Deep Auto-encoder-based Anomaly Detection of Electricity Theft Cyber-attacks in Smart Grids

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

***Abstract*—Designing an electricity theft cyber-attack detector for the advanced metering infrastructures (AMIs) is challenging due to the limited availability of electricity theft datasets (i.e., malicious datasets). Anomaly detectors, which are trained solely on honest customers' energy consumption profiles (i.e., benign datasets), could potentially overcome this challenge. Unfortunately, existing anomaly detectors in AMIs present shallow architectures and are incapable of capturing the temporal correlations as well as the sophisticated patterns present in the electricity consumption data, which impact their detection performance negatively. This paper proposes the adoption of deep (stacked) auto-encoders with a long-short-term-memory (LSTM)-based sequence to sequence (seq2seq) structure. The depth of the auto-encoders' structure helps capture the sophisticated patterns of the data and the seq2seq LSTM model enables exploitation of the time-series nature of data. We investigate the performance of simple auto-encoder (SAE), variational auto-encoder (VAE), and auto-encoder with attention (AEA), in which improved detection performance is observed when seq2seq structures are adopted compared to fully connected ones. Based on the simulation results, the AEA detector offers an enhancement of $4-21\%$ and $4-13\%$ in terms of detection rate and false alarm rate, respectively, compared to existing state-of-the-art detectors.**

***Index Terms*—electricity stealth, auto-encoders, seq2seq, deep machine learning, hyper-parameter optimization.**

I. Introduction

Electricity stealth represents a primary concern in power grids as it results in high financial losses. In the United States, such losses are approximated to be \$6 billion [3] yearly. Besides overloading the power grid, electricity stealths negatively impact the grid's performance in terms of stability and efficiency. To monitor the energy consumption and detect electricity thefts, utility companies have deployed advanced metering infrastructures (AMIs) where smart meters regularly record customers' electricity consumption in residential units. However, smart meters may be subject to different types of cyber stealths. The smart meters' authentication firmware allows malicious users to compromise the electricity consumption reports' integrity by lowering their electricity bills [4], [5].

A. Takiddin is with the ECEN Dept., Texas A&M University, College Station, TX, USA (email: abdulrahman.takiddin@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. Part of this work was presented in the 2020 28^{th} European Signal Processing Conference (EUSIPCO) [1] and the 2021 International Symposium on Signals, Circuits and Systems (ISSCS) [2].

A. Related Work and Limitations

Since smart meters provide lots of energy consumption data, machine learning models are proposed to identify electricity thefts [6], [7]. Machine learning-based detectors include both supervised classifiers and anomaly detectors.

Supervised classifiers employ both the benign and malicious energy consumption profiles of customers for training. Such classifiers include shallow machine learning-based classifiers such as Naïve Bayes [8] and multi-class support vector machines (SVMs) [3] that offer detection rates (DRs) of 80% and 94%, respectively. AdaBoost [9] and random forests [10] provide an accuracy of 80% and F1-score of 81%, respectively. Additionally, a two-step detector based on decision trees and SVM reports a 92% accuracy score [11], whereas an extreme gradient boosting classifier offers a precision score of 97% [12]. Deep machine learning-based classifiers including feed forward neural networks [13], recurrent neural networks (RNNs) [14], and vector embeddings [15] report 92% - 95% in DR. Hybrid detectors based on convolutional neural networks (CNNs) and RNN variations exhibit an accuracy score of 89% [16] [17]. Notice that the evaluation metrics mentioned above vary based on different authors' reports. The limitation of such supervised classifiers is that they rely on the availability of both benign and malicious samples for customers' energy consumption data. However, this condition is not a practical assumption in cases where the attack occurs for the first time (i.e., zero-day attacks). Hence, a supervised classifier is not a practical option in this case since the detection is exclusive for the attacks that the detector is trained on (seen attacks) only.

Anomaly detectors [18] utilize benign data to learn the honest customer consumption patterns during training. During testing, malicious usage is detected based on the deviation from the learned honest pattern. Unlike supervised classifiers, where both benign and malicious data are needed, anomaly detectors can detect zero-day attacks using only benign data. For example, using only benign data, shallow machine learning-based approaches such as single-class SVM [3], auto-regressive integrated moving average (ARIMA) [19], principal component analysis [20], and outlier detection [21] report a DR of 76% - 87%. Although such anomaly detectors are capable of detecting zero-day attacks, it is evident from the reported detection performance that they partially fail to capture the sophisticated consumed energy readings' patterns. This is because they present static (excluding ARIMA) and shallow structures. Therefore, there is a need to develop practical anomaly detectors that are trained only on benign load profiles and yet exhibit improved detection performance.

B. Contributions

Our motivation behind adopting auto-encoders as electricity theft detectors is to overcome the limitations of the aforementioned approaches and to improve the electricity theft detection performance. Auto-encoders present a feasible approach due to the following advantages. First, auto-encoders are applicable in various scenarios as they are unsupervised anomaly detectors that are only trained on benign customers' data. Hence, they are able to detect different malicious activities based on the deviation from the learned benign patterns without the need of accessing labeled malicious datasets. Such a feature not only overcomes the limited availability issue of malicious energy consumption data, but also helps in detecting unseen attacks since the detection is not exclusively limited to the attacks that the detectors are trained on, like the case with supervised learning. Such a generality plays an important role in practice since not all attacks can be modeled beforehand. Thus, having a detector that works even in the presence of unseen attacks brings a second major benefit. Third, The deep structure of the proposed auto-encoders helps extract more relevant and representative features from the energy consumption data. Fourth, the proposed auto-encoders are equipped with a sequence-to-sequence (seq2seq) dynamic structure to better comprehend and model the time-series nature of the data. Fifth, it turns out that such advantages contribute towards improving the detection performance. Next, we summarize our contributions.

- We examine both fully connected feed forward and seq2seq structures based on RNN, long-short-term-memory (LSTM) architectures for the auto-encoders. Auto-encoders that are based on fully connected feed forward structures serve as a benchmark due to their simple architecture and low computational complexity. On the other hand, LSTM-based RNN auto-encoders exploit deeper the nature of the time-series electricity consumption data and take into consideration the temporal correlations present in the data.
- We investigate the performance of simple auto-encoder (SAE), variational auto-encoder (VAE), and auto-encoder with attention (AEA) structures. The SAE serves as a benchmark for the achieved performance and offers the lowest complexity. Adopting VAE rather than SAE helps in capturing the variabilities within the latent space through a variance parameter, and hence, improves detection performance. Implementing AEA rather than VAE further improves the detection performance as it better handles long sequences and speeds up the learning process using the appropriate attention layer [22]. We validate this by assessing the models' performance using two datasets separately.
- The applicability of the proposed auto-encoders is verified using six different unseen electricity cyber-attacks. Then, we compare the proposed auto-encoders' performance to a wide variety of shallow/deep, supervised/unsupervised, and static/dynamic detectors including shallow supervised classifiers (Naïve Bayes and multi-class SVM), deep supervised classifiers (feed forward and LSTM), and shallow anomaly detectors (single-class SVM and ARIMA). The optimization of hyper-parameters is carried out to maximize the detection performance of the proposed auto-encoders as well as the benchmark supervised classifiers and anomaly detectors. The deep AEA anomaly detector yields the best performance with 94% in DR and 5% in false alarm (FA) rate, which enhances the DR by $4-8\%$ and reduces the FA rate by $4-7\%$. Further, this performance is achieved with low complexity in the offline training stage while offering almost an instant decision in the online implementation.

The remainder of the paper is organized as follows. Section II describes the energy consumption profiles (benign and malicious) and the data preparation. Section III discusses the architecture of the proposed auto-encoders and the conducted optimization. Section IV presents the results of the conducted experiments. Finally, conclusions are drawn in Section V.

II. Data Preparation

In this section, we present the energy consumption data utilized to train and test the investigated detectors. The developed anomaly detectors are trained on benign datasets only and tested on the benign and malicious datasets, whereas the developed supervised classifiers are trained and tested on benign and malicious datasets. To verify the applicability and feasibility of auto-encoders, we use two different datasets to capture a wide variety of realistic customers' electricity usage patterns of various appliances during weekdays, weekends, and during all seasons of the year, including vacations. The first dataset is acquired from the State Grid Corporation of China (SGCC) [23]; it has labeled benign and malicious daily energy consumption readings of real customers. The second dataset has real benign hourly energy consumption samples from the Irish Smart Energy Trial (ISET) [24] and artificially generated malicious samples generated using six attack functions.

A. State Grid Corporation of China Dataset

The SGCC dataset contains around 40,000 customers' daily electricity consumption data (both benign and malicious samples) distributed over three years. Sample daily energy consumption data for three benign and three malicious customers is illustrated in Figure 1 throughout ten days. Let $E_c(d,t)$ represent the value of the electricity consumption for customer c at a given day d. All these values designate entries of matrix $\boldsymbol{E}_c$. When the customer is honest, the recorded energy consumption by the customer's smart meter $R_c(d)$ satisfies the condition $R_c(d) = E_c(d)$. Thus, matrices $\boldsymbol{E}_c$ and $\boldsymbol{R}_c$ are equal. On the other hand, in the SGCC dataset, malicious customers trigger a simple attack where they report an energy consumption value of zero at specified hours to replace the real consumption value. Hence, the total energy consumption throughout the day is reduced as illustrated in Figure 1.

