

@PAD: Adversarial Training of Power Systems Against Denial-of-Service Attacks

Ali I Ozdagli
Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee
Ali.I.Ozdagli@vanderbilt.edu

Carlos Barreto
Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee
Carlos.A.Barreto@vanderbilt.edu

Xenofon Koutsoukos
Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee
Xenofon.Koutsoukos@vanderbilt.edu

ABSTRACT

In this work, we study the vulnerabilities of protection systems that can detect cyber-attacks in power grid systems. We show that machine learning-based discriminators are not resilient against Denial-of-Service (DoS) attacks. In particular, we demonstrate that an adversarial actor can launch DoS attacks on specific sensors, render their measurements useless and cause the attack detector to classify a more sophisticated cyber-attack as a normal event. As a result of this, the system operator may fail to take action against attack-related faults leading to a decrease in the operation performance. To realize a DoS attack, we present an optimization problem to determine which sensors to attack within a given budget such that the existing classifier can be deceived. For linear classifiers, this optimization problem can be formulated as a mixed-integer linear programming problem. In this paper, we extend this optimization problem to find attacks for more complex classifiers such as neural networks. We demonstrate that a neural network, in particular, with RELU activation functions, can be represented as a set of logic formulas using Disjunctive Normal Form, and the optimization problem can be used to efficiently compute a DoS attack. In addition, we propose a defense model that improves the resilience of neural networks against DoS through adversarial training. Finally, we evaluate the efficiency of the approach using a dataset for classification in power systems.

ACM Reference Format:

Ali I Ozdagli, Carlos Barreto, and Xenofon Koutsoukos. 2020. @PAD: Adversarial Training of Power Systems Against Denial-of-Service Attacks. In *Hot Topics in the Science of Security Symposium (HotSoS '20)*, April 7–8, 2020, Lawrence, KS, USA. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3384217.3385616>

1 INTRODUCTION

Power systems coordinate multiple and diverse components to operate efficiently (e.g., generate energy at the lowest costs) within given engineering constraints. For example, the system components operate at fixed frequencies, voltages, and have limits in the power

that they support. Power systems rely on communication technologies, such as *supervisory control and data acquisition* (SCADA) systems, to assess the system reliability and performance and take corrective actions. Data is critical to make decisions. In particular, power companies use sensor measurements to estimate the current state of the system, to forecast future load or generation, and to detect faults on the system.

Power systems also have quality requirements. For this reason, they analyze disturbances of the system to identify and correct faults. The disturbances occur due to typical events, such as connection or disconnection of large loads, large capacitors, and transformers, or rare events, such as faults (short circuits) or attacks. Disturbances can create disruptions in the voltage many orders of magnitude the nominal values and cause efficiency losses (e.g., the energy is dissipated as heat), which can deteriorate (and even can break) equipment. Works in the literature have proposed disturbance classification mechanisms based on machine learning algorithms, such as artificial neural networks (ANN) and support vector machines (SVM), decision trees [26, 32, 39, 40]. These algorithms also have been used to detect attacks against power grids [5, 20, 30].

The technologies used to classify disturbances are critical for the security of the system. Adversaries may specifically target these critical systems and exploit their vulnerabilities to cause harm. For example, SCADA systems may have vulnerabilities because some of their components have outdated operating systems and use poor security practices [47]. In fact, some attacks on critical infrastructures have exploited vulnerabilities in SCADA systems [3]. Moreover, adversaries can modify the inputs of ML algorithms to get a desired response, e.g., misclassify samples or induce wrong estimations. These vulnerabilities can endanger critical infrastructures.

In this work we analyze vulnerabilities of disturbance classification systems, which take information about the transitory state of the system to identify faults or attacks. We analyze how an adversary uses a Denial-of-Service (DoS) attack to induce errors in classifying disturbances. In this way, the system operator may fail to detect faults or attacks, which may delay the appropriate corrective action. For example, power companies may fail to identify components that cause failures or may try to correct faults that didn't occur.

In general, disturbance classification systems assume that the measurements from sensors are not corrupted. However, some of the sensors may be disabled through DoS attacks. In this paper, we discuss a DoS attack on a disturbance classification system. The objective of this model is determining the features to *delete* in a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

HotSoS '20, April 7–8, 2020, Lawrence, KS, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7561-0/20/04...\$15.00

<https://doi.org/10.1145/3384217.3385616>

given input data by disabling sensors instance within a budget such that the classifier will mispredict an attack event as normal. The features to be deleted correspond to the sensors that will receive the DoS attack. For linear classifiers, this is Linear Programming (LP) where the objective function has a convex solution [15]. However, for more complex classifiers, such as multi-perceptron neural networks with nonlinear activation functions, the solution space is not convex anymore and we cannot utilize LP approach for a solution. Our attack model describes a method to *linearize* a neural network with RELU activation functions and find features to delete that will cause misprediction. The core idea of this linearization approach is essentially rewriting the network in terms of logic formulas using Disjunctive Normal Form. For each clause in DNF, we can describe a convex Multi Integer Linear Programming (MILP) with proper constraints, and determine the features to delete. In addition to the attack model, we also discuss the defender model that is relatively robust against adversarial feature deletion attacks. The aim of this defense model is preventing the misprediction caused by the attack model. For this model, we propose adversarial training, i.e. training a new network with the attacked/modified data obtained from the attack model.

