allocation in the downlink of massive MIMO (maMIMO) networks. Yet, such deep learning models are vulnerable to adversarial attacks. In the context of maMIMO power allocation, adversarial attacks refer to the injection of subtle perturbations into the deep learning model’s input, during inference (i.e., the adversarial perturbation is injected into inputs during deployment after the model has been trained) that are specifically crafted to force the trained regression model to output an infeasible power allocation solution. In this work, we develop an autoencoder-based mitigation technique, which allows deep learning-based power allocation models to operate in the presence of adversaries without requiring retraining. Specifically, we develop a denoising autoencoder (DAE), which learns a mapping between potentially perturbed data and its corresponding unperturbed input. We test our defense across multiple attacks and in multiple threat models and demonstrate its ability to (i) mitigate the effects of adversarial attacks on power allocation networks using two common precoding schemes, (ii) outperform previously proposed benchmarks for mitigating regression-based adversarial attacks on maMIMO networks, (iii) retain accurate performance in the absence of an attack, and (iv) operate with low computational overhead. Code is publicly available at https://github.com/Jess-jpg-txt/DAE_for_adv_attacks_in_MIMO.

Index Terms—Adversarial attacks, deep learning, denoising autoencoder, massive MIMO, wireless security.

I. INTRODUCTION

M ASSIVE multiple-input-multiple-output (maMIMO) systems have incurred exciting developments in recent years, both in theory [1] and in practice [2]. A maMIMO system uses a wireless network topology where each base station (BS) is equipped with a large number of antennas to serve a multitude of user equipments (UEs) by using the spatial degrees-of-freedom [3]. One major task of the BS in maMIMO networks is feasible power allocation to each serviced UE. Accurate power allocation from the BS to each UE is vital for efficient communication (e.g., to increase the sum rate and reduce operational costs) in maMIMO networks. Although the equal power allocation policy can equally distribute power to all UEs, it is far from optimal and cannot always meet the power allocation needs of each UE simultaneously. Thus, power allocation algorithms that can meet each UE’s power needs while staying within the BS’s total power budget are required.

A variety of maMIMO power allocation algorithms have been developed, given a specific precoder, using convex optimization and geometric programming [4], [5], [6], [7], [8], [9]. Yet, such methods often computationally costly, which cannot always be afforded in real-time systems [10]. To address this, recent work has demonstrated the effectiveness of power allocation in maMIMO networks, given a specific precoder, using deep learning, which can attain near-optimal performance by using the positions of UEs as input into a deep neural network (DNN) [11]. Moreover, deep learning is capable of improving the complexity-performance trade-off in comparison to traditional optimization-based methods while also eliminating the need to compute any statistical averages usually needed in standard methods [12], [13], [14].

The utilization of deep learning for wireless systems, however, has introduced a new class of malicious attacks. Specifically, deep learning-based power allocation models in maMIMO have been shown to be susceptible to gradient-based adversarial evasion attacks—subtle perturbations injected into a DNN’s input at inference to induce erroneous behavior at the output [15], [16], [17], [18]. In the context of maMIMO power allocation, erroneous model behavior refers to the adversarial perturbation inducing an infeasible power allocation to the UEs. Such attacks cause the DNN to unfairly allocate resources to certain UEs, thus hindering the effectiveness and efficiency of deep learning in maMIMO networks. Contrary to deep learning-based power allocation, traditional optimization-based power allocation techniques are not hindered specifically by adversarial attacks (although other perturbations may degrade their performance [19], [20], [21]) since there is no
model that can be used to craft a gradient-based attack. Despite this advantage, optimization-based methods still incur high computational overhead, which may not always be affordable in real-time [11], [12], [13]. Thus, mitigating adversarial attacks in DNN-based power allocation is needed to allow for both the computational efficiency and model robustness of deep learning in maMIMO systems.

In this work, we consider a denoising autoencoder (DAE) framework for mitigating the effects of adversarial attacks instantiated on DNN regression models for power allocation in maMIMO networks. We begin by developing our DAE, which is a layer-based DNN architecture trained to map perturbed adversarial samples to their corresponding clean (non-adversarial) representations. In addition, the DAE is also trained to map clean, unperturbed data to itself in order to retain performance on non-adversarial samples. During inference, the received sample is first forward propagated through the DAE, and the DAE output is then inputted into the DNN regression model as the BS. We evaluate, in terms of both robustness and computational efficiency, the feasibility of our method against multiple attack generation algorithms on two precoding schemes.

Formally, our contributions of this work can be summarized as follows:

1) **DAE output mitigation strategy against adversarial attacks** (Sections III, V, VI-B, and VI-C): We develop a DAE architecture capable of filtering adversarial perturbations out of maMIMO input data on deployed DNNs. At test time, we show that forward propagating the input (i.e., UE positions) through the DAE prior to inputting it into the DNN significantly mitigates the effects of adversarial attacks.

2) **Effectiveness of defense in a multitude of environments** (Sections IV and VI-A–VI-C): We demonstrate the effectiveness of our defense in both semi-white-box and black box threat models on maMIMO networks using both maximum ratio (MR) and multicell-minimum mean square error (MMMSE) precoding schemes.

3) **Robustness against first order \(l_\infty\)-bounded adversarial attacks** (Sections VI-B and VI-C): We show that our proposed DAE defense provides robustness against multiple first-order \(l_\infty\)-bounded adversarial attacks on regression-based power allocation DNNs in multiple threat models and on multiple precoding schemes.

4) **Computational Feasibility** (Section VI-D): We quantify the computational speedup provided by our proposed method over comparative benchmarks. Specifically, we find that our proposed framework does not incur higher computational overhead as the parameter space of the employed DNN increases.

The remainder of the paper is organized as follows. Section II discusses prior work on adversarial attacks and defenses instantiated on deep learning-based massive MIMO power allocation as well as other relevant deep learning-based communications tasks. Section III describes our considered setup for performing power allocation in massive MIMO networks using deep learning. Section IV discusses our adversarial attack formulation, considered adversarial attacks, and the adversary’s various knowledge levels. Section V details our denoising autoencoder (DAE) defense and implementation details. Finally, Section VI presents the results of our methodology’s empirical evaluation. We present concluding remarks in Section VII.

## II. RELATED WORK

Deep learning has recently achieved cutting edge performance in the domain of wireless communications. The majority of these applications have consisted of classification problems including, but not limited to, automatic modulation classification [22], sensing and localization [23], [24], and channel estimation [25]. Such tasks have extensively been shown to be vulnerable to adversarial attacks [26], [27], with a particular emphasis on automatic modulation classification [28], [29], [30], [31], [32]. Similarly, previous studies have proposed methods to craft effective wireless eavesdroppers [33], while other works have examined circumventing such eavesdroppers by intentionally transmitting adversarially perturbed data that the receiver can subtract from the observed signal [34], [35], [36]. Our work can be considered orthogonal to such approaches in that our proposed framework is (i) based on regression and not classification and (ii) aiming to directly induce erroneous behavior at the BS and not fool an eavesdropper.

As a result of the susceptibility of wireless signal classifiers to adversarial perturbations, several defenses have been proposed such as autoencoder pre-training [37], detection and rejection [38], and assorted deep ensembles [39]. Despite the strides made in defending signal classifiers from adversarial attacks, these methods are difficult to directly adopt for defending adversarial attacks on regression DNNs for the task of power allocation in the downlink of maMIMO because they are specifically tailored for classification tasks. In this work, we address this shortcoming by developing a defense that is capable of mitigating adversarial attacks on DNNs used for regression tasks in maMIMO networks.