B. Irish Smart Energy Trial Dataset

1) Benign Dataset: The ISET dataset contains readings from smart meters that are installed at around 3,000 residential units that report electricity consumption every 30 minutes for

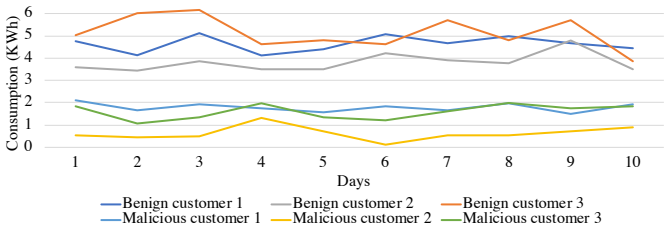
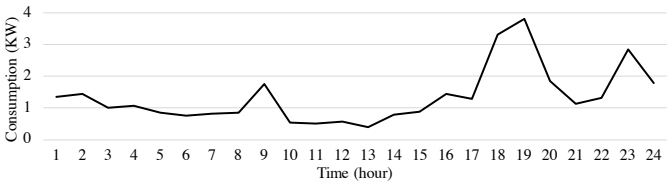
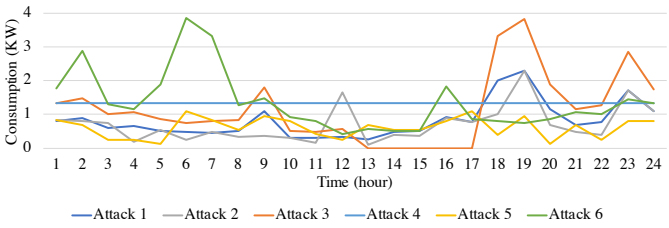


Fig. 1. SGCC energy consumption sample.



(a) Sample energy consumption of a benign customer.



(b) Sample energy consumption of a malicious customer.

Fig. 2. ISET energy consumption sample.

a period of a year and a half. This yields around 25,000 reports per customer, which are sufficient for training and testing the developed electricity stealth anomaly detectors. A sample energy consumption throughout a day of a benign customer from the ISET dataset is visualized in Figure 2a.

2) Malicious Dataset: In the ISET dataset, malicious customers trigger multiple attack types to exploit the integrity of the consumption reading so that they can have lower electricity bills when $R_c(d,t) \neq E_c(d,t)$. To construct the malicious dataset, we adopt the approach of false data injection of [3]. In this study, we assume that all customers are equipped with smart meters where malicious customers tend to locally perform malicious activities on their own smart meters that are installed at their own premises to manipulate their own readings to reduce their electricity bills. No action is performed by malicious customers to affect the smart meter readings of other customers within the grid. We consider various types of cyber-attacks that are split into three main categories.

Within the *partial reduction attacks*, cyber-attack function $f_1(E_c(d,t))$ reduces the real value of electricity consumption by a fixed random fraction, $\alpha = rand(0.1, 0.8)$ capturing low and high level attacks, for all samples, and hence, the value of the reported electricity consumption $R_c(d,t)$ is

$$f_1(E_c(d,t)) = \alpha E_c(d,t). \tag{1}$$

Cyber-attack function $f_2(E_c(d,t))$ multiplies each reading by a dynamic random fraction, $\beta = rand(0.1, 0.8)$ where

$$f_2(E_c(d,t)) = \beta(d,t) E_c(d,t). \tag{2}$$

In *selective by-pass attacks*, zero energy consumption is reported during an interval of time $[t_i(d), t_f(d)]$, and outside that interval, the true energy consumption is reported. Hence,

$$f_3(E_c(d,t)) = \begin{cases} 0 & \forall t \in [t_i(d), t_f(d)] \\ E_c(d,t) & \forall t \notin [t_i(d), t_f(d)], \end{cases} \tag{3}$$

where the interval has an initial time $t_i(d) = rand(0, 23-4)$, length $t_l(d) = rand(4, 24)$, and final time $t_f(d) = t_i(d) - t_l(d)$. This range captures low-level attacks with minimum off-time of 4 hours and high level attacks with maximum off-time of 24 hours.

Price-based load control attacks occur if there are different prices for electricity during the day. An attack function reports a consumption value that is constant across the day such that

$$f_4(E_c(d,t)) = \mathbb{E}[E_c(d)], \tag{4}$$

in which $\mathbb{E}[\cdot]$ depicts the expected average consumption value. Since recording consumption values that are constant during the day is easily detectable, $\beta = rand(0.1, 0.8)$ is applied as

$$f_5(E_c(d,t)) = \beta(d,t)\mathbb{E}[E_c(d)]. \tag{5}$$

The last attack function records the values of energy consumption that are high during the time intervals where the electricity tariff is low and vice versa. Hence,

$$f_6(E_c(d,t)) = E_c(d, T-t+1). \tag{6}$$

For all customers, we apply all the aforementioned cyber-attack functions to the matrix of the customer's consumption profile $\boldsymbol{E}_c$. Each customer ends up with six malicious matrices. Figure 2b illustrates sample malicious energy consumption patterns generated using the six cyber-attack functions from the benign energy consumption pattern in Figure 2a.

C. Datasets Preparation

The detectors adopted herein are generalized, which means that all customers' data are merged together to train and test the detectors [14]. Then, the smart meter data is split (over customers) into train and test sets. Customers in the train set are not included in the test set so that all benign and malicious customers in the test set present completely unseen data by the trained network. Below, we discuss splitting the data into train and test sets for the investigated detectors for both datasets.

For the anomaly detectors, the concatenation of the energy consumption profile for all customers represents the benign class dataset, where a single row gives an energy consumption profile sample along the day. The concatenation of the malicious matrices for all customers represents the malicious class dataset. Then, we normalize both classes so that all feature values are brought to a common scale that presents zero mean and unit variance. Let $\boldsymbol{B}$ and $\boldsymbol{M}$ denote the normalized benign and malicious data, respectively. First, with a ratio of 2:1, we split $\boldsymbol{B}$ over the rows (customers) into two disjoint subsets, $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$. $\boldsymbol{X}_{\text{TR}}$ denotes the training data, which is the first subset $\boldsymbol{B}_1$. Then, to build the test data, $\boldsymbol{M}$ is concatenated with the second subset $\boldsymbol{B}_2$. For this concatenation, each sample is associated with either a zero-label for a benign sample or a one-label for a malicious sample. Accumulating

more malicious data than benign data might lead to deceptive results in terms of performance. To avoid this, the adaptive synthetic (ADASYN) sampling method [25] is employed so that the benign and malicious samples are balanced through oversampling the minor class (malicious readings in the SGCC dataset and benign readings in the ISET dataset) in the testing set. Hence, we obtain the test data $\boldsymbol{X}_{\text{TST}}$ with the label $\boldsymbol{Y}_{\text{TST}}$.

For the supervised learning detectors, we concatenate the benign and malicious classes for all customers and apply the ADASYN to balance them. Then, we split the concatenated dataset into disjoint train and test sets with a $2:1$ ratio. We perform the same normalized feature scaling for the train and test sets to form the final disjoint sets $\boldsymbol{X}_{\text{TR}}$ with the label $\boldsymbol{Y}_{\text{TR}}$ and $\boldsymbol{X}_{\text{TST}}$ with the label $\boldsymbol{Y}_{\text{TST}}$, respectively.

III. Electricity Stealth Detector Design

Herein, we present the design of the data-driven anomaly detectors that recognize electricity theft cyber-attacks on smart meters installed at residential units. We investigate various auto-encoder architectures along with hyper-parameter optimization to enhance the detection performance.

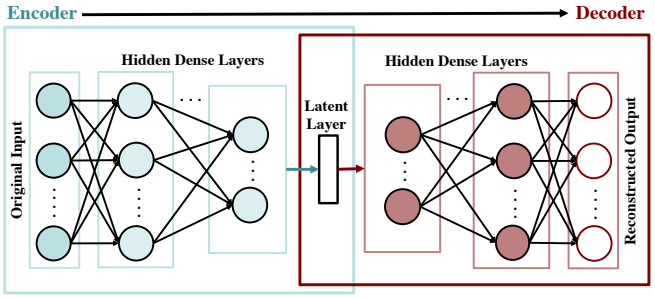
A. Simple Auto-encoder Architecture

The architecture of the SAE represents the simplest implementation of the anomaly detector. In Figure 3, we illustrate the structure of the SAE. The anomaly detector first learns the behavioral patterns of benign energy consumption data. Then, it detects abnormality by assessing the deviation from such patterns. This deviation is then used as an indication of the presence of an electricity theft cyber-attack. An effective approach to define anomalies is based on the reconstruction errors, which can be achieved using stacked SAEs. Stacked SAEs are used to reduce the dimension of the data during the encoding stage and then to reconstruct the data in the decoding stage. The difference between the original data and the reconstructed data is denoted as the reconstruction error [26]. The SAE is trained using the benign data to determine the encoder and decoder parameters that minimize the reconstruction error. Formally, we have $\boldsymbol{H} = f_\Theta(\boldsymbol{x})$ for the encoder and $\boldsymbol{R} = g_\Theta(\boldsymbol{x})$ for the decoder, where $\boldsymbol{x}$ represents the rows of $\boldsymbol{X}_{\text{TR}}$, and Θ denotes the SAE parameters determined by

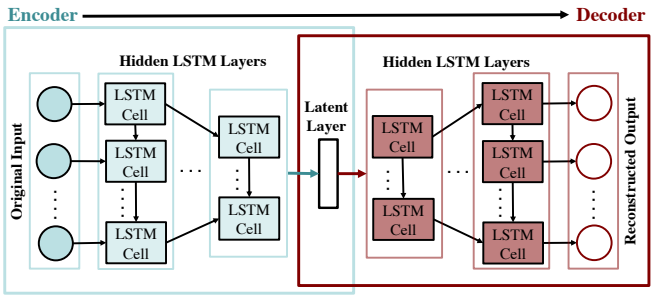
$$\min_{\Theta} C(\boldsymbol{x}, g_\Theta(f_\Theta(\boldsymbol{x}))), \qquad \boldsymbol{x} \in \boldsymbol{X}_{\text{TR}}, \tag{7}$$

where $C(\boldsymbol{x}, g_\Theta(f_\Theta(\boldsymbol{x})))$ denotes a cost function (i.e., the mean squared error (MSE)) that penalizes $g_\Theta(f_\Theta(\boldsymbol{x}))$ for being dissimilar from $\boldsymbol{x}$. Following the cost function (7), the reconstruction error is expected to be small for benign data and large for anomalies. We use the reconstruction error to indicate how familiar the model is with a given test instance; whenever it exceeds a certain threshold, an anomaly indicating an electricity theft is detected. We adopt two SAE structures.

