Adversarial Examples on Power Systems State Estimation

Ali Sayghe, Olugbenga Moses Anubi, Charalambos Konstantinou
Department of Electrical and Computer Engineering, FAMU-FSU College of Engineering
Center for Advanced Power Systems, Florida State University
Tallahassee, FL, USA
E-mail: {asayghe, oanubi, ckonstantinou}@fsu.edu

***Abstract*—The number of cyber-attacks targeting power system infrastructures is increasing at an alarming rate. Among them, False Data Injection Attacks (FDIAs) can disturb the normal operation of state estimation routines in power systems and potentially lead to outages. Several studies utilize machine learning algorithms to detect FDIAs with high accuracy. However, such algorithms can be susceptible to adversarial examples able to lower the accuracy of the detection model. Adversarial examples are crafted inputs intentionally designed to falsify machine learning algorithms. In this paper, we examine the effect of adversarial examples on machine learning algorithms that are used to detect FDIAs in state estimation. Specifically, we demonstrate the impact on Support Vector Machines (SVM) and Multilayer Perceptrons (MLP) against poisoning and evasion adversarial attacks. The algorithms are tested on IEEE 14 bus system using load data collected from the New York Independent System Operator (NYISO).**

***Index Terms*—False data injection attacks, machine learning, adversarial examples.**

I. Notation

The following notions and conventions are employed throughout the paper: $\mathbb{R}, \mathbb{R}^m, \mathbb{R}^{m\times n}$ denote the space of real numbers, real vectors of length m and real matrices of m rows and n columns respectively. $\mathcal{H}$ denotes the Hilbert space with the corresponding Hilbert norm $\|.\|_{\mathcal{H}}$. The particular Hilbert space used will be clear from context. $X^\top$ denotes the transpose of the quantity X. Normal-face lower-case letters ($x \in \mathbb{R}$) are used to represent real scalars, bold-face lower-case letter ($\mathbf{x} \in \mathbb{R}^m$) represents vectors, normal-face upper case ($X \in \mathbb{R}^{m\times n}$) represents matrices, while calligraphic upper case letters (e.g $\mathcal{S}$) represent sets.

II. Introduction

Malicious attacks against critical infrastructures could lead to catastrophic results. They can target various parts of the cyber-physical system and disturb its normal operating condition [1], [2]. For example, cyber-attacks on power systems through the Supervisory Control And Data Acquisition (SCADA) network could have severe effects as they could mislead operators into making erroneous decisions. SCADA systems continuously collect measurements from Remote Terminal Units (RTUs) to ensure normal operation via the Energy Management System (EMS). EMS performs various essential functions, including contingency analysis, optimal power flow, State Estimation (SE), etc. These routines assist the system and control operators in maintaining stable and secure grid operation.

SE estimates the state variables of the power system (the vector of the voltage magnitudes and angles at all network buses) from RTU measurements. Modules within SE can detect and remove any bad data measurements. However, SE can be susceptible to False Data Injection Attacks (FDIAs) in which adversaries aim to hack the readings of multiple RTUs in order to mislead the EMS decision making process [3]–[5]. Existing literature on FDIAs against SE focuses on developing different FDIA strategies [6], [7] as well as addressing the issue by introducing robust detection methods [8]–[10]. Among the FDIA detection methods, machine learning algorithms have been recently utilized due to their high effectiveness and accuracy [11]–[13]. Despite the demonstrated success of many data-driven methodologies used for detecting FDIAs, they remain vulnerable to adversarial examples [14]–[16].

Adversarial examples (also called adversarial attacks) are malicious inputs carefully designed to mislead machine learning algorithms. There exist two primary attack strategies: poisoning and evasion attacks. Poisoning attacks occur during the training phase in which the adversary provides incorrect training data to the machine learning routine. On the other hand, evasion attacks are staged during the testing phase in which the adversary attempts to misdirect the learning algorithms to make wrong decisions [17]. Based on the attackers' capabilities, the adversarial attack can be further classified as either a white-box or a black-box attack. In white-box attacks, the adversary is assumed to have full knowledge about the learning algorithm including the model, parameters, architecture, etc. In the black-box case, the attacker does not have prior knowledge of algorithms.

