Data Integrity Attack Detection in Smart Grid: A Deep Learning Approach

Sunitha Basodi*, Song Tan and Yi Pan

Department of Computer Science,
Georgia state University,
Atlanta, GA, USA
E-mail: sbasodi1@cs.gsu.edu
E-mail: songtancs@gmail.com
E-mail: yipan@gsu.edu
*Corresponding author

WenZhan Song

School of Electrical and Computer Engineering ,
University of Georgia,
Atlanta, GA, USA
E-mail: wsong@uga.edu

Abstract:
Cybersecurity in smart grids plays a crucial role in determining its reliable functioning and availability. Data integrity attacks at the physical layer of smart grids are mainly addressed in this paper. State Vector Estimation(SVE) methods are widely used to detect such attacks, but such methods fail to identify attacks that comply with physical properties of the grid, known as unobservable attacks. In this paper, we formulate a distance measure to be employed as the cost function in deep-learning models using feed-forward neural network architectures to classify malicious and secured measurements. Efficiency and performance of these models are compared with existing state-of-the-art detection algorithms and supervised machine learning models. Our analysis shows better performance for deep learning models in detecting centralized data attacks.

Keywords: smart grids; bad data detection; state vector estimation; deep learning; IEEE test bus systems; matpower; keras with tensorflow.

Reference to this paper should be made as follows: Sunitha Basodi Song Tan, WenZhan Song and Yi Pan (2019) ''*Data Integrity Attack Detection in Smart Grid: A Deep Learning Approach*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Sunitha Basodi is currently a PhD student in the Department of Computer Science, Georgia State University. Her current research is focused on machine learning and deep learning methods, and their applications in Bioinformatics and SmartGrid domains. She has worked in Amazon Inc., India as a developer and received her Bachelors degree from Jawaharlal Nehru Technological University, Hyderabad, India.

Dr. Song Tan pursued his PhD at Department of Computer Science, Georgia State University. His research is focused on the cyber-physical security in Smart Grid system, which includes bad data detection, electrical market security and design of cyber-physical security testbed for Smart Grid. He has a MS from Georgia State University, and a BS from Northeast Normal University, China.

Dr. WenZhan Song is a Chair Professor of Electrical and Computer Engineering at the University of Georgia. Dr. Songs research focuses on cyber-physical systems informatics and security and their applications in energy, health and environment. Dr. Song has an outstanding record of leading large multidisciplinary research projects on those issues with multi-million grant support from NSF, NASA, USGS, and industry.

Dr. Yi Pan is currently a Regents Professor and Chair of Computer Science at Georgia State University, USA. He has served as an Associate Dean and Chair of Biology Department during 2013-2017 and Chair of Computer Science during 2006-2013. Dr. Pan joined Georgia State University in 2000, was promoted to full professor in 2004, named a Distinguished University Professor in 2013 and designated a Regents' Professor in 2015.

1 Introduction

Cybersecurity plays a vital role in determining the reliability and availability of the entire smart grid infrastructure. Potential intrusions can make the system vulnerable to various types of attacks, each of which may lead to serious outcomes, ranging from the leakage of customer private information to cascade of system failures as described in Aloul et al. [2012], Metke and Ekl [2010], Wang and Lu [2013]. Liu et al.[2012] provide an overview of the potential attacks and suggest the importance of cyber security in smart grids. In the current study, we primarily focus on data integrity attacks in which, measurements of a set of compromised meters are altered by the attacker. Such attacks can mislead the operational metrics at the control center causing the operators to make fatal decisions and disrupting the reliability of measured data. State Vector Estimation (SVE) methods, detailed in Abur and Exposito [2004], are generally employed to identify such malicious data attacks. In these methods, the state vector, which corresponds to an optimal power system state, is maintained, and a new state vector is reconstructed from measurements. If the difference between the actual and the reconstructed state vector is above a threshold value τ, then the data is considered to be attacked data. T~~he study Li~~u et al. [2011] suggests a way of constructing data integrity attack vectors that fall in the column space of the Jacobian matrix($\boldsymbol{H}$) of the measurements. In those cases, there exist cooperative attacks on meters, known as unobservable attacks, that cannot be identified using state vector estimation or any other known bad data detection techniques. There have been many algorithms proposed to detect such attacks in the literature , as in Kosut et al. [2010], Giani et al. [2011], Xiao et al. [2013]. Esmalifalak et al.[2014] and Ozay et al. [2016] use supervised machine learning classification models have also been propos~~ed to identify such attac~~ks [Esmalifalak et al. (2014); Ozay et al. (2016)]. The work in Esmalifalak et al. [2014] uses a series of measurements as time series data, preprocesses the data using principal component analysis (PCA), and then uses support vector machines (SVM) with Gaussian kernel to identify malicious data attacks. The authors in Ozay et al. [2016] employ a wide array of supervised and semi-supervised machine learning methods on the measurements, and analyze the performance of these methods. Kundur et al. [2011] analyze the impact of such physical and cyber attacks in smart grids using graph-based model.

Identifying attacked measurements from secure measurements using machine learning techniques is an active research area. In this paper, we use deep learning Feed-Forward Neural Network model to learn and classify malicious data from secured data. This is done by geometrically analyzing and visualizing the measurement space which helps in identifying a distance metric for measurements, that can be used in machine learning algorithms as the cost function, to optimally discriminate secured data from attacked(malicious) data. We employ deep learning models, in this case feed-forward neural networks, to classify secured vs. malicious data. We perform exhaustive performance analysis of the classification models for a range of attacked meters for multiple IEEE bus test systems and also compare our results with other physical and machine learning models.