1) Fully Connected SAE: In the Fully Connected SAE (FC-SAE), the encoder part comprises an input layer, a number of dense hidden layers in addition to a latent layer. The decoder part comprises a number of dense hidden layers as well as an output layer [26]. The input layer has 48 neurons that are fed with the energy consumption profile $\boldsymbol{x} \in \boldsymbol{X}_{\text{TR}}$. The



(a) Fully connected SAE architecture.



(b) Sequence-to-sequence SAE architecture.

Fig. 3. Illustration of the SAE architecture.

hidden dense layers in the encoder part compress the input into the latent layer l', which is then filtered through the decoder's hidden dense layers to reconstruct the output. The encoder and decoder's hidden layers consist of L layers, each layer being equipped with N_l neurons. Let $w^l_{nn'}$ represent the connection weight from neuron n', which is in layer $l-1$ to neuron n that is in layer l. The weight matrix is denoted by $\boldsymbol{W}^l$. Define b^l_n as the bias of neuron n in layer l. The bias vector of layer l is denoted by $\boldsymbol{b}^l$. The weighted sum of inputs to neuron n is $z^l_n = \sum_{n'} w^l_{nn'} a^{l-1}_{n'} + b^l_n$, where $\boldsymbol{a}^l = \varphi(\boldsymbol{z}^l)$ and $\varphi(\cdot)$ stands for the activation function. The training of the SAE aims to find the model parameters $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$, denoted by Θ, that satisfy (7). The minimization of (7) is achieved using the iterative gradient descent algorithm depicted in Algorithm 1, which assumes a stochastic gradient descent (SGD) implementation with learning rate η. Notation $\bigtriangledown_y$ denotes the partial derivative given y and K represents the total number of training samples.

2) Sequence-to-Sequence SAE: The customer's energy consumption profile presents time-series data, which exhibits obvious temporal correlation. While the FC-SAE offers low computational complexity, such a structure does not have the capability of exploiting the temporal correlations within the electricity consumption profiles. Thus, a deep RNN-based auto-encoder is expected to offer a better detection performance compared to FC-SAEs. We utilize an RNN variation based on LSTM [27] in order to handle the vanishing gradient problem during the learning process of temporal correlations throughout the long intervals. Although gated recurrent units (GRU) might offer faster training time using less number of gates, LSTM tends to provide better accuracy than GRU at the expense of training time [27], [28]. Training time should not be an obstacle to implementing LSTM since the training is mainly

Algorithm 1: FC-SAE Training

Input Data: $\boldsymbol{X}_{\text{TR}}$
Initialization: Weights $\boldsymbol{W}^l$ and biases $\boldsymbol{b}^l$ for all l
while *not converged* **do**
  for *each training sample* $\boldsymbol{x}$ **do**
    Encoder and Decoder:
    Feed forward: Compute:
    $\boldsymbol{z}^l(\boldsymbol{x}) = \boldsymbol{W}^l \boldsymbol{a}^{l-1}(\boldsymbol{x}) + \boldsymbol{b}^l$ and
    $\boldsymbol{a}^l(\boldsymbol{x}) = \varphi(\boldsymbol{z}^l(\boldsymbol{x}))$ for all layers
    $l = 1, \ldots, l', \ldots, L$
    Back propagation: Compute:
    $\nabla_{\boldsymbol{W}^l} C$ and $\nabla_{\boldsymbol{b}^l} C$
  end
  Weight and bias update:
  $\boldsymbol{W}^l = \boldsymbol{W}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{W}^l} C$, $\boldsymbol{b}^l = \boldsymbol{b}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{b}^l} C$
end
Output: Optimal parameters $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ for all layers

done offline and not in real time; for example, during the billing period. Hence, we adopt LSTM. A deep LSTM-SAE model consists of two LSTM-RNNs, where the first LSTM-RNN is a deep LSTM encoder and the second one is a deep LSTM decoder [29], [30]. The input to the LSTM encoder is a time-series of the energy consumption profile ($\boldsymbol{x} \in \boldsymbol{X}_{\text{TR}}$), and hence, consists of 48 neurons. This time-series vector is then encrypted into a hidden state by the LSTM encoder, an operation that can be interpreted as finding a more compact representation for the time-series data. Hence, the encoder's input layer is followed by a number of hidden LSTM layers L each with N_l LSTM cells. The LSTM encoder output is then used as an input to the LSTM decoder, which reconstructs the original time-series data. The LSTM-SAE minimizes the reconstruction mean-square-error.

An LSTM cell (memory unit) has a state c_t at time instant t and yields as an output a hidden state h_t. Accessing such a cell is managed by a set of gates, namely, input, forget, and output gates, $i_{\text{E},t}$, $f_{\text{E},t}$, and $o_{\text{E},t}$, respectively, for the encoder. For the decoder, the set of gates are input, forget, and output gates, $i_{\text{D},t}$, $f_{\text{D},t}$, and $o_{\text{D},t}$, respectively. The LSTM cell receives two external inputs, namely, the energy consumption value at time instant t, $\boldsymbol{x}_t$ that is the value of $\boldsymbol{x}$ at time instant t, and the LSTM cells' preceding hidden states that are in the same layer, which are $h_{\text{E},t-1}$ and $h_{\text{D},t-1}$ for the encoder and decoder, respectively. In addition, an internal input is included, that is the cell state $c_{\text{E},t-1}$ for the encoder and $c_{\text{D},t-1}$ for the decoder. All these inputs are added along with a bias, and activation functions are applied so that the input, forget, and output gates are activated. The states h' and c' are present at the rearmost time step of the encoder, which are then used as the decoder's initial hidden and cell states. In Algorithm 2, lines 9 - 13 and 18 - 22 present the computations of $i_{\text{E/D},t}$, $f_{\text{E/D},t}$, and $o_{\text{E/D},t}$ where $\boldsymbol{W}^l_{(\cdot)}$, $\boldsymbol{U}^l_{(\cdot)}$, $\boldsymbol{V}^l_{(\cdot)}$, and $\boldsymbol{b}^l_{(\cdot)}$ denote the learnable weight matrices and bias vectors. The optimal learnable parameters are obtained by solving (7), a step which is achieved via the iterative gradient descent approach shown in Algorithm 2.

Algorithm 2: LSTM-SAE Training

1 **Input Data:** $\boldsymbol{X}_{\text{TR}}$
2 **Initialization:** Weights $\boldsymbol{U}^l_{(\cdot)}$, $\boldsymbol{W}^l_{(\cdot)}$, $\boldsymbol{V}^l_{(\cdot)}$, and bias $\boldsymbol{b}^l_{(\cdot)}$ $\forall l$
3 **while** *not converged* **do**
4 **for** *each training sample* x **do**
5 **Feed forward:**
6 **Encoder:**
7 **for** *each hidden layer* $l = 1, \ldots, L/2$ **do**
8 **for** *each time step* t **do**
9 $\boldsymbol{i}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_i \boldsymbol{x}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_i)$,
10 $\boldsymbol{f}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_f \boldsymbol{x}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_f)$,
11 $\boldsymbol{c}^l_{\text{E},t} = f^l_{\text{E},t} c^l_{\text{E},t-1} + i^l_{\text{E},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t + \boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{b}^l_c)$,
12 $\boldsymbol{o}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_o \boldsymbol{x}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{E},t} + \boldsymbol{b}^l_o)$,
13 $\boldsymbol{h}^l_{\text{E},t} = \boldsymbol{o}^l_{\text{E},t} \tanh(\boldsymbol{c}^l_{\text{E},t})$,
14 **end**
15 $\boldsymbol{h}'^l = \boldsymbol{h}^l_{\text{E},t}$,
16 $\boldsymbol{c}'^l = \boldsymbol{c}^l_{\text{E},t}$.
17 **end**
18 **Decoder:**
19 At initial time step, the decoder hidden and cell states are equal to h'^l and c'^l and the output of the encoder is fed as the input to the decoder $\breve{\boldsymbol{x}}$
20 **for** *each hidden layer* $l = L/2 + 1, \ldots, L$ **do**
21 **for** *each time step* t **do**
22 $\boldsymbol{i}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_i \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_i)$,
23 $\boldsymbol{f}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_f \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_f)$,
24 $\boldsymbol{c}^l_{\text{D},t} = f^l_{\text{D},t} c^l_{\text{D},t-1} + i^l_{\text{D},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t + \boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{b}^l_c)$,
25 $\boldsymbol{o}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_o \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{D},t} + \boldsymbol{b}^l_o)$,
26 $\boldsymbol{h}^l_{\text{D},t} = \boldsymbol{o}^l_{\text{D},t} \tanh(\boldsymbol{c}^l_{\text{D},t})$,
27 **end**
28 **end**
29 **Back propagation:** Compute:
30 $\nabla_{\boldsymbol{W}^l_{(\cdot)}} C$, $\nabla_{\boldsymbol{U}^l_{(\cdot)}} C$, $\nabla_{\boldsymbol{V}^l_{(\cdot)}} C$, and $\nabla_{\boldsymbol{b}^l_{(\cdot)}} C$
31 **end**
32 **Weight and bias update:**
  $\boldsymbol{W}^l_{(\cdot)} = \boldsymbol{W}^l_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{W}^l_{(\cdot)}} C$
  $\boldsymbol{U}^l_{(\cdot)} = \boldsymbol{U}^l_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{U}^l_{(\cdot)}} C$
  $\boldsymbol{V}^l_{(\cdot)} = \boldsymbol{V}^l_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{V}^l_{(\cdot)}} C$
  $\boldsymbol{b}^l_{(\cdot)} = \boldsymbol{b}^l_{(\cdot)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{b}^l_{(\cdot)}} C$
33 **end**
34 **Output:** Optimal $\boldsymbol{U}^l_{(\cdot)}$, $\boldsymbol{W}^l_{(\cdot)}$, $\boldsymbol{V}^l_{(\cdot)}$, and $\boldsymbol{b}^l_{(\cdot)}$ $\forall l$.