Furthermore, in addition to the host of defenses studied in classification-based wireless communications, defenses against adversarial attacks have been investigated to a large extent in computer vision [40]. In particular, DAEs have shown success in mitigating adversarial attacks for computer vision tasks such as image classification [41], [42]. However, DAEs have not been explored for mitigating adversarial attacks in regression-based computer vision tasks. For mitigating adversarial attacks in regression-based maMIMO networks, adversarial training [43], [44], [45] has been shown to effectively reduce the adversary’s success rate in white-box [46] and black box settings [47]. Despite its success, adversarial training remains vulnerable to high-bounded perturbations and, further, suffers from overfitting to adversarial examples, which, in turn, reduces performance on unperturbed data [48]. Our proposed DAE, on the other hand, remains effective at high perturbation magnitudes without significantly reducing baseline prediction performance.

Lastly, it is important to note that prior work has demonstrated the ineffectiveness of pre-processing defense mechanisms in defending adversarial attacks in computer
vision applications [49]. However, contrary to computer vision, adversaries in the context of mmMIMO networks will rarely have complete knowledge of each processing module employed at the BS [17], [50]. Thus, pre-processing defense mechanisms, as we will see, serve as effective mitigation techniques in the context of wireless communications, where the adversary is typically knowledge-limited while operating.

III. DEEP LEARNING FOR POWER ALLOCATION

In this section, we describe the mmMIMO power allocation objective formulated for the downlink and describe how to perform power allocation using DNNs.

A. Downlink Power Allocation

We begin by considering a mmMIMO system with $L$ cells, where each cell contains a BS with $M$ antennas and $K$ UEs per cell. The downlink signal transmitted from BS $j$ is given by

$$ y_{jk} = \left( h_{jk}^H \right) H f_k s_k + \sum_{i \neq k}^{K} \left( h_{jk}^H \right) H f_i s_i + \sum_{l \neq j}^{L} \sum_{i = 1}^{K} \left( h_{jk}^H \right) H f_i s_i + \sigma^2_n $$

where $s_k$ is the downlink data signal for UE $k$ in cell $j$, and $f_k \in \mathbb{C}^M$ is the precoding vector, which determines the directivity of the transmission. Furthermore, $\|f_k\|^2 = 1$ and thus $\rho_{jk}$ corresponds to the transmit power. Following (1) as the transmitted downlink signal, we denote the received signal of UE $k$ in cell $j$ as

$$ y_{jk} = \left( h_{jk}^H \right) H f_k s_k + \sum_{i \neq k}^{K} \left( h_{jk}^H \right) H f_i s_i + \sum_{l \neq j}^{L} \sum_{i = 1}^{K} \left( h_{jk}^H \right) H f_i s_i + \sigma^2_n $$

where $(\cdot)^H$ denotes the Hermitian operator, $h_{jk}^H$ is the channel between BS $l$ and UE $k$ in cell $j$ (i.e., the superscript of $h_{jk}^H$ corresponds to the transmit BS and the subscripts identify the cell and index of the receive UE), and $n_{jk} \sim \mathcal{C}\mathcal{N}(0, \sigma^2)$ is complex additive white Gaussian noise (AWGN) with variance $\sigma^2$.

We assume that the uplink and downlink channels are reciprocal within a coherence block and the UE can thus approximate the precoded channel through its expected value, $E[\left( h_{jk}^H \right) H f_k]$, where $E[\cdot]$ denotes the expectation operator. This is a deterministic value that the UE has access to (as assumed in [51]). The remaining precoded channels in (2) (i.e., the inter-cell and intra-cell interference) are treated as realizations of random variables whose expectations are taken at the UE with respect to different channel realizations. Each of these quantities are then used to calculate the signal-to-interference-plus-noise ratio (SINR) of (2) in the downlink for UE $k$ in cell $j$. The SINR, which is a deterministic scalar, is given by (see [51])

$$ \gamma_{jk}^{dl} = \frac{\rho_{jk} a_{jk}}{\sum_{l=1}^{L} \sum_{i=1}^{K} \rho_{li} b_{lijk} + \sigma^2}, $$

where $a_{jk} = E \left[ \left( f_{jk}^H h_{jk}^f \right)^2 \right]$, (4) and

$$ b_{lijk} = \begin{cases} E \left[ \left( f_{jk}^H h_{jk}^i \right)^2 \right], & (l, i) \neq (j, k) \\ E \left[ \left( f_{ij}^H h_{jk}^i \right)^2 \right] - E \left[ \left( f_{ij}^H h_{jk}^l \right)^2 \right], & (l, i) = (j, k) \end{cases}, $$

are the expected channel and interference gains, respectively.

Selecting the optimal precoder, $f_k$, is a challenging task since (3) depends on the precoders of all UEs in the mmMIMO network. Thus, following prior work (see [11, Sec. II-C]), we use the UL-DL duality and design the precoder based on the UL combining vector (used to detect the UL signal transmitted by the UE), where we consider MR combining [52] and MMSE combining [53]. Thus, throughout this work, we will consider the performance of our framework when the precoder is based on both MR and MMSE combining.

Using the above definitions of SINR, the optimal power allocation to UEs is the solution of the max-product policy given by [51]

$$ \max_{\rho_{jk} : \forall j, k} \prod_{j=1}^{L} \prod_{k=1}^{K} \gamma_{jk}^{dl} $$

where $P_{max}^{dl}$ is the downlink transmit power budget. The optimal power allocation solutions to (6) are obtained by computing (i) the large-scale fading coefficients and channel correlation matrices, (ii) the random channel vectors, (iii) $f_k$, $\{a_{jk}\}$, and $\{b_{lijk}\}$ and, lastly, (iv) allocating power to UEs to satisfy (6) [11], [51]. This algorithm is implemented using geometric programming, which requires polynomial complexity. However, such complexity may not be affordable in real-time systems. By calculating the optimal power allocation for UEs at various locations ahead of time, a mapping between UE positions and their optimal power allocation can be learned for faster real-time power allocation prediction.

B. Deep Learning Implementation

The objective of employing a DNN at the BS is to (i) learn the optimal power allocation, given by the max-product policy, in the downlink of a mmMIMO network and (ii) reduce the latency incurred from solving (6) when UE positions change in real-time. To do this, training data is first generated by solving the formulation given in (6) using geometric programming for faster real-time power allocation prediction.
also eliminates the need to solve (6) when the $K$ UEs have moved within a cell. As a result, the overall runtime during deployment can be reduced by up to 99.9% [13].

Although generating the training data requires polynomial computational complexity, since (6) has to be solved for each training sample, this can be done offline, and thus higher complexity can be afforded since real-time constraints do not apply. Specifically, UE positional data and their corresponding optimal power allocations obtained from solving (6) are assumed to be collected and stored in a central server where the DNNs are trained (corresponding to the offline development) before the trained models are deployed at the BSs for real-time power allocation (corresponding to the online deployment). Thus, the trained DNN eliminates the need to re-calculate the needed power for different sets of UEs as they move around the cells, since their new power requirement can be predicted by forward propagating the UE locations through the trained DNN at the BS, leading to significantly improving latency.

Formally, we define the $n$th input to the DNN as $x(n) \in \mathbb{R}^{2KL}$, where $x(n) = [x_1, y_1, \ldots, x_{KL}, y_{KL}]$ represents the geographical coordinates, in a Euclidean space, of each of the $(KL)$ UEs. The corresponding DNN target output is denoted by $\mathbf{p}_j(n) \in \mathbb{R}^{K+1}$, which is the optimal power allocation for each UE in cell $j$ obtained for training by solving (6) with knowledge of (4) and (5). Note that, in addition to the power allocation for each UE, $\mathbf{p}_j(n)$ also contains the power constraint $\sum_{k=1}^{K} \rho_{jk}$, where $\sum_{k=1}^{K} \rho_{jk} = P_{\text{max}}$ is always fixed, such that $\mathbf{p}_j(n) = [\rho_1, \ldots, \rho_K, \sum_{k=1}^{K} \rho_{jk}]$, in order to improve the DNN estimation ability during training and inference. In addition, each BS learns the mapping between UE positions, $x(n)$, and power allocation, $\mathbf{p}_j(n)$, for the individual cell $j$.

