Evasion Attacks with Adversarial Deep Learning Against Power System State Estimation

Ali Sayghe[*], **Junbo Zhao**[†], **Charalambos Konstantinou**[*]
[*]Department of Electrical and Computer Engineering, FAMU-FSU College of Engineering,
Center for Advanced Power Systems, Florida State University
[†]Department of Electrical and Computer Engineering, Mississippi State University
E-mail: {asayghe, ckonstantinou}@fsu.edu, junbo@ece.msstate.edu

Abstract—**Cyberattacks against critical infrastructures, including power systems, are increasing rapidly. False Data Injection Attacks (FDIAs) are among the attacks that have been demonstrated to be effective and have been getting more attention over the last years. FDIAs can manipulate measurements to perturb the results of power system state estimation without being detected, leading to potentially severe outages. In order to protect against FDIAs, several machine learning algorithms have been proposed in the literature. However, such methods are susceptible to adversarial examples which could significantly reduce their detection accuracy. In this paper, we examine the effects of adversarial examples on FDIAs detection using deep learning algorithms. Specifically, the impacts on Multilayer Perceptron (MLP) against two different adversarial attacks are investigated, namely the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) and the Jacobian-based Saliency Map Attack (JSMA). Numerical results tested on the IEEE 14-bus system using load data collected from the New York Independent System Operator (NYISO) demonstrate the effectiveness of the proposed methods.**

Index Terms—**State estimation, false data injection attacks, deep learning, adversarial examples.**

I. Introduction

A successful cyberattack on power system infrastructure could target various parts of the cyber-physical energy system, and thus disturb the normal power grid operation and even lead to catastrophic consequences [1]. In 2015, a cyberattack successfully compromised the information systems of three energy distribution companies in Ukraine and caused power outages that affected around 230,000 customers [2]. Following Stuxnet in 2010, cyberattacks targeting the power system became a global concern.

One of the critical parts of the power system that could be targeted is the Supervisory Control and Data Acquisition (SCADA) system. SCADA continuously collects measurements from Remote Terminal Units (RTUs). These measurement data is then used by State Estimation (SE) to estimate state variables of the current system topology. The state variables are represented as a set of voltage magnitudes and angles for each bus. The SE results are further leveraged by the Energy Management System (EMS) to perform various essential functions including contingency analysis, optimal power flow, etc.

The accuracy of any SE algorithm depends on the quality of measurements; any malicious ones, if not being detected, could have severe effects as they could mislead operators into making erroneous decisions. Indeed, studies have shown that SE can be vulnerable to False Data Injection Attacks (FDIAs) in which adversaries aim to hack multiple RTUs and inject malicious measurements to mislead the EMS decision making process [3]. The malicious measurements are well-coordinated so that they could not be detected by the widely used residual-based bad data detectors.

Current research on FDIAs against SE focuses on developing different attacking strategies [4], [5], as well as addressing the issue via robust detection methods [6]–[9]. Among the FDIA detection methods, machine learning algorithms have been recently utilized due to their effectiveness and proper detection accuracy [10]–[12]. The deep learning algorithms, which are a subset of machine learning algorithms [13]–[16], have also been investigated for FDIAs detection. Despite the demonstrated success of many data-driven methods for detecting FDIAs, they remain vulnerable to adversarial examples that can fool machine learning algorithms [17]–[19].

Adversarial examples (also called adversarial attacks) are well-designed malicious inputs to machine learning algorithms that could result in incorrect output. They can occur during multiple stages in the learning process. Depending on the learning stage, adversarial examples fall within two attack categories: poisoning attacks and evasion attacks. Evasion attacks are staged during the testing phase in which the adversary attempts to misdirect the learning algorithms to make wrong decisions [20]. On the other hand, poisoning attacks occur during the training phase in which the adversary provides incorrect training data to manipulate the training process. Depends on the attacker's capabilities, the attacker can manipulate testing data (which is called exploratory) or both training and testing data (which is called causative) [21]. Based on the adversary's knowledge, the attack can be further classified into the white-box and black-box. For the former one, the attacker is assumed to have full knowledge of the target classifier including the training data, feature sets, the learning algorithms, and the algorithm's parameters.