The remainder of this paper is organized as follows. State estimation and centralized attack data generation methods used in this paper are briefly described in Section 2. The problem formulation is presented in Section 3. Deep learning models employed, and their characteristics are described in Section 4. Implementation and experimental results and comparisons are covered in Sections 5 and 6, followed by conclusions in Section 7.

2 Preliminaries

2.1 State Estimation

State Estimation is the process of identifying the current state of the system from measurements gathered across various meters. Monitoring such information is necessary for the reliability of the system, as the operators at the control center use this information to re-dispatch power, call on operating/contingency reserves, re-balance control areas, and for auditing and settlement(Giani et al. [2011]). In this process, the control center collects real time measurements $\boldsymbol{z}$ across various meters of the smart grid system and combines the network topology and parameter information, to calculate the real time estimates of the unknown system state variables $\boldsymbol{x}$. Let $\boldsymbol{z} = (z_1, z_2, ..., z_m)^T$ denote the meter measurements genertated using bus power and branch power flow measurements and $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ denote the system state variables computed using voltage phases at all buses, where m is the number of meters and n is the number of unknown state variables, and $m \geq n$. Let $\boldsymbol{e} = (e_1, e_2, ..., e_m)^T$ denote the measurement errors with Guassian noise with zero mean and a covariance matrix $\sigma^2 I$. The measurement model for DC power flow is given as follows:

$$\boldsymbol{z = Hx + e} \tag{1}$$

where $\boldsymbol{H}$ is an $m \times n$ full rank jacobian matrix of the measurement model. The matrix $\boldsymbol{H_{m \times n}}$ signifies the relationship between the vector $\boldsymbol{z_{m \times 1}}$ of measurements consisting of m meters readings, and the state vector $\boldsymbol{x_{n \times 1}}$ consisting of n state variables $x_1, ..., x_n$. From the vector of measurements, the estimated state vector $\hat{\boldsymbol{x}}$ is computed using

$$\hat{\boldsymbol{x}} = (\boldsymbol{H^T W H})^{-1} \boldsymbol{H^T W z} \tag{2}$$

where $\boldsymbol{W}$ is a diagonal matrix whose elements are reciprocals of the variances of meter errors.

2.2 Bad Data Detection

Bad measurements may be introduced due to some faulty transmission lines, meter failures, or due to malicious attacks. Estimated state vector of normal measurements is close to the actual state vector, while deviates from it for malicious measurements. The difference between the observed measurements $\boldsymbol{z}$ and the computed measurements using $\hat{\boldsymbol{x}}$ i.e., $\hat{\boldsymbol{z}} = \boldsymbol{H}\hat{\boldsymbol{x}}$ is known as the measurement residual, $\boldsymbol{z} - \hat{\boldsymbol{z}}$. $L2$-Norm of this residual $\|\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}\|_2$ is generally used to detect bad measurements using a threshold value τ. If the measurement residue exceeds this threshold value i.e., $\|\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}\|_2 > \tau$, the measurements are considered to be compromised; otherwise they are considered as secured measurements.

2.3 Unobservable Attacks

Injecting malicious data involves modifying the meter readings so that the actual measurements are augmented with an attack vector $\boldsymbol{a} = (a_1, a_2, ..., a_m)^T$, that have non-zero elements for compromised meters. The compromised vector of measurements can then be constructed as:

$$\hat{\boldsymbol{z}} = \boldsymbol{z} + \boldsymbol{a} \tag{3}$$

The attacker can be assumed to have access to the matrix $\boldsymbol{H_{m \times n}}$, in addition to some or all of the meters. Let $\mathcal{A} = \{t_1, t_2, ..., t_k\}$, where $t_i \in \{1, 2, .., m\}$, be the set of meters that are compromised and $\bar{\mathcal{A}}$ be the uncompromised meters. When the compromised measurements are coherent with the physical power flow constraints of the system, such attacks are called as *unobservable* attacks. These attacks cannot be detected by State Estimation methods or any other bad data detection algorithms. Such an unobservable attack vector $\boldsymbol{a} = (a_1, a_2, ..., a_m)^T$ can be generated using the following condition:

$$\boldsymbol{a} = \boldsymbol{H}\boldsymbol{c} \tag{4}$$

where $\boldsymbol{c}$ is a sparse vector using guassian distribution, and $a_i \neq 0$ for $i \in \mathcal{A}$ and $a_i = 0$ for $i \notin \mathcal{A}$. The properties and constraints for generating unobservable data integrity attacks are briefly described in Liu et al. [2011].

The compromised measurements generated after adding the above attack vector is as follows:

$$\hat{\boldsymbol{z}} = \boldsymbol{H}(\boldsymbol{x} + \boldsymbol{c}) + \boldsymbol{e} \tag{5}$$

where the injected vector $\boldsymbol{c}$ is the residue error introduced into the measurements, which goes undetected.

There are two types of attacks possible for centralized data integrity attacks, namely, random false data injection attacks and targeted false injected attacks. In the former type of attacks, the attacker can corrupt measurements from any of the meters to make the attack undetectable in state estimation, whereas in the latter, the attacker has access to only a specific set of meters. In this paper, we use LASSO optimization methods suggested in Ozay et al. [2013] to generate attack vectors.

2.4 Deep learning

Deep learning is a branch of machine learning that uses artificial neural networks, a complex layered network of non-linear processing units called neurons, to capture multiple levels of hidden representations of the data. These extracted representations can be used for feature extraction, classification or, any other related tasks. Artificial Neural Networks work very well with high-dimensional data, both for classification and regression problems. Deep learning has been successful in a wide variety of areas such as automatic speech recognition, natural language processing, etc. Several architectures exist for deep learning such as feed-forward neural networks, recurrent neural networks and LSTMs. Detailed survey of deep learning methods and models, along with their applications, can be found in LeCun et al. [2015], Schmidhuber [2015].