Next, we define $X_{\text{tr}} = \{x(n), \mathbf{p}_j(n); n = 1, \ldots, N\}$ as the training dataset containing $N$ samples. Using this dataset, at each BS, we train a DNN, $f_l(\cdot, \mathbf{\theta}_l) : \mathbb{R}^{2KL} \rightarrow \mathbb{R}^{K+1}$, parameterized by $\mathbf{\theta}_l$, to map each input sample, $x(n)$, (i.e., the position of each UE among all $L$ cells) to the appropriate power allocation vector, $\hat{\mathbf{p}}_j$, for each UE in the cell. Here, the inputs only contain the positional information of the UEs; the target label, containing the optimal power allocation vector obtained from (6), captures the remaining effects of the channel. The DNN takes input $(\cdot)$, and outputs the predicted power, $\hat{\mathbf{p}}_j$, to allocate to each UE in its corresponding cell, i.e., the estimated power allocation by the DNN to the UEs serviced by BS $j$ is given by $\hat{\mathbf{p}}_j(n) = f_l(x(n), \mathbf{\theta}_l)$. During evaluation, we employ the dataset $X_{\text{te}} = \{x(b), \mathbf{p}_j(b); b = 1, \ldots, B\}$, where $X_{\text{tr}} \cap X_{\text{te}} = \emptyset$ (i.e., no samples overlap in the training and evaluation datasets).

We evaluate our methodology on two considered DNN architectures. The first model, referred to as Model 1, is a custom architecture, which we found capable of learning the desired mapping between UEs and optimal power allocation. The second model, Model 2, is adopted from prior work [11], [15] that has shown it to be effective for power allocation in mmMIMO systems. Both considered models are shown in Table I and Table II, respectively. Each model is a fully connected neural network (FCNN) comprised of individual units, where each unit contains a set of trainable weights, whose input dimensionality is equal to the number of units in the preceding layer, and the number of units in each layer is an adjustable hyper-parameter. The output of the $i$th layer of the DNN, prior to applying the activation, is given by $z^{(i)} = W^{(i)}o^{(i−1)} + b^{(i)}$, where $W^{(i)}$ is the weight matrix of layer $i$, $o^{(i−1)}$ is the input vector to layer $i$ containing the output from the previous layer, and $b^{(i)}$ is a vector of threshold biases.

Each intermediate layer in both Model 1 and Model 2 applies the unit-wise exponential linear unit (ELU) activation function, thus resulting in the output of each unit in hidden layer $i$ given by $o^{(i)}(p) = \nu(z^{(i)}(p))$ where

$$\nu(z^{(i)}(p)) = \begin{cases} z^{(i)}(p), & z^{(i)}(p) \geq 0 \\ \exp(z^{(i)}(p)) - 1, & z^{(i)}(p) < 0 \end{cases}$$

and the output of the final layer applies the linear activation function resulting in $\nu(z^{(i)}(p)) = z^{(i)}(p)$, where $\nu(\cdot)$ denotes the activation function, $z^{(i)}(p)$ and $o^{(i)}(p)$ denote the $p$th unit of layer $i$ prior to and after applying the activation, respectively, and $z^{(i)}$ and $o^{(i)}$ are the output vectors of layer $i$ prior to and after applying the activation, respectively.

Using $X_{\text{tr}} = \{x(n), \mathbf{p}_j(n); n = 1, \ldots, N\}$, both Model 1 and Model 2 are trained by minimizing the relative mean squared error, given by

$$\ell(x(n), \mathbf{p}_j(n), \mathbf{\theta}_l) = \frac{1}{N} \sum_{n=1}^{N} \left( \frac{\mathbf{p}_j(n) - \hat{\mathbf{p}}_j(n)}{\mathbf{p}_j(n)} \right)^2,$$

using stochastic gradient descent (SGD) so that the model parameters on epoch $t + 1$ are updated according to

$$\mathbf{\theta}_l^{t+1} = \mathbf{\theta}_l^t - \eta \frac{\partial \ell(x(n), \mathbf{p}_j(n), \mathbf{\theta}_l)}{\partial \mathbf{\theta}_l},$$

1In practice, mini-batches are used to accelerate training.
where $\eta$ is the model learning rate and the parameters for the first epoch, i.e., $\theta^0$, are randomly initialized. Rather than using a fixed learning rate, our training process starts with a learning rate of $10^{-1}$ and gradually reduces to $10^{-4}$ as the model gets closer to convergence. When multiple consecutive epochs yield indistinguishable relative mean squared errors for their validation sets, we reduce the learning rate by a factor of 0.1 to ensure the model takes smaller steps towards convergence. We consider this training process for both Model 1 and Model 2 in the cases of both MR and MMMSE precoding.

IV. ADVERSARIAL ATTACKS ON POWER ALLOCATION

In this section, we will begin by introducing the adversarial attack formulation on the power allocation DNNs described in Section III. Then, we will discuss two approaches of crafting adversarial attacks for inducing infeasible power allocation solutions by the DNN. Finally, we will discuss the adversarial threat model, where we will state the various knowledge levels we consider the adversary to operate in and specify the available system information known by the adversary.

A. Attack Formulation

Adversarial attacks on DNNs can be crafted in a multitude of ways (as further discussed in Sections IV-B and IV-C) by introducing subtle perturbations into an input sample. In this work, we assume that the BS first obtains each UE’s positional information for power allocation. After obtaining the UE positions at the BS, and prior to its input to the DNN, an adversary crafts and injects a perturbation into the DNN input sample. Physically, an attack crafted to perturb an input in this fashion corresponds to changing the UEs current position to a new position by a small amount (on the scale of tens of meters). Such an attack can be instantiated via spoofing attacks by using global navigation satellite system (GNSS) receivers, where an adversary calculates a minimum distance perturbation that will result in erroneous outputs made by the DNN [54]. Furthermore, such attacks may also induce from position equivocation algorithms leading to inexact UE positional information at the BS.

Formally, the adversary aims to inject the $n^{th}$ sample with a perturbation, $\delta(n) \in \mathbb{R}^{2KL}$, thus yielding

$$\tilde{x}(n) = x(n) + \delta(n),$$

where $\tilde{x}(n)$ represents the perturbed adversarial sample. As the adversary is aiming to operate undetected, while simultaneously trying to degrade the DNN performance, an effective perturbation can be generated by solving

$$\min_{\delta(n)} ||\delta(n)||_{\infty}$$

subject to

$$\sum_{k=1}^{K} \hat{p}_{jk} > P_{\text{max}}^{dl},$$

$$||\delta(n)||_{\infty} \leq \epsilon,$$

$$x(n) + \delta(n) \in \mathbb{R}^{2KL},$$

where $||\cdot||_{\infty}$ is the $l_{\infty}$ norm and $\epsilon$ is the maximum allowed perturbation for each feature in $x(n)$ (i.e., $||\tilde{x}(n) - x(n)||_{\infty} \leq \epsilon$), thus directly corresponding to the positional displacement of the UE. Although (11a) specifically constrains the $l_{\infty}$ norm of $\delta(n)$, note that the adversary aims to inject its $\epsilon$-constrained perturbation into each feature (positional coordinate) of $x(n)$ (i.e., $\delta(n) \in \mathbb{R}^{2KL}$). The ultimate objective of this design, from the adversary’s perspective, is to force the DNN to output an infeasible power allocation solution as in (11b), while remaining in (i) an $\epsilon$-neighborhood of the original sample as in (11c) and (ii) the same dimensional space as the input as given by (11d). Furthermore, note that the perturbation is added after normalizing the sample and prior to inputting it to the DNN and, therefore, the normalization does not aid in mitigating the attack since the perturbation is added after normalization.

