Robust Electricity Theft Detection Against Data Poisoning Attacks in Smart Grids

Abdulrahman Takiddin, Muhammad Ismail, *Senior Member, IEEE*, Usman Zafar, Erchin Serpedin, *Fellow, IEEE*

***Abstract*—Data-driven electricity theft detectors rely on customers' reported energy consumption readings to detect malicious behavior. One common implicit assumption in such detectors is the correct labeling of the training data. Unfortunately, these detectors are vulnerable against data poisoning attacks that assume false labels during training. This paper addresses three major problems: What is the impact of data poisoning attacks on the detector's performance? Which detector is more robust against data poisoning attacks, i.e., generalized or customer-specific detectors? How to improve the detector's robustness against data poisoning attacks? Our investigations reveal that: (a) Shallow and deep learning-based detectors suffer from data poisoning attacks that may lead to a significant deterioration of detection rate of up to 17%. Furthermore, deep detectors offer 12% performance improvement over shallow detectors. (b) Generalized detectors present 4% performance improvement over customer-specific detectors even in the presence of data poisoning attacks. To enhance the detectors' robustness against data poisoning attacks, we propose a sequential ensemble detector based on a deep auto-encoder with attention (AEA), gated recurrent units (GRUs), and feed forward neural networks. The proposed robust detector retains a stable detection performance that is deteriorated only by $1-3\%$ in the presence of strong data poisoning attacks.**

***Index Terms*—electricity theft, data poisoning, robust detector, machine learning, data-driven detection.**

I. Introduction

Electricity theft is a serious threat for power grids as it incurs high financial losses. For instance, the annual losses are up to $6 billion in Canada and the United States [1]. Moreover, since electricity thefts overload the power grid, they have a negative effect on its performance [2]. Recently, utility companies have started to deploy advanced metering infrastructures (AMIs) that are equipped with smart meters to regularly monitor the customer's energy consumption and to reduce traditional (physical) electricity thefts [3]. However, the implementation of smart meters has introduced a variety of electricity theft cyber-attacks where malicious customers hack into the meter to reduce their electricity bills by manipulating their electricity consumption readings [4].

A. Takiddin is with the ECEN Program, Texas A&M University at Qatar, Doha, Qatar, (email: abdulrahman.takiddin@qatar.tamu.edu).

M. Ismail is with the Department of Computer Science, Tennessee Tech University, Cookeville, TN, USA (email: mismail@tntech.edu).

U. Zafar is with Qatar Environment and Energy Research Institute, Hamad Bin Khalifa University, Doha, Qatar (email: uzafar@hbku.edu.qa).

E. Serpedin is with the ECEN Dept., Texas A&M University, College Station, TX, USA (e-mail: eserpedin@tamu.edu).

This publication was made possible by NPRP10-1223-160045 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

A. Related Work and Limitations

So far, several shallow and deep learning-based attempts have been carried out to adopt data-driven approaches for electricity theft detection[1]. For instance, AdaBoost-based models presented accuracy of approximately 80% [5]. An anomaly detector that employs an auto-regressive integrated moving average (ARIMA) model offered a detection rate (DR) of 89% and false alarm (FA) rate of 11% [6]. Outlier detection techniques such as local outlier factor, mutual k-nearest neighbor, and indegree number offered an area under the curve (AUC) of up to 90% [7]. By exploiting decision trees and a support vector machine (SVM), a two-step electricity theft detector showed an accuracy of 92.5% [8]. A slight improvement in performance is achieved by the multi-class SVM model with DR and FA values of 94% and 11%, respectively [1]. More recently, deep learning-based detection techniques were investigated. For example, a hybrid convolutional neural network (CNN) and gated recurrent unit (GRU) detector presented an AUC of 89% [9]. A feed forward-based detection presented a DR of 92% and FA of 2.3% [10]. A recurrent neural network (RNN)-based detector presented a DR of 94% and FA of 4.7% [11]. A random forest and CNN-random forest-based model offered F1-score of approximately 81% and 96%, respectively [12]. A residual neural network-based detection scheme provided a DR of 97% [13]. A bidirectional RNN offered an accuracy of 97% [14]. Another hybrid CNN and RNN-based detector presented a DR of 99.3% and FA of 0.22% [15].

All of the aforementioned detectors rely on the customers' reported energy consumption readings during the training stage. In this context, two detection mechanisms are adopted, namely, generalized or customer-specific detection. When datasets of all customers are merged to train a single detector that can be used by any customer to detect electricity theft, we say that a generalized detection mechanism is employed. However, when a detector is trained (and then used) for each individual customer using only the dataset of that customer, then we say that a customer-specific detection mechanism is employed. Unfortunately, all existing detectors, whether generalized or customer-specific, implicitly assume that the detector has been trained using correct labels. Specifically, if a customer is carrying out electricity theft and has not been detected before, the utility company will be training the electricity theft detector using the customer's data while

[1]It should be highlighted that the aforementioned results are hard to compare to each other since different metrics and datasets have been used in the relevant works.

assuming it is benign data. Such a training that is based on false labels is referred to as a data poisoning attack [16], [17], and is accompanied by a shift in the detector's decision boundaries, which results in a deterioration of the detector's ability to distinguish benign from malicious data. Hence, our paper focuses on addressing the following three major questions:

- What is the impact of data poisoning attacks on the electricity theft detector's performance?
- Which electricity theft detector is more robust to data poisoning attacks, generalized or customer-specific detectors?
- How to develop a robust electricity theft detector that is marginally affected by data poisoning attacks?

While the problem of data poisoning is overlooked in smart grid literature, traditional techniques that deal with this problem in other domains usually apply data filtering first to remove any false labels, and then the detection step is carried out [18]. The limitation of such a solution is that it adds to the detector an additional stage dedicated only for data filtering. In this paper, we investigate the possibility of developing a robust detector that is marginally affected by data poisoning, and at the same time, provides a stable classification of the readings to either benign or malicious (electricity theft detection) without requiring an additional data filtering operation.

B. Contributions

In order to address the aforementioned questions, this paper reports the following contributions:

- To quantify the impact of data poisoning attacks, we test the performance of customer-specific and generalized benchmark detectors along with their ability of detecting data poisoning attacks as well as cyber-attack electricity thefts in one step. The benchmark detectors include random forest, adaBoost, ARIMA, SVM, deep feed forward, GRU, and auto-encoder with attention (AEA) detectors at different levels of data poisoning attacks. Our simulation results reveal that both customer-specific and generalized detectors suffer from severe performance degradation by 17%. Generalized detectors tend to be more robust against data poisoning compared to customer-specific detectors and exhibit a performance improvement of 4%.
- To enhance the robustness of electricity theft detectors against data poisoning attacks, we investigate detection schemes based on a combination of AEA, GRUs, and feed forward layers using ensemble averaging and sequential ensemble learning methods. Ensemble averaging builds its decision based on the average of the outputs of individual AEA, GRU, and feed forward models. On the other hand, the output of each individual model in the sequential ensemble is passed over to the next model for further processing until a final decision is made. Our investigations pointed out that sequential ensemble is more robust against data poisoning attacks compared to ensemble averaging (up to 10% improvement) since it takes full advantage of its subcomponents.
- Overall, the sequential ensemble detector is capable of detecting electricity theft with high DR (95.2%) and low FA (2.9%). Additionally, the sequential ensemble detector's performance is deteriorated by only $1-3\%$ under data poisoning attacks. Thus, the proposed sequential ensemble-based detector is robust against electricity theft and data poisoning attacks, without the need of a separate data filtering stage.

Hyper-parameter optimization is carried out for all of the investigated detectors using a sequential grid search.

This paper is organized as follows. Section II presents the used benign and malicious datasets. Section III investigates the impact of data poisoning attacks on customer-specific and generalized detectors. Section IV presents the design of the proposed robust detector along with the experimental results corresponding to the ensemble learning-based detection schemes. Finally, conclusions are made in Section V.

II. Benign and Malicious Datasets

This section presents the electricity consumption data that we use to train and test the detectors under investigation. The benign energy consumption data is taken from the publicly available Irish Smart Energy Trail dataset [19]. The malicious data is simulated using six general cyber-attack functions adopted from [1]. The data poisoning attack is simulated by flipping some of the labels of malicious data to benign labels.

