Variational Auto-encoder-based Detection of Electricity Stealth Cyber-attacks in AMI Networks

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

***Abstract*—Current efforts to detect electricity theft cyber-attacks in advanced metering infrastructures (AMIs) are hindered by the lack of malicious electricity theft datasets. Therefore, anomaly detectors trained with the energy consumption profiles of honest customers appear as a plausible solution to overcome the lack of malicious datasets. Taking into account this constraint, this paper examines the performance of two structures of variational auto-encoders (VAEs); fully-connected (FC) VAE and long-short-term-memory (LSTM) VAE in detecting electricity thefts. The proposed structures are promising and exhibit an improvement of $11-15\%$ in detection rate, $9-22\%$ in false alarm rate, and $27-37\%$ in the highest difference compared to existing state-of-the-art anomaly detectors that are shallow and static, such as single-class support vector machine (SVM) and auto-regressive integrated moving average (ARIMA) models.**

***Index Terms*—electricity theft, auto-encoders, deep learning.**

I. Introduction

Electricity theft is a major problem for power companies not only because of the financial loss but also the grid overload and negative influence. To enable the energy consumption monitoring task in power grids, power companies are currently deploying advanced metering infrastructures (AMIs) with smart meters mounted in the customers' premises to track the energy consumption data. Regrettably, AMIs are subject to cyber electricity thefts. Malicious customers can hack AMIs and alter the integrity of energy consumption readings [1] - [4]. Thus, there is an acute need to develop algorithms to detect such cyber-attacks.

A. Related Work and Limitations

Two classes of machine learning (ML) approached were proposed in the literature to identify electricity thefts. The first class is based on supervised learning and employs both benign and malicious energy consumption data for training. For example, the support vector machine (SVM)-based classifier in [5] exhibits a low detection accuracy of 72%. This approach suffers from the lack of malicious datasets to train the classifier. Moreover, it is not practical in case of attacks that the model is not trained to detect, such as zero-day attacks that take place for the first time. Therefore, the second class, that is based on anomaly detection, is used. Anomaly detectors are trained using benign datasets to learn the honest customer's data consumption profile. Then, it identifies electricity theft patterns by measuring the deviation from the normal profile. For example, the auto-regressive integrated moving average (ARIMA)-based model presents a 77% detection rate [6].

Still, the existing anomaly detectors [1], [6], [7] suffer from three major weaknesses. First, the existing approaches rely mostly on shallow techniques with limited detection capabilities due to their inability to apprehend the complex patterns present in the energy consumption data. Second, most anomaly detectors do not make use of the temporal correlation in the energy consumption data. Third, the existing approaches exhibit a relatively low detection performance. Thus, new approaches are needed to capture the complex patterns and temporal correlations of the energy consumption readings to enhance the detection performance.

B. Contributions

This paper proposes cyber-attack anomaly detectors based on variational auto-encoders (VAEs) and investigates their detection performance. The rationale behind the deep structure of the auto-encoders is to capture the complex patterns present in the energy consumption profiles. We consider sequence-to-sequence (seq2seq) structures based on long-short-term-memory (LSTM) recurrent neural networks (RNNs) to model the time-series nature of the energy consumption data. The contributions of this paper are as follows:

- VAE detectors are proposed to identify electricity cyber thefts. The advantage of using VAE lies in the latent variables, which are stochastic. Also, the probabilistic encoder of the VAE models the distribution of these latent variables. The variations in the latent space are captured by the VAE anomaly detector via the variance parameter.
- The performance of fully connected feed forward variational auto-encoders (FC-VAE) and LSTM-based RNN variational auto-encoders (LSTM-VAE) is investigated. FC-VAE presents a simple architecture with low computational complexity, while LSTM-VAE captures the temporal correlations in the energy consumption data [8].
- Sequential grid search hyper-parameter optimization [8] is adopted to optimize one hyper-parameter at a time to reduce the computational complexity and augment the overall detection performance.
- The proposed anomaly detectors are tested against six types of electricity theft cyber-attacks. The models' performance is compared to shallow architectures, namely,

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.

single-class SVM and ARIMA. The developed VAE detector improves the detection rate (DR), false alarm (FA), and highest difference (HD) by $11-15\%$, $9-22\%$, and $27-37\%$, respectively.

The rest of this paper is structured as follows. Section II describes the benign and malicious load profiles and the data preparation step for training, validation, and testing. Section III explores the design and optimization of the anomaly detectors. Section IV depicts the experimental results. Section V concludes this paper's contributions.