Due to the excessive nonlinearity of the considered DNN architectures as well as the non-convex nature of the adversary’s objective function, solving (11) directly is very challenging. Thus, several gradient-based adversarial attacks have been proposed in order to approximate a solution that ideally solves (11) in effective and computationally efficient ways [55], [56]. Since solutions to (11) are typically approximated, they may not always be effective (i.e., there may be designs of $\delta(n)$ for which certain constraints in (11) are not satisfied). Specifically, an adversarial attack is effective when the DNN outputs an infeasible power allocation to the UEs as a result of the perturbation and ineffective otherwise. Moreover, the adversary’s primary objective is to force an infeasible output by the DNN and, thus, an effective approximation to (11) may not necessarily be realized when $||\delta(n)||_{\infty}$ is minimized.

As shown in (11), we consider the class of adversarial attacks bounded by the $l_{\infty}$ norm constraint. However, adversarial attacks can be constrained by other $l_p$ norms as well such as when $p = 0$, $p = 1$, or $p = 2$. In this work we focus exclusively on $l_{\infty}$ norm since the input to the DNN is the position of the UE in a Euclidean space, and thus the $l_{\infty}$ norm corresponds to the maximum positional displacement of the UE. Note that the distance of the UE displacement for an attack bound of $\epsilon$ is given by $d_{\epsilon} = \sqrt{2}\epsilon$ since the perturbation displaces the UE along both dimensions of the Euclidean space.

We consider two adversarial attack algorithms that approximate solutions to (11): the fast gradient sign method (FGSM) [44] and projected gradient descent (PGD) [43]. The FGSM is a single step attack making it computationally efficient whereas PGD is iterative thus requiring higher complexity to implement but resulting in a more potent attack compared to the FGSM. Both attacks rely on adding a perturbation in the direction of some loss function’s gradient to achieve their objective. In order to maximize the probability of crafting an effective attack, and following in line with prior works [17], [47], we define the loss function used by the adversary to be

$$L(x(n), \theta) = \sum_{k=1}^{K} \hat{p}_{jk},$$

where $\hat{p}_{jk}$ corresponds to the predicted power to be allocated to UE $k$ in cell $j$ on the input sample $x(n)$. The adversary may
either use the DNN deployed at the BS to measure this loss, or they may alternatively use a surrogate DNN. The DNN used by the adversary will depend on the amount of available system knowledge, which is further discussed in Section IV-D. Next, we will discuss how the adversary uses (12) to craft \( \delta(n) \) using the FGSM and PGD algorithms. Each considered attack is applied cell-wise, where the adversary aims to degrade performance of the DNN in the cell serviced by the respective BS.

### B. Fast Gradient Sign Method (FGSM)

The FGSM is a single step method that is given by

\[
\delta(n) = \epsilon \text{sign} \left( \nabla_{x(n)} \mathcal{L}(x(n), \theta_l) \right),
\]

where \( \mathcal{L}(x(n), \theta_l) \) is the loss function used by the adversary to craft the perturbation and the model that is used to calculate the loss function’s gradient is parameterized by \( \theta_l \). Thus, the adversarial sample generated using the FGSM targeted at cell \( l \) is given by

\[
\tilde{x}(n) = x(n) + \epsilon \text{sign} \left( \nabla_{x(n)} \mathcal{L}(x(n), \theta_l) \right).
\]

Intuitively, the FGSM aims to add the \( \epsilon \)-bounded perturbation in the direction that results in the highest loss on the model. This can be considered a step of gradient ascent, where after calculating the direction of the gradient, the perturbation is added in that direction instead of subtracted as is done when training the model using SGD in (9). This attack is calculated for each DNN deployed at each of the separate BSs since the attacks considered here are cell-wise.

### C. Projected Gradient Descent (PGD)

PGD is an iterative extension of FGSM. Specifically, instead of adding \( \epsilon \) in a single step to each feature in the direction of the loss function’s gradient, as in (13), a smaller perturbation, \( \alpha = \epsilon / Q \), is added over \( Q \) iterations, where the direction of the gradient is recalculated on each iteration. Formally, the total perturbation is initialized to zero on the \( n^{th} \) sample, \( \Delta^{(0)}(n) = 0 \), and on each iteration, \( q \), the perturbation is crafted according to

\[
\Delta^{(q)}(n) = \Delta^{(q-1)}(n) + \alpha \text{sign} \left( \nabla_{x(n)} \mathcal{L}(x^{(q-1)}(n), \theta_l) \right),
\]

where the final perturbation is given by

\[
\delta(n) = \Delta^{(Q)}(n),
\]

thus resulting in the \( n^{th} \) adversarial sample crafted according to PGD being

\[
\tilde{x}(n) = x(n) + \Delta^{(Q)}(n).
\]

The increased potency of the PGD attack, over the FGSM, is a result of its iterative recalculation of the optimal direction of attack. This not only makes PGD more potent of an attack, but it also makes it much harder to mitigate at the BS.

### D. Threat Model and Adversary Knowledge Level

As shown in both FGSM and PGD formulations, the generation of adversarial attacks requires the adversary to calculate the gradient of the loss function with respect to a particular model. However, a successful adversarial attack (i.e., an attack that induces an erroneous power solution by the DNN) does not need to be crafted using the gradient of the DNN under attack. This is because adversarial attacks are transferable [57], [58], [59] in that attacks crafted to fool one specific DNN are effective on disparate models trained to perform the same task. Due to the transferability property of adversarial attacks, and considering that adversaries will realistically not have precise information about the DNN at the BS [17], we consider two distinct knowledge levels, termed threat models, of the adversary.

Each threat model differs in the amount of system knowledge available to the adversary. The semi-white box threat model assumes that the adversary is aware of the trained DNN model and its parameters but is blind to the defense. The black box threat model assumes that the adversary is blind to both the defense and power prediction DNN. In a white box threat model, by contrast, the adversary would have full knowledge of the DAE and the power allocation DNN as well as their parameters. However, in the context of wireless communications, it is rare for the adversary to have exact knowledge of each signal processing module deployed at the BS [17].

The semi-white box and black box threat models, on the other hand, are more practical to consider because (i) the semi-white box threat model assumes that the adversary has knowledge of the underlying DNN, which may be possible if an off-the-shelf network has been deployed (e.g., residual DNNs for ImageNet [60]) and (ii) the black box threat model assumes the adversary does not have access to the defense or the DNN model, which is the most likely attack scenario given that an adversary will rarely have precise information about the signal processing modules at any BS. Thus, due to the impractical nature of white box attacks in massive MIMO networks, we exclude white box attacks from consideration and, instead, show that our defense is robust in semi-white box and black box threat models, which are more pragmatic in real-world mmMIMO environments.

**Semi-white box attack:** In the semi-white box threat model, we assume that the adversary has full access to the DNN model architecture and its parameters but is blind to any pre-processing functions and algorithms applied to the UE positions prior to its input to the DNN. This type of threat model would hold when the DNN is an off-the-shelf mobile made publicly available (as several DNN models are, e.g., residual DNNs for ImageNet [60]) giving an adversary access to it but simultaneously leaving them blind to other system modules. Therefore, under the semi-white box threat model, the adversary would use the gradient of the model deployed at the BS to craft the attack. This attack would directly target the model deployed at the BS since its gradient would be used to craft the attacks shown in (14) and (17). The semi-white box attack results in a more potent attack in comparison to the black box attack. Higher attack potency, in this context, refers to requiring a smaller
perturbation magnitude (i.e., smaller UE positional displacement) to induce an infeasible power allocation prediction from the BS.

Black box attack: In the black box threat model, the adversary is aware of the power allocation regression task at the BS, but they are blind to both the DNN (including its architecture and parameter values) and the pre-processing applied to the input prior to forward propagating it through the DNN. Here, the adversary leverages the transferability property of adversarial attacks to the input processed by the DNN at the BS. Specifically, the adversary constructs their surrogate model by first obtaining samples that can be used to train a DNN to perform power allocation prediction. Then, the adversary trains their own DNN architecture, which is different than the adversary leverages the transferability property of adversarial attacks generated using PGD; this expected to induce an infeasible power allocation solution on the model at the BS with high probability, albeit with lower potency (i.e., the same perturbation displacement may not be as effective in a black box attack compared to a semi-white box attack).