Deep learning models use artificial neural networks which capture multiple levels of hidden representations of the data to perform the classification task. It can also perform other tasks such as object recognition, speech recognition, series prediction and so on. An artificial neural network consists of an input layer, an output layer, and a set of hidden layers that connect them using computation units called neurons. Neurons of each layer are connected to its preceding and succeeding layers via weighted links. Each neuron takes inputs from the neurons of its preceding layer, computes weighted sum of the corresponding inputs, and then applies an activation function to generate the transformed output. Activation functions are non-linear functions that help determine the output states of neurons. This output is passed on as input to the neurons in the next layer. This process is repeated until the values are propagated to the output layer which predicts the final result. The weights of the connected links are adjusted to minimize the error between expected and predicted outputs. The connections between layers can be uni-directional or bi-directional.

3 Problem Formulation

Attacked measurements cannot be identified using state estimation methods (SVE) when the attack vectors are in the column space of $\boldsymbol{H}$ as in equation (4). Such vectors can always be constructed when the number of attacked meters is ~~more than $m + n - 1$~~ atleast $m - n + 1$, as per *Theorem 2* of Liu et al. [2011]. These generated attack vectors are closely located around the normal values as shown in Fig.1, for IEEE 57-bus test system, which generally holds good for any system. Clearly, the compromised measurements are

spaced very closely to secure measurements, separated by short distances. These inter-measurement distances can be used in machine learning models to separate attacked and unattacked measurements in space, and to identify the discriminating boundaries that help classify new measurements efficiently. Since the distances between the measurements are usually small, firstly, the measurements are transformed to logarithmic space, and then the distances are computed.

Formally, let $\boldsymbol{z} = (z_1, z_2, ..., z_m)^T$ denote the actual meter measurements, and $\boldsymbol{a^p} = (a_1^p, a_2^p, ..., a_m^p)^T$ and $\boldsymbol{a^q} = (a_1^q, a_2^q, ..., a_m^q)^T$ be the potential attack vectors generated for $\boldsymbol{z}$ at timestamps t_p and t_q respectively. Let $\hat{\boldsymbol{z}}^{\boldsymbol{p}} = \boldsymbol{z} + \boldsymbol{a^p}$ and $\hat{\boldsymbol{z}}^{\boldsymbol{q}} = \boldsymbol{z} + \boldsymbol{a^q}$. Clearly, $\hat{\boldsymbol{z}}^{\boldsymbol{p}} = (\hat{z}_1^p, \hat{z}_2^p, ..., \hat{z}_m^p)^T$, where $\hat{z}_i^p = z_i + a_i^p$. Let M_i^p, M_i^q represent the meter i of measurement vector $\hat{\boldsymbol{z}}^{\boldsymbol{p}}$, $\hat{\boldsymbol{z}}^{\boldsymbol{q}}$ respectively. As the meter measurements can be attacked independently at t_p and t_q, either none or one or both of the M_i^p and/or M_i^q values can be attacked. The distance measure computation ~~problem~~ can be formulated as below:

$$d(\hat{z}_i^p, \hat{z}_i^q) = \begin{cases} \|log_(|a_i^p|) - log(|a_i^q|)\|_2, & if M_i^p, M_i^q \in \mathcal{A} \\ \|log(|a_i^p|)\|_2, & if M_i^p \in \mathcal{A}, M_i^q \notin \mathcal{A} \\ 0, & \text{if } M_i^p, M_i^q \notin \mathcal{A} \end{cases}$$

where $d(.,.)$ is the distance measure between the two points. Since the meters of the measurement vectors can be attacked independently, M_i^p and/or M_i^q can be attacked. The above formulation handles all these cases . Whenever there is attack on the meter readings, the distance between the estimated and the actual measurements is not zero, irrespective of the type of the attack, which helps in identifying that the attacks. ~~, and this~~ This distance measure can be used in any machine learning algorithms as the loss or optimization function to train the models so as to effectively separate secure vs. attacked measurements in vector space and to detect attacks on new measurements. We use the above formulation in our deep learning model as the loss function and optimize it to build a robust model for classifying normal vs. malicious measurements.

Given $\mathcal{S} = \{s_i\}_{i=1}^m$, the set of measurements from all the meters, and the corresponding label set $\mathcal{Y} = \{y_i\}_{i=1}^m$, where $s_i \in \mathbb{R}$, $y_i \in \{0, 1\}$. The value of y_i corresponding to the measurement s_i is defined as follows:

$$y_i = \begin{cases} 1, & \text{if the measurement } s_i \text{ is compromised} \\ 0, & \text{otherwise} \end{cases}$$

The input data $(s_i, y_i) \in \mathcal{S} \times \mathcal{Y}$ is assumed to be independent and identically distributed to a joint distribution $\mathcal{P}$ (Bousquet et al. [2004]). The sampling from $\mathcal{P}$ is done independently and identically, as per the assumptions in Ozay et al. [2016]. The aim is to determine a learning model defined as a function mapping $f : \mathcal{S} \to \mathcal{Y}$, which determines the relationships between $\mathcal{S}$ and $\mathcal{Y}$ using the distance measure $d(.,.)$ and separates the attacked and secure measurements.

4 Deep Learning For Attack Detection

In this section, we briefly describe the design of the software, the architecture of the deep learning model employed to build a classification model to identify secure data and attacked data.