A. Benign Dataset

We use the Irish Smart Energy Trail dataset as the benign dataset to train and test the electricity theft detectors under investigation. This dataset is released by the Sustainable Energy Authority of Ireland and it is available to the public since 2012. The electricity readings in this dataset are taken from $3,000$ residential units' smart meters that took readings every 30 minutes over 18 months. Thus, the benign dataset has a record of $25,000$ reports per customer. The energy consumption value of customer c at a specific day d and time period t is defined as the entry $E_c(d,t)$ of matrix $\boldsymbol{E}_c$. For honest customers, the energy consumption that is recorded by the customer's meter $R_c(d,t)$ and $E_c(d,t)$ are equal. Hence, matrices $\boldsymbol{E}_c$ and $\boldsymbol{R}_c$ coincide in such a case.

B. Malicious Dataset

Malicious customers launch cyber-attacks and exploit the integrity of the electricity readings to lessen their electricity bills. Thus, $R_c(d,t) \neq E_c(d,t)$. Herein, in order to construct a malicious dataset, we use the false data injection approach [1]. As shown in Table I, we consider six cyber-attack functions that are divided into three classes. The cyber-attack function is represented by $f(.)$.

In partial reduction attacks, $f_1(.)$ decreases the real electricity consumption by a constant fraction α, while $f_2(.)$ involves a dynamic fraction $\beta(d,t)$. For selective by-pass attacks, $f_3(.)$ assumes an electricity consumption of zero during a specific time interval, $[t_i(d), t_f(d)]$, and reports the real consumption outside that interval. Price-based load control attacks take

TABLE I
ELECTRICITY THEFT CYBER-ATTACK FUNCTIONS

Attack Class	Equation
Partial reduction	$\mathrm{f}_1(E_c(d,t)) = \alpha E_c(d,t)$
	$\mathrm{f}_2(E_c(d,t)) = \beta(d,t)E_c(d,t)$
Selective by-pass	$\mathrm{f}_3(E_c(d,t)) = \begin{cases} 0 & \forall t \in [t_i(d), t_f(d)] \\ E_c(d,t) & \forall t \notin [t_i(d), t_f(d)] \end{cases}$
Price-based load control	$\mathrm{f}_4(E_c(d,t)) = \mathbb{E}[E_c(d)]$
	$\mathrm{f}_5(E_c(d,t)) = \beta(d,t)\mathbb{E}[E_c(d)]$
	$\mathrm{f}_6(E_c(d,t)) = E_c(d, T-t+1)$

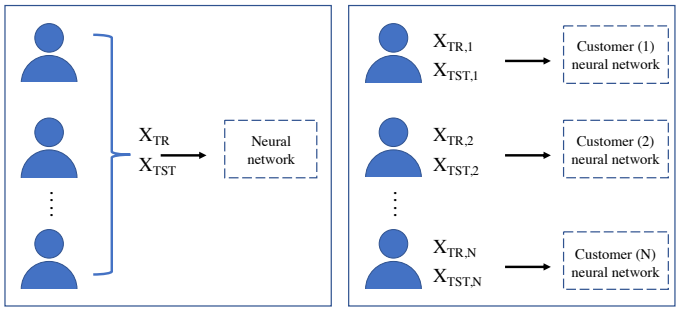
place where the price of electricity is different throughout the day. Hence, $f_4(.)$ reports a constant value of electricity consumption throughout the day. Notation $\mathbb{E}[\cdot]$ denotes the expectation (averaging) operator. Reporting a constant value of electricity consumption throughout the day can be detected easily. In order to avoid that, $f_5(.)$ employs a dynamic fraction $\beta(d,t)$. Finally, $f_6(.)$ is considered to be a reverse function as it reorders the electricity consumption reports across the day such that higher electricity consumption is reported when the tariff is low. Each of these cyber-attack functions is applied to the customer's electricity consumption profile matrix $\boldsymbol{E}_c$, and results in six malicious matrices per customer. The aggregation of all six attack matrices creates a comprehensive malicious dataset that captures a wide range of electricity theft behaviors. For each matrix, each row represents a sample of the electricity consumption profile across the day. Each sample is labeled. If the sample is benign, the label is 0, whereas if the sample is malicious, the label is 1.

III. IMPACT OF DATA POISONING ATTACKS

This section studies the impact of data poisoning attacks on generalized and customer-specific benchmark detectors. We first discuss the data preparation for generalized and customer-specific detectors. Next, we discuss how to launch a data poisoning attack. Then, we present a set of shallow and deep anomaly (novelty) detectors and classifiers to detect electricity thefts. Anomaly detection is a broad term that includes outlier and novelty detection. Outlier detectors are trained on benign and outlier data, whereas novelty detectors are trained only on benign data [20], which represents the technique adopted herein. Finally, we present the performance results to quantify the impact of data poisoning attacks on electricity theft detectors.

A. Dataset Preparation

This subsection discusses the dataset preparation for generalized and customer-specific detectors along with launching data poisoning attacks. For each detection type, we consider two approaches. The first is novelty detection where the detector is trained using benign data only and tested on benign and malicious samples. The second is two-class detection where the detector is trained and tested on benign and malicious datasets.



(a) Generalized neural network model (b) Customer-specific neural network model

Fig. 1. Illustration of generalized and customer-specific detectors.

1) Novelty Detectors: For generalized novelty detectors, data samples are concatenated from all customers as illustrated in Figure 1(a). The concatenated benign samples from all customers are further split into a disjoint training set and test set with ratio $2:1$. Malicious samples are then concatenated with the benign test set to form the final test set. As a result, we end up with imbalanced data since we accumulate more malicious data (six malicious matrices per customer) than the original benign data in the test set, which may result in misleading performance results. To overcome this, we adopt the adaptive synthetic sampling approach (ADASYN) [21] to balance the benign and malicious samples by over-sampling the minor (benign) class in the test set. Feature scaling is applied to the training set so that the consumption samples from all customers at all periods present equal influence during the detector's training. The scaled training set X_{TR} has zero-mean and unit-variance. The same scale is applied to the test set X_{TST} with labels Y_{TST}. Since the same scale is applied to both the train and test set, the change in the signal shape does not impact the overall process.

For customer-specific novelty detectors, a detector is developed for each customer as illustrated in Figure 1(b). Each detector is trained and tested using the dataset of a specific customer c, with a unique identification for each customer. Thus, the same steps as above are carried out using the dataset of each customer c to develop the scaled training set $X_{\text{TR},c}$ and test set $X_{\text{TST},c}$ with labels $Y_{\text{TST},c}$.

2) Two-class Detectors: For generalized two-class detectors, we concatenate both benign and malicious samples for all customers. Then, we employ the ADASYN approach to balance the benign and malicious samples. The concatenated dataset is then split into disjoint training and test sets with ratio $2:1$. Feature scaling is applied to the training set, which yields the final scaled version X_{TR} with label Y_{TR}. The same scale is then applied to the test set, which yields the final scaled version X_{TST} with label Y_{TST}. The same steps are applied to customer-specific two-class detectors but using only the dataset of customer c to develop training samples $X_{\text{TR},c}$ with labels $Y_{\text{TR},c}$ and test samples $X_{\text{TST},c}$ with labels $Y_{\text{TST},c}$.

3) Data Poisoning Attacks: In order to investigate the impact of data poisoning attacks on the detector's performance, malicious samples are falsely identified to be benign. Hence, under data poisoning attacks, part of the data used during

the training stage (whether novelty or two-class detection is adopted) is considered to be benign, while this data is actually malicious. Along with the six malicious attacks discussed in Section II.B, we launch data poisoning attacks using different attack penetration levels. Specifically, for generalized detectors, we consider the cases where 0%, 10%, 20%, and 30% of customers present poisoned data. For customer-specific detectors, we consider the cases where 0%, 10%, 20%, and 30% of customer c samples are poisoned.

B. Benchmark Detectors

This subsection presents a series of shallow and deep learning detectors to study the impact of data poisoning attacks.