V. THE DECODING AUTOENCODER DEFENSE

In this section, we discuss our proposed methodology for defending maMIMO power allocation DNNs from adversarial attacks. We begin by introducing a deeper learning-based denoising function, which is used to pre-process inputs prior to their input to the DNN. After the DAE has been trained, samples at inference time (when the DAE is deployed at the BS) are forward propagated through the DAE before being inputted into the regression DNN. The DAE is trained to mitigate the effect of any potential additive adversarial perturbation as well as retain performance on unperturbed signals.

Adversarial attacks introduce subtle perturbations into the input that lead to erroneous outputs by the DNN. Since such attacks are often gradient-based, they will be added in a manner consistent with maximizing some loss function regardless of the type of attack and threat model adopted by the adversary. We leverage this intuition and propose training a Denoising Autoencoder (DAE), which is capable of filtering noise from a consistent corruption processes [61]. The construction of our proposed DAE is detailed in Algorithm 1. We begin, in lines 3 – 12, by using the samples comprising $X_{\text{tr}}$ to generate adversarial examples using PGD according to (15) – (17). Our DAE is specifically tailored to defend PGD because, as a result of its iterative recalculation of the gradient as discussed in Section IV-C, adversarial examples generated using PGD capture an exhaustive set of gradient-based adversarial attacks. Therefore, as we show in Section VI, defenses trained against PGD attacks effectively generalize to mitigate other classes of gradient-based attacks such as FGSM. Furthermore, prior work [43] has shown that, for classification-based DNNs, robustness against this attack guarantees robustness against other first-order $l_\infty$-bounded adversarial attacks as well. In Section VI, we show that this property extends to regression-based DNNs as well in the specific context of mitigating adversaries in maMIMO power allocation.

In the implementation of Algorithm 1, PGD perturbations are generated using the gradient of the DNN at BS $l$. This corresponds to a semi-white box adversarial sample generation method, which is chosen since semi-white box attacks are more potent than black box attacks. Thus, training to defend a semi-white box attack is expected to increase robustness to black box attacks, despite the gradient used to generate such attacks being in a suboptimal direction of attack (different than the direction of the gradient of the loss calculated using the DNN under attack at the BS).

Following the generation of PGD adversarial examples, at BS $l$, we train a DAE denoted by $D^l(\cdot, \phi_l, \zeta_l): \mathbb{R}^{2KL} \rightarrow \mathbb{R}^{2KL}$, which consists of an encoder function, $u^l(\cdot, \phi_l): \mathbb{R}^{2KL} \rightarrow \mathbb{R}^z$, parameterized by $\phi_l$, and a decoder function $v^l(\cdot, \zeta_l): \mathbb{R}^z \rightarrow \mathbb{R}^{2KL}$ parameterized by $\zeta_l$. The encoder compresses the 2KL dimensional input into a $z$ dimensional representation, and the decoder reconstructs an approximation of the uncorrupted input from the $z$ dimensional representation to the original 2KL dimensional space. The full DAE, comprised of both the encoder and decoder, are constructed using fully connected layers.

| Algorithm 1 Denoising Autoencoder Construction |
|------------------------------------------------|
| 1: input: $X_{\text{tr}}$: Set of input samples: $x(n) \forall n$ |
| $f^l(\cdot, \theta_l)$: Trained regression DNN at BS $l$ |
| $\alpha_l$; $l_\infty$-norm bound used at BS $l$ |
| $v^l(\cdot, \phi_l)$: Randomly initialized encoder function |
| $u^l(\cdot, \zeta_l)$: Randomly initialized decoder function |
| 2: initialize: $X_{\text{DAE}} \leftarrow \emptyset$ |
| 3: for $x(n) \in X_{\text{tr}}$ do |
| 4: $\Delta^{(0)}(n) = 0$ |
| 5: for $q = 1, \ldots, Q$ do |
| 6: $\mathcal{L}(x^{(q-1)}(n), \theta_l) = \sum_{k=1}^{K} \rho_{jk}$ |
| 7: $\kappa_{(q-1)}(n) = \text{sign}(\nabla_{x(n)} \mathcal{L}(x^{(q-1)}(n), \theta_l))$ |
| 8: $\Delta^{(q)}(n) = \Delta^{(q-1)}(n) + \alpha_l \kappa_{(q-1)}(n)$ |
| 9: end for |
| 10: $\tilde{x}(n) = x(n) + \Delta^{(Q)}(n)$ |
| 11: append $\tilde{x}(n)$ to $X_{\text{DAE}}$ |
| 12: end for |
| 13: $X_{\text{DAE}} \leftarrow \text{concatenate}(X_{\text{DAE}}', X_{\text{tr}})$ |
| 14: $X_{\text{target}} \leftarrow \text{concatenate}(X_{\text{target}}', X_{\text{tr}})$ |
| 15: for $x'(n), x(n) \in X_{\text{DAE}}, X_{\text{target}}$ do |
| 16: $e = x(n) - v^l(u^l(x'(n)))$ |
| 17: $D^l(\cdot, \phi_l, \zeta_l) \leftarrow \min_{\phi_l, \zeta_l} \frac{1}{2KL} \sum_{s=1}^{2KL} (e(s))^2$ |
| 18: end for |
| 19: return $D^l(\cdot, \phi_l, \zeta_l)$ |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
layers. The architecture of our proposed model is shown in Table III. After random initialization, the DAE is trained, using SGD, to map the generated PGD adversarial examples to an approximation of the unperturbed positional input by minimizing the mean squared error. Here, we first calculate the error between the DAE’s reconstruction of an adversarial example, and its corresponding clean sample given by

$$e = x(n) - v^l(u^l(\tilde{x}(n)))$$

where $v^l(u^l(\tilde{x}(n)))$ is the output of the DAE on input $\tilde{x}(n)$ and $e \in \mathbb{R}^{2KL}$, corresponds to the difference being taken element-wise. Next, we train the DAE by minimizing the squared error given by

$$\min_{\phi, \zeta} \frac{1}{2KL} \sum_{s=1}^{2KL} (e(s))^2,$$

where $e(s)$ denotes the $s^{th}$ element in the error vector $e$. Although the formulation shown here corresponds to training on one sample at a time (i.e., a batch size of 1), mini-batches consisting of multiple samples are used in practice to accelerate training.

During the deployment of the DAE defense, the BS will be unaware whether the input is corrupted or not (i.e., the BS will not know whether $x(n)$ of $\tilde{x}(n)$ was received). Therefore, while the trained DAE should map $\tilde{x}(n)$ to an approximation of $x(n)$, it should also be capable of mapping $x(n)$ to an approximation of itself so that benign inputs do not suffer performance degradation by the defense. During training, we enforce this mapping by including all uncorrupted training samples (i.e., all samples in $\mathcal{X}_{12}$) in the training set for the DAE. This is shown in lines 13–18 of Algorithm 1.

After training the DAE, each input is pre-processed using $D^l(\cdot, \phi, \zeta)$ before being inputted to the DNN for power allocation. Specifically, given a received sample, $\tilde{x}(n)$, we first denoise it to subtract any potential adversarial perturbation. The denoised input is given by $\tilde{x}(n) = D^l(\tilde{x}(n), \phi, \zeta)$. Then, $\tilde{x}(n)$ is forward propagated through the DNN regression model to yield the prediction power allocation solution to each UE in cell $j$ given by $\tilde{p}_j(n) = f^l(\tilde{x}(n), \theta_l)$. Our overall system diagram is shown in Fig. 1.