To evaluate the performance of the attack and defense schemes, this study employs a dataset designed for power grid disturbance classifiers. The dataset contains sensor measurements from natural events and disturbances caused by attacks. The dataset contains multiple measurement collected by phasor measurement units. Each data instance is classified as normal or attack. Using this dataset, we developed a neural network which constitutes our baseline model. Using the proposed attack model, we generated a modified dataset that contains instances with deleted features. We tested the performance of the attack model by looking at the decrease in the prediction accuracy. Additionally, as prescribed in the defense model, we trained a new network using the modified data set and checked the improvement in the accuracy of the new network under attack. The overall results demonstrate that the attack model works very efficiently even within constrained budgets. The defense model increases the accuracy against adversarial attacks to some degree but the improvement is limited.

In summary, the major contributions of this study are summarized as below:

- A new attack model is proposed that is capable of devising an effective DoS attack to deceive disturbance classifiers and disguise more complex cyber-attacks.
- In previous literature, DoS attacks were formulated for linear classifiers which is not applicable to nonlinear classifiers, such as neural networks. The attack scheme proposed herein can compute a DoS attack solution for neural networks effectively.
- A defense based on adversarial training is proposed such that the classifier can be more robust against DoS attacks.
- Results demonstrate that the attack model can be very effective for deceiving the classifiers under limited attack budget. The defense model improves the resilience of the classifier considerably.

It should be noted that the proposed attack and defense model is generalizable for many machine learning-based classifiers in various domains. In this paper, we specifically focus on the security of power grid systems. Since the sensors of grid systems are susceptible to DoS attacks, the methodology discussed in this paper aligns well with the research needs.

Note: The codes to generate the results presented in this paper can be accessed from <https://github.com/aliirmak/ATPAD>.

2 BACKGROUND

Power equipment has tight operational constraints. Thus, deviations from the nominal values can result in energy lost as heat, which can deteriorate and even can break the equipment. As a result of this, to extend the equipment life-cycle and reduce the maintenance costs, electric companies must offer a good power quality, that is, a continuous supply of voltage with some desired properties (e.g., symmetry, frequency, magnitude, and waveform) [1].

Power companies analyze disturbances (i.e., voltage changes) to assess whether they occurred due to changes in the system or faults (or as response to faults) [37]. It is possible to distinguish events because they create distinct disturbances. For example, some faults cause sudden voltage dips (due to the activation of protections), while other events (transformer saturation) may create voltage dips with high harmonics. The following events can cause voltage disturbances: energizing components (capacitors, motors), transformer saturation, transformer saturation followed by protection, transformer's step change, (fault or non-fault) interruptions, or single or multi stage dips due to faults [11].

For large sensor arrays, the data from sensors can be multi-dimensional to the point that human operator may not be able to detect abnormal events manually. For this reason, pattern recognition and data mining algorithms (e.g., ANN, fuzzy logic, expert systems, SVM) have been used to classify and identify power quality problems.

Let us consider n labeled samples (x_i, y_i) , with $i = 1, \dots, n$, where $x_i \in \mathbb{R}^d$ represents an input vector with d features (attributes of the power grid) and $y_i \in \{-1, 1\}$ denotes the type of event of the i^{th} sample (-1 for normal event; and 1 for attack event). Let us denote the output of the function $F(x_i, w) \in \{-1, 1\}$ as the classification label obtained with some algorithm, such as ANN or SVM. Here, w represents the weights used by the algorithm. We select the weight vector w to minimize the average error classifying the disturbances. We express the problem of designing the classifier (choosing the weight vector w) as

$$w^* \in \min_w \frac{1}{n} \sum_{i=1}^d l(y_i, F(x_i, w)), \quad (1)$$

where the loss function $l(y_i, f(x_i, w))$ measures the model's classification error. Let us denote the predicted label as $\hat{y}_i = F(x_i, w^*)$. Throughout the text, we will refer to $F(x, w)$ as $F(x)$ for simplicity.

3 METHODOLOGY

In this section, we describe the fundamental components of the attack and defense models in detail. First, we introduce the attack model, which is essentially a maximization problem. Then, we consider the challenges of the maximization problem for nonlinear

classifiers and suggest some relaxations, such as Disjunctive Normal Form approach, to compute attacks efficiently. The second section focuses on the defense model and describes it as a mini-max problem. This section also discusses adversarial training strategies for improving the resiliency of the classifier.

3.1 Attack Model

Suppose we have a power system and this system has an internal discriminator component capable of predicting and classifying events as *normal* and *attack* during operation. DoS attack is a type of attack where an adversary prevents users or cyber-infrastructures from accessing the necessary information [29]. For power systems, the malicious actor can devise a DoS attack such that sensors collecting operation-critical information, such as voltage and phase values, become inaccessible by real-time monitoring systems. In such cases, the sensor data will likely be registered as zero.

DoS attacks can be considered as a feature deletion from machine learning perspective [15, 36]. The discriminator can be deceived if relevant features are *deleted* from the input data. Accordingly, an adversarial actor can DoS-attack some sensors and make the sensor measurements inaccessible by the operator. When specific features are inaccessible, the predictor can mislabel attack events as normal. As a result, critical faults or more sophisticated cyber-attacks may be obscured and the misprediction could eventually lead to system failure and service interruption.

Here, we consider an adversarial model where the attacker attempts to determine features to delete, which will minimize the capability of the discriminator to detect attack events. In other words, the attacker model can be formulated as finding an optimal solution within a given budget (how many sensors can be disabled using DoS) to maximize the prediction error. While this problem is solved for linear classifiers, such as support vector machines (SVMs), to the authors' best knowledge, there is little research for more complex learning algorithms, such as neural networks.