1) Novelty Detectors: The following detectors are trained only on benign data but tested against benign and malicious data. These detectors are robust against zero-day attacks.

a) ARIMA-based Detection: This model represents a shallow anomaly detector that is trained to predict future energy consumption with minimum prediction mean square error (MSE). The model then detects malicious samples in the testing stage whenever the MSE is above some threshold [6].

b) Auto-encoder with Attention: This model represents a deep novelty detector. The structure of the AEA consists of an encoder and a decoder, based on long-short-term memory (LSTM) recurrent layers [22], along with an attention layer. The input to the LSTM encoder is an energy consumption sample ($\boldsymbol{x} \in \boldsymbol{X}_{\text{TR}}$ for the generalized detector or $\boldsymbol{x} \in \boldsymbol{X}_{\text{TR},c}$ for the customer-specific detector). Then, the LSTM encoder encodes that time-series vector into a hidden state. The input layer in the encoder is followed by L_{A} hidden LSTM layers, and each layer presents N_{A} LSTM cells. Then, the output of the LSTM encoder as well as the hidden state of the decoder are used as inputs to an attention layer. This is done to assign different weights and scores to each time step such that higher importance is given to time steps that contribute more to obtaining the needed output [23], [24]. After that, the concatenation of the output from the attention layer and the reconstructed output in the decoder are fed as input to the decoder. The detailed structure of the AEA is shown in Figure 2. Since the AEA is trained on benign samples, the reconstruction error will be minimum for benign test samples and large for malicious test samples. Hence, a certain threshold is used to differentiate benign from malicious test samples.

Overall, an LSTM cell presents a state c_t at a given time t and outputs a hidden state h_t. The access to this cell is controlled by input, forget, and output gates, $i_{\text{E},t}$, $f_{\text{E},t}$, and $o_{\text{E},t}$ for the encoder and $i_{\text{D},t}$, $f_{\text{D},t}$, and $o_{\text{D},t}$ for the decoder, respectively. The electricity consumption value at a given time t, $\boldsymbol{x}_t$, and the prior hidden states of all LSTM cells that are in the same layer $h_{\text{E},t-1}$ for the encoder and $h_{\text{D},t-1}$ for the decoder are received by the LSTM cell. Additionally, the LSTM cell receives the cell state, $c_{\text{E},t-1}$ for the encoder and $c_{\text{D},t-1}$ for the decoder. The calculations of $i_{\text{E/D},t}$, $f_{\text{E/D},t}$, $o_{\text{E/D},t}$, $c_{\text{E/D},t}$, and $h_{\text{E/D},t}$ are shown in lines 9 – 13 and 28 – 32 of Algorithm 1. As shown in Figure 2, the attention layer receives the hidden states $\boldsymbol{h}_{\text{E},t}^{L_A}$ and $\boldsymbol{h}_{\text{D},t-1}^{L_A}$ of the encoder and decoder, respectively. The attention layer outputs a context vector $\boldsymbol{c}_{\text{v},t}$ that is calculated using: (a) an alignment scoring function $\boldsymbol{m}$, which is a feed forward neural network trained on $\boldsymbol{h}_{\text{E},t}^{L_A/2}$ and $\boldsymbol{h}_{\text{D},t-1}^{L_A}$, (b) a softmax function $\boldsymbol{s}$, and (c) a multiplication layer. The calculations are given in Lines 16 – 18 of Algorithm 1. The decoder's hidden layers receive the concatenation of $\boldsymbol{c}_{\text{v},t}$ and the reconstructed output, x_{A}, $\sum(\boldsymbol{c}_{\text{v},t}, x_{\text{A}})$.

2) Two-class Detectors: The following detectors are trained and tested on both benign and malicious data.

a) Random forest-based Detection: This classifier integrates multiple decision trees that control overfitting while handling high-dimensional data and maintaining the computational efficiency [12].

b) AdaBoost-based Detection: This classifier uses decision trees as weak classifiers. It works by setting more weights on the samples that are harder to classify and less weight on the samples that can be easily classified.

c) SVM-based Detection: This represents a shallow and static classifier. The SVM detector is trained using the benign and malicious samples along with their labels in order to learn to predict the sample's label during the testing stage.

d) Feed Forward Neural Network: This represents a deep classifier, but it is still static, i.e., it does not capture well the temporal correlation that is present in the electricity consumption time-series data. The classifier consists of a set of L_{F} hidden layers and each layer has N_{F} neurons. The energy consumption data ($\boldsymbol{x} \in \boldsymbol{X}_{\text{TR}}$ for the generalized detector or $\boldsymbol{x} \in \boldsymbol{X}_{\text{TR},c}$ for the customer-specific detector) is fed to an input layer and then processed by the L_{F} hidden layers. An activation function $A_{\text{H,F}}$ is defined for the neurons in the hidden layers. An output layer with $A_{\text{O,F}}$ activation function then yields the predicted label. The detector is trained using benign and malicious data and labels in order to learn the weight matrix $\boldsymbol{W}_F$ for the connections between the neurons of different layers and the bias vector $\boldsymbol{b}_F$ that minimize the cross-entropy function:

$$C = \min_{\Theta} \frac{-1}{|X_{\text{TR}}|} \sum_{X_{\text{TR}}} \{y^{\text{T}}(\boldsymbol{x}) \ln(\tilde{y}) + (1 - y^{\text{T}}(\boldsymbol{x})) \ln(\tilde{y})\}, \quad (1)$$

where Θ denotes the model parameters ($\boldsymbol{W}_F$ and $\boldsymbol{b}_F$), $|X_{\text{TR}}|$ denotes the total number of training samples, $\tilde{y}$ represents the predicted output (label) of the detector, and T stands for the transposition operation. For customer specific-detectors, X_{TR} in (1) is replaced by $X_{\text{TR},c}$. Further details about the feed forward detector's training are given in [10].

e) Gated Recurrent Neural Network: This represents a deep classifier. Unlike the previous two classifiers, GRU layers are efficient in capturing patterns of temporal correlation and sequential information that are present in customers' electricity consumption time-series data [25]. In the detector, the hidden recurrent part contains L_{G} hidden layers, each with N_{G} GRUs. Hence, the electricity consumption data ($\boldsymbol{x} \in \boldsymbol{X}_{\text{TR}}$ for the generalized detector or $\boldsymbol{x} \in \boldsymbol{X}_{\text{TR},c}$ for the customer-specific detector) is fed to an input layer and then processed by the L_{G} hidden layers. All of the GRU layers, except for the last one, receive a sequence vector as input and yield a sequence output vector. The following parameters are presented for each of the GRU hidden layers $l_{\text{G}} \in \{2, \ldots, L_{\text{G}} - 1\}$:

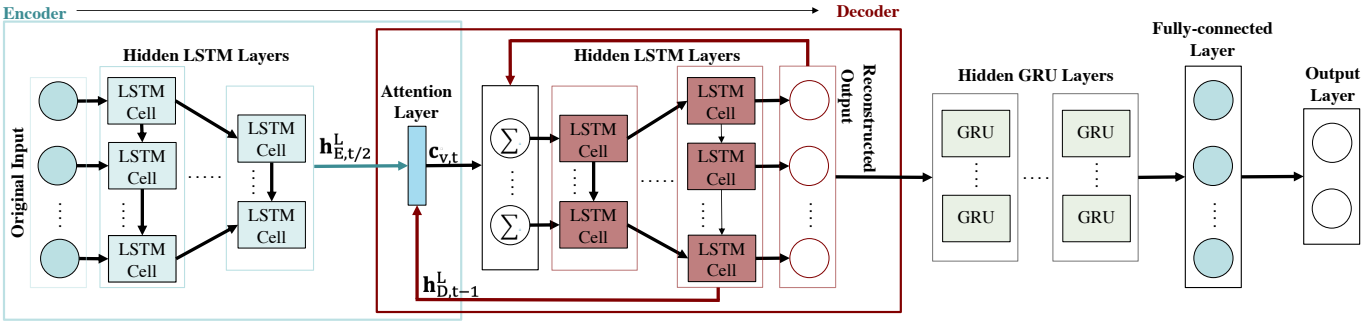


Fig. 2. Illustration of the proposed robust detector based on sequential ensemble.