The threat model under which the adversary operates determines the type and potency of the launched attack. In the semi-white box threat model, the adversary uses the gradient of the model at the BS to craft its attack using either (14) or (17). In this threat model, the adversary is blind to the DAE parameters and processing function at the BS. Moreover, this threat model results in the most potent attack since it directly targets the DNN employed at the BS. Contrarily, in the black box threat model, the adversary is blind to both the DAE and the DNN employed at the BS. As a result, the adversary instantiates an attack using the gradient of a surrogate model (i.e., a model that the adversary obtained access to by training it to perform the same task as the DNN at the BS) and relies on the transferability property of adversarial attacks to induce infeasible power allocation predictions at the BS. Although this attack is less potent than the semi-white box attack, it still reduces the DNN’s performance and is the most likely attack that the adversary will induce, given that adversaries will typically be knowledge-limited about the signal processing modules at the BS.

Finally, we analyze the online computational complexity of our proposed DAE defense. For a general DNN, a single matrix followed by an element-wise activation function is needed to characterize the weights and transformations between one layer to the next. Assuming an $A \times B$ weights matrix with $A > B$, where $A$ and $B$ are the number of units in the former and latter layers, respectively, the computational complexity of forward propagating a sample through a single layer with an activation function is given by $O(A^3)$. Thus, the computational complexity for a $p$ layered DNN is upper bounded by $O(pA^3)$ (assuming that the largest number of units in any layer of the network is $A$). Our DAE introduces an additional complexity of $O(sa^2)$, where the complexity is squared instead of cubed because the activation function is linear, $a < A$ since the largest layer of the DAE has less units than the largest layer in the DNN (e.g., as in Tables I, II, and III), and $s$ is the number of layers in the DAE. Therefore, the total time complexity of our proposed method is given by $O(sa^2 + pA^3)$. This leads to computationally efficient runtimes, as we demonstrate empirically in Section VI-D.

### VI. Performance Evaluation

In this section, we conduct an empirical evaluation of our methodology. We describe our empirical setup, show the
predicts a total power allocation less than or equal to the feasible power allocation solution is achieved when the DNN serves as the defense against adversarial attacks. Defensive distillation applied for regression tasks, however, may potentially hinder baseline performance, since it forces re-training of the model on inaccurate power allocation outputs (i.e., the “soft labels”). Our proposed method aims to address these shortcomings of both adversarial training and defensive distillation by avoiding model retraining.

To evaluate and compare our proposed method, we show four performance curves per environment. First, we show the baseline with no attack, which is the proportion of samples whose power predictions naturally result in infeasible power solutions without any added perturbations. This is analogous,

Table IV: The Parameter Values Used to Train Our Proposed Defense in Our Empirical Evaluation

| Parameter     | Value       |
|---------------|-------------|
| L             | 4 cells     |
| M             | 100 antennas|
| K             | 5 UEs       |
| Cell area     | 250m × 250m |
| \(P_{\text{max}}\) | 500 mW     |
| \(\sigma^2\)  | -94 dBm    |
| Bandwidth     | 20MHz       |
| \(z\)         | 16 units    |
| \(\alpha\)    | 0.03        |
| Q             | 10 iterations|
| N             | 329000 samples |
| B             | 500 samples  |

![Visualize Data Samples](image)

Fig. 2. 40 testing samples shown in their original unperturbed form, perturbed form, and denoised form after forward propagating through the DAE. Each point denotes the position of the UE. We see that perturbed samples deviate substantially from their unperturbed counterparts, whereas denoised samples are much closer to unperturbed samples.

A. Dataset and Performance Metrics

To assess the efficacy of our methodology, we employ the Power Allocation in Multi-Cell Massive MIMO dataset, which is publicly available at https://data.ieeemlc.org/Ds2Detail. All parameter values used in our evaluation are presented in Table IV. We consider \(L = 4\) cells in a square grid layout of \(2 \times 2\) cells. In each cell, the BS is located in the center of the cell. The presented results are averaged across all BSs. A feasible power allocation solution is achieved when the DNN predicts a total power allocation less than or equal to \(P_{\text{max}}\) to all UEs in the cell. Otherwise, the solution is infeasible.

The DAE is trained according to Algorithm 1 using the architecture and parameter values shown in Tables III and IV, respectively. Fig. 2 visually compares some examples of the original, adversarial, and denoised samples. At test time, we vary the perturbation magnitude of each considered attack and measure the attack success rate, which is the number of samples in the testing set that exceed the power budget of \(P_{\text{diff}} = 500\) mW in the DNN prediction output (i.e., the proportion of samples that satisfy each constraint in (11)). The performance of our defense in both the presence and absence of adversarial attacks, and discuss the computational efficiency of our method.

B. Semi-White Box Performance

We begin by evaluating the efficacy of our proposed defense in the semi-white box threat model, where the adversary has access to the classifier (either Model 1 or Model 2) under attack and uses its gradient to generate an adversarial perturbation. We consider attacks on both Model 1 and Model 2 using both the MMMSE and MR precoding schemes. In addition, we compare our results to adversarial training on maMIMO DNN models [47] as well as defensive distillation [62] in each case as a performance benchmark. Adversarial training, as proposed in [47], consists of generating adversarial perturbations on the target DNN (i.e., the model whose gradient will be used to generate the attack) and retraining it with the corrected labels. Although this method is effective in mitigating low-bounded perturbations, it remains vulnerable to higher bounded perturbations and, furthermore, is highly susceptible to overfitting on adversarial samples thus lowering its prediction performance on unperturbed inputs [48]. Defensive distillation first trains each DNN model on the true “hard labels” (i.e., the true power allocation vectors). Then, training samples are forward propagated through the DAE to obtain each samples’ “soft labels,” which are the predicted labels from the trained model. The original input samples and the “soft labels” are then used to train a new model, which serves as the defense against adversarial attacks. Defensive distillation applied for regression tasks, however, may potentially hinder baseline performance, since it forces re-training of the model on inaccurate power allocation outputs (i.e., the “soft labels”).
in a classification setting, to samples that are misclassified due to the DNN’s inability to correctly predict them rather than due to any additive perturbations. Thus, since the baseline is not dependent on the perturbation magnitude, it stays constant throughout the considered perturbation range. Second, we show the performance curves for the baseline case, which is the performance of the DNN without any defense in place to mitigate the attack. Note that the attack success rate for the baseline case, in the majority of settings, increases with higher perturbation magnitudes thus confirming that the approximated attack solution is not guaranteed to effectively induce an infeasible power allocation solution from the DNN. Third, we show the performance of our DAE defense, and lastly, we show the performance of adversarial training and defensive distillation as previously described.

Figs. 3 and 4 show the DNN performance of Model 1 and Model 2, respectively. In general, we can see from both models that the PGD attack is more potent than the FGSM attack irrespective of the precoding scheme. In addition, we see that both adversarial training and defensive distillation remain susceptible to high-bounded adversarial attacks which is consistent with prior works that have considered using them as a defense [47], [48], [63]. This trend is observed under both FGSM and PGD perturbations, where adversarial training and defensive distillation deliver slightly improved performance over no defense. Our proposed defense, on the other hand, is able to significantly mitigate semi-white box attacks across the entire range of considered perturbation magnitudes. For example, at a perturbation magnitude of $\alpha = 0.10$, the PGD attack on MR pre-coded samples on both Models 1 and 2 results in nearly 100% samples being predicted with infeasible power allocation solutions on undefended models, models with adversarial training, and defensive distillation. However, with our proposed DAE in place, only approximately 10% of the samples produce infeasible solutions. Although the FGSM attack yields a smaller proportion of infeasible solutions in comparison to the PGD perturbation, our defense is still able to mitigate its effects to a large degree (e.g., from 60% to 10% on Model 1 with MR precoding). These observations are consistent across both Model 1 and Model 2 in defending both considered attacks on both considered precoding schemes.