4.1 Workflow Design

The design of our tool is shown in the Fig. 3. *Data Generator* module is responsible for generating the metered data of the smart grids. Any software that simulates power grid models to generate meter readings (or measurements) at different time intervals or the actual meter readings available from power grid websites like PJM [2017] can be used in the *Data Generator*. Measurements of all the meters for each timestamp are stored into a matrix. This matrix data is passed to the *Attack Simulator* which has two modules *Attack Identifier* and *Attack Generator*. *Attack Identifier* determines timestamps for the attack, this determination is done by randomly selecting the timestamps. Once a timestamp is considered for the attack, then all the meter readings corresponding to that timestamp are passed to the *Attack Generator*. An attack vector is generated for the given measurement vector and the attacked measurements along with the attacked meters are passed to the *Data Persistor*. Here, the module looks the attacked meters passed from the *Attack Generator*. If any of the meters are passed as attacked meters, then it labels the corresponding measurements with label 1 otherwise, the measurements are labeled as 0. This measurement data along with the labels are stored and also passed to the *Attack Predictor*. This module is reads the labeled measurements and builds a classification model to predict if an unseen measurements are secured or attacked. In our work, we use deep learning models to perform this classification.*Attack Predictor* module can also be independently executed using the data stored by the *Data Persistor*.

4.2 Deep Learning Model Architecture

There are many neural network architectures such as Convolution deep neural networks (CNNs), Recurrent neural networks (RNNs), and LSTM networks which have outperformed in image recognition, object identification, natural language processing, next sequence prediction. However, our data consists of measurements and therefore a simple feed forward network architecture is employed. Feed-forward networks are uni-directional networks, in which, the connections from the input layer to the output layer are all in the forward direction, without any cycles. The information flows from input layer through the hidden layers, and finally to the output layer without any looping back of the information (see Fig.2). In our classification model, the neurons on the input layer accept the actual meter measurements, transform them using activation

functions, and pass it on to its successive hidden layers, which in turn transform the data, and finally pass it to the output layer which transforms and outputs a binary value, 0 or 1, classifying the data as secure or attacked respectively.

The feed-forward architecture of our neural network is shown in the Fig. 4. The input layer which is fully connected to the subsequent layer and accepts the input data. The output layer is fully connected to its preceding layer and gives us the classification result. Each of the hidden layers is fully connected to its subsequent layer, however dropout method is used to drop a few neurons and network connections to prevent the network from overfitting. Batch normalization is also performed during training the model and overfitting of the model is prevented using dropout (Srivastava et al. [2014]) for each of the hidden layers. The number of neurons in each layer is controlled by a parameter K and the dropout factor for each of the layer is determined using the parameter p.

5 Implementation

Data Generator module is implemented in Matlab and the simulated measurements for different power grid bus architectures are generated using MATPOWER toolbox (Zimmerman et al. [2011]). We use MATPOWER to simulate IEEE 9-bus, 14-bus, 30-bus and 57-bus test systems. *Attack Identifier* randomly determines if the measurements obtained from the *Data Generator* should be attacked or not. If it determines the measurements should not be attacked, then the measurements are sent to the *Data Persistor* module and the label for these measurements are marked as 0. Otherwise, the attack vector for the given measurements is generated using false data injection attacks. The *Attack Generator* module is implemented using by Algorithm 1 of Ozay et al. [2013]. This module generates the attack vector of different sparseness based on the ratio $\kappa/m \in [0, 1]$. In our implementation, we generate 22 ratio values equally spaced between (0,1). The ratio parameter value multiplied by the number of meters for the test system determines that number of meters to be attacked. This value is added as one of the stopping constraints of Algorithm 1 of Ozay et al. [2013] so that it corresponds to the sparseness of the attack vector. These generated unobservable attack vectors are added to the original measurements to get the corrupted or attacked measurements. These attacked measurements are passed to the *Data Persistor* module and are stored with the label 1. *Data Persistor* module stores these measurements along with the corresponding labels in the file system as CSV files. These CSV files along with their labels are used by the *Attack Predictor* module to perform classification. Our deep learning models are implemented using Keras (Chollet [2015]), a high-level API for neural networks that works on top on TensorFlow (Abadi et al. [2015]) and Theano. This API provides a simple interface to implement complex neural network models. Our code implements the architecture shown in Fig.4 to train a feed forward neural network with one input layer, one output layer, and multiple hidden layers. Weight initialization methods, activation functions, loss functions, optimizer methods, and metrics which need to be improved, can be passed as method parameters. In our code, we train the model to achieve best accuracy metric using our distance formulation as the loss function. The weights in the network are regulated so as to achieve the best possible accuracy.

Table 1 Calculation of performance measures

	Attacked	Secure
Predicted as Attacked	tp	fp
Predicted as Secure	fn	tn

5.1 Sampling and Metrics

Stratified 10-fold cross validation is used for training and testing the model. Accuracy, precision, and recall are the metrics used to study the performance of the model. Accuracy (Acc) is defined as the proportion of test cases correctly classified as either attacked or secure. Precision (Prec) measures the fraction of the test cases which are classified as attacked, have truly been attacked. Recall (Rec) measures the fraction of the test cases that are correctly predicted as attacked. To calculate these metrics, we measure true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn), as shown in Table 1.

$$\text{Acc}=\frac{tp+tn}{tp+tn+fp+fn}\ ,\ \text{Prec}=\frac{tp}{tp+fp}\ ,\ \text{Rec}=\frac{tp}{tp+fn}$$

In our models, the input data $(s_i, y_i) \in \mathcal{S} \times \mathcal{Y}$ is sampled and used for training feed-forward neural networks. Each measurement s_i is fed as input to the model and the output layer with a single neuron results in either 0 or 1, to represent if the data is secure or attacked respectively. Problem formulation defined in Section 3 is employed into the cost function and the model is trained so as to optimize the cost function and reduce the prediction errors.