II. Data Preparation

The data offered by the public Irish Smart Energy Trail [9] is adopted herein as the benign energy consumption data. The malicious data is created by employing the six cyber-attack models proposed in [1]. Each cyber-attack function models the electricity theft behavior of a malicious user.

A. Benign Dataset

The Irish Smart Energy Trail dataset [9] is used to train and test the proposed electricity theft detector. This dataset consists of readings from 3,000 smart meters placed at residential units and that are recorded every half an hour for a 1.5-year period.

B. Malicious Dataset

Let $E_c(d,t)$ denote the energy consumption value for customer c at day d and time t. All these consumption values are regarded as entries of matrix $\boldsymbol{E}_c$. The energy consumption value reported by an honest customer's smart meter is denoted by $R_c(d,t)$ ($R_c(d,t) = E_c(d,t)$). Thus, $\boldsymbol{E}_c = \boldsymbol{R}_c$ for honest customers. Malicious customers manipulate the integrity of the energy consumption readings and reduce their electricity bills such that $R_c(d,t) \neq E_c(d,t)$. We employ the false data injection approach [1] to build the malicious dataset. Next, we describe the set of cyber-attack functions, which can be classified into three main classes. The first class refers to *partial reduction attacks*. As a representative example of such a partial reduction attack, cyber-attack function $f_1(E_c(d,t))$ decreases the actual energy consumption via a penalty constant factor α, and the reported energy consumption $R_c(d,t)$ is given by

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

As a more general example, cyber-attack function $f_2(E_c(d,t))$ considers the dynamic penalty factor $\beta(d,t)$:

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

The second class consists of *selective by-pass attacks*. In this class, malicious customers claim zero energy consumption at a given time window, $[t_i(d), t_f(d)]$, and report actual energy consumption data the rest of the time. Thus, it is modelled as:

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

The third class comprises the *price-based load control attacks*. These attacks are applicable to cases where the electricity price varies during the day. In this set-up, one possible attack function may report a flat energy consumption value across the day:

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

where $\mathbb{E}[\cdot]$ denotes the expectation operator. To avoid reporting a constant value across the day, which can be easily detected, a dynamic fraction $\beta(d,t)$ is employed:

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

Finally, another attack function that we consider reports high energy consumption values during periods of low electricity tariffs and vice versa:

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

Each of the above cyber-attack functions is applied to the customer energy consumption profile matrix $\boldsymbol{E}_c$. This operation leads to six malicious matrices per customer. The benign and malicious datasets are normalized to bring the values of all features to a common scale. The normalized dataset presents zero mean and unit variance. The normalized benign dataset $\boldsymbol{B}$ is then divided into two disjoint subsets at the ratio 2:1. The first subset is used as training data $\boldsymbol{X}_{\text{TR}}$. The second subset is concatenated with the normalized malicious dataset $\boldsymbol{M}$ to construct the test data. In this concatenation, each sample is associated with a label that takes value '0' if the sample is benign and value '1' if the sample is malicious. As much more malicious data is generated than benign data, and to avoid misleading performance results, we employ the adaptive synthetic sampling approach (ADASYN) [10] to balance the sets of benign and malicious data by over-sampling the minor (benign) class within the test set. Thus, we obtain test data $\boldsymbol{X}_{\text{TST}}$ with label $\boldsymbol{Y}_{\text{TST}}$.

III. Design of Electricity Theft Detector

Next, we focus on designing the electricity theft detectors. We analyze the adoption of two types of VAE architectures.

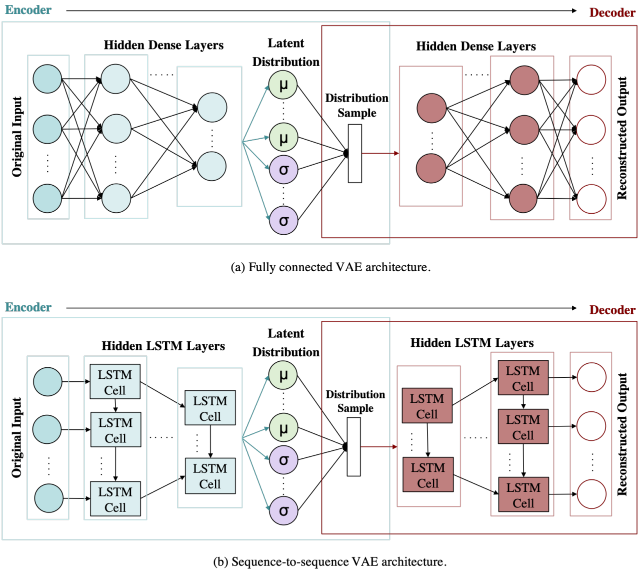
A. Variational Auto-encoder Architecture