Next, we consider the performance of our defense in the absence of adversarial attacks. Note that, upon receiving a sample of UE positional locations, the BS will be unaware of whether it contains an adversarial perturbation or not. Therefore, any employed defense should keep the baseline performance on unperturbed signal intact. In Figs. 3 and 4, unperturbed signals correspond to $\epsilon = 0$ and $\alpha = 0$ for the FGSM and PGD perturbations, respectively (this results in $d(n) = 0$ in (13) and (16) for FGSM and PGD, respectively). From Figs. 3 and 4, we see that, at a perturbation magnitude of zero, our proposed defense retains approximately the same performance as the baseline model with no attack (i.e., the performance on the original DNN at the BS before employing any defense). In fact, the DAE slightly improves the baseline power prediction performance in the absence of adversarial attacks in black box setups with MR precoding. The performance of our model in the absence of attacks is shown in Table V. Ultimately, in all cases, the proposed defense provides strong attack mitigation abilities on a wide range of perturbation magnitudes.

We more closely examine the distribution of predicted power by the DNN under different defenses in Fig. 5. Here, we instantiate a PGD attack using $\alpha = 0.05$ on Model 2. Once again noting that $P_{\text{max}} = 500 \text{ mW}$, we see that the original set of unperturbed testing samples only result in 9.15% of samples being predicted with infeasible power solutions whereas, after instantiating the attack with no defense, 99.27% of predicted samples are infeasible. Our proposed DAE defense significantly reduces the proportion of samples predicted with infeasible solutions to 24.01%, which outperforms adversarial training and more closely approaches the performance of the model with no attack.

Finally, as shown in Algorithm 1, only PGD adversarial examples are generated to train the DAE. However, as shown in Figs. 3 and 4, the DAE is still able to defend the FGSM
Fig. 5. The power prediction distributions under each considered defense scenario on Model 2 in the semi-white box environment. We see that our proposed DAE defense (bottom right) significantly lowers the proportion of samples predicted with infeasible power solutions (i.e., when $P_{\text{max}} > 500 \text{ mW}$) over using no defense (top right) and adversarial training (bottom left).

Table V

|                  | Model 1 | Model 1 | Model 2 | Model 2 |
|------------------|---------|---------|---------|---------|
|                  | MMMSE   | MR      | MMMSE   | MR      |
| Baseline DNN     | 5.31%   | 4.63%   | 9.15%   | 20.64%  |
| DAE Defense      | 14.57%  | 6.67%   | 25.97%  | 5.97%   |

p perturbation. Thus, we see that training to defend the PGD attack, as our proposed defense does, provides robustness against other first-order $l_\infty$-bounded perturbations on power allocation regression-based DNNs in maMIMO. Furthermore, the PGD attack generated to defend the DNN only employs one perturbation bound ($\alpha = 0.03$ and $Q = 10$ as discussed earlier). Yet, it is able to provide mitigation against a wide range of adversarial perturbation magnitudes as demonstrated in Figs. 3 and 4. This provides a substantial improvement over adversarial training, where training on multiple bounds may be required for better robustness at the cost of lower performance on non-adversarial samples. Training our proposed DAE on a single perturbation bound not only retains the performance on unperturbed samples (i.e., when $\delta(n) = 0$), but it also does not incur costly computational overhead during deployment, which will be further discussed in Section VI-D.

C. Black Box Performance

Now, we discuss the effectiveness of our defense in the black box threat model, where the adversary is blind to the DNN model used for power allocation at the BS. Here, we consider two attack scenarios. The first is where the adversary crafts an adversarial attack using the gradient of Model 2 but Model 1 is used to predict the power allocation at the BS using the perturbed sample. The second case is where the attack is crafted using the gradient of Model 1 but the perturbed sample is evaluated on Model 2 at the BS. Results for the former and latter attack scenarios are shown in Figs. 6 and 7, respectively, where we use the term victim model to refer to the model under attack and the term adversary model to refer to the model whose gradient is used to generate the perturbation.

From Figs. 6 and 7, we see that black box attacks are generally less potent than semi-white box attacks, but our proposed method is still the most effective in reducing the adversary’s success rate in each considered scenario.
in Fig. 8, where we plot the distribution of the sum of the predicted power for all testing samples. In comparison to Fig. 5, corresponding to the semi-white box threat model, we see that the same perturbation magnitude ($\alpha = 0.05$) results in less infeasible samples on defended models, thus corroborating the lower potency of black box attacks in maMIMO DNNs. Despite the lower potency, we still see that our proposed defense, in general, is able to lower the proportion of samples predicted with infeasible power solutions. In certain cases, such as black box attacks induced on MMSME precoded signals using the FSGM perturbation, the proposed defense slightly improves the attack success rate. Thus, in these cases, the BS experiences a slight performance degradation, which is traded off for significant attack mitigation in all other threat models.

Moreover, we see that the general trends observed on the effectiveness of our defense in the semi-white box threat model hold in the black box threat model as well despite the attacks being less transferable (and thus less potent in the black box threat model). For example, the DAE retains its prediction performance on unperturbed samples as shown in Figs. 6 and 7 where $\epsilon = 0$ and $\alpha = 0$ for the FGSM and PGD attack, respectively. In addition, the DAE significantly reduces the proportion of samples predicted with an infeasible power solution in each case over the considered perturbation magnitude range in comparison to no defense, adversarial training, and defensive distillation. For example, as shown in Fig. 7, a PGD adversarial attack at the maximum considered perturbation on samples with MR precoding results in an adversary success rate of approximately 70%. Adversarial training and defensive distillation result in an adversarial success rate to approximately 58% and 82%, respectively, while our proposed DAE reduces the adversarial success rate to approximately 11%, which is approaching the expected performance of the DNN when no attacks are present (approximately 10% of samples predicted with infeasible solutions). Lastly, we see that our proposed defense is able to mitigate FGSM attacks despite only being trained on PGD perturbations on one perturbation magnitude. This reinforces our proposed method’s ability to provide robustness against other first order $l_{\infty}$-bounded adversarial attacks that the DAE was not trained to defend.

D. Computational Efficiency

Lastly, we discuss the computational feasibility of our proposed defense during both offline and online usage. Offline computational overhead is primarily associated with the training costs of the defense. During training, our proposed DAE takes, on average, 2.01 secs. per epoch to train whereas adversarial training and defensive distillation require, on average, 3.10 secs. per epoch. In all cases, these average training times assume that the defense only requires one perturbation bound for training. Advantageously, our proposed DAE defense can significantly mitigate a wide range of perturbation magnitudes when only trained on one bound (as described in Algorithm 1 and shown in Sec VI-B and VI-C), while adversarial training may require additional bounds leading to even higher offline computational costs. In addition, the computational cost of adversarial training and defensive distillation is dependent on the DNN architecture and its total number of trainable parameters (since they both require retraining) whereas the training cost of the DAE is independent of the DNN architecture. For example, the DAE trained to defend both Model 1 and Model 2 incur identical computational costs despite Model 2 having more parameters than Model 1. As a result, our proposed DAE can be implemented on high-parameter DNNs for strong adversarial attack mitigation without incurring high offline computational costs.

Online computational costs correspond to the total time required at the BS to predict the power allocation to each UE in the cell. The forward propagation time through the DNN at the BS for power prediction on one sample (i.e., one particular distribution of UE positions) is $1.25 \times 10^{-3}$ secs. Forward propagation through the DAE prior to inputting the sample into the DNN adds an additional $0.87 \times 10^{-3}$ secs. for both Model 1 and Model 2, thus resulting in an online prediction time of our proposed defense of $2.12 \times 10^{-3}$ secs. Thus, with our proposed defense in place, the prediction time during online implementation remains very low and allows for rapid recalculation of power allocation as UEs quickly move within cells.