6 Experimental Results

The classification models are analyzed for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems in our experiments. MATPOWER toolbox (Zimmerman et al. [2011]) is used to compute the measurement Jacobian matrices $\boldsymbol{H}$. The operating state of the system $\boldsymbol{x}$ is obtained from MATPOWER, and then the measurements for $\boldsymbol{z}$ are computed. For each of the measurement vectors, we independently determine if the measurements need to be attacked. Following this, the corresponding attacked measurements using random and targeted false injection attacks are generated (Liu et al.

[2011], Ozay et al. [2013]). Data is labeled '1' if the measurements are attacked, or '0' otherwise. In the experiments, we assume that the attacker has access to κ meters, and these meters are randomly chosen to generate a κ-sparse attack vector. These attack vectors have non-zero values with the same mean and variance as $\boldsymbol{z}$, and follow a Gaussian distribution. We evaluate the behavior of the models to classify both observable and unobservable attacks for different values of $\kappa/m \in [0, 1]$. Specifically, we analyze the performance of the model for $\kappa \geq m - n + 1$, where unobservable attacks are generated, and the attack vectors are not observable by SVE (Liu et al. [2011]). Otherwise, the generated attacks are observable.

Feed-forward neural networks are implemented using Keras API (Chollet [2015]) in python with Tensorflow as the backend library. There are multiple parameters to learn in the deep learning models, such as the number of hidden layers, the number of neurons in each of these layers, the type of activation functions for neurons, the dropout factor and, the loss functions. In our models, we integrate the distance metric in the problem formulation as the loss function, and use Exponential Linear Units (elu) (Clevert et al. [2015]) as the activation function for all the neuronal units of the hidden layers. Sigmoid is used as the activation function for the output layer so that the output result can be directly related to the class label. The number of neurons in the input and output layers are fixed as it depends on the dimension of the input data and the type of output. To determine the appropriate number of hidden layers, the model is parameterized to have upto 100 hidden layers, and the number of neurons (f) in each layer is taken from the set $\{0, 50, 100, .., 450, 500\}$. Dropout factors (p) are chosen from the set $\{0.25, 0.50\}$. Scikit (Pedregosa et al. [2011]) library from python is used to perform grid search through the parameter space, and determine the best possible parameters for the model. We also train Linear SVM models using scikit library in python for comparison studies.

Fig.5 shows the accuracies of SVE, SVM (Linear), and deep learning models for IEEE 57-bus test system. Accuracies are measured using a stratified 10-fold cross validation method. It can be seen that SVM performs comparable to the feed-forward deep neural network model till the ratio κ/m reaches 0.5, after which, the feed-forward network (deep learning) model performs better. SVE (State Vector Estimation) performance increases linearly with the number of meters, but it also increases the true negatives (number of outputs predicted as attacked, but are actually unattacked). This can be inferred from the precision and recall comparison figures shown in Fig. 6, 7. Linear SVM models have better recall values when less than half of the meters are attacked, but have very low precision values due to the presence of false positives and vice versa, when more than half of meters are attacked. SVE models have good precision performance, but have less recall performance which shows that not all the attacked meters are identified correctly. Deep learning models have less precision performance when compared to SVE, but have better recall performance than SVE. They also have better accuracy performance when compared to SVE and SVM models.

The performance of neural network model slightly increases with an increase in the number of layers, after which it stabilizes or sometimes decreases after a maximum value. One of the reasons for this reduction in accuracy with the increase in number of layers is due to overfitting of the data with the increase in number of layers. Fig. 11 shows the accuracy of models with the increase in number of layers. Clearly, accuracy of some IEEE bus test systems reach highest accuracy with almost two layers which some models require more layers. In Fig. 11, we observe that IEEE 9-bus, 14-bus test systems achieve their maximum accuracy around 50 hidden layers, IEEE 30-bus test system achieves similar highest accuracies around 2 or 90 hidden layers and IEEE 57-bus test system achieves it around 2 or 40 hidden layers.

The accuracies of the models vary based on the number of attack meters and also the other hyperparameters. We show metrics for neural network models with the best performance values of all trained models for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems. The performance charts can be seen in Fig. 8, 9, 10. It is observed that all the models start to converge to high accuracy when more than 55% of the meters are attacked. IEEE 9-bus test system achieves around 90% accuracies with just 2 hidden layers. The number of hidden layers and neurons increase with the number of buses in the IEEE test systems. Clearly, feed-forward neural networks achieve more than 90% accuracies when the ratio of the attacked meters is greater than 0.3 for IEEE 9-bus, 11-bus test systems, and is greater than 0.7 for IEEE 30-bus, 57-bus test systems.

We also analyze the computational time for each of the models with the increase in the number of layers. In Fig. 12, we observe that the average training time increases with the increase in the number of layers. The horizontal axis in the Fig. 12 represents the number of hidden layers in the model and the vertical axis represents the average time taken for the model to train the across the parameter space. The interesting observation is that as the complexity of the IEEE bus test system increases, the average training time is less compared to the simplest bus system. That is, as the number of layers increases, IEEE 9-bus system has more average training times compared to IEEE 14-bus test system which in turn has higher average training times compare to IEEE 30-bus test system. IEEE 30-bus and 57-bus has comparable (close) training times which is less than IEEE-9, IEEE 14-bus systems.

Since we experiment with multiple IEEE bus test systems, namely 9-bus, 14-bus, 30-bus, and 57-bus systems, to achieve fast computation results, we perform the data generation for all the bus systems in parallel using multiple processes. For each of the test systems,

we analyze the scenario when different number of meters attacked and analyze their classification performance. This means for each of the datasets corresponding to different ratio of attacked meters, a model needs to trained and tested. To accelerate the training process for all the models, we train these models in parallel on a server and persist the optimal network models.