Before going forward, there are few assumptions we have made regarding the attack model:

- The adversary has access to the industrial control system (ICS) and the sensor readings.
- The adversary can attack a limited number of sensors. The DoS attack causes ICS to register the readings from attacked sensors as zero.
- At ICS level, there is an attack detection mechanism utilizing a machine learning algorithm that can label events as normal and attack. The adversary has access to the machine learning algorithm. In other words, this is a white-box attack.
- The adversary attacks only the events labeled as an attack.
- Lastly, we assume neither data nor attack is time-correlated. In other words, the DoS attack deals with each measurement individually.

Under the given assumptions, we construct an optimization problem to determine which features to attack. Suppose that we have a neural network, $F(x)$ that can classify sensor measurement as normal or attack. Here, the input to $F(x)$ is x_i , the i^{th} electric transmission measurement instance. Each feature of the input data corresponds to a sensor reading in x_i such that $x_i \in \mathbb{R}^d$. The network is expected to predict the class label, \hat{y}_i where $\hat{y}_i \in \{-1, +1\}$,

-1 being normal events, and $+1$ being attack events. Then, the attack problem can be formulated as a maximization problem:

$$\begin{aligned} \alpha_i^{\max} &= \arg \max [1 - y_i \hat{y}_i]_+ \\ &= \arg \max [1 - y_i F(x_i \circ (1 - \alpha_i))]_+ \\ \text{s.t. } \alpha_i &\in \{0, 1\}^d \\ \sum_{j=1}^d \alpha_{ij} &\leq K \end{aligned} \quad (2)$$

where y_i is the true label; \hat{y}_i is the prediction; α_i is a vector describing which features to be deleted and can be described as $[\alpha_{i1}, \dots, \alpha_{id}]$ and j corresponds to the j^{th} feature. K is budget, i.e. the number of features attacker can delete. The operator, $[\cdot]_+$ is simply defined as $[y] = \max\{0, x\}$. When a particular element of α_i is zero (i.e. $\alpha_{ij} = 0$), the j^{th} feature should be deleted or attacked. Finally, the symbol, \circ represents the element wise product. According to the Equation 2, the attacker tries to determine α_i that will maximize the prediction error of the detector. If the adversary does not cause any misprediction, then the error is zero since $[1 - y_i \hat{y}_i]_+ = 0$. Ideally, the adversary should be able to find such a vector, α that should switch the predicted class from attack to normal.

3.1.1 Finding a solution for neural networks. For linear classifiers, the optimization problem presented in Equation 2, is a convex mixed-integer LP (MILP) [31]. While MILP is an NP-hard problem [21], it can be solved still efficiently using heuristic methods [43]. For neural networks (NN) with nonlinear activation functions, the solution space is not convex and Equation 2 cannot be written as a MILP. A solution could be still found by applying computationally exhaustive nonlinear programming (NILP) approaches [4].

We can consider some relaxations and assumptions regarding the NN structure to find a solution faster than a NILP does. Specifically, NN with linear and rectified linear unit (RELU) activations holds a piece-wise linearity characteristic. It is possible to reconstruct NN as a set of logic formulas using the Disjunctive Normal Form (DNF) [34] and solve Equation 2 using MILP. Each formula corresponds to the state whether a RELU is activated or not. This idea is sought in detail for the formal verification of neural networks [22]. Suppose we have a NN with a single RELU-activated neuron. We omit the bias for presentation purposes. Then, we can describe the NN as following:

$$\hat{y} = \text{RELU}(w x) \quad (3)$$

$$= \max(0, w x) \quad (4)$$

where the DNF clauses would be:

$$(\hat{y} == w x \wedge y > 0) \quad (5)$$

$$\vee (\hat{y} == 0 \wedge w x \leq 0) \quad (6)$$

Using these clauses, we can write NN as an MILP problem and each clause can be regarded as a collection of constraints that needs to be satisfied based on the state of the activation [13]. For example, clause given in Equation 5 implies that the RELU is active and $w x$ should be larger than zero. Similarly, the next clause (Equation 6) describes a state where RELU is not activated, thus, the $w x$ should be smaller than zero.

For each clause, we can find α that can maximize the error according to Equation 2. For example, for the first clause, the maximization problem will be written as:

$$\begin{aligned} \alpha_{i,1} &= \arg \max [1 - y_i \hat{y}_i]_+ \\ \text{s.t. } \alpha_i &\in \{0, 1\}^d \\ \sum_{j=1}^d \alpha_{ij} &\leq K \\ \hat{y}_i &= w x_i \circ (1 - \alpha_i) \\ \hat{y}_i &> 0 \end{aligned} \quad (7)$$

where $\alpha_{i,1}$ is the vector of features to attack for the i^{th} data instance using the 1^{st} clause given in Equation 5. As seen above, in addition to the original formulation in Equation 2, the two clause components are added and the problem is properly constrained for a single perceptron. If the problem is still feasible, then we should be able to find α_i . For the second clause, the last two constraints will be substituted with the components from Equation 6. Eventually, for a system that has k number of clauses, we can determine α_i^{\max} that will maximize the prediction error as following:

$$\alpha_i^{\max} = \arg \max_{\alpha_{i,1}, \dots, \alpha_{i,k}} [1 - y_i F(x_i \circ (1 - \alpha_i))]_+ \quad (8)$$

For NN with k neurons, there will 2^k clauses since the activation state of each neuron is independent of each other. To find an optimal solution, we need to go through all clauses and compute the α and the corresponding prediction error, and test whether there a switch in the labels upon prediction. For large networks, this search is extremely exhaustive and infeasible as the number of clauses is growing exponentially. Instead, we can go through each clause until we find a solution that will cause the misprediction, labeling an attack event as normal. In other words, we don't need to maximize the error among all clauses as the maximization problem presumes. Instead, we only need one clause that will cause mislabeling. In the worse case, this approach will still go through all possibilities. Most of the time, if the problem is feasible, the solution is usually found when all RELUs is activated. Algorithm 1 outlines the core idea of the proposed attack model.