Adversarial examples have been widely investigated in image recognition applications with demonstrations of white-box and black-box attacks [17]–[19], [22]. Recently, there exist research efforts on performing adversarial examples on machine learning algorithms used for power system applica-

TABLE I
NOTATIONS.

Notation	Parameter
m	The number of measurements
n	The number of state variables
$\boldsymbol{H}$	$m \times n$ Jacobian matrix representing the topology
$\boldsymbol{x}$	$n \times 1$ vector of state variable
$\boldsymbol{z}$	$m \times 1$ vector of measurements
$\boldsymbol{e}$	$m \times 1$ vector of measurement errors
$\hat{\boldsymbol{x}}$	$n \times 1$ vector of estimated state variables
$\boldsymbol{W}$	$m \times m$ diagonal matrix, s.t., $w_i, i = \sigma_i^{-2}$, where σ_i^2 is the variance of the i-th measurement $(1 \leq i \leq m)$
τ	Threshold for L_2-norm based detection of bad measurements
$\boldsymbol{z_a}$	$m \times 1$ Measurement vector with bad measurement
$\boldsymbol{a}$	$m \times 1$ Attack vector, $s.t.$, $z_a = z + a$
$\boldsymbol{c}$	$n \times 1$ Vector of estimation errors $s.t.$, $a = Hc$

tions such as load forecasting and SE [23]–[25]. In this paper, we extent this direction to examine the effects of adversarial examples on deep learning-based algorithms developed to detect FDIAs. We demonstrate the impacts of two adversarial attacks on the Multilayer Perceptron (MLP) scheme, namely the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) and the Jacobian-based Saliency Map Attack (JSMA). We show that adversarial attacks could dramatically reduce the detection accuracy of the deep learning algorithms used for FDIAs detection.

The rest of the paper is organized as follows: Section II provides the problem description and formulation; Section III describes the adversarial attack methods on MLP; Section IV illustrates the simulation process and results, and finally Section V concludes the paper. Common notations used in the paper are listed in Table I.

II. Problem Formulation

A. State Estimation and FDIAs

The SE is a process of using redundant system measurements to estimate the most likely states of the system. The linear SE model is formalized as:

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

where $\boldsymbol{z} \in \mathbb{R}^m$ denotes measurement vector which could include active and reactive power flow, active and reactive power injection and voltage magnitudes, $\boldsymbol{x} \in \mathbb{R}^n$ denotes the state variables (voltage magnitudes and angles), and $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ denotes the measurement noise which is assumed to follow the Gaussian distribution. The Weighted Least Squares (WLS) is a commonly used method for SE that can be mathematically formulated as:

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

where $\hat{\boldsymbol{x}}$ is the estimated state; $\boldsymbol{H}_{m\times n}$ is the Jacobian matrix which denotes the functional dependency between the measurements and the state variables, and $\boldsymbol{W}$ is the corresponding measurement error covariance matrix:

$$\boldsymbol{W} = \text{diag}\left\{\frac{1}{\sigma_1^2}, \quad \frac{1}{\sigma_2^2}, \quad \cdots, \frac{1}{\sigma_m^2}\right\} \tag{3}$$

In order to detect bad data that may be caused by meter malfunctions, broken sensors, or even bad communication, the Chi-square statistical test is often used:

$$J(\boldsymbol{z}) \triangleq \|\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}\|_{\boldsymbol{W}}^2 \leq \tau \tag{4}$$

where the function $J(\boldsymbol{z})$ is assumed to follow the Chi-squared distribution at most $m - n$ degree of freedom [3]. $\tau \in \mathbb{R}$ is a pre-determine threshold.