B. Variational Auto-encoder Architecture

The VAE structure is shown in Figure 4. Since SAEs represent a deterministic mapping, they can be modeled as a mapping to the mean value of a Dirac delta distribution. Hence, they cannot capture the variance. If both normal and anomalous (electricity theft) data share the same mean but present different variance behavior, an SAE-based model might not be able to recognize the anomalies. On the other hand, a VAE represents a better alternative for anomaly detection. VAEs are directed probabilistic graphical models (PGMs) in which the posteriors are determined using neural networks [1]. VAEs assume that any point x in the data is generated based on the continuous random variable k, which is unobserved. First, through the prior distribution $p(k)$, we generate a particular value k'. Next, based on the conditional distribution $p(x|k)$, we generate the data instance x'. Because the values of k are unknown and inferring the exact distribution $p(k|x)$ is intractable, both, $p(k|x)$ for the encoder and $p(x|k)$ for the decoder neural networks are approximated by the VAE. $q(k|x)$ depicts the approximation of the true posterior $p(k|x)$. For data point x, the log-likelihood is

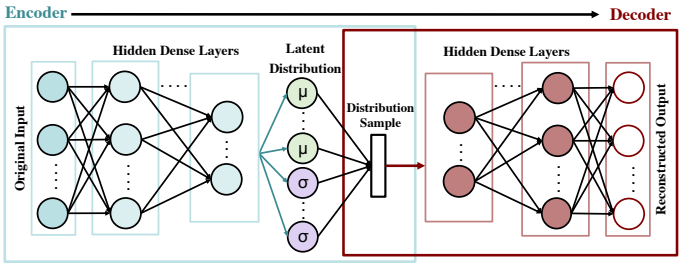
$$\log p(x) = D_{\mathrm{KL}}(q(k|x)||p(k|x)) + \mathcal{L}(\Theta; x), \tag{8}$$

where the Kullback–Leibler (KL) divergence is denoted by D_{KL}, the model parameters are expressed by Θ, and the log-likelihood's variational lower bound is given by

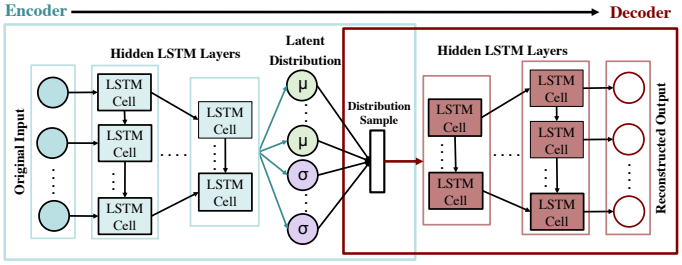
$$\mathcal{L}(\Theta; x) = -D_{\mathrm{KL}}(q(k|x)||p(x)) + \mathbb{E}_{q(k|x)}[\log p_\Theta(x|k)]. \tag{9}$$

Since $\mathcal{L}(\Theta; x) \leq \log p(k)$, Θ are learned through optimizing $\mathcal{L}(\Theta; x)$. To carry out this step, a re-parameterization trick [1] that models the random variable $k \sim q(k|x)$ as a deterministic variable is applied. $k = \mu + \sigma\mathcal{N}$ in which $\mathcal{N}$ refers to a normal distribution that has zero mean and unit variance if the latent variables are represented as univariate Gaussian. Hence, the mean (μ) and variance (σ^2) of the Gaussian distribution are determined by the encoder forward-pass's activations [31]. Similarly, the distribution parameters of $p(x|k)$ are determined by the decoder forward-pass's activations. Hence, a VAE models the data distribution parameters and not the data itself.

In Section III.A, for the anomaly score, the reconstruction error was utilized. However, we utilize the reconstruction probability to reflect the anomaly score. Specifically, let the probabilistic encoder and decoder assume a multi-variant Gaussian distribution in the latent variable space and the original input space, respectively. The probabilistic encoder that is trained using the benign energy consumption profile data will determine the distribution parameters of the input data. Then, a number of samples will be generated using the distribution parameters and the probabilistic decoder will recover the distribution parameters. Using the recovered parameters, the probability that the original data is generated from that distribution is calculated. The average of such probabilities over all samples is referred to as the reconstruction probability. Since the VAE model parameters are based on benign data, such a reconstruction probability will be low for anomalous data. Hence, a threshold is defined, and if the reconstruction probability is below that threshold, an electricity theft is detected. Next, two structures are discussed for the VAEs.



(a) Fully connected VAE architecture.



(b) Sequence-to-sequence VAE architecture.

Fig. 4. Illustration of the VAE architecture.

1) Fully Connected VAE: The Fully Connected VAE (FC-VAE) and FC-SAE present a similar structure in terms of the input layer and the hidden dense layers. However, when it comes to generative modeling, VAE presents an advantage over the SAE because its latent space is continuous in a way that makes random sampling simpler for interpolation. This is carried out by producing the mean $\boldsymbol{\mu_x}$ and variance vectors $\boldsymbol{\sigma_x^2}$ to form the parameters of the random variable vector, which is then passed to the decoder. For the generation of the consumption data samples, the decoder uses $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$. To detect stealth, the reconstruction probability is calculated and compared to a threshold. The loss function of the VAE constrains the reconstruction and regularisation terms as

$$\hat{C} = ||\boldsymbol{x} - \tilde{\boldsymbol{x}}_\Theta||^2 + D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x^2})||\mathcal{N}(0, 1)), \tag{10}$$

where $\tilde{\boldsymbol{x}}_\Theta$ is the reconstructed output as function of the model's parameter Θ, $\mathcal{N}(\mu_x, \sigma_x^2)$ is a general distribution, and $\mathcal{N}(0, 1)$ stands for a standard normal distribution.

FC-VAE uses a similar training algorithm structure as the FC-SAE shown in Algorithm 1 to determine the model parameters Θ. However, the VAE generates the parameters of the probability distribution ($\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$). With these parameters and assuming a Gaussian distribution, the decoder generates the output. Using (10), the loss is computed. To update the encoder and decoder's parameters, back propagation is used. Algorithm 3 is similar to Algorithm 1, with the addition of generating $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$ in lines 11 and 12 as well as computing the optimal parameters given the iterative gradient descent.

2) Sequence-to-Sequence VAE: LSTM-VAEs can be introduced by considering LSTM structures for the probabilistic encoders and decoders [32]. Originally, LSTM-VAEs were used to generate data for language modeling applications [33]. However, herein, we utilize them for anomaly detection. The structure of LSTM-VAE has the same components as LSTM-SAE with the addition of a latent distribution as in FC-VAE.

Algorithm 3: FC-VAE Training

Input Data: $\boldsymbol{X}_{\text{TR}}$
Initialization: Weights $\boldsymbol{W}^l$ and biases $\boldsymbol{b}^l$ for all layers l and weights $\boldsymbol{V}_\mu$ and $\boldsymbol{V}_\sigma$ and biases $\boldsymbol{b}_\mu$ and $\boldsymbol{b}_\sigma$ for the latent layer
while *not converged* **do**
  for *each training sample* $\boldsymbol{x}$ **do**
    Feed forward:
    Encoder:
    for *all layers* $l = 1, \ldots, L/2$ **do**
      $\boldsymbol{z}^l(\boldsymbol{x}) = \boldsymbol{W}^l \boldsymbol{a}^{l-1}(\boldsymbol{x}) + \boldsymbol{b}^l$ and $\boldsymbol{a}^l(\boldsymbol{x}) = \varphi(\boldsymbol{z}^l(\boldsymbol{x}))$
    end
    Generate $\boldsymbol{\mu_x}$ **and** $\boldsymbol{\sigma_x^2}$**:**
    $\boldsymbol{\mu_x} = \varphi(\boldsymbol{V}_\mu \boldsymbol{a}^{l-1}(\boldsymbol{x})) + \boldsymbol{b}_\mu^l$
    $\boldsymbol{\sigma_x^2} = \varphi(\boldsymbol{V}_\sigma \boldsymbol{a}^{l-1}(\boldsymbol{x})) + \boldsymbol{b}_\sigma^l$
    Sample data $\breve{\boldsymbol{x}}$ from $\mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x^2})$
    Decoder:
    for *all layers* $l = L/2+1, \ldots, L$ **do**
      $\boldsymbol{z}^l(\breve{\boldsymbol{x}}) = \boldsymbol{W}^l a^{l-1}(\breve{\boldsymbol{x}}) + \boldsymbol{b}^l$ and $\boldsymbol{a}^l(\breve{\boldsymbol{x}}) = \varphi(\boldsymbol{z}^l(\breve{\boldsymbol{x}}))$
    end
    Back propagation: Compute:
    $\nabla_{\boldsymbol{W}^l} \hat{C}$, $\nabla_{\boldsymbol{V}_{(.)}} \hat{C}$ and $\nabla_{\boldsymbol{b}_{(.)}^l} \hat{C}$
  end
  Weight and bias update:
  $\boldsymbol{W}^l = \boldsymbol{W}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{W}^l} \hat{C}$,
  $\boldsymbol{V}_{(.)} = \boldsymbol{V}_{(.)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{V}_{(.)}} \hat{C}$,
  $\boldsymbol{b}_{(.)}^l = \boldsymbol{b}_{(.)}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{b}_{(.)}^l} C$
end
Output: Optimal parameters $\mathbf{W}^l$, $\boldsymbol{V}_\mu$, $\boldsymbol{V}_\sigma$, $\mathbf{b}^l$, $\boldsymbol{b}_\mu$, and $\boldsymbol{b}_\sigma$ for all layers