In this paper, we examine the effect of adversarial examples on machine learning algorithms designed to detect FDIAs. Specifically, we demonstrate the impact of poisoning and evasion attacks on Support Vector Machine (SVM) and Multilayer Perceptron (MLP) schemes. Our work shows that adversarial attacks on such algorithms dramatically reduce their accuracy of detecting FDIA for power systems SE routines.

The rest of the paper is organized as follows: Section III provides the problem description. Section IV describes the adversarial attacks on SVM and MLP. Section V presents the simulations results, while Section VI concludes the paper.

III. Problem Formulation

A. State Estimation and False Data Injection Attack

The linear (DC) SE model can be formalized as:

$$\mathbf{z} = H\mathbf{x} + \mathbf{e} \tag{1}$$

where $\mathbf{z} \in \mathbb{R}^m$ denotes system measurements, including active and reactive power flow, active and reactive power injection and voltage magnitudes, $\mathbf{x} \in \mathbb{R}^n$ denotes the state variables of voltage magnitudes and angles, and $\mathbf{e} \sim \mathcal{N}(0, \Sigma)$ denotes the measurement noise assumed to be normally distributed with zero mean and covariance matrix $\Sigma \in \mathbb{R}^{m\times m}$. One of the most widely used methods for computing the state estimates based on the measurement model above is the Weighted Least Squares (WLS) given by [18]:

$$\begin{aligned}\hat{\mathbf{x}} &= \operatorname{argmin} \|\mathbf{z} - H\mathbf{x}\|_W^2 \\ &= \left(H^\top W H\right)^{-1} H^\top W \mathbf{z}\end{aligned}$$

where $W = \mathsf{diag}\left\{\sigma_1^{-2}, \sigma_2^{-2}, \ldots, \sigma_m^{-2}\right\}$. Bad data can exist in the measurements due to meter malfunctions, defected sensors, communication errors, cyber-intrusions, etc. Residual-based approaches are commonly used to detect bad data measurements. Given a detection threshold $\tau \in \mathbb{R}$, $\mathbf{z}$ is declared a bad data if $J(\mathbf{z}) \triangleq \|\mathbf{z} - H\hat{\mathbf{x}}\|_W^2 \leq \tau$. The function $J(\mathbf{z})$ is expected to follow the Chi-squared distribution with degree of freedom of at most $m - n$ [19].

B. False Data Injection Attack

Let $\mathbf{z}_a \in \mathbb{R}^m$ represent the observed measurement vector poisoned with malicious data $\mathbf{a} \in \mathbb{R}^m$ as follows:

$$\mathbf{z}_a = \mathbf{z} + \mathbf{a}, \tag{2}$$

If $\mathbf{a}$ is comprised of a linear combination of the column vectors of H, then

$$\mathbf{a} = H\boldsymbol{\beta} \Rightarrow \mathbf{z}_a = H(\mathbf{x} + \boldsymbol{\beta}) \Rightarrow \hat{\mathbf{x}}_a = \mathbf{x}^* + \boldsymbol{\beta} \tag{3}$$

where $\boldsymbol{\beta}$ is a non-zero vector representing the impact on the estimate from the malicious injection and $\mathbf{x}^* \in \mathbb{R}^n$ is the unknown true state of the system. It is well known [3], [8] that the residual-based bad data detection method cannot detect this class of FDIA, as $J(\mathbf{z}_a) = J(\mathbf{z})$. In order to build the attack vector, we assume that the attacker has full knowledge of the system including the H matrix and system parameters [3].

C. FDIA Detection using Machine Learning Algorithms

Detecting FDIA is considered a supervised binary classification problem [11]. Given a prior data set $\mathcal{S} = \{s_i\}_{i=1}^M$, with M features, and $\mathcal{Y} = \{y_i\}_{i=1}^M$ labels, the goal of the binary classifier is to determine if new data are either normal measurements (negative class) or attacked measurements (positive class) [20]. In what follows, we expatiate on two specific machine learning algorithms for FDIA detection.