FDIAs aim to inject a well-coordinated vector of bad data into meters without being detected and is formulated as:

$$\boldsymbol{z}_a = \boldsymbol{z} + \boldsymbol{a}, \tag{5}$$

where $\boldsymbol{a}$ is a non-zero vector and a linear combination of the column vector of $\boldsymbol{H}$ [3].

$$\boldsymbol{a} = \boldsymbol{H}\boldsymbol{c} \tag{6}$$

$$\boldsymbol{z}_a = \boldsymbol{H}(\boldsymbol{x} + \boldsymbol{c}) \tag{7}$$

$$\hat{\boldsymbol{x}}_a = \boldsymbol{x} + \boldsymbol{c} \tag{8}$$

The attacker can choose a random vector $\boldsymbol{c}$ to inject random false data into the meter as follows:

$$\begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ a_m \end{bmatrix} = c_1 \begin{bmatrix} H_{11} \\ H_{21} \\ \cdot \\ H_{m1} \end{bmatrix} + \cdots + c_n \begin{bmatrix} H_{1n} \\ H_{2n} \\ \cdot \\ H_{mn} \end{bmatrix} \tag{9}$$

Note that the carefully chosen vector $\boldsymbol{c}$ can manipulate specific measurements to the desired values without changing measurement residuals via:

$$\begin{aligned}\|\boldsymbol{z}_a - \boldsymbol{H}\hat{\boldsymbol{x}}_a\| &= \|\boldsymbol{z} + \boldsymbol{a} - \boldsymbol{H}(\hat{\boldsymbol{x}} + \boldsymbol{c})\| \\ &= \|\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}} + (\boldsymbol{a} - \boldsymbol{H}\boldsymbol{c})\| \\ &= \|\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}\| \leq \tau\end{aligned} \tag{10}$$

To build the attack vector, we assume that the attacker has full local knowledge of the interested areas including local $\boldsymbol{H}$ matrix and system parameters as in [26]. After completing the successful FDIAs, measurement samples $\boldsymbol{z}$ and $\boldsymbol{z}_a$ can be collected, labeled, and prepared for further analysis by the machine learning-based detection model.

B. FDIA Detection using Deep Learning Algorithms

In this paper, detecting FDIAs is considered as a supervised binary classification problem [27]. The objective of the binary classifier is to decide whether the given data $\mathcal{S} = \{s_i\}_{i=1}$ with M features and label $\mathcal{Y} = \{y_i\}_{i=1}$ is either $\boldsymbol{z}$, a normal measurement (negative class) or $\boldsymbol{z}_a = \boldsymbol{z} + \boldsymbol{a}$, an attacked measurement (positive class) for all $\boldsymbol{z}_a$ and $\boldsymbol{z} \in \mathcal{S}$. The output class labels are:

$$y = \begin{cases} +1 \text{ for } a_i \neq 0 \\ -1 \text{ for } a_i = 0 \end{cases} \tag{11}$$

where a_i is the i_{th} element of the attack vector. The distance between normal and attacked measurement vectors is defined by the attack vector in the measurement set $\mathcal{S}$ [27]. The distance between two unobservable attack vectors, i.e., $a_i = \boldsymbol{H}_i\boldsymbol{c}$ and $a_j = \boldsymbol{H}_j\boldsymbol{c}$ is computed as follows:

$$\left\|z_{a_i} - z_{a_j}\right\|_2 = \begin{cases} \left\|z_i - z_j\right\|_2 + \left\|a_i - a_j\right\|_2, \text{ if } i,j \in \mathcal{A} \\ \left\|z_i - z_j\right\|_2 + \left\|a_i\right\|_2, \text{ if } i \in \mathcal{A}, j \in \overline{\mathcal{A}} \\ \left\|z_i - z_j\right\|_2, \text{ if } i,j \in \overline{\mathcal{A}} \end{cases} \tag{12}$$

$$a_i \neq 0, \quad \forall i \in \mathcal{A} \tag{13}$$

$$a_i = 0, \quad \forall i \in \overline{\mathcal{A}} \tag{14}$$

where $z_{a_i} = z_i + a_i$, $z_{a_j} = z_j + a_j$, and $\mathcal{A}$ represents the indices of the measurements that will be attacked.