- $\boldsymbol{o}_t^{l_\mathrm{G}-1}$ represents the input at time step t and the output of the previous layer $l_\mathrm{G}-1$.
- $\boldsymbol{s}_t^{l_\mathrm{G}}$ represents the hidden state at time step t, which is the computed memory in the previous hidden state $\boldsymbol{s}_{t-1}^{l_\mathrm{G}}$ of the same layer.
- $\boldsymbol{z}_t^{l_\mathrm{G}} = \sigma(\boldsymbol{o}_t^{l_\mathrm{G}-1}\boldsymbol{U}_z^{l_\mathrm{G}} + \boldsymbol{s}_{t-1}^{l_\mathrm{G}}\boldsymbol{W}_z^{l_\mathrm{G}} + \boldsymbol{b}_z^{l_\mathrm{G}})$ is the update gate that is specified by the new input $\boldsymbol{o}_t^{l_\mathrm{G}-1}$ and previous memory $\boldsymbol{s}_{t-1}^{l_\mathrm{G}}$; $\boldsymbol{U}_z^{l_\mathrm{G}}$ and $\boldsymbol{W}_z^{l_\mathrm{G}}$ are learnable weights, $\sigma(.)$ is an activation function, and $\boldsymbol{b}_z^{l_\mathrm{G}}$ is a bias vector.
- $\boldsymbol{r}_t^{l_\mathrm{G}} = \sigma(\boldsymbol{o}_t^{l_\mathrm{G}-1}\boldsymbol{U}_r^{l_\mathrm{G}} + s_{t-1}^{l_\mathrm{G}}\boldsymbol{W}_r^{l_\mathrm{G}} + \boldsymbol{b}_r^{l_\mathrm{G}})$ is the reset gate that specifies the amount of the prior memory $\boldsymbol{s}_{t-1}^{l_\mathrm{G}}$ that takes part in the next state using the equation $\boldsymbol{h}_t^{l_\mathrm{G}} = \tanh(\boldsymbol{o}_t^{l_\mathrm{G}-1}\boldsymbol{U}_h^{l_\mathrm{G}} + (\boldsymbol{s}_{t-1}^{l\mathrm{G}} \odot \boldsymbol{r}^{l\mathrm{G}})\boldsymbol{W}_h^{l\mathrm{G}} + \boldsymbol{b}_h^{l\mathrm{G}})$, where $\boldsymbol{U}_r^{l\mathrm{G}}$, $\boldsymbol{W}_r^{l\mathrm{G}}$, $\boldsymbol{U}_h^{l\mathrm{G}}$, and $\boldsymbol{W}_h^{l\mathrm{G}}$ are learnable weight matrices, $\boldsymbol{b}_r^{l\mathrm{G}}$ and $\boldsymbol{b}_h^{l\mathrm{G}}$ are bias vectors, and $\odot$ is the Hadamard product.
- $\boldsymbol{s}_{t+1}^{l\mathrm{G}} = (1-\boldsymbol{z}^{l\mathrm{G}}) \odot \boldsymbol{h}^{l\mathrm{G}} + \boldsymbol{z}^{l\mathrm{G}} \odot \boldsymbol{s}_t^{l\mathrm{G}}$ is the next state and the output at time $t+1$ is $\boldsymbol{o}_{t+1}^{l\mathrm{G}} = \mathrm{softmax}(\boldsymbol{V}^{l\mathrm{G}}\boldsymbol{s}_{t+1}^{l\mathrm{G}} + \boldsymbol{b}_o^{l\mathrm{G}})$, where $\boldsymbol{V}^{l\mathrm{G}}$ is a learnable weight matrix.

Denote the hidden layers' activation functions by $A_{\mathrm{H,G}}$. The activation function $A_{\mathrm{O,G}}$ of the output layer predicts a label for the input sample. During the training stage, the detector uses the benign and malicious data and labels in order to learn the model parameters (matrices $\boldsymbol{W}_{G,*}, \boldsymbol{U}_{G,*}, \boldsymbol{V}_G$ and bias vectors $\boldsymbol{b}_{G,*}$) that minimize the cross-entropy function defined in (1). Further details about the training of the GRU-based detector are found in [26].

C. Hyper-parameter Optimization

To ensure that we capture the best performance out of each detector, hyper-parameter optimization is implemented [27]. For the random forest and AdaBoost classifiers, the optimal number of estimators are 100 and 50, respectively. For SVM, the optimal regularization parameter and kernel turn out to be 1.0 and Sigmoid, respectively. The following hyper-parameters are considered for the deep detectors: (a) number of hidden layers (L_A, L_F, L_G for AEA, feed forward, and GRU detectors), (b) number of neurons (cells/units) in the hidden layers (N_A, N_F, N_G for AEA, feed forward, and GRU detectors), (c) type of optimizer O to find the model parameters, (d) dropout rate B, (e) weight constraints G, (f) type of hidden and output activation functions. To reduce the computational complexity, we implement a sequential grid search analysis where each of the hyper-parameters is optimized in sequential stages separately by finding the best hyper-parameter that provides the best reported DR during each stage. More details about sequential grid search can be found in [10]. The search space for the hyper-parameter values are: Number of layers: $\mathcal{L}_{(.)}$ = {2, 4, 6, 8}. Number of neurons: $\mathcal{N}_{(.)}$ = {100, 200, 300, 500, 1000}. Optimizers: $\mathcal{O}$ = {SGD, Adam, Adadelta, Adamax}. Dropout rates: $\mathcal{B}$ = {0, 0.2, 0.4, 0.5}. Weight constraints: $\mathcal{G}$ = {0, 1, 3, 5}. Hidden activation functions: $\mathcal{A}_\mathrm{H}$ = {ReLU, Sigmoid, Linear, tanh}. Output activation functions: $\mathcal{A}_\mathrm{O}$ = {Softmax, Sigmoid}. Cross-validation is implemented over X_TR for the generalized detectors and $X_{\mathrm{TR},c}$ for the customer-specific detectors.

D. Performance Evaluation

This subsection evaluates the impact of data poisoning attacks on the benchmark detectors presented in Section III.B.

1) Evaluation Metrics: When the model correctly detects a malicious sample as malicious, it is considered to be true positive (TP). When it correctly detects a benign sample as benign, it is represented as true negative (TN). When it incorrectly detects a benign sample as malicious, it is referred to as false positive (FP). When it incorrectly detects a malicious sample as benign, it is considered as false negative (FN). To assess the performance of the detectors from different aspects, we adopt seven performance evaluation metrics: (1) Detection rate (DR = TP/(TP + FN)) specifies the number of malicious samples that the detector correctly identifies as malicious. (2) False alarm (FA = FP/(FP + TN)) computes the number of benign samples that are incorrectly identified as malicious by the detector. (3) Specificity (SP = 100-FA). (4) Precision (PR = TP/(TP+FP)) determines the ratio of correctly detected malicious samples to the total number of malicious samples. (5) Accuracy (ACC = (TP + TN)/(TP + TN + FP + FN)) determines how well the model correctly detects benign and malicious samples. (6) F1-score provides the harmonic mean of PR and DR. (7) The area under the curve (AUC) of the receiver operating characteristic (ROC) plots the TP versus the FP.

2) Evaluation Results: Keras sequential API is used for the training of the detectors. We adopt the following for deep learning detectors: the number of epochs $I = 50$, batch size $K = 100$, initial hyper-parameters of the models are: optimizer = SGD, dropout rate and weight constraint = 0, hidden layer activation = ReLU, and output layer activation = Sigmoid.