The training algorithms for LSTM-SAE and LSTM-VAE are also similar in terms of the logic, but LSTM-VAE, similar to Algorithm 3, generates $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$ that are utilized to generate the samples to be used within the decoder as shown in Algorithm 4 in lines 19 - 32.

C. Auto-encoder with Attention Architecture

For the aforementioned models, there is a limitation with regards to the size of the input and output sequences, which restricts the detection performance. The AEA structure is shown in Figure 5. In the SAE and VAE models, the encoder's responsibility is to examine the time steps of the input as well as encode the whole sequence into a context vector with fixed length. The decoder's responsibility is to examine the time steps of the output while looking through the context vector. Thus, the capability of such models is limited to the fixed-sized internal representation developed by the encoder. The AEA model extends the architecture and addresses this limitation [34]. Attention helps overcome the performance loss of the traditional seq2seq networks by assigning different weights and scores for each time step. This is carried out to produce

Algorithm 4: LSTM-VAE Training

1 **Input Data:** $\boldsymbol{X}_{\text{TR}}$
2 **Initialization:** Weights $\boldsymbol{U}_{(\cdot)}^l$, $\boldsymbol{W}_{(\cdot)}^l$, $\boldsymbol{V}_{(\cdot)}^l$, and bias $\boldsymbol{b}_{(\cdot)}^l$ $\forall l$
3 **while** *not converged* **do**
4 **for** *each training sample* $\boldsymbol{x}$ **do**
5 **Feed forward:**
6 **Encoder:**
7 **for** *each hidden layer* $l = 1, \ldots, L/2$ **do**
8 **for** *each time step* t **do**
9 $\boldsymbol{i}_{\text{E},t}^l = \varphi(\boldsymbol{W}_i^l \boldsymbol{x}_t^l + \boldsymbol{U}_i^l \boldsymbol{h}_{\text{E},t-1}^l + \boldsymbol{V}_i^l \boldsymbol{c}_{\text{E},t-1}^l + \boldsymbol{b}_i^l)$,
10 $\boldsymbol{f}_{\text{E},t}^l = \varphi(\boldsymbol{W}_f^l \boldsymbol{x}_t^l + \boldsymbol{U}_f^l \boldsymbol{h}_{\text{E},t-1}^l + \boldsymbol{V}_f^l \boldsymbol{c}_{\text{E},t-1}^l + \boldsymbol{b}_f^l)$,
11 $\boldsymbol{c}_{\text{E},t}^l = f_{\text{E},t}^l c_{\text{E},t-1}^l + i_{\text{E},t}^l \tanh(\boldsymbol{W}_c^l \boldsymbol{x}_t^l + \boldsymbol{U}_c^l \boldsymbol{h}_{\text{E},t-1}^l + \boldsymbol{b}_c^l)$,
12 $\boldsymbol{o}_{\text{E},t}^l = \varphi(\boldsymbol{W}_o^l \boldsymbol{x}_t^l + \boldsymbol{U}_o^l \boldsymbol{h}_{\text{E},t-1}^l + \boldsymbol{V}_o^l \boldsymbol{c}_{\text{E},t}^l + \boldsymbol{b}_o^l)$,
13 $\boldsymbol{h}_{\text{E},t}^l = \boldsymbol{o}_{\text{E},t}^l \tanh(\boldsymbol{c}_{\text{E},t}^l)$,
14 **end**
15 $\boldsymbol{h}'^l = \boldsymbol{h}_{\text{E},t}^l$,
16 $\boldsymbol{c}'^l = \boldsymbol{c}_{\text{E},t}^l$.
17 **end**
18 **Generate** $\boldsymbol{\mu_x}$ **and** $\boldsymbol{\sigma_x^2}$**:**
19 $\boldsymbol{\mu_x} = \varphi(\boldsymbol{V}_\mu \boldsymbol{a}^{l-1}(x)) + \boldsymbol{b}_\mu^l$
20 $\boldsymbol{\sigma_x^2} = \varphi(\boldsymbol{V}_\sigma \boldsymbol{a}^{l-1}(x)) + \boldsymbol{b}_\sigma^l$
21 Sample data $\breve{\boldsymbol{x}}$ from $\mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x^2})$
22 **Decoder:**
23 At initial time step, the decoder hidden and cell states are equal to $\boldsymbol{h}'$ and $\boldsymbol{c}'$
24 **for** *each hidden layer* $l = L/2+1, \ldots, L$ **do**
25 **for** *each time step* t **do**
26 $\boldsymbol{i}_{\text{D},t}^l = \varphi(\boldsymbol{W}_i^l \breve{\boldsymbol{x}}_t^l + \boldsymbol{U}_i^l \boldsymbol{h}_{\text{D},t-1}^l + \boldsymbol{V}_i^l \boldsymbol{c}_{\text{D},t-1}^l + \boldsymbol{b}_i^l)$,
27 $\boldsymbol{f}_{\text{D},t}^l = \varphi(\boldsymbol{W}_f^l \breve{\boldsymbol{x}}_t^l + \boldsymbol{U}_f^l \boldsymbol{h}_{\text{D},t-1}^l + \boldsymbol{V}_f^l \boldsymbol{c}_{\text{D},t-1}^l + \boldsymbol{b}_f^l)$,
28 $\boldsymbol{c}_{\text{D},t}^l = f_{\text{D},t}^l c_{\text{D},t-1}^l + i_{\text{D},t}^l \tanh(\boldsymbol{W}_c^l \boldsymbol{x}_t^l + \boldsymbol{U}_c^l \boldsymbol{h}_{\text{D},t-1}^l + \boldsymbol{b}_c^l)$,
29 $\boldsymbol{o}_{\text{D},t}^l = \varphi(\boldsymbol{W}_o^l \breve{\boldsymbol{x}}_t^l + \boldsymbol{U}_o^l \boldsymbol{h}_{\text{D},t-1}^l + \boldsymbol{V}_o^l \boldsymbol{c}_{\text{D},t}^l + \boldsymbol{b}_o^l)$,
30 $\boldsymbol{h}_{\text{D},t}^l = \boldsymbol{o}_{\text{D},t}^l \tanh(\boldsymbol{c}_{\text{D},t}^l)$,
31 **end**
32 **end**
33 **Back propagation:** Compute $\nabla_{\boldsymbol{W}_{(.)}^l} C$, $\nabla_{\boldsymbol{U}_{(.)}^l} C$, $\nabla_{\boldsymbol{V}_{(.)}^l} C$, and $\nabla_{\boldsymbol{b}_{(.)}^l} C$
34 **end**
35 **Weight and bias update:**
  $\boldsymbol{W}_{(.)}^l = \boldsymbol{W}_{(.)}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{W}_{(.)}^l} C$
  $\boldsymbol{U}_{(.)}^l = \boldsymbol{U}_{(.)}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{U}_{(.)}^l} C$
  $\boldsymbol{V}_{(.)}^l = \boldsymbol{V}_{(.)}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{V}_{(.)}^l} C$
  $\boldsymbol{b}_{(.)}^l = \boldsymbol{b}_{(.)}^l - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{b}_{(.)}^l} C$
36 **end**
37 **Output:** Optimal $\boldsymbol{U}_{(\cdot)}^l$, $\boldsymbol{W}_{(\cdot)}^l$, $\boldsymbol{V}_{(\cdot)}^l$, and $\boldsymbol{b}_{(\cdot)}^l$ $\forall l$.

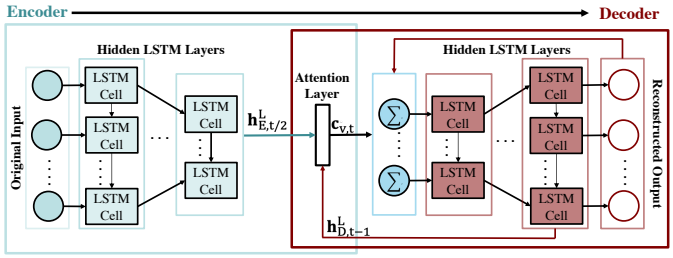


Fig. 5. Illustration of the AEA architecture.