C. The Multilayer Perceptron (MLP)

The MLP, also called the feedforward neural networks or deep feedforward networks, is a network consisting of multiple layers of perceptrons. They are called feedforward because the information flows in one direction from the input through the hidden layer to the output. MLP has the capability of learning any mapping function, and it has been proven to be a universal approximation algorithm. The main equation for a single perceptron is as follows:

$$y = \varphi(\sum_{i=1}^{M_{Tr}} w_i s_i + b) \tag{15}$$

where y is the estimated output from the activation function, w is the weight, s_i is the input, b is the bias, and φ is the non-linear activation function. The activation function is used to define the output of the node for a given input or a set of inputs. Note that, the activation function is a critical feature of deep learning since it determines whether a neuron should be fired or not by calculating the weighted sum of inputs and adding a bias to it. MLPs utilize back-propagation training algorithms to update the weights by using gradient descent to minimize the following error function:

$$E(\boldsymbol{w}, b, \{\mathcal{S}, \mathcal{Y}\}) = \sum_{j=1}^{M_{Tr}} \left(y_j - \varphi(\sum_{i=1}^{n} w_i s_{j_i} + b) \right)^2 \tag{16}$$

The training process for deep learning algorithms requires a large amount of data to achieve a proper trade-off between generalization and training performance. Unlike other machine learning algorithms, deep neural networks are trained by using iterative, gradient-based optimizers, which drive the cost function to a very low value, and this allows increasing the number of correctly classified data points.

III. Proposed Methodology

In this section, we describe the adversarial attacks utilized against the MLP-based detection method. The attacks are based on the white-box methodology where the intruder has partial knowledge of the MLP detection model in order to construct the adversarial examples. In both attack scenarios, the adversary needs to have access to the testing samples to successfully attack the MLP [28], [29].

A. Limited-memory BFGS (L-BFGS)

Szegedy *et al.* defined the adversarial example as an input that is very similar to their real counterparts according to a distance metric [28], and it can cause a classifier to misclassify it. As part of this work, the L-BFGS is advocated to generate adversarial examples. Note that the generated adversarial examples could also be generalized to different models and different training datasets. The L-BFGS is a non-linear gradient-based numerical optimization algorithm that uses a limited amount of memory. It is a popular algorithm for parameter estimation in the machine learning field. Given a measurement sample s, an attacker can generate an adversarial example s' that is close to s as follows:

$$s' = s + \epsilon \tag{17}$$

Then, the following optimization problem is formulated:

$$\begin{aligned} \text{Minimize}: \quad & ||s - s^{'}||_2^2 \\ \text{Subject to}: \quad & f(s+\epsilon) = l \\ & s + \epsilon \in [0,1]^m \end{aligned} \tag{18}$$

where $||s - s^{'}||_2^2$ is the L_2-norm and l is the target class. The attacker's objective is to train the model on:

$$f(s') = l, s' \in [0,1]^m \tag{19}$$

where Eq. (19) is a non-linear optimization problem and mathematically difficult to solve. To this end, L-BFGS is utilized and the optimization problem is re-formulated as follows:

$$\begin{aligned} \text{Minimize}: \quad & \lambda \cdot ||s - s^{'}||_2^2 + \text{loss}_{F,l}(s') \\ \text{Subject to}: \quad & s + \epsilon \in [0,1]^m \end{aligned} \tag{20}$$

where λ is a hyper-parameter and $\text{loss}_{F,l}$ is the loss function. During the implementation process, the algorithm performs a binary search to find the optimal values for λ with multiple iterations. The L-BFGS algorithm can be used for estimating the inverse Hessian matrix to steer its search through the variable space. The memory parameters for the algorithm, i.e., the maximum number of correction pairs that defines the approximation of the Hessian matrix, are randomly selected and updated in every iteration.