7 Conclusion & Future Work

The data integrity attack detection problem in smart grids is redefined as a machine learning problem, and a feed-forward neural network model is employed for the classification of attacked data and secured data. The performance of this model is analyzed and compared with State Vector Estimation (SVE) and supervised machine learning algorithms for a range of attacked meters. This analysis shows that feed-forward neural networks can detect attacks with better performance than attack detection algorithms that employ state vector estimation methods for centralized data integrity attacks. Also, the performance of feed-forward deep neural networks against various IEEE-bus test systems is compared. It is clearly seen that feed-forward neural networks perform better than State Vector Estimation methods and have comparable (often better) performance with other supervised machine learning algorithms. In future work, we would like to work on the real-time measurement data from PJM and employ other deep learning models such as recurrent networks and LSTMs which are known to work better for sensor data, time series and high-dimensional data to improve the performance of the classification of secured and attacked data.

Acknowledgement

This work is supported by the National Science Foundation under Award Number DUE-1303359. We would like to thank Mete Ozay for the productive discussions and comments, and Krishna Pusuluri for valuable feedback.

References

Mohammad Esmalifalak, Lanchao Liu, Nam Nguyen, Rong Zheng, and Zhu Han. 'Detecting stealthy false data injection using machine learning in smart grid.' *IEEE Systems Journal*, 2014.

Annarita Giani, Eilyan Bitar, Manuel Garcia, Miles McQueen, Pramod Khargonekar, and Kameshwar Poolla. 'Smart grid data integrity attacks: characterizations and countermeasures π.' In *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 232–237. IEEE, 2011.

Oliver Kosut, Liyan Jia, Robert J Thomas, and Lang Tong. 'Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures.' In *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 220–225. IEEE, 2010.

Deepa Kundur, Xianyong Feng, Salman Mashayekh, Shan Liu, Takis Zourntos, and Karen L Butler-Purry. 'Towards modelling the impact of cyber attacks on a smart grid.' *International Journal of Security and Networks*, 6(1):2–13, 2011.

Ali Abur and Antonio Gomez Exposito. *Power system state estimation: theory and implementation*. 1st ed., CRC press, 2004.

Fadi Aloul, AR Al-Ali, Rami Al-Dalky, Mamoun Al-Mardini, and Wassim El-Hajj. 'Smart grid security: Threats, vulnerabilities and solutions.' *International Journal of Smart Grid and Clean Energy*, 1(1):1–6, 2012.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Advanced lectures on machine learning*, pages 169–207. Springer, 2004.

François Chollet. Keras. [online] `https://github.com/fchollet/keras`, 2015. (Accessed 15 February 2017).

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. [online] URL `http://tensorflow.org/`. Software available from tensorflow.org (Accessed 11 March 2017).

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 'Deep learning.' *Nature*, 521(7553):436–444, 2015.

Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and CL Philip Chen. 'Cyber security and privacy issues in smart grids.' *IEEE Communications Surveys & Tutorials*, 14(4):981–997, 2012.

Yao Liu, Peng Ning, and Michael K Reiter. 'False data injection attacks against state estimation in electric power grids.' *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

Anthony R Metke and Randy L Ekl. 'Smart grid security technology.' In *Innovative Smart Grid Technologies (ISGT), 2010*, pages 1–7. IEEE, 2010.

Mete Ozay, Inaki Esnaola, Fatos TYarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. 'Sparse attack construction and state estimation in the smart grid: Centralized and distributed models.' *IEEE Journal on Selected Areas in Communications*, 31(7): 1306–1318, 2013.

Mete Ozay, Inaki Esnaola, Fatos Tunay Yarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. 'Machine learning methods for attack detection in the smart grid.' *IEEE transactions on neural networks and learning systems*, 27(8):1773–1786, 2016.

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. 'Scikit-learn: Machine learning in Python.' *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

PJM : Markets and Operations , 2017. [online] URL `http://www.pjm.com/markets-and-operations.aspx`. (Accessed 23 June 2017).

Jürgen Schmidhuber. 'Deep learning in neural networks: An overview.' *Neural networks*, 61:85–117, 2015.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 'Dropout: a simple way to prevent neural networks from overfitting.' *Journal of machine learning research*, 15 (1):1929–1958, 2014.

Wenye Wang and Zhuo Lu. 'Cyber security in the smart grid: Survey and challenges.' *Computer Networks*, 57 (5):1344–1371, 2013.

Zhifeng Xiao, Yang Xiao, and David Hung-Chang Du. 'Exploring malicious meter inspection in neighborhood area smart grids.' *IEEE Transactions on Smart Grid*, 4(1):214–226, 2013.

Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. 'Matpower: Steady-state operations, planning, and analysis tools for power systems research and education.' *IEEE Transactions on power systems*, 26(1):12–19, 2011.

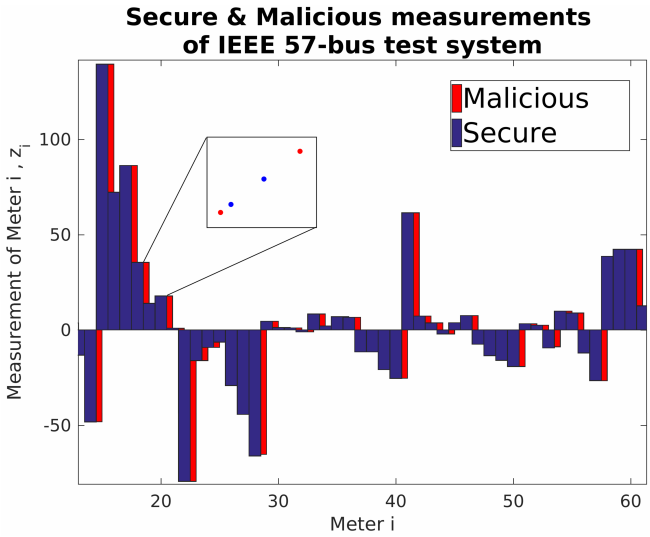

Figure 1 Secure and malicious measurements of IEEE 57-bus test system. All the meter measurements of vector $\boldsymbol{z} = (z_1, z_2, ..., z_m)^T$ are plotted using $(i,\ z_i)$ for $i \in 1, 2, .., m$. Y-axis shows the measurement values for the meters on X-axis. The attacked and secure measurement values are close to each other, meaning the difference between them is very small. We demonstrate the difference by plotting those values on the logarithmic scale, as shown in the small rectangular box. For case-57 bus system, m=137. Here, we only show a part of the plot for values of $i \in \{13, .., 62\}$ to clearly demonstrate that the malicious and the secure measurements are very closely placed in vector space, separated by very short distances, as shown in the rectangular box. The points on the histogram connected to the rectangle correspond to the zoomed regions showing how these points are separated by some distance.