3.2 Defense Model

Usually, the simplest solution to defend against the adversarial feature deletion would be training the classifier with the adversarial examples [16]. Ideally, we want to train the network based on the worse prediction error over the entire training set [15]. Then, we can formalize the defense model as a minimization problem:

$$\begin{aligned} w &= \arg \min \frac{1}{n} \sum_{i=1}^n [1 - y_i \hat{y}_i]_+ \\ &= \arg \min \frac{1}{n} \sum_{i=1}^n [1 - y_i F(x_i \circ (1 - \alpha_i^{\max}))]_+ \end{aligned} \quad (9)$$

We can rewrite this problem more explicitly as given below:

$$\min_w \max_{\alpha_1, \dots, \alpha_n} \frac{1}{n} \sum_{i=1}^n [1 - y_i F(x_i \circ (1 - \alpha_i))]_+ \quad (10)$$

Algorithm 1: Proposed Attack Model

Input: $(x_i, y_i), w, F(x)$
Output: α_i

- 1 Generate DNF clauses for the given weights of the network
- 2 **foreach** DNF clause set **do**
- 3 Assign clause components as constraints to Equation 2
- 4 Solve Equation 2 with new constraints
- 5 **if** Problem is infeasible **then**
- 6 **continue** with the next clause set
- 7 **else**
- 8 Obtain α_i
- 9 Predict the label $\rightarrow \hat{y}_i = F(x_i \circ (1 - \alpha_i))$
- 10 **if** $\hat{y}_i == \text{normal}$ **then**
- 11 /* there is a successfully attack! */
- 11 **continue** with the next input (x_{i+1}, y_{i+1})
- 12 **if** $\hat{y}_i == \text{normal for all DNF clause sets}$ **then**
- 13 /* there is no successfully attack! */
- 13 $\alpha_i = 0$
- 14 **continue** with the next input (x_{i+1}, y_{i+1})

This problem is now, in essence, a mini-max where our aim is finding the weights that will minimize the average maximum prediction error caused by the adversarial examples. There are few important considerations regarding the training. In the previous section, it has been argued that finding the optimal solution for the worse attack can be sometimes challenging, and a sub-optimal solution can be used as an attack if it is powerful enough to cause a misclassification. A similar logic can be pursued here where we can train the network with the sub-optimal yet powerful adversarial examples. Here, there are three strategies to pursue adversarial training:

- **One-shot Training:** First, we train an NN with the given data set which constitutes the baseline reference. Let's denote this network as $F(x)$. Then, we compute α for each example with given $F(x)$ and derive the adversarial example dataset, \hat{x} . Finally, we train a new network using \hat{x} as the input dataset. This approach is used mainly in [16].
- **Iterative Training:** Here, we start with a NN with random weights. At the beginning of each batch or epoch, we compute α for each example with given $F(x)$ and derive the adversarial example dataset to be used in the training until next batch or epoch. After enough training iterations, the weights of the NN are expected to converge. Iterative training is similar to Bundle Methods for Regularized Risk Minimization (BMRM) approach discussed by [38] and [15].
- **Training and Retraining:** We perform one-shot training and obtain a NN. We perform an attack on the training data using this NN and derive the adversarial example dataset. Then, we train a completely new NN. We perform this process until network performance converges.

This paper focuses on the one-shot training and the effectiveness of other training methods is left for future research.

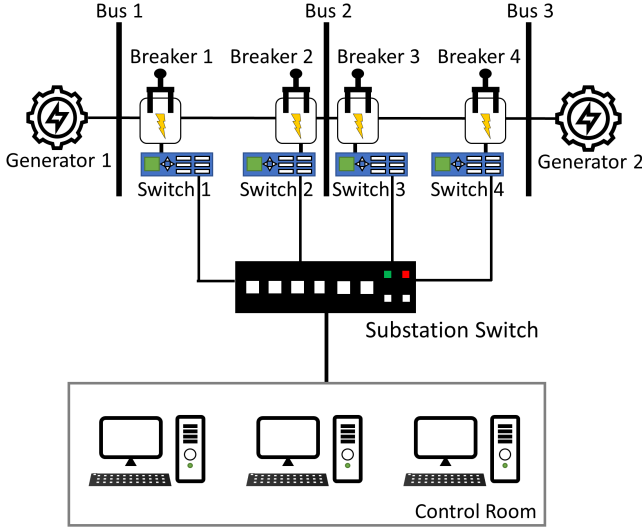


Figure 1: ICS overview, adopted from [18]

4 EVALUATION

4.1 Dataset

The efficiency of the proposed DoS attack and defense scheme is investigated using a dataset that includes measurements from an electric transmission system network [18]. This network contains four breakers, each controlled by an intelligent electronic relay (IED) individually. The breakers can be manually and automatically tripped. A substation switch monitors all the relay and breakers and sends this information to a SCADA system (see Figure 1). An IED is expected to alert the operator and trip the breaker when an abnormal (natural or man-made) event occurs. However, SCADA does not have an internal validation detection system to assess if the fault is fake or valid. Additionally, data can be multi-dimensional and complex enough such that human operator may not be able to detect abnormal events manually either. The purpose of the original dataset is to provide reference data for researchers to develop effective cyber-attack classification methods that can discriminate power system disturbances. Accordingly, the aim of the DoS attack explained in this paper is disguising attack events, such that the SCADA will not be able to discriminate against cyber-attacks and the network will operate under undesirable conditions.