B. Jacobian-based Saliency Map Attack (JSMA)

The JSMA is developed based on the L_0-norm distance [29], and its goal is to build a saliency map with the gradients. The saliency map is a standard method used to visualize the network. An adversarial saliency map indicates which features should be perturbed to most effectively achieve an adversarial attack. For the JSMA, the gradients are modeled based on the effect of each input feature, where the gradients are directly proportional to the probability that the input feature is correctly classified as the target class. The JSMA allows the attacker to find and select the most important feature that maximizes the gradient regarding the saliency map. Then, noise is added to the feature to increase the likelihood of labeling the measurements as the target class. The following steps describe the JSMA method:

1) Compute the forward derivative $\bigtriangledown f(s)$:

$$\bigtriangledown f(s) = \frac{df(s)}{ds} = \left[\frac{df_j(s)}{ds_i}\right]_{i\in 1\cdots m, j\in 1\cdots n} \tag{21}$$

2) Build up a saliency map D based on the forward derivative:

$$D^{+}(s_{(i)},\delta) = \begin{cases} 0 & if\ \frac{\partial df(s)_{(\delta)}}{\partial ds_i} < 0\ or \sum_{\delta\neq\delta'} \frac{\partial df(s)_{(\delta')}}{\partial ds_i} > 0 \\ -\ \frac{\partial df(s)_{(\delta)}}{\partial ds_i} \cdot \sum_{\delta\neq\delta'} \frac{\partial df(s)_{(\delta')}}{\partial ds_i} & \text{otherwise} \end{cases} \tag{22}$$

where $D^{+}(\cdot)$ measures how the feature $s_{(i)}$ is positively correlated with $\delta = y'(s)$ while being negatively correlated with all other classes $\delta' \neq \delta$. If either condition is violated, then saliency is reset to zero.

3) Adjust the most important feature based on the saliency map and repeat this process until the output becomes the target class. Note that the previous optimal value found for D is used as an initial value for the next attempt.

The non-target JSMA is another approach of the JSMA proposed by Wiyatno *et al.* [29]. Its idea is to increase the prediction confidence of the true class label instead of increasing the prediction confidence of the adversarial example's label.

IV. Simulation Results

In this section, we evaluate the effects of adversarial examples against FDIA detection algorithms based on MLP. All tests are performed on the IEEE 14-bus test system. The load data utilized in the simulation are gathered from the New York Independent System Operator (NYISO) [30]. We utilized the 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 [31]. Then, we fit the normalized load data into the 14-bus case study. We run a power flow analysis to collect true measurement sets and the true state vector $\hat{x}$. The FDIA samples are simulated based on the DC SE; the number of measurements k ranges from 1 to $m = 54$ for the 14-bus system. FDIAs are performed based on two different attack scenarios: random and targeted ones [3]. For the targeted attack, we simulated different injection amounts of the measurements represented within the vector $\boldsymbol{c}$ of Eqs. (7), (8), ranging from 2-5% in terms of the difference between true value and false value of the state variable.

Two different sets of attacked measurements of $\boldsymbol{z}_a$ are generated for one day to speed up the process with a total of 3168 labeled samples. To evaluate the performance of the detection algorithms, we measure the accuracy index as the ratio of the total correct predicted observations over the total observations. Recall and precision are also used for the JSMA attack since the data is unbalanced.

A. Detecting FDIAs based on MLP

The MLP networks are constructed and implemented using Keras API in python with the TensorFlow library. The Scikit library is also used to determine the critical parameters for the model throughout the parameter space. The MLP is trained with stochastic gradient descent using back-propagation. The Exponential Linear Units (ELU) activation function is used for the hidden layer while the Sigmoid is utilized for the output layer. The latter is directly related to the output result of the class label. For the input and output layers, the number of neurons is fixed (1 hidden layer of 100 neurons). The MLP hyper-parameters are selected by performing *GridSearch* and utilizing Python's Scikitlearn library. *GridSearch* is an optimization technique that constructs and evaluates the MLP model for each combination of parameters and finds the optimal hyper-parameters for the model [32]. Besides the accuracy index, the cross-entropy loss or loss function is also used to measure the performance of the MLP binary classification model, where a perfect classifier model has a loss of 0. The test results are shown in Fig. 1. It can be seen that MLP is able to achieve an accuracy rate of almost 99% with a loss of ≈ 0.1.