TABLE II
OPTIMAL HYPER-PARAMETERS OF THE GENERALIZED DETECTORS

Hyper-parameter	AEA	Feed-forward	GRU
$L_{(.)}$	6	6	8
$N_{(.)}$	$500, 300, 200$	500	300
O	SGD	Adamax	Adam
B	0	0	0.2
G	1	3	5
A_{H}	Sigmoid	ReLU	ReLU
A_{O}	Sigmoid	Sigmoid	Softmax

a) Optimal Hyper-parameters: Table II shows the optimal hyper-parameters' combination for the generalized detectors. For the AEA, 3 layers are used in the encoder and additional 3 layers are used in the decoder. The number of LSTM cells in the encoding layers is $(500, 300, 200)$ and $(200, 300, 500)$ in the decoding layers. The hyper-parameters in Table II are also used by the majority of the customer-specific detectors.

b) Optimal Threshold Values for Novelty Detectors: For the novelty detectors, ARIMA and AEA, the calculated labels $\boldsymbol{Y}_{\text{CAL}}$ and $\boldsymbol{Y}_{\text{TST}}$ are compared against each other to produce a confusion matrix that is used to calculate the different evaluation metrics. To determine $\boldsymbol{Y}_{\text{CAL}}$, thresholds are introduced for each developed detector. As discussed earlier, comparing the prediction MSE/reconstruction error against such a threshold is done to distinguish between benign and malicious samples. For each novelty detector, the median of the interquartile range (IQR) of the receiver operating characteristic (ROC) curve determines the threshold, where scores under that specific threshold value represent benign samples and the scores above it denote malicious samples. Hence, each ROC curve is divided into three quartiles to take the median of the IQR. Thus, the threshold value for ARIMA is 0.58 and 0.51 for the AEA.

c) Computational Complexity: The training of all the detectors is done offline using the NVIDIA GeForce RTX 2070 hardware accelerator. Approximately, it takes an hour to train the shallow detectors and $1.5 - 3$ hours to train the deep detectors. For all the detectors, the testing is done online, which requires around 2 seconds to report a decision regarding individual readings.

d) Detection Performance: Simulation results of generalized and customer-specific detectors are shown in Tables III and IV, respectively. The results are reported using a set of performance metrics. Along with the six malicious attacks discussed in Section II.B, the detectors are tested without data poisoning (0%) as well as with 10%, 20%, and 30% of data poisoning. From the results in Tables III and IV, we can see that the DR of the detectors ranges from $80 - 94\%$ when all labels are correct (i.e., 0% penetration level of data poisoning). Hence, roughly $6 - 20\%$ of malicious users go undetected and so they will be falsely labeled by the utility company as benign users. Similarly, FA when all labels are correct ranges from $5 - 20\%$, and hence, $5 - 20\%$ honest users may be falsely labeled as malicious. Hence, based on a set of state-of-the-art benchmarks that cover shallow and deep classifiers as well as novelty detectors, we conclude that false labeling may occur, on average, up to 20%. Since this is on average, we study the impact of different levels of poisoning attack strengths from $0 - 20\%$ and we even consider the more extreme case of 30% attack penetration level to gain more insights.

Table III shows the performance results of the generalized detectors. Without data poisoning, the AEA-based novelty detector outperforms the rest of the models roughly by $1.6 - 11.9\%$ in DR. For all of the generalized detectors, with 10% data poisoning, the average reduction in detection performance is 4%. After poisoning 20% and 30% of the training data, the average degradation compared to the 0% case is 9% and 14.8%, respectively. The average performance degradation of the generalized detectors per data poisoning stage is roughly 4.8%. The impact of data poisoning attacks on the DR of the generalized detectors throughout the different data poisoning percentages is visualized in Figure 3(a).

Table IV shows the performance of customer-specific detectors (averaged over the customers). Without data poisoning, the customer-specific AEA-based model outperforms the rest of the models by $2 - 12.9\%$ in DR. For all of the customer-specific detectors, with 10% of data poisoning, the degradation is roughly 4.4%. After poisoning 20% and 30% of the training data, the average degradation compared to the 0% case is 9.8% and 16%, respectively. The average performance degradation of the customer-specific models per data poisoning stage is 5.3%. The impact of data poisoning attacks on the DR of the customer-specific detectors throughout the different data poisoning percentages is visualized in Figure 3(b).

3) Remarks: The following conclusions can be drawn:

- Overall, generalized detectors perform better than customer-specific detectors whether or not the system encounters data poisoning attacks. The average improvement in detection performance is roughly $2 - 3\%$.
- Generalized detectors tend to be more robust than customer-specific detectors. Generalized detectors use data from all customers in the neighborhood to do the training. For example, consider the case with penetration level of 10% data poisoning, in this case, while 10% of the customers have false labels, the remaining 90% customers have correct labels. Each single customer adds up roughly $24,000$ data points. Consider the very simple scenario of 10 customers. Now we have $216,000$ data points with correct labels and $24,000$ data points with false labels. On the other hand, with customer specific detectors, training is done using only the customer's data. Hence, when 10% of the data is poisoned, we have $2,400$ data points with false labels and $21,600$ data points with correct labels. While the poisoning percentage is the same in both generalized and customer specific detectors, the amount of data is different. The amount of data with correct and false labels affects the detector's decision boundary. Hence, one advantage of generalized detectors is that they aggregate data from many customers, and hence they are able to better capture distinctive features of honest customers, which eventually enhance the detector's robustness.
- For the generalized detectors, deep learning-based approaches outperform shallow techniques by roughly 5% in DR when there is no data poisoning attack. Under

TABLE III
IMPACT OF DATA POISONING ON GENERALIZED DETECTORS (%)

Model	Metric	Poisoning Percentage			
		0%	10%	20%	30%
Random forest	**DR**	82.2	77.8	72.7	66.8
	FA	17.6	20.3	26.4	33.3
	SP	82.4	79.7	73.6	66.7
	PR	82.1	77.7	73.8	66.6
	ACC	82.3	78.7	73.1	66.7
	F1	82.1	77.7	72.7	66.7
	AUC	81.4	78.6	73.9	66.7
AdaBoost	**DR**	85.7	81.2	76.2	70.1
	FA	14.1	18.4	23.3	29.9
	SP	85.9	81.6	76.7	70.1
	PR	85.3	81.1	76.1	70.0
	ACC	85.8	81.4	76.4	70.1
	F1	85.5	81.1	76.1	70.0
	AUC	85.0	82.1	77.3	70.0
ARIMA	**DR**	87.8	83.3	78.1	72.0
	FA	12.4	16.8	22.1	28.4
	SP	87.6	83.2	77.9	71.6
	PR	87.1	83.0	78.2	72.1
	ACC	87.7	83.2	78.0	71.8
	F1	87.4	83.1	78.1	72.0
	AUC	87.1	84.2	79.1	72.1
SVM	**DR**	89.2	84.8	79.7	73.7
	FA	10.2	14.5	19.6	25.7
	SP	89.8	85.5	80.4	74.3
	PR	89.0	84.5	79.5	74.0
	ACC	89.5	85.1	80.0	74.0
	F1	89.1	84.6	79.6	73.8
	AUC	89.5	85.7	79.8	74.0
Feed forward	**DR**	90.8	86.5	81.6	76.0
	FA	9.3	13.5	18.5	24.4
	SP	90.7	86.5	81.5	75.6
	PR	90.0	86.5	81.5	75.8
	ACC	90.7	86.5	81.5	75.8
	F1	90.4	86.5	81.5	75.9
	AUC	91.1	86.4	81.3	76.1
GRU	**DR**	92.4	88.3	83.7	78.5
	FA	6.8	10.8	15.3	20.6
	SP	93.2	89.2	84.7	79.4
	PR	92.3	88.7	84.0	79.0
	ACC	92.8	88.7	84.2	78.9
	F1	92.3	88.5	83.8	78.7
	AUC	92.1	88.2	83.8	79.4
AEA	**DR**	94.1	90.2	85.8	80.8
	FA	5.2	9.2	13.5	18.4
	SP	94.8	90.8	86.5	81.6
	PR	94.5	90.3	85.2	80.6
	ACC	94.4	90.5	86.1	81.2
	F1	94.3	90.2	85.5	80.7
	AUC	94.0	90.1	85.9	80.3

TABLE IV
IMPACT OF DATA POISONING ON CUSTOMER-SPECIFIC DETECTORS (%)