The original dataset contains six types of scenarios categorized under normal and attack events. Those are *i)* normal operation, short-circuit fault, and line maintenance for normal events summing up to 9 scenarios; and *ii)* remote tripping command injection, relay setting change, data injection for 30 different attack scenarios. There are 128 features for each record in the dataset. There are 4 phasor measurement units (PMU) measuring the electrical waves on the electricity grid, each measuring 29 features summing up to 116 features. Those features include phase angle and magnitude values for current and voltage; frequency and impedance values for relays. Additionally, there are 12 columns for control panel logs. The data does not contain any time-stamp information. Finally, each data entry contains a label marking if an event is normal or

attack. We focus on a subset of this dataset for training and testing. The *training dataset* contains 880 normal events and 3092 attack instances. Likewise, for the *testing dataset*, 220 normal events and 774 attack events are used. The ratio of normal events to attack events is about 28 percent.

In our attack model, the adversary has access to the sensor readings. A DoS attack defines the interruption between the sensor and the control room. We assume a DoS attack will result in a zero sensor reading in the control room. The discriminator evaluates this data to predict the correct class.

4.2 Implementation

This study uses a single-hidden-layer neural network to discriminate against cyber-attacks. The neural network is trained with the *training dataset* using MLPClassifier from scikit-learn v0.22 [35] running on Python v3.6.7 [41]. The hidden layer has 5 neurons and RELU is used as the activation function for all hidden neurons. This network is optimized using Limited-memory BFGS while training. The network is denoted as the Original Model. At this stage, the accuracy of the Original Model for training and testing dataset constitutes the baseline.

For the attack model formulation, we assumed the output layer is linearly activated. When the network contains n neurons for the hidden layer, there are 2^n independent RELU activation state. In other words, there are 2^n clauses we should go through, to compute α for each data instance. We only focused on data instances that are labeled as (ground-truth) *attack* since the ultimate aim is disguising only the cyber-attacks. At each data instance, x_i , for each clause, we adopted Equation 7 with the respective constraints. We went through all the clauses and computed α_i until the label predicted by the Original Model is switched from *attack* to *normal* for a given $x_i \circ (1 - \alpha_i)$ as prescribed in Algorithm 1. Once we found an adversarial example, we stopped further solution-seeking process for the particular data instance and moved on to the next data instance, x_{i+1} . The MILP problem for each clause is formulated using a Python-based modeling language for convex optimization problems, CVXPY v1.0 [2, 10]. CVXPY allows utilizing Gurobi Optimizer as the solver for MILP problems [27]. To accelerate the attack modeling, we parallelized the optimization over 8 cores. For an Intel 9900K CPU, finding attacks over the entire training dataset usually takes less than 3 minutes.

After we go through all the data instances and compute α for each instance, we obtained the modified data set, \hat{x} . At this stage, the accuracy of the undefended network, Original Model is tested with \hat{x}_{test} derived from the original testing dataset. This accuracy shows how vulnerable the Original Model is against attacks.

For the adversarial training, we trained a new network, Resilient Model with the same properties as the first model. However, here, the modified training dataset (\hat{x}_{train} derived from the original dataset) is used for the adversarial training. The accuracy of the defended model is tested using a new modified dataset to demonstrate the improvement of the defense model against attacks. This set is derived from x_{test} by attacking to the Resilient Model. Additionally, we tested the generalizability of the Resilient Model with original datasets.

4.3 Results and Discussion

For this study, we attacked a shallow network with 5 neuron on a single hidden layer using a variety of attack budgets ($K = 1, 3, 6$) corresponding to 1, 2.5 and 5 percent of all features. To evaluate the attack and defense success, we computed the accuracy of the undefended (Original Model) and the defended (Resilient Model) networks against original (x_{train} and x_{test}) and modified (\hat{x}_{train} and \hat{x}_{test}) datasets.

Table 1 summarizes the prediction accuracy values of the Original Model for the efficiency of the attack when it is introduced to the original dataset using the Original Model. Here, the Original Model (baseline) is first trained with the original training dataset. After training, the Original Model has an accuracy above 80 percent for both training and testing datasets. The success of the model is comparable to the classifiers tested by [18]. Then, we introduced the attack to the original testing dataset with various budgets. For this attack, we used the Original Model network structure when computing the features to be deleted. Eventually, we derived the modified testing dataset for each budget. When the attack was launched with a budget of $K = 1$ (i.e. the adversary can only disrupt one sensor), the accuracy decreased to about 30 percent (see *Modified Testing Dataset* ($K = 1$)). This observation clearly shows that the attack is very efficient even under a very constrained budget. For higher budgets ($K = 3$ and $K = 6$), the attacker can generate more severe attacks that can reduce the prediction accuracy even further. For budgets higher than 6, almost all *attack*-labeled events are predicted as *normal*. Overall, results show that for the given Original Model, the attack scheme can effectively disguise the attacks such that the classifier cannot properly discriminate against the modified entries.