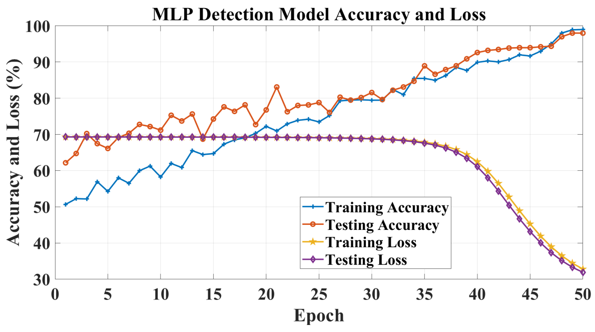

Fig. 1. Accuracy and loss for MLP detection model.

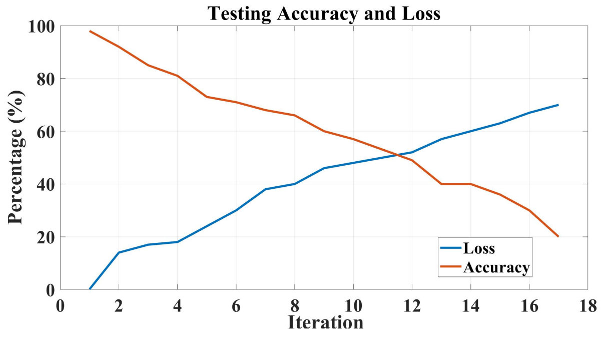

Fig. 2. Testing loss and Accuracy after L-BFGS.

B. Limited-Memory BFGS

In this section, we show how the L-BFGS adversarial attack can pose significant challenges to the MLP detection algorithm. Specifically, the L-BFGS adversarial attacks are first performed on the testing data. For the limited memory matrix, the number of corrections pairs and iterations are set to be 15 and 17, respectively. This allows us to yield the maximum effects on FDIAs detection accuracy. Pytorch is used to model the network and extract the gradients with respect to the input features. Next, a bisection search is performed to determine the optimal value for ϵ. The test results are shown in Fig. 2.

TABLE II
TESTING ACCURACY, RECALL AND PRECISION VS. DIFFERENT δ AMOUNT.

δ	Testing Accuracy (%)	Recall (%)	Precision (%)
0.2	87.65 %	80.02 %	100 %
0.3	79.77 %	83.3 %	89.3 %
0.02	66.32 %	79.5 %	78.13 %
0.03	50.09 %	63.4 %	70.66%
0.01	10.76 %	40.32 %	55.79%

The optimal value of ϵ is applied to a small amount of the testing samples. We observe from Fig. 2 that at each iteration the loss function is increasing, meaning that the detection accuracy is gradually decreasing. After 15 iterations, the detection accuracy reaches $\approx 20\%$. This indicates that the detection results from MLP are no longer reliable.

C. Jacobian-based Saliency Map Attack (JSMA)

The non-target JSMA attack is also used to investigate the impacts of adversarial attacks on MLP detection performance. Unlike the target JSMA, the non-target JSMA attack formulation does not depend on any specific class irrespective of increasing or decreasing the feature values. First, we train the MLP model f on the original training dataset s. Then, the adversarial data samples s' based on the output from the saliency map D are generated. This indicates that the valuable features are modified by adding δ amount to them (see Eq. (22)). Table II shows the effects of different amounts of δ on the critical features. Note that the less amount of δ, the higher impacts on the MLP detection accuracy will be. Despite that MLP is able to achieve very high detection accuracy (Fig. 1), its performance has been dramatically reduced after the non-target JSMA, yielding detection accuracy to be around 10% with recall almost 40%.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we evaluated the robustness of MLP for the detection of FDIAs in the presence of adversarial examples. Two different adversarial attack methods are investigated, namely L-BFGS and JSMA. Our results show that adversarial attacks could dramatically reduce the detection accuracy of the MLP method. For future work, we will investigate more adversarial attacks considering black-box attacks where the adversary has no prior knowledge of the system and study the effects of different scenarios on the power system non-linear SE. Another direction is to build robust detection algorithms to mitigate performance degradation issues.