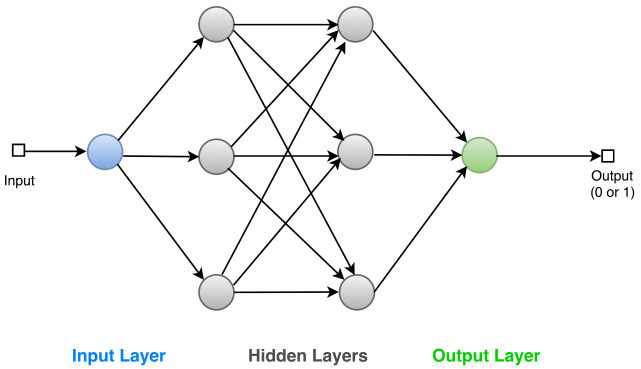

Figure 2 Example of a feed-forward neural network with one input on the input layer(blue) and one output on the output layer(green) and 2 hidden layers with three neurons each(grey). This is a fully connected feed-forward network in which a neuron receives inputs from all the neurons in the previous layer and then uses weights on its connections and its activation function to determine the value to be passed to the next layer.

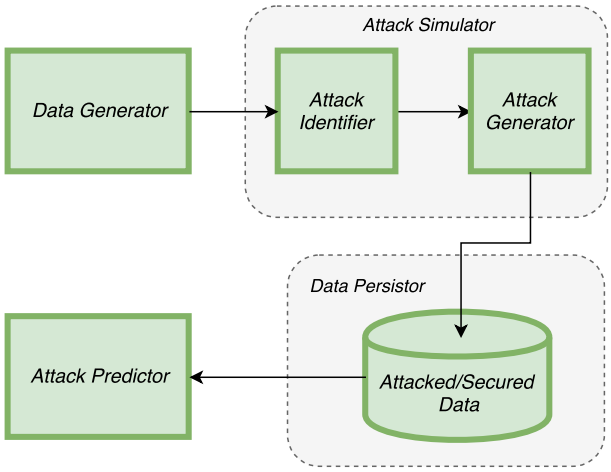

Figure 3 Workflow design of the smart grid data generation and prediction model.

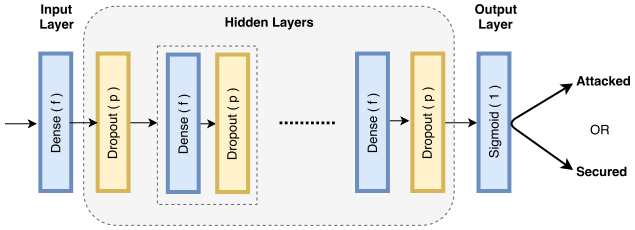

Figure 4 Architecture of deep learning model

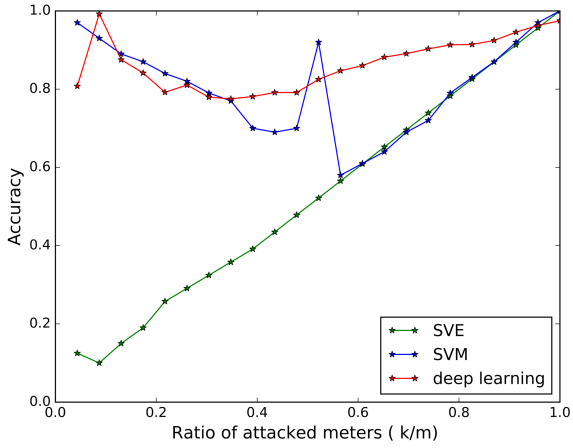

Figure 5 Comparison of accuracies among state vector estimation (SVE), SVM (Linear) and deep learning models for IEEE 57-bus test system.

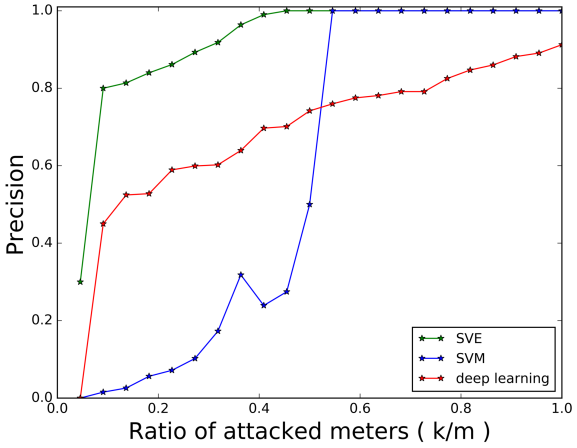

Figure 6 Comparison of precision values among State Vector Estimation (SVE), SVM (Linear) and deep learning models for IEEE 57-bus test system.