Table 1: Efficiency of the attack model on the Original Model

Dataset	Accuracy in Percentage		
Original Training Dataset	87.47		
Original Testing Dataset	83.23		
	K = 1	K = 3	K = 6
Modified Testing Dataset	31.08	16.29	12.77

Next, the efficiency of the defense model is summarized in Table 2. Here, the Resilient Model is trained with an adversarial training dataset derived from the original training data set. Namely, we introduced attacks with various budgets on the original training dataset using the Original Model network structure. Then, we aggregated 25 percent of the original training dataset and 75 percent of the attacked dataset. This new dataset constitutes the adversarial training dataset for the Resilient Model. The accuracy of the Resilient Model is above 85 percent for the three budgets. Additionally, we tested the generalizability of the Resilient Model against the original training and testing dataset. For both original sets, accuracy is above 80 percent which is consistent with the results from Table 1. Please note that prediction error is slightly larger for the Resilient Model when original datasets are considered (about 1 to 3 percent). This implies that the increased resiliency may cause a minor drop in the

overall prediction performance of the classifier. As for resiliency, we launched DoS with various budgets on the original testing dataset using the Resilient Model network structure under various budget constraints. For a budget of $K = 1$, the accuracy of the Resilient Model is about 39 percent. The increase of the accuracy relative to the Original Model (~ 31 percent for $K = 1$) corresponds to 8 percent. When the budget increases, the prediction performance of the Resilient Model decreases as the DoS attack has more room to be efficient. On the other end, the accuracy of the Resilient Model is always higher than the one of the Original Model against attacks for all budgets. The improvement varies between 7 to 10 percent.

Table 2: Efficiency of the defense model on the Resilient Model

Dataset	Accuracy in Percentage		
	K = 1	K = 3	K = 6
Adversarial Training Dataset	86.12	86.70	88.06
Original Training Dataset	85.14	85.32	86.58
Original Testing Dataset	81.89	82.69	80.78
Modified Testing Dataset	39.23	26.05	19.51

The confusion matrices for the Original Model before and after the DoS attack over the Testing Dataset ($K = 1$) are given in Tables 3, and 4, respectively. Similarly, Table 5 presents the confusion matrix for the Resilient Model against DoS attacks over Testing Dataset. Accordingly, the Original Model predicts 708 out of 774 attack events as *attack* which corresponds to 90 percent correct labeling. When the Original Model is attacked by DoS, the class for 506 attack events switched to *normal* label summing to 572 false positives. As a result, only 26 percent of the attack events are predicted correctly. After retraining using the defense scheme, the Resilient Model was able to predict 266 out of 774 attack events correctly and the recall value for the attack events increased to 34 percent. The increase in recall demonstrates that the defense mechanism may improve the prediction accuracy and reduce the impact of cyber-attacks on the performance of the grid systems, albeit the defense is limited.

Table 3: Confusion Matrix for Original Model before attack

		Prediction	
		Normal	Attack
Actual	Normal	107	113
	Attack	66	708

Table 4: Confusion Matrix for Original Model after attack

		Prediction	
		Normal	Attack
Actual	Normal	107	113
	Attack	572	202

Table 5: Confusion Matrix for Resilient Model

		Prediction	
		Normal	Attack
Actual	Normal	124	96
	Attack	508	266

Overall, results show that the attack model is very efficient in deceiving the classifier. While the defense model increased the prediction performance against DoS attacks, the improvement is limited. It is possible that one-shot training strategy generates a resilient network that can only defend against a limited number of DoS attacks. For example, the accuracy of the Resilient Model against Modified Testing Set created for the Original Model yields about 80 percent (not shown in this work). But the accuracy of the Resilient Model against Modified Testing Set created for the Resilient Model is still low. This shows the Resilient Model cannot generalize well over all types of DoS attacks. Iterative training or training-retraining strategies discussed in Section 3.2 may produce classifiers more robust against adversarial feature deletion.

5 LITERATURE REVIEW

5.1 Vulnerabilities of ML

ML algorithms have some vulnerabilities, which makes them unsafe for security critical applications. Adversaries may use different attacks depending on their objectives [23].

- Poisoning attacks affect the training phase to contaminate the model generated and misclassify specific examples. For example, an adversary may mimic legitimate applications (e.g., traffic behavior) so that systems that learn signatures (e.g., antivirus) learn to filter out legitimate applications [9].
- Model inversion attacks allow the recovery of the (private) features used in the model. Previous works have shown that attackers can infer sensitive information (e.g., facial images) from facial recognition systems (provided that they know the person's name) [14].

- Perturbation attacks modify the inputs to get a desired response (bypass content filters or malware detection algorithms). For example, it is possible to attack voice controlled devices through inaudible commands [7, 46]. Likewise, adversaries can induce misclassification of images by adding noise to them [16]. Lastly, [12] modify physical traffic signs to mislead the algorithms that classify them.

An adversary needs limited information to craft *adversarial examples*, the instances/features/inputs designed to mislead ML models. This happens because the attacks are *transferable*, that is, adversarial examples that affect a particular model often affect other models that perform the same task, even if they have different architectures [19, 33]. Thus, an adversary can train a substitute model and find adversarial examples to attack the victim's model.

In some cases, the adversary does not need access to model nor the data set used in training. An adversary able to submit queries to the victim's model create a data set using the labels that the victim's model assigns to some chosen instances.

Most of the research on adversarial examples focus on classification problems; however, other applications also have vulnerabilities. [6, 8] show how an adversary can manipulate sensor measurements to increase or decrease load forecasts.

5.2 Defenses

Works in the literature have explored several mechanisms to mitigate the impact of adversarial examples. Some works train the models using adversarial examples to anticipate the adversary's actions [28]. Likewise, [45] use bounded RELU activation neurons and augment the data set adding Gaussian noise to the inputs to limit the effect of perturbations and increase the generalization capabilities.