REFERENCES

[1] X. Liu *et al.*, "Assessment of low-budget targeted cyberattacks against power systems," in *IFIP/IEEE*. Springer, 2018, pp. 232–256.

[2] Liang *et al.*, "The 2015 ukraine blackout: Implications for false data injection attacks," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3317–3318, 2016.

[3] Y. Liu, P. Ning, and M. K. Reiter, "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] Liang *et al.*, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, 2016.

[5] Zhao *et al.*, "Short-term state forecasting-aided method for detection of smart grid general false data injection attacks," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1580–1590, 2015.

[6] O. M. Anubi and C. Konstantinou, "Enhanced resilient state estimation using data-driven auxiliary models," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 639–647, Jan 2020.

[7] Li *et al.*, "Detecting false data injection attacks against power system state estimation with fast go-decomposition (godec) approach," *IEEE Transactions on Industrial Informatics*, 2018.

[8] O. M. Anubi, C. Konstantinou, and R. Roberts, "Resilient optimal estimation using measurement prior," *arXiv preprint arXiv:1907.13102*, 2019.

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

[10] 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.

[11] 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.

[12] 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.

[13] S. A. Foroutan and F. R. Salmasi, "Detection of false data injection attacks against state estimation in smart grids based on a mixture gaussian distribution learning method," *IET Cyber-Physical Systems: Theory & Applications*, vol. 2, no. 4, pp. 161–171, 2017.

[14] James *et al.*, "Online false data injection attack detection with wavelet transform and deep neural networks," *IEEE Transactions on Industrial Informatics*, 2018.

[15] He *et al.*, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, 2017.

[16] J. Wei and G. J. Mendis, "A deep learning-based cyber-physical strategy to mitigate false data injection attack in smart grids," in *CPSR-SG Workshop*. IEEE, 2016, pp. 1–6.

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

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

[19] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," 2014.

[20] Biggio *et al.*, "Evasion attacks against machine learning at test time," in *ECML PKDD*. Springer, 2013, pp. 387–402.

[21] Demontis *et al.*, "Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks," in *28th {USENIX} Security Symposium*, 2019, pp. 321–338.

[22] C. Xie *et al.*, "Adversarial examples for semantic segmentation and object detection," in *IEEE ICCV*, 2017, pp. 1369–1378.

[23] Y. Chen, Y. Tan, and D. Deka, "Is machine learning in power systems vulnerable?" in *2018 IEEE SmartGridComm*. IEEE, 2018, pp. 1–6.

[24] Y. Chen, Y. Tan, and B. Zhang, "Exploiting vulnerabilities of load forecasting through adversarial attacks," in *Proceedings of the Tenth ACM e-Energy*. ACM, 2019, pp. 1–11.

[25] Bor *et al.*, "Adversarial machine learning in smart energy systems," *arXiv preprint arXiv:1702.04267*, 2019.

[26] J.-M. Lin and H.-Y. Pan, "A static state estimation approach including bad data detection and identification in power systems," in *2007 IEEE Power Engineering Society General Meeting*. IEEE, 2007, pp. 1–7.

[27] 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, 2015.

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

[29] R. Wiyatno and A. Xu, "Maximal jacobian-based saliency map attack," *arXiv preprint arXiv:1808.07945*, 2018.

[30] "NYISO." [Online]. Available: https://www.nyiso.com/

[31] 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.

[32] "Tuning the hyper-parameters of an estimator." [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html