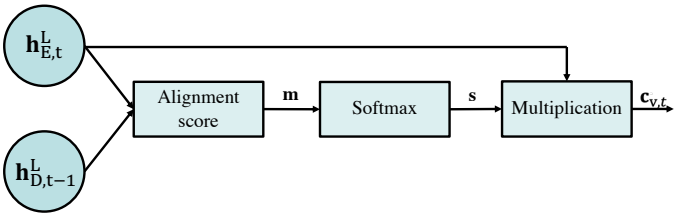


Fig. 6. Illustration of the attention layer.

a context vector that presents a view of the whole sequence balanced by the learned weights.

Since the AEA model, by nature, is sequential, it only makes sense to use a seq2seq algorithm when adding the attention layer. The AEA architecture is similar to LSTM-SAE, with the addition of a new layer, the attention layer, which is present between the encoder and decoder [35]. Two inputs are fed to the attention layer, namely, the encoder's hidden state $\boldsymbol{h}^{L}_{\mathrm{E},t}$ at time step t and the decoder's hidden state $\boldsymbol{h}^{L}_{\mathrm{D},t-1}$ at time $t-1$. The attention layer gives weights to different time steps and hence gives higher importance to time steps that provide higher contribution towards acquiring the desired output. Thus, in case of a theft, the reconstruction error is higher. This is achieved using alignment scoring, Softmax, and multiplication.

The alignment scoring function of the attention layer, Γ, is a neural network with a feed forward architecture that is jointly trained with the model using two inputs $\boldsymbol{h}^{L/2}_{\mathrm{E},t}$ and $\boldsymbol{h}^{L}_{\mathrm{D},t-1}$. The neural network Γ learns weights that give significance to important time steps. The alignment score, $\boldsymbol{m}$, is the output of this feed forward neural network and is given by

$$\boldsymbol{m} = \Gamma(\boldsymbol{h}^{L/2}_{\mathrm{E},t}, \boldsymbol{h}^{L}_{\mathrm{D},t-1}). \tag{11}$$

The attention weight is then a Softmax of alignment scores

$$\boldsymbol{s} = \frac{\exp(\boldsymbol{m})}{\sum_{|\boldsymbol{m}|}\exp(\boldsymbol{m})}. \tag{12}$$

$|\boldsymbol{m}|$ denotes the cardinality of vector $\boldsymbol{m}$. The context vector is calculated as the weighted sum of the encoder's hidden state vectors:

$$\boldsymbol{c}_{\mathrm{v},t} = \sum_{T} \boldsymbol{s} \times \boldsymbol{h}^{L/2}_{\mathrm{E},t}. \tag{13}$$

The input is passed through the hidden LSTM layers in sequence before going to the attention layer where the aforementioned process of linear combination takes place. Then, they are passed to the latent layer. Finally, the reconstruction takes place in cycled seq2seq LSTM layers. The training algorithm of the AEA is described in Algorithm 5, which presents a similar logic as Algorithms 2 and 4, but includes the attention layer in the encoder section in lines 14 - 17. The concatenation of $\boldsymbol{c}_{\mathrm{v},t}$ and the reconstructed output $\tilde{\boldsymbol{x}}$, $\sum(\boldsymbol{c}_{\mathrm{v},t}, \tilde{\boldsymbol{x}})$ is fed to the hidden layers of the decoder.

For all of the built architectures, after training is completed using $\boldsymbol{X}_{\mathrm{TR}}$, we apply the test dataset $\boldsymbol{X}_{\mathrm{TST}}$. If the cost function, which computes the MSE between the original and reconstructed electricity consumption for SAE and AEA, or reconstruction probability for VAEs, is greater than a specific threshold, a $y =$ '1' label will denote a malicious sample. Else, $y =$ '0' label will be assigned to a benign sample.

D. Performance Evaluation of the Detectors

A true positive (TP) is a rightly detected malicious sample and a true negative (TN) is a rightly detected benign sample. A false positive (FP) is a wrongly detected benign sample and a false negative (FN) is a wrongly detected malicious sample. To assess the investigated detectors' performance, we use multiple evaluation metrics. Sensitivity (DR) is calculated using the correctly identified malicious samples (DR = TP/(FN+TP)). FA is computed using the incorrectly marked benign samples (FA = FP/(FP+TN)). Specificity (SP) is expressed as (SP = 100-FA). Precision (PR) refers to the proportion of the rightly identified malicious readings to all malicious readings (PR = TP/(FP+TP)). Accuracy (ACC) is the arithmetic mean of DR and SP. F1-score represents the harmonic mean of DR and PR. The area under the receiver operating characteristic (ROC) curve (AUC) plots TP against FP.

For each developed detector, we compare the predicted label $\boldsymbol{Y}_{\mathrm{CAL}}$ against $\boldsymbol{Y}_{\mathrm{TST}}$ to build a confusion matrix that we use to compute the performance evaluation metrics' scores. To obtain $\boldsymbol{Y}_{\mathrm{CAL}}$ for the anomaly detectors, we introduce thresholds. Comparing the reconstruction error/probability versus this threshold differentiates between a benign and malicious sample. For anomaly detectors, the threshold is set based on the median of the interquartile range (IQR) of the ROC curve. Any score below that threshold value denotes a benign sample and scores that are above that value denote malicious samples.

E. Hyper-parameter Optimization

The optimal choices of hyper-parameters enhance the detection performance of the electricity stealth detectors. We tuned a set of hyper-parameters as follows, the number of hidden (dense or LSTM) layers (L); the number of layers in the encoder and decoder is the same, the optimal number of neurons in each layer (N_l), the optimizer (O), the dropout rate (D), the hidden activation (A_{H}), and the output activation function (A_{O}). Algorithm 6 shows that the optimization of hyper-parameter is implemented using multiple steps in sequence. Since we aim to optimize a large number of hyper-parameters, implementing an exhaustive grid search is not practical since it results in high computational complexity. Thus, we perform a sequential grid search [15]. We perform a cross-validation over $\boldsymbol{X}_{\mathrm{TR}}$ of the ISET dataset to select the hyper-parameters. The optimal setting of hyper-parameters is denoted by P^{*}, which leads to an improved DR against the validation set. MD is the resultant model of a given combination of hyper-parameters.

Algorithm 5: LSTM-AEA Training

Input Data: $\boldsymbol{X}_{\text{TR}}$
Initialization: Weights $\boldsymbol{U}^l_{(.)}$, $\boldsymbol{W}^l_{(.)}$, $\boldsymbol{V}^l_{(.)}$, and bias $\boldsymbol{b}^l_{(.)}$ $\forall l$, $\boldsymbol{h}^L_{\text{D},t-1}$ and $\tilde{\boldsymbol{x}}$
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**
        $\boldsymbol{i}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_i \boldsymbol{x}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_i)$,
        $\boldsymbol{f}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_f \boldsymbol{x}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_f)$,
        $\boldsymbol{c}^l_{\text{E},t} = f^l_{\text{E},t} c^l_{\text{E},t-1} + i^l_{\text{E},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t + \boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{b}^l_c)$,
        $\boldsymbol{o}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_o \boldsymbol{x}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{E},t} + \boldsymbol{b}^l_o)$,
        $\boldsymbol{h}^l_{\text{E},t} = \boldsymbol{o}^l_{\text{E},t} \tanh(\boldsymbol{c}^l_{\text{E},t})$,
        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
    $\breve{\boldsymbol{x}} = \sum(\boldsymbol{c}_{\text{V},t}, \tilde{\boldsymbol{x}})$
    Decoder:
    At initial time step, the decoder hidden and cell states 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**
        $\boldsymbol{i}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_i \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_i)$,
        $\boldsymbol{f}^l_{\text{D},t} = \varphi(W^l_f \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_f)$,
        $\boldsymbol{c}^l_{\text{D},t} = f^l_{\text{D},t} c^l_{\text{D},t-1} + i^l_{\text{D},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t + \boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{b}^l_c)$,
        $\boldsymbol{o}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_o \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{D},t} + \boldsymbol{b}^l_o)$,
        $\boldsymbol{h}^l_{\text{D},t} = \boldsymbol{o}^l_{\text{D},t} \tanh(\boldsymbol{c}^l_{\text{D},t})$,
      end
    end
    Back propagation: Compute $\nabla_{\boldsymbol{W}^l_{(.)}} C$, $\nabla_{\boldsymbol{U}^l_{(.)}} C$, $\nabla_{\boldsymbol{V}^l_{(.)}} C$, and $\nabla_{\boldsymbol{b}^l_{(.)}} C$
  end
  Weight and bias update:
  $\boldsymbol{W}^l_{(.)} = \boldsymbol{W}^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{W}^l_{(.)}} C$
  $\boldsymbol{U}^l_{(.)} = \boldsymbol{U}^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{U}^l_{(.)}} C$
  $\boldsymbol{V}^l_{(.)} = \boldsymbol{V}^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{V}^l_{(.)}} C$
  $\boldsymbol{b}^l_{(.)} = \boldsymbol{b}^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{\boldsymbol{b}^l_{(.)}} C$
end
Output: Optimal $\boldsymbol{U}^l_{(.)}$, $\boldsymbol{W}^l_{(.)}$, $\boldsymbol{V}^l_{(.)}$, and $\boldsymbol{b}^l_{(.)}$ $\forall l$.