Model	Metric	Poisoning Percentage			
		0%	10%	20%	30%
Random Forest	**DR**	79.1	74.3	68.7	62.0
	FA	19.4	24.6	29.9	35.1
	SP	80.6	75.4	70.1	64.9
	PR	78.6	74.7	68.9	62.5
	ACC	79.8	74.8	69.4	63.4
	F1	78.8	74.5	68.8	62.2
	AUC	79.1	75.0	69.3	62.1
AdaBoost	**DR**	82.5	77.8	72.2	65.5
	FA	16.6	21.2	26.8	32.9
	SP	83.4	78.8	73.2	67.1
	PR	82.0	78.1	72.2	65.9
	ACC	82.9	78.3	72.7	66.3
	F1	82.2	77.9	72.2	65.7
	AUC	82.3	78.3	72.7	65.4
ARIMA	**DR**	84.6	79.9	74.3	67.7
	FA	14.6	19.3	24.8	31.2
	SP	85.4	80.7	75.2	68.8
	PR	84.1	80.0	74.0	67.9
	ACC	85.0	80.3	74.7	68.2
	F1	84.3	79.9	74.1	67.8
	AUC	84.3	80.1	74.2	67.8
SVM	**DR**	87.1	82.6	77.2	70.8
	FA	12.4	17.0	22.4	28.7
	SP	87.6	83.0	77.6	71.3
	PR	87.0	82.0	77.5	71.1
	ACC	87.3	82.8	77.4	71.0
	F1	87.0	82.3	77.3	70.9
	AUC	87.2	82.5	77.3	70.9
Feed forward	**DR**	88.7	84.3	79.1	72.8
	FA	11.7	16.2	21.5	27.7
	SP	88.3	83.8	78.5	72.3
	PR	88.5	84.0	78.9	72.4
	ACC	88.5	84.0	78.8	72.5
	F1	88.6	84.1	79.0	72.6
	AUC	88.6	85.8	78.5	72.4
GRU	**DR**	90.0	85.6	80.4	74.2
	FA	10.4	14.7	19.9	26.1
	SP	89.6	85.3	80.1	73.9
	PR	89.8	85.5	80.3	74.0
	ACC	89.9	85.4	80.2	74.0
	F1	89.9	85.5	80.3	74.1
	AUC	89.7	85.8	80.3	74.0
AEA	**DR**	92.0	87.7	82.7	76.7
	FA	8.3	12.4	17.4	22.5
	SP	91.7	87.6	82.6	77.5
	PR	91.8	87.5	82.6	77.9
	ACC	91.8	87.6	82.6	77.1
	F1	91.9	87.6	82.6	77.2
	AUC	91.4	87.3	82.6	77.3

strong data poisoning attack (30%), the performance improvement is up to 8% in DR.

- Out of the three deep learning-based detectors, AEA performs the best, followed by GRU-based, and then by feed forward detectors. Both AEA and GRU-based detectors capture (and hence leverage) the temporal correlation present in the electricity consumption time-series data.
- All investigated detectors suffer roughly from 17% degradation in detection performance when the system encounters strong data poisoning attacks (30%).

IV. ROBUST ELECTRICITY THEFT DETECTION

Since generalized deep learning-based detectors offer better performance whether or not the system encounters data poisoning attacks, we rely on them to propose a more robust detector that presents marginal deterioration in detection performance even in the presence of strong data poisoning attacks. Herein, ensemble learning is used to design a robust detector that combines deep AEA, GRU-based RNN, and feed forward neural network to extract more distinctive representative fea-

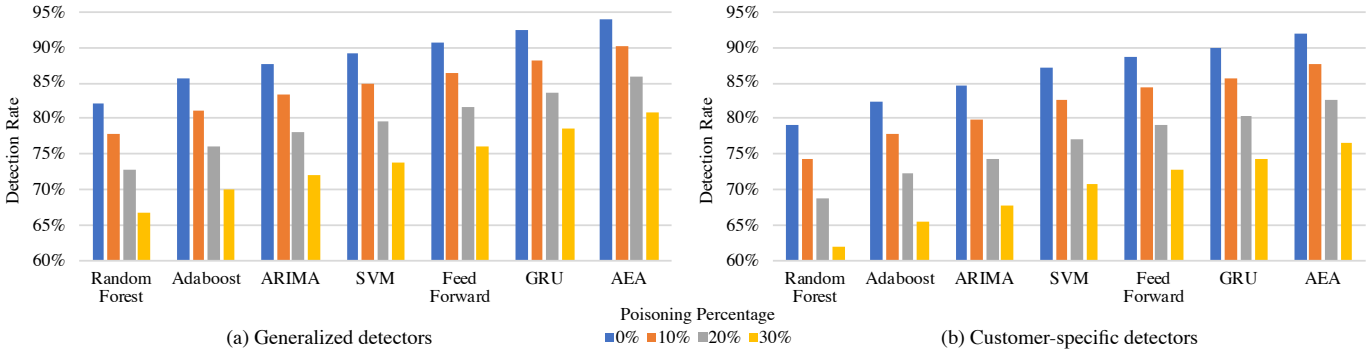


Fig. 3. Impact of data poisoning attacks on the DR of generalized and customer-specific benchmark detectors.

tures and hence improve the overall detection performance against data poisoning attacks. Below, we investigate two ensemble learning techniques, namely, ensemble averaging and sequential ensemble. By extracting more distinctive representative features, we expect that ensemble-based models will improve the detection performance against data poisoning attacks.

A. Ensemble Averaging

Ensemble averaging involves taking the average of the outputs of AEA, GRU, and feed forward detectors into another fully-connected layer to make a final decision. Such an approach helps to improve the performance of stand-alone models since the various errors of models are averaged out. However, the performance of the ensemble average is not significantly better than the best stand-alone detector (i.e., AEA). The ensemble averaging detector is included here for comparison and validation purposes.

B. Sequential Ensemble

Figure 2 illustrates the proposed architecture of a sequential ensemble-based generalized detector that consists of an input layer, an AEA with LSTM cells, recurrent layers with GRUs, a fully-connected layer, and an output layer, all placed in sequence. The rationale behind placing these layers in this specific sequential order is highlighted next. Since AEA gives the best detection result, we place it as the front layers of the detector. This benefits from two advantages: (a) differentiating between normal and anomalous behavior and (b) capturing temporal correlations within the data. The output from the AEA is then fed to recurrent layers based on GRUs to further extract hidden features within the reconstructed data. Finally, the fully connected layer reshapes the output from the recurrent GRU layers to make a final decision at the output layer. As can be observed, ensemble averaging is more like treating the detectors in parallel, while sequential ensemble treats the detectors in series to extract more distinctive features that help in improving the detection performance.

During the training stage, the optimal weights and bias values for the AEA, GRU, and fully connected layers are learned. The optimization objective consists of minimizing the cross-entropy cost function (1). Algorithm 1 presents the proposed robust electricity theft detector's training process, which is carried out using an iterative gradient descent optimization algorithm. This is achieved by splitting X_{TR} into equal-sized M mini-batches and executing feed forward as well as back-propagation steps for I (total) iterations. In the feed forward stage, the training samples in the mini-batch pass through all of the layers in the network to compute the predicted output vectors. Lines 6 – 34, 26 – 34, and 49 in Algorithm 1 correspond to the AEA, recurrent GRU, and fully connected sections of the robust detector, respectively. In the back-propagation stage, to calculate the gradient of cost function (1) with respect to the network weights, the mini-batches are used [27]. Thus, the calculated gradients are used to update each iteration's biases and weights.

C. Performance Evaluation

The same evaluation metrics are used as depicted in Section III.D. Furthermore, the same initialization values are used for the detector's hyper-parameters as discussed in Section III.D. Sequential grid search is conducted to find the detector optimal hyper-parameters, which are found to be: 3 layers in the encoder part and 3 layers in the decoder part of the AEA, the number of LSTM cells in the encoding layers is $(500, 300, 200)$ and $(200, 300, 500)$ in the decoding layers, 8 recurrent layers each with 300 GRUs, 500 neurons in the fully connected layer, the used optimizer is Adam, no dropout rate, weight constraint of 1, ReLU hidden activation function, and Sigmoid output activation function. It takes around 3 and 4 hours to train offline the ensemble averaging and sequential ensemble-based detectors, respectively. The ensemble averaging-based detector is faster since the models can be trained in parallel. For both detectors, the online testing of the models requires around 2 seconds to report a decision.