1) Support Vector Machine: SVM is a commonly used classifier often applied for the detection of FDIAs in SE. SVM constructs a hyperplane or sets of hyperplanes that are used to segregate safe and attacked measurements in a high, possibly infinite, dimension using nonlinear boundary generated from kernel functions. Given the data and labeled measurement sets $\mathcal{S}$ and $\mathcal{Y}$, the decision boundaries for the linear separable data are usually given by two parallel hyperplanes of the form $\mathbf{w}^\top \mathbf{s}_{1,2} + b = \pm 1$, where $\mathbf{w}$ and b are the optimal weight vector and bias which specify the position of the decision hyperplane in feature space, with corresponding *support vectors* $\mathbf{s}_1, \mathbf{s}_2$. The area located between the two boundaries is called the margin. It's width is given by $D = 2/\mathbf{w}^\top\mathbf{w}$. To increase the SVM margin, $\mathbf{w}^\top\mathbf{w}$ is often minimized. Consequently, linear SVM is obtained by solving the following optimization problem:

$$\begin{aligned}\textsf{Minimize}: \quad & \tfrac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{M_{Tr}} \xi_i \\ \textsf{Subject to}: \quad & y_i\left(\mathbf{w}^\top\mathbf{s}_i + b\right) \geq 1 - \xi_i, i = 1, 2, \ldots, M_{Tr},\end{aligned}$$

where M_{Tr} is the number of training samples and C is the cost function. The above problem is a quadratic program with the corresponding dual problem.

$$\begin{aligned}\textsf{Maximize}: \quad & \sum_{i=1}^{M_{Tr}} \Psi_i - \tfrac{1}{2}\sum_{i,j=1}^{M_{Tr}} y_i y_j \Psi_i \Psi_j \mathbf{s}_i^\top \mathbf{s}_j \\ \textsf{Subject to}: \quad & \sum_{i=1}^{M_{Tr}} \Psi_i y_i = 0, \\ & 0 \leq \Psi_i \leq C,\ i = 1, \cdots, M_{Tr},\end{aligned} \tag{4}$$

where the dual variables $\Psi_i \geq 0$ are the corresponding Lagrange multipliers for the constraints $y_i\left(\mathbf{w}^\top\mathbf{s}_i + b\right) \geq 1$ in the primal problem. The optimal weight vector is given by

$$\mathbf{w}^* = \sum_{i=1}^{M_{Tr}} y_i \Psi_i \mathbf{s}_i.$$

It is noteworthy that the dual problem is sparsity encouraging. Thus, $\Psi_i = 0$ exactly when $\mathbf{s}_i$ lies on the correct side of the resulting hyperplanes. The value of the bias is calculated by enforcing $y_k\left(\mathbf{w}^\top\mathbf{s}_k + b\right) = 1$ for the point $\mathbf{s}_k$ on the hyperplane:

$$b^* = {\mathbf{w}^*}^\top \mathbf{s}_k - y_k.$$

For non-linearly separable data, a kernel function $\kappa(\mathbf{s}_i, \mathbf{s}_j)$ is used, which helps the mapping of the non-linear data into a higher, possibly infinite, dimensional space. The associated primal and dual problems are:

$$\begin{aligned}\textsf{Minimize}: \quad & \tfrac{1}{2}\|\mathbf{w}\|_{\mathcal{H}}^2 + C\sum_{i=1}^{M_{Tr}} \xi_i \\ \text{Subject to}: \quad & \mathbf{w}\cdot\Phi(\mathbf{s}_i) + b \geq 1 - \xi_i\end{aligned} \tag{P}$$

$$\begin{aligned}\textsf{Maximize}: \quad & \sum_{i=1}^{M_{Tr}} \Psi_i - \tfrac{1}{2}\sum_{i,j=1}^{M_{Tr}} y_i y_j \Psi_i \Psi_j \kappa(\mathbf{s}_i, \mathbf{s}_j) \\ \textsf{Subject to}: \quad & \sum_{i=1}^{M_{Tr}} \Psi_i y_i = 0, \\ & 0 \leq \Psi_i \leq C,\ i = 1, \cdots, M_{Tr},\end{aligned} \tag{D}$$

where $\Phi : \mathbb{R}^m \mapsto \mathcal{H}$ maps from the input space to a given Hilbert space with Hilbert norm denoted by $\|.\|_{\mathcal{H}}$. The *kernel trick* is also used to express the dot product $\Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{s}_j)$ as $\kappa(\mathbf{s}_i, \mathbf{s}_j)$. In this paper, we use the Gaussian radial basis function, $e^{-\gamma\|\mathbf{s}_i - \mathbf{s}_j\|_2^2}$, $\gamma > 0$, as a kernel for SVM. This choice is widely used in literature and generally improves the separation boundary between normal and attacked measurements.