VII. CONCLUSION

Power allocation from base stations (BSs) to user equipments (UEs) is a vital task in maMIMO networks for low-interference and low-cost communications. Recent work has shown that deep neural networks (DNNs) can perform effective power allocation with reduced computational overhead in comparison to convex optimization techniques. Yet, DNNs are susceptible to adversarial attacks, which cause trained DNNs to predict infeasible power allocation solutions to UEs, thus depleting their effectiveness outweighing their computational efficiency. In this work, we proposed defending power allocation DNN models from adversarial attacks using denoising autoencoders (DAEs), where input sample were first forward propagated through the DAE to filter any potentially added adversarial attack before being inputted to the DNN for power allocation prediction. Our proposed method
significantly improves and outperforms previously considered baselines for power allocation prediction in the presence of adversarial attacks, and it can be adopted in existing systems with relatively little computational overhead. Future work will consider the scalability of our work in very large federated machine learning networks [64], where layers of devices are used between BSs to train and defend power allocation DNNs.

REFERENCES
[1] K. Zheng, L. Zhao, J. Mei, B. Shao, W. Xiang, and L. Hanzo, “Survey of large-scale MIMO systems,” IEEE Commun. Surveys Tuts., vol. 17, no. 3, pp. 1738–1760, 3rd Quart., 2015.
[2] S. Parkvall, E. Dahlman, A. Furuskar, and M. Frenne, “NR: The new 5G radio access technology,” IEEE Commun. Standards Mag., vol. 1, no. 4, pp. 24–30, Dec. 2017.
[3] T. L. Marzetta, “Noncooperative cellular wireless with unlimited numbers of base station antennas,” IEEE Trans. Wireless Commun., vol. 9, no. 11, pp. 3590–3600, Nov. 2010.
[4] L. Zhao, H. Zhao, F. Hu, K. Zheng, and J. Zhang, “Energy efficient power allocation algorithm for downlink massive MIMO with MRT precoding,” in IEEE 78th VTC, 2013, pp. 1–5.
[5] Q. Zhang, S. Jin, M. McKay, D. Morales-Jimenez, and H. Zhu, “Power allocation schemes for multicell massive MIMO systems,” IEEE Trans. Wireless Commun., vol. 14, no. 11, pp. 5941–5955, Nov. 2015.
[6] T. Van Chien, E. Björnson, and E. G. Larsson, “Joint power allocation and user association optimization for massive MIMO systems,” IEEE Trans. Commun., vol. 65, no. 12, pp. 6288–6295, Dec. 2017.
[7] P. Liu, S. Jin, T. Jiang, Q. Zhang, and M. Matthaiou, “Pilot power allocation through user grouping in multi-cell massive MIMO systems,” IEEE Trans. Commun., vol. 65, no. 4, pp. 1561–1574, Apr. 2017.
[8] Y. Dai and X. Dong, “Power allocation for multi-pair massive MIMO two-way AF relaying with linear processing,” IEEE Trans. Wireless Commun., vol. 15, no. 9, pp. 5932–5946, Sep. 2016.
[9] Y. Shi, Y. E. Sagduyu, T. Erpek, K. Davasioglu, Z. Lu, and J. H. Li, “Adversarial deep learning for cognitive radio security: Jamming attack and defense strategies,” in Proc. IEEE ICC Workshops, 2018, pp. 1–6.
[10] A. Berian, K. Staab, N. Teku, G. Ditzler, T. Bose, and R. Tandon, “Adversarial filters for secure modulation classification,” 2020, arXiv:2010.11351.
[11] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, “Adversarial attacks in modulation recognition with convolutional neural networks,” IEEE Trans. Rel., vol. 70, no. 1, pp. 389–401, Mar. 2021.
[12] M. Z. Hameed, A. György, and D. Gündüz, “The best defense is a good offense: Adversarial attacks to avoid modulation detection,” IEEE Trans. Inf. Forensics Security, vol. 15, no. 1, pp. 1074–1087, 2020.
[13] S. Kokalj-Filipovic, R. Miller, N. Chang, and C. L. Lau, “Mitigation of adversarial examples in RF deep classifiers utilizing AutoEncoder pre-training,” in Proc. ICMCS, 2019, pp. 1–6.
[14] L. Zhang, S. Lambotharan, G. Zheng, B. AsSadhan, and F. Roli, “Countermeasures against adversarial examples in radio signal classification,” IEEE Wireless Commun. Lett., vol. 10, no. 8, pp. 1830–1834, Aug. 2021.
[15] R. Sahay, C. G. Brinton, and D. J. Love, “A deep ensemble-based wireless receiver architecture for mitigating adversarial attacks in automatic modulation classification,” IEEE Trans. Cogn. Commun. Netw., vol. 8, no. 1, pp. 71–85, Mar. 2022.
[16] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey,” IEEE Access, vol. 6, pp. 14410–14430, 2018.
[17] R. Sahay, R. Mahfuz, and A. E. Gamal, “Combating adversarial attacks through denoising and dimensionality reduction: A cascaded autoencoder approach,” in Proc. 53rd Annu. CISS, 2019, pp. 1–6.
[18] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” in Proc. IEEE CVPR, Jun. 2018, pp. 1778–1787.
[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–23.
[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2014, arXiv:1412.6572.
[21] R. Sahay, D. J. Love, and C. G. Brinton, “Robust automatic modulation classification in the presence of adversarial attacks,” in Proc. 55th Annu. CISS, 2021, pp. 1–6.
[22] L. Zhang, S. Lambotharan, and G. Zheng, “A countermeasure against adversarial attacks on power allocation in a massive MIMO network,” in Proc. IEEE Symp. Wireless Technol. Appl. (ISWTA), 2022, pp. 13–16.
Minjun Zhang received the B.S. degree (with Distinction) in computer engineering from Purdue University, West Lafayette, in 2022. She was a Software Engineer Intern with Caterpillar, Inc. in Summer 2021, and a Data Science Intern with Chan Zuckerberg Initiative, LLC in Summer 2022. Since Spring 2020, she has been a Teaching Assistant with Purdue University for courses, such as Data Structures and Algorithms, and Python for Data Science. Her research interests include deep learning for networks, adversarial machine learning, and computer vision. While at Purdue University, she received the Dave and Marsha Meyer Undergraduate Teaching Award in April 2022, and the Eli Shay Scholarship in May 2022.

David J. Love (Fellow, IEEE) received the B.S. (with highest Hons.), M.S.E., and Ph.D. degrees in electrical engineering from the University of Texas at Austin in 2000, 2002, and 2004, respectively. Since 2004, he has been with the Elmore Family School of Electrical and Computer Engineering, Purdue University, where he is currently the Nick Trbovich Professor of Electrical and Computer Engineering. He holds 32 issued U.S. patents. His research interests are in the design and analysis of broadband wireless communication systems, beyond-5G wireless systems, multiple-input multiple-output (MIMO) communications, millimeter wave wireless, software-defined radios and wireless networks, coding theory, and MIMO array processing. Along with his coauthors, he won best paper awards from the IEEE Communications Society (2016 Stephen O. Rice Prize and 2020 Fred W. Ellersick Prize), the IEEE Signal Processing Society (2015 IEEE Signal Processing Society Best Paper Award), and the IEEE Vehicular Technology Society (2010 Jack Neubauer Memorial Award). He served as a Senior Editor for IEEE Signal Processing Magazine, the Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and the Guest Editor for special issues of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS and the EURASIP Journal on Wireless Communications and Networking. He was a member of the Executive Committee for the National Spectrum Consortium. He is a Fellow of the American Association for the Advancement of Science and was named a Thomson Reuters Highly Cited Researcher from 2014 to 2015.

Christopher G. Brinton (Senior Member, IEEE) received the M.S. and Ph.D. (with Hons.) degrees in electrical engineering from Princeton University in 2013 and 2016, respectively. He is an Assistant Professor with the School of Electrical and Computer Engineering, Purdue University. Prior to joining Purdue University, he was the Associate Director of the EDGE Lab and a Lecturer of Electrical Engineering with Princeton University. His research interest is at the intersection of networked systems and machine learning, specifically in distributed machine learning, fog/edge network intelligence, and data-driven network optimization. He was the recipient of the 2022 NSF CAREER Award, the 2022 ONR Young Investigator Program Award, and the 2022 DARPA Young Faculty Award. He currently serves as an Associate Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, in the ML and AI for wireless area.