Some papers design robust NNs in image classification applications introducing randomness in the system. For example, [44] modifies images randomly (e.g., rotate or scale the images) before their processing. Likewise, [24] injects noise in the inputs using *differential privacy* (DP) to guarantee that it will induce bounded changes in the output, preventing the misclassification. [25] adds noise in the layers of the NN. In this way, a single NN acts as multiple models, which combined conform an ensemble of models.

Some papers estimate the robustness of models to attacks examining their sensitivity to changes in the inputs. In particular, *certified defenses* measure the Lipschitz constant of the models, which gives an upper bound on how fast the output changes with respect to changes in the inputs [17, 42]. In other words, the Lipschitz constant gives a measure of the sensitivity of the model, which gives information of the perturbation that the model can withstand. Hence, it is possible to improve the resiliency of models regularizing the training to reduce the Lipschitz constant, i.e., make the model less sensitive to attacks.

6 CONCLUSIONS

With rapidly growing demand for electricity, the power grid systems became a critical infrastructure and backbone of the US economy. Today, with the integration of the Internet, the power systems serve as a cyber-physical platform capable of accessing and monitoring multiple sensors and controlling the state of the operation quality.

for the continued delivery of electricity. While such interconnected systems are exposed to new vulnerabilities, such as cyber-attack, internal detection mechanisms, such as classifiers exist to discriminate adversaries.

The aim of this study is deceiving attack detection systems using Denial-of-Service type attacks. By interrupting the communication between the control room and sensors using DoS attack, it is possible to disguise a more mature cyber-attack such that the detection system fails to predict the correct label for the event. To achieve this aim, we formulated an attack model utilizing DoS. According to this model, the objective of the problem is finding the most effective features to *delete* to maximize the prediction error, such that the classifier will mispredict an attack event as a normal event. This is essentially a MILP for linear classifiers. For more complex classifiers, such as multi-perceptron neural networks with nonlinear activation functions, MILP is not applicable as the solution space is not convex. However, for neural networks specifically with RELUs, we can write the network in terms of logic formulas using Disjunctive Normal Form. For each clause in DNF, we can describe a convex MILP with proper constraints, and determine the features to delete. Using this approach, we can optimize the search for a DoS attack.

Another aim of this paper is defending the classifier against DoS attacks such that the predictions can be still accurate under DoS attack. The essential idea of the defense model is minimizing the maximum prediction error caused by the attack model. Thus, it can be considered as a minimax problem. For the defender model, we proposed three defense strategies, (one-shot, iterative, and train-and-retrain) and focused on the first one. In the one-shot scheme, we train the network with the adversarial attacks obtained from the attack model.

To evaluate the results, this study utilized a power system dataset specifically created for the development of power disruption and cyber-attack classification. This dataset contains a set of sensor measurement collected from multiple PMU along with a label indicating if the record is normal or attack. Using this dataset, we developed a shallow neural network with 5 neurons. Additionally, we derived an attacked dataset using the proposed attack model. Three different attack budgets have been tested corresponding to 1, 2.5, and 5 percent of the total number of features. The overall results show that the network is prone to DoS attack even under limited budget constraints. The low accuracy implies that the attack detection cannot predict the labels under DoS attack properly. To defend against DoS attacks, we trained a new network with one-shot adversarial training approach proposed in the defense model. As a result of this, the prediction accuracy against attacks increased but the improvement was limited.

Overall, this paper has shown that a power system can be attacked using DoS, and a classifier may be deceived. Equally, it is possible to defend against such attacks by training a new network using the defense scheme up to some point. It should be noted that the attack scheme discussed here assumes the adversarial actor has access to the modeling parameters of the machine learning classifier. While, in reality, the attacker may not have full access to the system all the time, we assume that the system is eventually prone to white-box attack. For future research, we want to explore deeper and more complex networks to obtain better generalization and

higher accuracy values for the defense. Additionally, the effectiveness of the attack should be investigated for large networks, since it may be time-consuming to find attacks due to the exponentially growing worst-case time complexity of the attack model. As an alternative adversarial training strategy, the iterative training and train-and-retrain procedure should be investigated and resistance of those training schemes against adversarial examples should be studied. Last but not least, we want to compare the DoS attacks to more complicated attack schemes such as false data injection attacks in terms of the cost of devising the attack and its effectiveness.

7 ACKNOWLEDGMENTS

This work was supported in part by the National Institute of Standards and Technology under Grant 70NANB18H198. The conclusions of this research represent solely that of the authors'.