Table V summarizes the performance evaluation results for the ensemble learning-based detectors compared to the AEA model (which exhibited the best detection performance among the benchmark detectors discussed in Section III). The ensemble averaging-based detector presents a slightly improved performance compared to the benchmark detectors. However, as expected, since the individual detectors run in parallel, the performance of the ensemble average detector is limited by the performance of the best detector, i.e., the

Algorithm 1: Training of the robust detector

Input Data: $\boldsymbol{X}_{\text{TR}}$
Initialization: Weights $\boldsymbol{U}^l_{(\cdot)}$, $\boldsymbol{W}^l_{(\cdot)}$, $\boldsymbol{V}^l_{(\cdot)}$, and bias $\boldsymbol{b}^l_{(\cdot)}$ $\forall l$, $\boldsymbol{h}^L_{\text{D},t-1}$ and x_{A}
while *not converged* **do**
  for *each training sample* $\boldsymbol{x}$ **do**
    Feed forward
    Encoder:
    for *each hidden layer* $l = 1, \ldots, L/2$ **do**
      for *each time step* t **do**
        $\boldsymbol{i}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_i \boldsymbol{x}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_i)$,
        $\boldsymbol{f}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_f \boldsymbol{x}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_f)$,
        $\boldsymbol{c}^l_{\text{E},t} = \boldsymbol{f}^l_{\text{E},t} \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{i}^l_{\text{E},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t + \boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{b}^l_c)$,
        $\boldsymbol{o}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_o \boldsymbol{x}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{E},t} + \boldsymbol{b}^l_o)$,
        $\boldsymbol{h}^l_{\text{E},t} = \boldsymbol{o}^l_{\text{E},t} \tanh(\boldsymbol{c}^l_{\text{E},t})$,
        Attention Layer:
        if $l = L/2$ **then**
          $\boldsymbol{m} = \Gamma(\boldsymbol{h}^{L/2}_{\text{E},t}, \boldsymbol{h}^L_{\text{D},t-1})$
          $\boldsymbol{s} = \exp(\boldsymbol{m}) / \sum_{|\boldsymbol{m}|} \exp(\boldsymbol{m})$
          $\boldsymbol{c}_{\text{V},t} = \sum_T \boldsymbol{s} \times \boldsymbol{h}^{L/2}_{\text{E},t}$.
        end
      end
      $\boldsymbol{h}'^l = \boldsymbol{h}^l_{\text{E},t}$, $\boldsymbol{c}'^l = \boldsymbol{c}^l_{\text{E},t}$.
    end
    $\breve{\boldsymbol{x}} = \sum(\boldsymbol{c}_{\text{V},t}, x_{\text{A}})$
    Decoder:
    The decoder hidden and cell states at initial time step are equal to $\boldsymbol{h}'^l$ and $\boldsymbol{c}'^l$
    for *each hidden layer* $l = L/2 + 1, \ldots, L$ **do**
      for *each time step* t **do**
        $\boldsymbol{i}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_i \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_i)$,
        $\boldsymbol{f}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_f \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_f)$,
        $\boldsymbol{c}^l_{\text{D},t} = \boldsymbol{f}^l_{\text{D},t} \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{i}^l_{\text{D},t} \tanh(\boldsymbol{W}^l_c \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{b}^l_c)$,
        $\boldsymbol{o}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_o \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{D},t} + \boldsymbol{b}^l_o)$,
        $\boldsymbol{h}^l_{\text{D},t} = \boldsymbol{o}^l_{\text{D},t} \tanh(\boldsymbol{c}^l_{\text{D},t})$,
      end
    end
    x_{A} denotes the reconstructed output in the AEA
    GRU Layers:
    $o^{l_{\text{G}}-1}_t = x_{\text{A}}$ for $l_{\text{G}} = 1$
    for *each recurrent layer lG* **do**
      for *each time step* t **do**
        $\boldsymbol{z}^{l_{\text{G}}}_t = \sigma(\boldsymbol{o}^{l_{\text{G}}-1}_t \boldsymbol{U}^{l_{\text{G}}}_z + \boldsymbol{s}^{l_{\text{G}}}_{t-1} \boldsymbol{W}^{l_{\text{G}}}_z + \boldsymbol{b}^{l_{\text{G}}}_z)$
        $\boldsymbol{r}^{l_{\text{G}}}_t = \sigma(\boldsymbol{o}^{l_{\text{G}}-1}_t \boldsymbol{U}^{l_{\text{G}}}_r + \boldsymbol{s}^{l_{\text{G}}}_{t-1} \boldsymbol{W}^{l_{\text{G}}}_r + \boldsymbol{b}^{l_{\text{G}}}_r)$
        $\boldsymbol{h}^{l_{\text{G}}}_t = \tanh(\boldsymbol{o}^{l_{\text{G}}-1}_t \boldsymbol{U}^{l_{\text{G}}}_h + (\boldsymbol{s}^{l_{\text{G}}}_{t-1} \odot \boldsymbol{r}^{l_{\text{G}}}) \boldsymbol{W}^{l_{\text{G}}}_h + \boldsymbol{b}^{l_{\text{G}}}_h)$
        $\boldsymbol{s}^{l_{\text{G}}}_t = (1 - \boldsymbol{z}^{l_{\text{G}}}) \odot \boldsymbol{h}^{l_{\text{G}}} + \boldsymbol{z}^{l_{\text{G}}} \odot \boldsymbol{s}^{l_{\text{G}}}_{t-1}$
        $\boldsymbol{o}^{l_{\text{G}}}_t = \text{softmax}(\boldsymbol{V}^{l_{\text{G}}} \boldsymbol{s}^{l_{\text{G}}}_t + \boldsymbol{b}^{l_{\text{G}}}_o)$
      end
    end
    x_{G} denotes the GRU output, $o^{L\text{G}}_t$
    Fully-connected Layer:
    Compute: $z^l(x_{\text{G}}) = W^l \sigma(x_{\text{G}}) + b^l$
    Back propagation: Compute:
    $\nabla_{\boldsymbol{U}^{l_{(\cdot)}}_{(\cdot)}} C(x)$, $\nabla_{\boldsymbol{V}^{l_{(\cdot)}}_{(\cdot)}} C(x)$, $\nabla_{\boldsymbol{W}^{l_{(\cdot)}}_{(\cdot)}} C(x)$, and $\nabla_{\boldsymbol{b}^{l_{(\cdot)}}_{(\cdot)}} C(x)$
  end
  Weight and bias update:
  $\boldsymbol{U}^{l_{(\cdot)}}_{(\cdot)} = \boldsymbol{U}^{l_{(\cdot)}}_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{U}^{l_{(\cdot)}}_{(\cdot)}} C(x)$
  $\boldsymbol{V}^{l_{(\cdot)}}_{(\cdot)} = \boldsymbol{V}^{l_{(\cdot)}}_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{V}^{l_{(\cdot)}}_{(\cdot)}} C(x)$
  $\boldsymbol{W}^{l_{(\cdot)}}_{(\cdot)} = \boldsymbol{W}^{l_{(\cdot)}}_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{W}^{l_{(\cdot)}}_{(\cdot)}} C(x)$
  $\boldsymbol{b}^{l_{(\cdot)}}_{(\cdot)} = \boldsymbol{b}^{l_{(\cdot)}}_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{b}^{l_{(\cdot)}}_{(\cdot)}} C(x)$
end
Output: Optimal $\boldsymbol{U}^{l_{(\cdot)}}_{(\cdot)}$, $\boldsymbol{W}^{l_{(\cdot)}}_{(\cdot)}$, $\boldsymbol{V}^{l_{(\cdot)}}_{(\cdot)}$, and $\boldsymbol{b}^{l_{(\cdot)}}_{(\cdot)}$ $\forall l$.