Algorithm 6: Optimization of Hyper-parameter

Initialization: Optimizer = SGD, dropout rate = 0, hidden activation = Relu, output activation = Softmax
Output: Optimized hyper-parameters
Input: Training set $\boldsymbol{X}_{\text{TR}}$
for $L \in \mathcal{L}$ **do**
  for $N_l \in \mathcal{N}$ **do**
    Apply algorithms 1-5 with L, N_l, and other initial hyper-parameters;
    Record DR and FA;
  end
end
L^*, N^*_l, and other initial hyper-parameters initiate MD1
for $O \in \mathcal{O}$ **do**
  Apply algorithms 1-5 with MD1's hyper-parameters and o;
  Record DR and FA;
end
L^*, N^*_l, O^*, and other initial hyper-parameters initiate MD2
for $D \in \mathcal{D}$ **do**
  Apply algorithms 1-5 with MD2's hyper-parameters and D;
  Record DR and FA;
end
L^*, N^*_l, O^*, D^*, and other initial hyper-parameters initiate MD3
for $A_{\text{H}} \in \mathcal{A}_{\text{H}}$ **do**
  for $A_{\text{O}} \in \mathcal{A}_{\text{O}}$ **do**
    Apply algorithms 1-5 with MD3's hyper-parameters, $A_{\text{H}}$and A_{O};
    Record DR and FA;
  end
end
L^*, N^*_l, O^*, D^*, A^*_{H}, and A^*_{O} define the optimal parameters.

IV. Experimental Results

Keras sequential API is utilized for the purpose of training and testing the detectors that we are investigating. All developed detectors (and benchmarks) are first trained offline at the utility. Then, in real-time, online detection of malicious samples takes place at the utility.

A. Benchmark Detectors

We compare the performance of the proposed detectors against supervised classifiers and anomaly detectors. Supervised classifiers utilize both benign and malicious samples for training and testing; they include shallow classifiers, namely, multi-class SVM and Naïve Bayes as well as deep classifiers, namely, feed forward and LSTM. Anomaly detectors are trained only on benign samples and then tested on both benign and malicious samples and include shallow single-class SVM and ARIMA. The SVM, Naïve Bayes, and feed forward-based

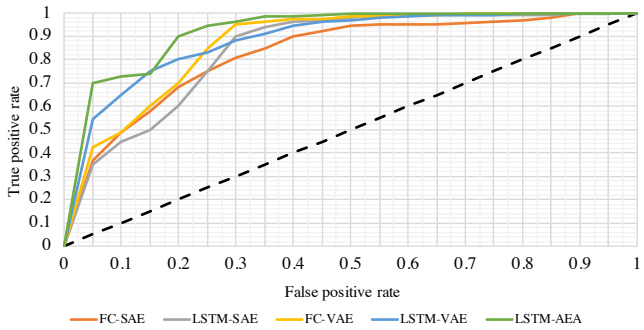


Fig. 7. ROC curves for the proposed auto-encoders.

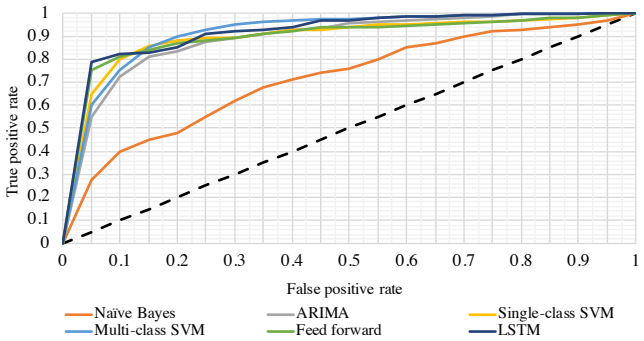


Fig. 8. ROC curves for the benchmark detectors.

detectors are static classifiers that are incapable of capturing the time-series nature of the energy data. The ARIMA model is dynamic and capable of capturing temporal dependencies, but has a shallow architecture. The LSTM model is dynamic and has a deep structure, but it represents a supervised model.

B. Threshold Values

To plot the ROC curves and determine the threshold values, we utilize the same cross-validation over $\boldsymbol{X}_{\text{TR}}$ of the ISET dataset discussed in Section III.E. Figures 7 and 8 plot the ROC curves for the benchmark detectors and auto-encoders, respectively. Figures 7 and 8 are utilized to obtain the optimal threshold values for the anomaly detectors to recognize benign and malicious samples. After dividing each curve into three quartiles and taking the median of IQR, the optimal threshold values are: 0.58 and 0.45 for ARIMA and single-class SVM, respectively, 0.58 and 0.61 for FC-SAE and LSTM-SAE, respectively, 0.43 and 0.47 for FC-VAE and LSTM-VAE, respectively, and 0.51 for AEA. Figure 8 also plots the ROC curves for the rest of the supervised benchmark detectors for comparison purposes.

C. Hyper-parameter Optimization

Sequential grid search is carried out to optimize the hyper-parameters of the benchmark and proposed detectors. For the shallow detectors, for the Naïve Bayes classifier, the optimal variance is found to be 10^{-9}. For ARIMA, the differencing degree and moving average values are 1 and 0, respectively. For both SVM-based detectors, the optimal kernel and gamma are scale and sigmoid, respectively. For the multi-class SVM, the optimal regularization parameter is 1.0.

TABLE I
OPTIMAL HYPER-PARAMETER VALUES

Detector	**SAE**		**VAE**		**AEA**
hyper-parameter	**FC**	**LSTM**	**FC**	**LSTM**	**LSTM**
L^*	8	4	8	4	6
O^*	Adam	Adam	Adam	SGD	SGD
D^*	0.4	0.2	0.4	0	0
A_{H}^*	Sigmoid	Sigmoid	Relu	Tanh	Sigmoid
A_{O}^*	Softmax	Sigmoid	Softmax	Sigmoid	Sigmoid

For the deep benchmark detectors and the proposed auto-encoders, the optimal values are chosen from the following ranges: for the number of layers $\mathcal{L} = \{2, 3, 4, 5\}$, for the number of neurons $\mathcal{N} = \{100, 200, 300, 400, 500\}$, the optimizer $\mathcal{O} = \{$SGD, Adam, Adamax, and Rmsprop$\}$, the dropout rate $\mathcal{D} = \{0, 0.2, 0.4, 0.5\}$, the hidden activation functions $\mathcal{A}_{\text{H}} = \{$ Relu, Sigmoid, Linear, Tanh$\}$, and the output activation function $\mathcal{A}_{\text{O}} = \{$Softmax, Sigmoid$\}$. For the feed forward model, $L^* = 5$, $N_l^* = 500$, O^*: Adamax, $D^* = 0$, A_{H}^*: ReLU, and A_{O}^*: Softmax. For the LSTM model, $L^* = 4$, $N_l^* = 300$, O^*: Adam, $D^* = 0.2$, A_{H}^*: ReLU, and A_{O}^*: Sigmoid. The optimized hyper-parameter values for each developed auto-encoders are summarized in Table I, which shows that the LSTM-based detectors tend to have fewer layers than the fully connected counterparts. Also, fully connected-based detectors perform better with Softmax output activation function, while Sigmoid is more suitable for the LSTM-based detectors. The number of neurons N_l, for the FC-SAE, in the four encoding layers are: $(400, 300, 200, 100)$ with opposite order in the decoder's part. For the LSTM-SAE, the number of neurons in the two encoder's layers are $(500, 300)$ with opposite order in the decoder's part. The number of neurons is $(500, 400, 300, 100)$ for the encoder part of the FC-VAE and $(400, 300)$ for the encoder side in the LSTM-VAE model, with opposite order in the decoder side. The number of neurons in the three layers within the encoder of the AEA model is $(500, 300, 200)$, with opposite order in the decoder side.

D. Performance Evaluation

Tables II and III summarize the performance of the developed detectors using the SGCC and ISET, respectively. The reported performance is based on completely unseen data (test set), which is different from the data used for selecting hyper-parameters and constructing the ROC curves (validation set).

Using the SGCC dataset, in SAE, an improvement of 3% in DR and 2% in FA is observed when LSTM-SAE is utilized compared to FC-SAE since it captures better the time-series nature of the electricity consumption data. This results in an improved detection performance. When VAE detectors are employed, the performance is further improved by $4-7\%$ in DR and $3-5\%$ in FA. This is because VAE captures better the variability within the electricity consumption data compared to SAE models. Compared to the fully connected models, the LSTM-based model improves the detection performance by 3% in DR and FA. Compared to the VAE models, the AEA

TABLE II
PERFORMANCE EVALUATION USING THE SGCC DATASET

Detector/Metric	DR	FA	SP	PR	ACC	F1	AUC
FC-SAE	83	14	86	83	84.5	83	83
LSTM-SAE	86	12	88	87	87	86.5	85
FC-VAE	90	9	91	91	90.5	90.5	88
LSTM-VAE	93	6	94	93	93.5	93	90
LSTM-AEA	**96**	**4**	**96**	**95**	**96**	**95.5**	**93**
Benchmark Detectors							
Naïve Bayes	75	16	84	75	79.5	77	73
ARIMA	88	10	90	87	89	87	88
Single-class SVM	91	8.5	91.5	90	91	90	89
Feed forward	91	9.5	90.5	90	91	90.5	89
LSTM	91.5	9	91	90.5	91	91	90
Multi-class SVM	92	7.5	92.5	91	92	91.5	90

TABLE III
PERFORMANCE EVALUATION USING THE ISET DATASET

Detector/Metric	DR	FA	SP	PR	ACC	F1	AUC
FC-SAE	81	15	85	81	83	81	81
LSTM-SAE	85	13	87	85	86	85	82
FC-VAE	88	11	89	89	88.5	88.5	85
LSTM-VAE	91	7	93	91	92	91	86
LSTM-AEA	**94**	**5**	**95**	**93**	**94.5**	**93.5**	**90**
Benchmark Detectors							
Naïve Bayes	73	18	82	73	77.5	73	70
ARIMA	86	12	88	86	87	86	87
Single-class SVM	90	9	91	89	90.5	89.5	87
Feed forward	90	11	89	89	89.5	89.5	88
LSTM	90.5	10	90	89.5	90	90	89
Multi-class SVM	91	8	92	90	91.5	90.5	89

further improves the DR and FA by $3-6\%$ and $2-5\%$, respectively. Compared to the SAE models, the AEA provides an improvement of up to $10-13\%$ and $8-10\%$ in DR and FA, respectively. This is due to the included attention layer that enhances the overall detection performance. These results show that the LSTM-based models enhance the detection performance compared to fully connected models. The AEA-based detector offers the best detection performance. Compared to benchmark detectors, AEA improves the DR and FA by $4-21\%$ and $3.5-12\%$, respectively.