2) Multilayer Perceptron: MLP, also called feed-forward neural network, is a common type of an artificial neural network. It is made up of multiple layers of interconnected units called perceptron. MLPs have the capability of computing non-linearly separable functions. The main equation for a single perceptron is:

$$y = \varphi(\sum_{i=1}^{n} w_i s_i + b) \tag{5}$$

where y is the estimated output from the activation function, φ is the non-linear activation function, w is the weight, s_i is the input variable, b is the bias, and n is the input dimension. The activation function determines whether a neuron output should be considered or not by calculating the weighted sum of inputs and adding the bias to it. Sigmoid, hyperbolic tangent, and Rectified Linear Unit (RELU) are examples of activation functions [21]. MLP uses back-propagation training algorithms in which weights are updated by using gradient descent to minimize the error function:

$$E(\mathbf{w}, b, \{\mathcal{S}, \mathcal{Y}\}) = \sum_{j=1}^{M_{Tr}} \left(y_j - \varphi(\sum_{i=1}^{n} w_i s_{j_i} + b) \right)^2$$

MLP training process often requires large amount of data to achieve proper trade-off between generalization and training performance. Higher level metrics, such as the F_1-score, are often used to computationally balance these competing objectives.

IV. Adversarial Attacks

In this section, we highlight two specific methods for generating adversarial examples to falsify the ML algorithms described in the previous sections.

A. Adversarial Label Flipped Attack on SVM

Adversarial label flipped attacks are poisoning attacks in which the adversary contaminates the training data through flipping labels. It was introduced by Xiao *et al.* who experimentally explained how adversarial label flipped attack can reduce the accuracy of SVMs [15]. The classification hypothesis for a given hypothesis space $\mathcal{H}$ and a loss function V can be found by solving the Tikhonov regularized problem:

$$f_{\mathcal{S}} := \underset{f}{\operatorname{argmin}}\, \zeta \sum_{i=1}^{M_{Tr}} V(y_i, f(\mathbf{s}_i)) + \|f\|_{\mathcal{H}}^2 \tag{6}$$

where $f_{\mathcal{S}}$ the resulting classifier trained on the training set $\mathcal{S}$, ζ is a positive parameter to quantifying the objectives trade-off. In order to demonstrate the label flipped attack, we begin by initiating a set of variables $l_i \in \{0, 1\}, i = 1, \cdots, M_{Tr}$. Then, y_i is replaced with $y_i' := y_i\,(1 - 2l_i)$ so that if $l_i = 1$ then the label is flipped $y_i' = -y_i$, otherwise $y_i' = y_i$. The tainted training set $\mathcal{S}' := \{(\mathbf{s}_i, y_i')\}_{i=1}^{M_{Tr}}$ shares the same instances as the original training set with some flipped labels. The original problem formulated to identify the near-optimal label flips is a bi-level optimization problem which is difficult to solve. Hence, a more relaxed formulation was proposed by assuming that the attacker can only increase the loss rate of the classification algorithm on the initial training set. Let $\mathcal{A}$ and $\mathcal{B}$ be two sets of labeled instances, an auxiliary loss function is defined as:

$$g\,(\mathcal{B}, f_{\mathcal{A}}) := \zeta \sum_{(s,y)\in\mathcal{B}} V\,(y, f_{\mathcal{A}}(s_i)) + \|f_{\mathcal{A}}\|_{\mathcal{H}}^2 \tag{7}$$

which reflects the empirical loss incured by $f_{\mathcal{A}}$ over $\mathcal{B}$. Consequently, the label-flipping problem is then defined as:

$$\begin{aligned} \underset{l}{\text{Minimize}}: \quad & g\,(\mathcal{S}', f_{\mathcal{S}'}) - g\,(\mathcal{S}', f_{\mathcal{S}}) \\ \text{Subject to}: \quad & \textstyle\sum_{i=1}^{M_{Tr}} c_i l_i \le C, \quad l_i \in \{0, 1\}, \end{aligned} \tag{8}$$

where c_i is the cost of flipped label y_i. The label-flipping thereby obtained will result in highest loss rate under the original classifier $f_{\mathcal{S}}$ but lowest loss rate under the tainted classifier $f_{\mathcal{S}'}$. Furthermore, after introducing an indicator function for $\mathcal{S} \cup \mathcal{S}'$, namely $q_i \in \{0, 1\}, i = 1, \ldots, 2M_{Tr}$, the problem is further simplified to:

$$\begin{aligned} \underset{q,f}{\text{Minimize}}: & \zeta \sum_{i=1}^{2M_{Tr}} q_i \left[V\,(y_i, f\,(s_i)) - V\,(y_i, f_{\mathcal{S}}\,(s_i))\right] + \|f\|_{\mathcal{H}} \\ \text{Subject to}: & \sum_{i=M_{Tr}+1}^{2M_{Tr}} c_i q_i \le C \\ & q_i + q_{i+M_{Tr}} = 1, i = 1, \ldots, M_{Tr} \end{aligned}$$

Upon substitution of the loss function of the SVM, the above problem becomes a mixed-integer quadratic program. This is further relaxed by changing the binary constraints as $0 \le q_i \le 1$. The resulting problem is then decomposed into the following sub-problems which are then solved iteratively until convergence.

1) Sub-problem 1: First, fixing the indicator variables results in the quadratic program:

$$\begin{aligned} \underset{\mathbf{w},\boldsymbol{\epsilon},b}{\text{Minimize}}: \quad & \zeta \sum_{i=1}^{2M_{Tr}} q_i \epsilon_i + \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{Subject to}: \quad & y_i \left(w^\top s_i + b\right) \ge 1 - \epsilon_i, \\ & \epsilon_i \ge 0,\ i = 1, \ldots, 2M_{Tr} \end{aligned} \tag{9}$$

2) Sub-problem 2: Next, fixing the weights and optimizing over the relaxed indicator variables results in the linear program:

$$\begin{aligned} \underset{\mathbf{q}}{\text{Minimize}}: \quad & \zeta \sum_{i=1}^{2M_{Tr}} q_i\,(\epsilon_i - \xi_i) \\ \text{Subject to}: \quad & \sum_{i=M_{Tr}+1}^{2M_{Tr}} c_i q_i \le C \\ & q_i + q_{i+M_{Tr}} = 1, i = 1, \ldots, M_{Tr} \\ & 0 \le q_i \le 1, i = 1, \ldots, 2M_{Tr} \end{aligned} \tag{10}$$

By minimizing Eq. (9) and Eq. (10), the original objective will be decreased.

B. Targeted Fast Gradient Sign Method Attack on MLP

Targeted Fast Gradient Sign Method (TFGSM) is a target evasion attack where the adversary is adding noise in the same direction as the gradient of the classification model [16]. The TFGSM attack is the result of solving the following optimization problem [22]:

$$\begin{aligned} \text{Minimize}: \quad & d(\mathbf{s}_i, \mathbf{s}_i + \delta) \\ \text{Subject to}: \quad & \lambda(\mathbf{s}_i + \delta) = t \\ & \mathbf{s}_i + \delta \in [-1, 1]^{M_{Tr}} \end{aligned} \tag{11}$$

where δ is the added noise, λ is the MLP classification model with Sigmoid activation function, d is the distance between normal data and the adversarial example, and t is the target label (which is not the true label for s_i) needed to predict the adversarial example $s_i' = s_i + \delta$. The optimal linear noise can be formulated as follows:

$$\delta = \varepsilon . \text{sign}\left(\nabla_s E(\eta, s_i, t)\right) \tag{12}$$

where $\nabla_s E$ is the gradient of loss function with respect to s_i, $\eta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$ is the model parameter, and ε is a tuneable parameter controlling the tradeoff between added noise and the success rate of TFGSM attack. The crafted input s_i' is computed as follows:

$$s_i' = s_i + \varepsilon . \text{sign}\left(\nabla_s E(\eta, s_i, t)\right) \tag{13}$$

The TFGSM uses the gradient of the loss function to determine in which direction should the data be changed in order to minimize the loss function.