TABLE V
IMPACT OF DATA POISONING ON THE PROPOSED DETECTORS

Model	Metric	Poisoning Percentage 0%	10%	20%	30%
AEA	DR	94.1	90.2	85.8	80.8
	FA	5.2	9.2	13.5	18.4
	SP	94.8	90.8	86.5	81.6
	PR	94.5	90.3	85.2	80.6
	ACC	94.4	90.5	86.1	81.2
	F1	94.3	90.2	85.5	80.7
	AUC	94	90.1	85.9	80.3
Ensemble Averaging	DR	94.6	91.3	87.1	81.3
	FA	4.8	8.6	12.9	17.8
	SP	95.2	91.4	87.1	82.2
	PR	94.9	91.1	87.1	81.8
	ACC	94.9	91.3	87.1	81.7
	F1	94.7	91.2	87.1	81.5
	AUC	94.6	91.2	87.0	81.4
Sequential Ensemble	DR	95.2	94.3	93.3	92.2
	FA	2.9	3.7	4.7	5.8
	SP	97.1	96.3	95.3	94.2
	PR	95.6	95.0	94.3	92.7
	ACC	96.1	95.3	94.3	93.2
	F1	95.4	94.6	93.8	92.4
	AUC	97.4	95.2	93.1	92.0

AEA. Specifically, a degradation of roughly 4% is witnessed when 10% of training data is poisoned. This degradation is up to 8.4% when 20% of training data is poisoned. The performance is further deteriorated by 13.4% when 30% of training data is poisoned. As seen in Table V, the average performance degradation rate for the ensemble averaging-based model throughout the different data poisoning stages is 4.3%. Hence, the deterioration in performance for the ensemble average-based detector is still remarkable (roughly with 13% performance deterioration when 30% of training data is poisoned), which is very close to the AEA performance when subjected to data poisoning attacks.

The sequential ensemble-based detector presents a stable performance against data poisoning attacks and marginal deterioration in performance. More specifically, the detector experiences less than 1% deterioration in performance when 10% of training data is poisoned. The deterioration percentage is only 1.8% when 20% of training data is poisoned. The detector experiences only 3% deterioration in performance when 30% of the training data is poisoned. Hence, the average performance degradation rate of 1% per data poisoning stage. Overall, the robust detector outperforms the benchmark and ensemble-average detector roughly by $10 - 25\%$ when the system is subject to strong data poisoning attack (30%), which reflects a superior and robust detection performance under severe operating conditions. The sequential ensemble-based detector outperforms the rest of the stand-alone detectors since it is able to extract more distinctive features that improves the robustness of the detector against data poisoning attacks. Specifically, AEA differentiates malicious from benign behaviors while capturing well the temporal correlations within the data. GRUs further extract hidden features within the reconstructed data in the AEA. The fully connected layer reshapes the output from the GRUs to make a final decision at the output layer.

V. Conclusion

This paper provided answers to three major questions pertaining to the performance of electricity theft detectors in the presence of data poisoning attacks. Extensive simulation studies provide support for the following conclusions: (1) Generalized detectors offer generally superior performance compared to customer-specific detectors whether or not the system is subject to data poisoning attacks, and the performance improvement is up to 4%; (2) Deep learning-based detectors offer superior performance compared to shallow detectors and the performance improvement is up to 12%; (3) Electricity theft detectors generally experience severe deterioration in performance when subjected to data poisoning attacks, with degradation up to 17% under strong data poisoning attacks (e.g., when 30% of training data is poisoned); (4) Sequential ensemble detectors based on AEA, recurrent GRU layers, and a fully connected layer offer a stable and robust performance against data poisoning attacks with marginal deterioration in performance, which raises up to 3% under strong data poisoning attacks. Such a performance represents a $10-25\%$ improvement compared to shallow and deep learning-based benchmark detectors and ensemble average-based detectors in the presence of strong data poisoning attacks.

References

[1] P. Jokar, N. Arianpoo, and V. C. Leung, "Electricity theft detection in AMI using customers' consumption patterns," *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 216–226, 2016.

[2] C. Lin, S. Chen, C. Kuo, and J. Chen, "Non-cooperative game model applied to an advanced metering infrastructure for non-technical loss screening in micro-distribution systems," *IEEE Transactions on Smart Grid*, vol. 5, no. 5, pp. 2468–2469, Sep. 2014.

[3] V. B. Krishna, C. A. Gunter, and W. H. Sanders, "Evaluating detectors on optimal attack vectors that enable electricity theft and der fraud," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 4, pp. 790–805, 2018.

[4] T. Popovic, C. Blask, M. Carpenter, S. Chasko, G. Chason, G. Ciocarlie, F. Cleveland, B. Davison, D. DeBlasio, D. Dickinson, M. David, P. Duggan, M. Ellison, S. Eswarahally, I. Gassko, E. Gonzales, S. Griffin, V. Hammond, J. Henry, and S. Rosenberger, "Electric sector failure scenarios and impact analyses - version 3.0 (NESCOR)," 12 2015.

[5] R. Wu, L. Wang, and T. Hu, "Adaboost-svm for electrical theft detection and grnn for stealing time periods identification," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 3073–3078.

[6] V. Badrinath Krishna, R. K. Iyer, and W. H. Sanders, "ARIMA-Based modeling and validation of consumption readings in power grids," in *Critical Information Infrastructures Security*. Springer International Publishing, 2016, pp. 199–210.

[7] J. Yeckle and B. Tang, "Detection of electricity theft in customer consumption using outlier detection algorithms," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2018, pp. 135–140.

[8] A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar, and S. Mishra, "Decision tree and svm-based data analytics for theft detection in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1005–1016, 2016.

[9] A. Ullah, N. Javaid, O. Samuel, M. Imran, and M. Shoaib, "CNN and GRU based deep neural network for electricity theft detection to secure smart grid," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 1598–1602.

[10] M. Ismail, M. Shahin, M. Shaaban, M. Shahin, E. Serpedin, and K. Qaraqe, "Efficient detection of electricity theft cyber attacks in AMI networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[11] M. Nabil, M. Ismail, M. Mahmoud, M. Shahin, K. Qaraqe, and E. Serpedin, "Deep learning-based detection of electricity theft cyber-attacks in smart grid AMI networks," in *Deep Learning Applications for Cyber Security*. Springer, 2019, pp. 73–102.

[12] S. Li, Y. Han, X. Yao, S. Yingchen, J. Wang, and Q. Zhao, "Electricity theft detection in power grids with deep learning and random forests," *Journal of Electrical and Computer Engineering*, vol. 2019, 2019.

[13] Y. Chen, G. Hua, D. Feng, H. Zang, Z. Wei, and G. Sun, "Electricity theft detection model for smart meter based on residual neural network," in *2020 12th IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. IEEE, 2020, pp. 1–5.

[14] Z. Chen, D. Meng, Y. Zhang, T. Xin, and D. Xiao, "Electricity theft detection using deep bidirectional recurrent neural network," in *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, 2020, pp. 401–406.

[15] M. Ismail, M. F. Shaaban, M. Naidu, and E. Serpedin, "Deep learning detection of electricity theft cyber-attacks in renewable distributed generation," *IEEE Transactions on Smart Grid*, 2020.

[16] F. Zhang, P. P. K. Chan, and T. Tang, "L-gem based robust learning against poisoning attack," in *2015 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, 2015, pp. 175–178.

[17] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 19–35.

[18] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of adversarial training examples in poisoning attacks through anomaly detection," *arXiv preprint arXiv:1802.03041*, 2018.

[19] "Irish Social Science Data Archive," Last accesed: Nov 2017. [Online]. Available: http://www.ucd.ie/issda/data/commissionforenergyregulationcer/

[20] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215 – 249, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016516841300515X

[21] C. Lu, S. Lin, X. Liu, and H. Shi, "Telecom fraud identification based on adasyn and random forest," in *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, 2020, pp. 447–452.

[22] T.-W. Sun and A.-Y. A. Wu, "Sparse autoencoder with attention mechanism for speech emotion recognition," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2019, pp. 146–149.

[23] Z. Zhao, Z. Bao, Z. Zhang, J. Deng, N. Cummins, H. Wang, J. Tao, and B. Schuller, "Automatic assessment of depression from speech via a hierarchical attention transfer network and attention autoencoders," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 2, pp. 423–434, 2020.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[25] D. Lavrova, D. Zegzhda, and A. Yarmak, "Using GRU neural network for cyber-attack detection in automated process control systems," in *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2019, pp. 1–3.

[26] M. Nabil, M. Ismail, M. Mahmoud, M. Shahin, K. Qaraqe, and E. Serpedin, "Deep recurrent electricity theft detection in AMI networks with random tuning of hyper-parameters," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 740–745.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.