Using the ISET dataset, in SAE, an improvement of 4% in DR and 2% in FA is observed when LSTM-SAE is utilized compared to FC-SAE. When VAE detectors are employed, the performance is further improved by $3-7\%$ in DR and $2-4\%$ in FA. The LSTM-based model improves the detection performance compared to fully connected models by 3% in DR and 4% in FA. The AEA further improves the DR and FA by $3-6\%$ and $2-6\%$, respectively, compared to the VAE models, while the improvement is up to $9-13\%$ and $8-10\%$ in DR and FA, respectively, compared to the SAE models. Compared to benchmark detectors, the AEA detector improves the DR and FA by $3-21\%$ and $3-13\%$, respectively. This shows that the deep and recurrent architecture of AEA offers superior performance compared to shallow and static classifiers, as well as the rest of the investigated auto-encoder-based detectors.

Table IV shows the time taken to train the proposed detectors on different sizes of the ISET dataset, where $|\boldsymbol{X}_{\text{TR}}|$ size is 60 million and $|\cdot|$ denotes the cardinality. Such training is done offline and as can be observed, the AEA requires a maximum time of 3 hours, which is low. The online testing of all the developed detectors requires $1-2$ seconds to report a decision. The detection is done on individual readings rather than the aggregate consumption of all meters.

TABLE IV
COMPUTATIONAL COMPLEXITY USING THE ISET DATASET

Model	Metric	Dataset Size		
		0.5$\|\boldsymbol{X}_{\text{TR}}\|$	**0.75**$\|\boldsymbol{X}_{\text{TR}}\|$	$\|\boldsymbol{X}_{\text{TR}}\|$
FC-SAE	**Time (min)**	72	97	137
	ACC (%)	70	78.5	83
LSTM-SAE	**Time (min)**	90	127	183
	ACC (%)	75	83	86
FC-VAE	**Time (min)**	81	103	141
	ACC (%)	79.5	86	88.5
LSTM-VAE	**Time (min)**	97	132	188
	ACC (%)	83	90	92
LSTM-AEA	**Time (min)**	102	142	193
	ACC (%)	86	93	94.5

TABLE V
PERFORMANCE EVALUATION AGAINST SEPARATE ATTACKS

Model	Metric	Attack Function						AVG
		(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	
FC-SAE	**DR**	82.5	81	83	80	80	80	81
	FA	15	16	10	17	17	19	15.5
LSTM-SAE	**DR**	84.5	83	90	82	84	83	84.5
	FA	13	15	9	14	14	14	13
FC-VAE	**DR**	86	85	93	88	88	87	88
	FA	11	12	8	10	11	12	10.5
LSTM-VAE	**DR**	88.5	88	95	91	91	90	90.5
	FA	7.5	8	4.5	8	8.5	8.5	7.5
LSTM-AEA	**DR**	94	93	97	94	94	93	94
	FA	3.5	4	2.5	6.5	5.5	6.5	5

Table V shows the DR and FA of the proposed auto-encoders when tested separately on each of the attack functions discussed in Section II.B. Herein, the models are only trained on the benign samples from the ISET dataset using the same optimal hyper-parameters and threshold values discussed in Sections IV.B and IV.C, respectively, without the use of ADASYN since the data is already balanced. From the results, it is evident that using the threshold values obtained using Figure 7 provides consistent results in multiple experiments. Although the DR varies depending on the complexity of the attack, the average values are still consistent with Table III.

V. CONCLUSION

We introduced multiple deep auto-encoder anomaly detectors for electricity stealth detection in this paper. They were trained only on benign electricity consumption samples. This approach overcame the limited availability issue of malicious electricity consumption profiles. Two research questions were answered in this paper: (a) whether the deep architectures can offer superior detection performance compared to the shallow detectors and (b) whether the recurrent LSTM-based architectures can improve the detection performance compared to fully connected detectors that are static. Our investigations revealed that a notable improvement took place when the deep and recurrent anomaly detectors were utilized compared to shallow and static architectures. The best detection performance was exhibited by the LSTM-AEA, which offered an improvement of $4-21\%$ and $4-13\%$ in DR and FA, respectively.

REFERENCES

[1] A. Takiddin, M. Ismail, U. Zafar, and E. Serpedin, "Variational auto-encoder-based detection of electricity stealth cyber-attacks in AMI networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*. Amsterdam, Netherlands, Jan. 2021, pp. 1590–1594.

[2] A. Takiddin, M. Ismail, U. Zafar, and E. Serpedin, "Deep autoencoder-based detection of electricity stealth cyberattacks in AMI networks," in *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*. Iasi, Romania, Jul. 2021, pp. 1–6.

[3] 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, Jan 2016.

[4] 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, Aug 2018.

[5] "Electric sector failure scenarios and impact analyses - version 3.0," *National Electric Sector Cybersecurity Organization Resource (NESCOR)*, Dec 2015, [Online]. Available: https://smartgrid.epri.com/doc/NESCOR%20Failure%20Scenarios%20v3%2012-11-15.pdf.

[6] A. Takiddin, M. Ismail, U. Zafar, and E. Serpedin, "Robust electricity theft detection against data poisoning attacks in smart grids," *IEEE Transactions on Smart Grid*, vol. 12, no. 3, pp. 2675–2684, Dec 2020.

[7] R. Punmiya and S. Choe, "Energy theft detection using gradient boosting theft detector with feature engineering-based preprocessing," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2326–2329, Jan 2019.

[8] T. S. Murthy, N. Gopalan, and V. Ramachandran, "A naive bayes classifier for detecting unusual customer consumption profiles in power distribution systems - APSPDCL," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*. Coimbatore, India, Mar. 2020, pp. 673–678.

[9] 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*. Washington, DC, USA, Oct. 2018, pp. 3073–3078.

[10] 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, Oct 2019.

[11] 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, Mar 2016.

[12] Z. Yan and H. Wen, "Electricity theft detection base on extreme gradient boosting in AMI," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–9, Jan 2021.

[13] 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," pp. 73–102, Aug 2019.

[14] M. Nabil, M. Mahmoud, M. Ismail, and E. Serpedin, "Deep recurrent electricity theft detection in AMI networks with evolutionary Hyper-Parameter tuning," in *2019 International Conference on Internet of Things (iThings)*. Atlanta, GA, USA, Jul. 2019, pp. 1002–1008.

[15] A. Takiddin, M. Ismail, M. Nabil, M. Mahmoud, and E. Serpedin, "Detecting electricity theft cyber-attacks in AMI networks using deep vector embeddings," *IEEE Systems Journal*, pp. 1–10, Oct 2020.

[16] 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)*. Limassol, Cyprus, Jun. 2020, pp. 1598–1602.

[17] M. N. Hasan, R. N. Toma, A.-A. Nahid, M. M. M. Islam, and J.-M. Kim, "Electricity theft detection in smart grid systems: A cnn-lstm based approach," *Energies*, vol. 12, no. 17, Aug 2019.

[18] J. Wang, C. Roberts, G. Vidal, and S. Leichenauer, "Anomaly detection with tensor networks," Jun 2020.

[19] 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, May 2016, pp. 199–210.

[20] S. K. Singh, R. Bose, and A. Joshi, "PCA based electricity theft detection in advanced metering infrastructure," in *7th International Conference on Power Systems (ICPS)*. Pune, India, Dec. 2017, pp. 441–445.

[21] 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)*. South Padre Island, TX, USA, Apr. 2018, pp. 135–140.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," Jun 2017.

[23] Z. Zheng, Y. Yang, X. Niu, H.-N. Dai, and Y. Zhou, "Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1606–1615, Dec 2018.

[24] "Irish Social Science Data Archive." [Online]. Available: http://www.ucd.ie/issda/data/commissionforenergyregulationcer/

[25] 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)*. Shanghai, China, Jun. 2020, pp. 447–452.

[26] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*. Chuncheon, Korea (South), Feb. 2018, pp. 178–183.

[27] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, Oct 2016, vol. 1, no. 2.

[28] X. Liu, Z. Lin, and Z. Feng, "Short-term offshore wind speed forecast by seasonal ARIMA-A comparison against GRU and LSTM," *Energy*, vol. 227, p. 120492, Jul 2021.

[29] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in neural information processing systems*, Nov 2015, pp. 3079–3087.

[30] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *International conference on machine learning*. Lille, France, Jul. 2015, pp. 843–852.

[31] M. S. Kim, J. P. Yun, S. Lee, and P. Park, "Unsupervised anomaly detection of LM guide using variational autoencoder," in *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. Bucharest, Romania, May 2019, pp. 1–5.

[32] O. Fabius and J. R. Van Amersfoort, "Variational recurrent auto-encoders," in *3rd International Conf. Learning Representations (ICLR)*. San Diego, CA, USA, May 2015, pp. 1–5.

[33] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *20th SIGNLL Conference on Computational Natural Language Learning*, Berlin, Germany, Aug. 2016, pp. 10–21.

[34] 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.

[35] 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, Feb 2020.