V. Simulations and Results

In this section, we experimentally evaluate the effects of adversarial examples against FDIA detection algorithms based on SVM and MLP on IEEE 14 bus test system using Matpower toolbox. Load data utilized in the simulation are gathered from NYISO [23]. We utilized actual load flow data profiles for 11 NY state regions recorded every five minutes from January 2018 until June 2018 with a total of 288 readings daily. In order to prepare the data for SE, each load bus of the IEEE 14 bus system is linked with one region of NYISO as shown in [4]. Then, we fit the normalized load data into the 14 bus case study. We run a power flow analysis to collect the true measurement sets. After, we generate the estimated state $\hat{x}$ from the true measurement sets. We prepare FDIA samples based on DC state estimation; the number of measurements k range from 1 to $m = 54$ for the 14 bus system. FDIAs are performed based on two different attack scenarios: random and targeted, as described in [3]. Two different sets of attacked measurements z_a are generated for one day to speed up the process with a total of 3168 labeled samples. To evaluate the performance of the detection algorithm, we measure the accuracy as the ratio of the total correct predicted observations over the total observations.

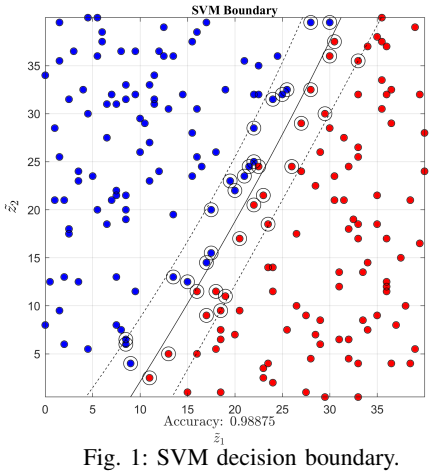

Fig. 1: SVM decision boundary.

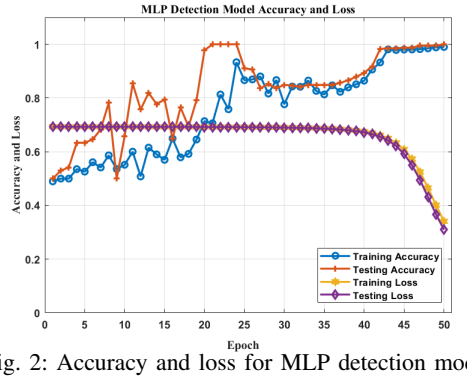

Fig. 2: Accuracy and loss for MLP detection model.

In the SVM detection model, two training sets are prepared based on two different FDIA scenarios: $\tilde{z}_1$ contains normal and random attacked measurements and $\tilde{z}_2$ contains normal and targeted attacked measurements. We consider a non-linear SVM with Gaussian radial basis function kernel due to the nature of training samples. For the non-linear SVM parameters C and γ, we empirically evaluated different values to analyze the detection accuracy of the SVM. For C and γ equal to 3 and 0.01, respectively, the accuracy rate reaches 98.8%. Fig. 1 presents the decision boundary between normal and attacked measurements: blue and red dots indicate the normal and attacked measurements, respectively.

The MLP networks are implemented by using Keras API in python with the TensorFlow library [24]. The Scikit library was used to determine the best parameters for the model throughout the parameter space [25]. The Exponential Linear Units (ELU) activation function is used for the hidden layers and Sigmoid for the output layer, which can be directly related to the output result of the class label. For the input and output layers, the number of neurons is fixed (depends on the dimension of the input data and the output data). Besides accuracy, cross-entropy loss or loss function is also used to measure the performance of an MLP binary classification model, where a perfect classifier model has a loss of 0. MLP achieved high performance with an accuracy rate of almost 99% and loss of ≈ 0.25, as illustrated in Fig. 2.

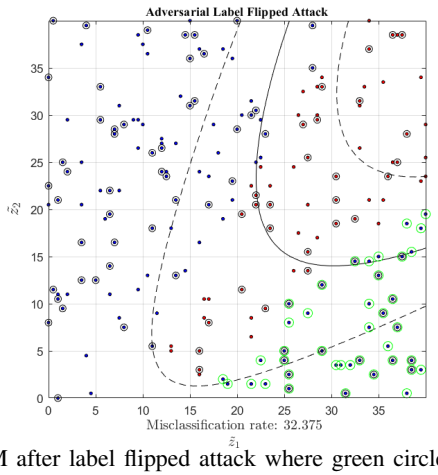

Fig. 3: SVM after label flipped attack where green circles represent label flipped instance.