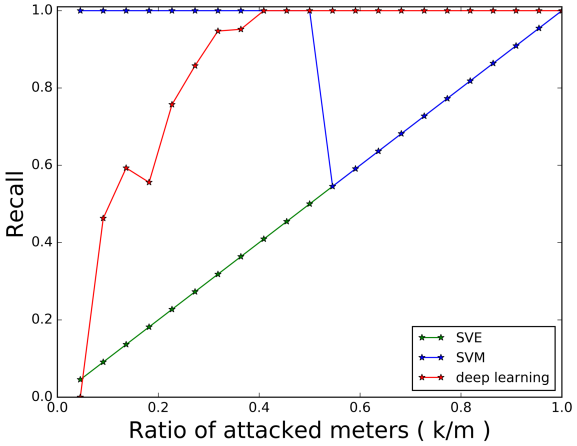

Figure 7 Comparison of recall values among State Vector Estimation (SVE), SVM (Linear) and deep learning models for IEEE 57-bus test system.

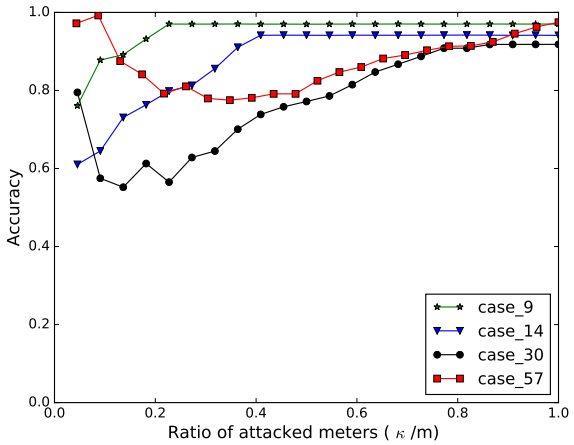

Figure 8 Accuracies of deep learning models for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems for a range of attacked meters.

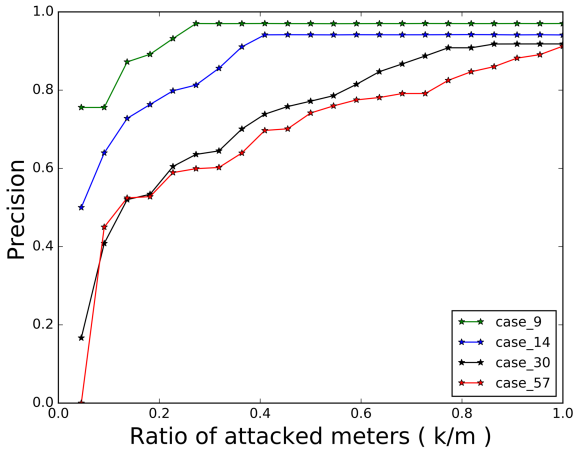

Figure 9 Precision values of deep learning models for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems for a range of attacked meters.

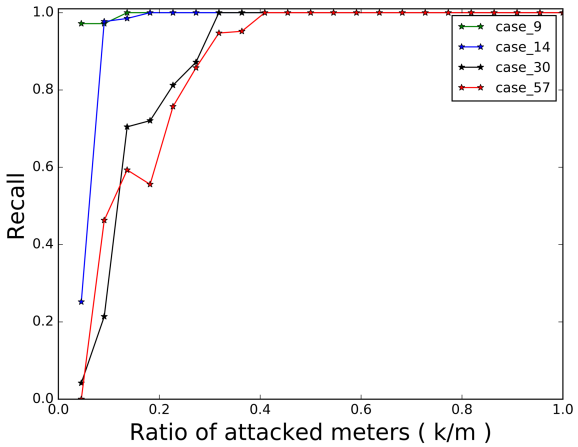

Figure 10 Recall performance values of deep learning models for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems for a range of attacked meters.

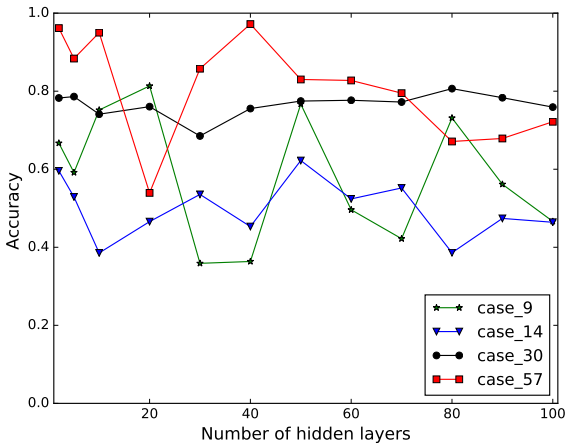

Figure 11 Accuracy of deep learning models with the increase in number of layers for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems when there are 10 attacked meters.

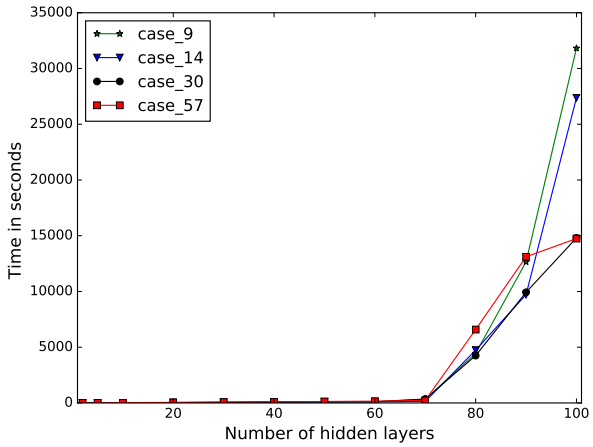

Figure 12 Average training times of deep learning models for IEEE 9-bus, 14-bus, 30-bus, and 57-bus test systems, where there are 10 attacked meters. The horizontal axis represents the number of hidden layers in the model and the vertical axis represents the average time taken for the model to train the across the parameter space.