REFERENCES

- [1] 2009. IEEE Recommended Practice for Monitoring Electric Power Quality. *IEEE Std 1159-2009 (Revision of IEEE Std 1159-1995)* (June 2009), 1–94. <https://doi.org/10.1109/IEEESTD.2009.5154067>
- [2] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2018. A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision* 5, 1 (2018), 42–60.
- [3] Simon Duque Anton, Daniel Fraunholz, Christoph Lipps, Frederic Pohl, Marc Zimmermann, and Hans D Schotten. 2017. Two decades of SCADA exploitation: A brief history. In *2017 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE, 98–104.
- [4] Mordecai Avriel. 2003. *Nonlinear programming: analysis and methods*. Courier Corporation.
- [5] Abdelrahman Ayad, Hany EZ Farag, Amr Youssef, and Ehab F El-Saadany. 2018. Detection of false data injection attacks in smart grids using recurrent neural networks. In *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 1–5.
- [6] Carlos Barreto and Xenofon Koutsoukos. 2019. Design of Load Forecast Systems Resilient Against Cyber-Attacks. In *Decision and Game Theory for Security*. Tansu Alpcan, Yevgeniy Vorobeychik, John S. Baras, and György Dán (Eds.). Springer International Publishing, Cham, 1–20.
- [7] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 513–530.
- [8] Yize Chen, Yushi Tan, and Baosen Zhang. 2019. Exploiting Vulnerabilities of Load Forecasting Through Adversarial Attacks. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems (e-Energy '19)*. ACM, New York, NY, USA, 1–11. <https://doi.org/10.1145/3307772.3328314>
- [9] Simon P Chung and Aloysius K Mok. 2006. Allergy attack against automatic signature generation. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 61–80.
- [10] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research* 17, 83 (2016), 1–5.
- [11] Sami Ekici. 2009. Classification of power system disturbances using support vector machines. *Expert Systems with Applications* 36, 6 (2009), 9859–9868.
- [12] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634.
- [13] Daniel S. Fava. 2018. Verification of Neural Networks via Linear Programming. <https://github.com/dfava/readingclub/wiki/Verification-of-Neural-Networks-via-Linear-Programming>.
- [14] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.
- [15] Amir Globerson, Choon-Hui Teo, Alexander Smola, Sam Roweis, et al. 2009. An adversarial view of covariate shift and a minimax approach. In *Dataset shift in machine learning*. MIT Press.
- [16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
- [17] Matthias Hein and Maksym Andriushchenko. 2017. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*. 2266–2276.

- [18] Raymond C Borges Hink, Justin M Beaver, Mark A Buckner, Tommy Morris, Uttam Adhikari, and Shengyi Pan. 2014. Machine learning for power system disturbance and cyber-attack discrimination. In *2014 7th international symposium on resilient control systems (ISRCs)*. IEEE, 1–8.
- [19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial Examples Are Not Bugs, They Are Features. *arXiv preprint arXiv:1905.02175* (2019).
- [20] JQ James, Yunhe Hou, and Victor OK Li. 2018. Online false data injection attack detection with wavelet transform and deep neural networks. *IEEE Transactions on Industrial Informatics* 14, 7 (2018), 3271–3280.
- [21] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.
- [22] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 97–117.
- [23] Ram Shankar Siva Kumar, David O'Brien, Kendra Albert, Salomé Viljōen, and Jeffrey Snover. 2019. Failure Modes in Machine Learning Systems. *arXiv preprint arXiv:1911.11034* (2019).
- [24] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2018. Certified robustness to adversarial examples with differential privacy. *arXiv preprint arXiv:1802.03471* (2018).
- [25] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 369–385.
- [26] Zhigang Liu, Yan Cui, and Wenhui Li. 2015. A classification method for complex power quality disturbances using EEMD and rank wavelet SVM. *IEEE Transactions on Smart Grid* 6, 4 (2015), 1678–1685.
- [27] Gurobi Optimization LLC. 2019. Gurobi Optimizer Reference Manual. <http://www.gurobi.com>
- [28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
- [29] M McDowell. 2004. Security tip (ST04-015): understanding denial-of-service attacks. *National Cyber Alert System* (2004).
- [30] Xiangyu Niu, Jiangnan Li, Jinyuan Sun, and Kevin Tomsovic. 2019. Dynamic detection of false data injection attack in smart grid using deep learning. In *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 1–6.
- [31] Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.
- [32] Zakarya Oubrahim, Vincent Choqueuse, Yassine Amirat, and Mohamed El Hachemi Benbouzid. 2017. Disturbances classification based on a model order selection method for power quality monitoring. *IEEE Transactions on Industrial Electronics* 64, 12 (2017), 9421–9432.
- [33] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).
- [34] Ali Payani and Faramarz Fekri. 2019. Learning algorithms via neural logic networks. *arXiv preprint arXiv:1904.01554* (2019).
- [35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
- [36] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2009. *Dataset shift in machine learning*. The MIT Press.
- [37] Emmanouil Styvaktakis, Math HJ Bollen, and Irene YH Gu. 2002. Expert system for classification and analysis of power system events. *IEEE Transactions on Power Delivery* 17, 2 (2002), 423–428.
- [38] Choon Hui Teo, SVN Vishwanthan, Alex J Smola, and Quoc V Le. 2010. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research* 11, Jan (2010), 311–365.
- [39] Karthik Thirumala, Sushmita Pal, Trapti Jain, and Amod C Umarikar. 2019. A classification method for multiple power quality disturbances using EWT based adaptive filtering and multiclass SVM. *Neurocomputing* 334 (2019), 265–274.
- [40] M. Valtierra-Rodriguez, R. de Jesus Romero-Troncoso, R. A. Osornio-Rios, and A. Garcia-Perez. 2014. Detection and Classification of Single and Combined Power Quality Disturbances Using Neural Networks. *IEEE Transactions on Industrial Electronics* 61, 5 (May 2014), 2473–2482. <https://doi.org/10.1109/TIE.2013.2272276>
- [41] Guido Van Rossum and Fred L Drake Jr. 1995. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.
- [42] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578* (2018).
- [43] H Paul Williams. 2009. Integer programming. In *Logic and Integer Programming*. Springer, 25–70.
- [44] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. 2017. Mitigating adversarial effects through randomization. *CoRR* abs/1711.01991 (2017). [arXiv:1711.01991](http://arxiv.org/abs/1711.01991) <http://arxiv.org/abs/1711.01991>
- [45] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 39–49.
- [46] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 103–117.
- [47] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. 2011. A taxonomy of cyber attacks on SCADA systems. In *2011 International conference on internet of things and 4th international conference on cyber, physical and social computing*. IEEE, 380–388.