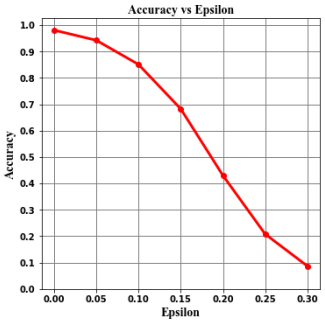

Fig. 4: Testing accuracy for different ϵ rates.

We performed the adversarial label flipped attack on SVM according to [15] with adversarial cost $c_i = 1$ for $i = 1, \cdots, n$. We randomly selected the same number of data samples with different classes. The SVM is trained with $\gamma = 1$ on the original data set. The classification error of the SVM is then evaluated on the test set. As a result, the accuracy of the SVM reduced to 67.62% with a flipped rate of 18%. Fig. 3 presents the SVM decision boundary between normal and attacked data; green circles indicate label flipped measurements.

For the TFGSM attack on MLP, we carefully generated a set of adversarial examples based on [16], and fed those instances to the MLP network in order to measure its effectiveness. The different ϵ values (Eq. (12)) are the only tunable parameter in the algorithm impacting the detection accuracy. Increasing the rate of ϵ will maximize the loss of MLP by adjusting the input data based on the same backpropagated gradients. Specifically, the gradient of the loss with respect to the input data is used and then adjusted to the input data maximizing the loss. Fig. 4 interprets the impact of ϵ in the testing accuracy.

VI. Conclusions

In this paper, we provided an overview of FDIAs data-driven detection models on power system SE against adversarial examples. We investigated two different attack scenarios against SVM and MLP: adversarial label flipped attack and TFGSM. Our results showed that such attacks significantly reduce the detection accuracy of machine learning algorithms. For future work, we will continue investigating more adversarial attacks on the FDIA detection model based on machine learning algorithms, and build a robust detection model.

References

[1] S. McLaughlin *et al.*, "The cybersecurity landscape in industrial control systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1039–1057, 2016.

[2] X. Liu and C. Konstantinou, "Reinforcement learning for cyber-physical security assessment of power systems," in *2019 IEEE Milan PowerTech*, June 2019, pp. 1–6.

[3] Y. Liu, Ning *et al.*, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[4] C. Konstantinou and M. Maniatakos, "A case study on implementing false data injection attacks against nonlinear state estimation," in *2nd ACM Workshop on CPS Security and Privacy*, 2016, pp. 81–92.

[5] ——, "Hardware-layer intelligence collection for smart grid embedded systems," *Journal of Hardware and Systems Security*, vol. 3, no. 2, pp. 132–146, 2019.

[6] Bobba *et al.*, "Detecting false data injection attacks on dc state estimation," in *1st Workshop on Secure Control Systems, CPSWEEK*, 2010.

[7] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1630–1638, 2016.

[8] L. Liu *et al.*, "Detecting false data injection attacks on power grid by sparse optimization," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 612–621, 2014.

[9] C. Konstantinou and M. Maniatakos, "A data-based detection method against false data injection attacks," *IEEE Design Test*, pp. 1–1, 2019.

[10] O. M. Anubi and C. Konstantinou, "Enhanced resilient state estimation using data-driven auxiliary models," *IEEE Transactions on Industrial Informatics*, 2019.

[11] M. Ozay *et al.*, "Machine learning methods for attack detection in the smart grid," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 8, pp. 1773–1786, 2016.

[12] Esmalifalak *et al.*, "Stealth false data injection using independent component analysis in smart grid," in *SmartGridComm, IEEE International Conference on*. IEEE, 2011, pp. 244–248.

[13] J. Yan *et al.*, "Detection of false data attacks in smart grid with supervised learning," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 1395–1402.

[14] X. Yuan *et al.*, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, 2019.

[15] H. Xiao *et al.*, "Adversarial label flips attack on support vector machines." in *ECAI*, 2012, pp. 870–875.

[16] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[17] Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[18] A. Monticelli, "Modeling circuit breakers in weighted least squares state estimation," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1143–1149, 1993.

[19] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, 2012.

[20] J. Yan *et al.*, "Detection of false data attacks in smart grid with supervised learning," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 1395–1402.

[21] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *13th nternational conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[22] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[23] "NYISO Home - NYISO." [Online]. Available: https://www.nyiso.com/

[24] "Deep Learning for humans." [Online]. Available: https://github.com/keras-team/keras

[25] Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.