A VAE is a directed probabilistic graphical model whose posterior is approximated by a neural network [11]. The VAE assumes that a data point x is generated according to the unobserved continuous random variable k. k' represents a specific value that is generated using a prior distribution $p(k)$. Then, x', which is the data instance is generated according to the conditional distribution $p(x|k)$. Since inferring the distribution $p(k|x)$ is hard and the values of k are unknown, VAE determines the probability $p(k|x)$ and $p(x|k)$ through the network of encoders and decoders, respectively. The approximation of the true posterior $p(k|x)$ is denoted by $q(k|x)$ and the log-likelihood of data point x is

$$\log p(x) = D_{\text{KL}}(q(k|x)||p(k|x)) + \mathcal{L}(\Theta; x). \tag{7}$$

Notation D_{KL} designates the Kullback–Leibler (KL) divergence, Θ denotes the model parameters, and $\mathcal{L}(\Theta; x)$ is a variational lower bound on the log-likelihood expressed as

$$\mathcal{L}(\Theta; x) = -D_{\text{KL}}(q(k|x)||p(x)) + \mathbb{E}_{q(k|x)}[\log p_{\Theta}(x|k)]. \tag{8}$$



(a) Fully connected VAE architecture.

(b) Sequence-to-sequence VAE architecture.

Fig. 1. Illustration of the VAE architecture.

The model parameters Θ are learned by optimizing the lower bound $\mathcal{L}(\Theta; x)$ ($\mathcal{L}(\Theta; x) \leq \log p(k)$). If the latent variables are modeled as univariate Gaussian, then $k = \mu + \sigma\mathcal{N}$ with $\mathcal{N}$ denoting a normal distribution with zero mean and unit variance [15]. Therefore, the activations of the encoder forward-pass determine the mean (μ) and variance (σ^2) of the Gaussian distribution. In VAE, the reconstruction probability is used as an anomaly score [12].

1) Fully Connected VAE: Figure 1a shows the structure of an FC-VAE. In VAE, the latent space is continuous and simplifies the random sampling for interpolation by outputting two vectors, mean vector $\boldsymbol{\mu_x}$ and variance vector $\boldsymbol{\sigma_x^2}$. To generate samples of the energy consumption data, the decoder uses $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$. Then, the reconstruction probability is calculated to detect electricity theft. The combined VAE loss function is expressed as follows

$$\hat{C} = ||\boldsymbol{x} - \tilde{\boldsymbol{x}}_{\Theta}||^2 + D_{\text{KL}}(\mathcal{G}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x^2})||\mathcal{N}(0,1)), \tag{9}$$

$\tilde{\boldsymbol{x}}_{\Theta}$ represents the reconstructed output as function Θ. $\mathcal{G}(\mu_x, \sigma_x^2)$ and $\mathcal{N}(0,1)$ denotes the general and standard normal distribution, respectively.

The probability distribution's parameters, $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$, are generated by the auto-encoder. The decoder then generates vectors that are used to construct an output via the decoder layers. The loss is then calculated using (9). After that, back propagation is used to update the parameters of the encoder and decoder. Algorithm 1 shows the calculation of the optimal parameters based on the iterative gradient descent.

2) Sequence-to-Sequence VAE: Originally, LSTM-VAEs were used as a way of data generation for language modeling applications [13]. Herein, we adopt an LSTM-VAE as an anomaly detector. Figure 1b shows the structure of an LSTM-VAE. The training algorithms of LSTM-VAE are supposed to generate $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x^2}$, which are used to generate the samples that the decoder uses as shown in Algorithm 2.

Algorithm 1: Training of FC-VAE

```
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: Compute:
        Encoder:
        for all layers $l = 1, \dots, L/2$ 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 $\mathcal{G}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x^2})$
        Decoder:
        for all layers $l = L/2 + 1, \dots, L$ 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
```

For both FC-VAE and LSTM-VAE, after the training is complete using $\boldsymbol{X}_{\text{TR}}$, the test dataset $\boldsymbol{X}_{\text{TST}}$ is applied. Whenever the cost function that calculates the MSE between the original and reconstructed energy consumption profile is larger than a threshold, a malicious sample is labelled with $y =$ '1', else a benign sample is labelled with $y =$ '0'.

B. Performance Evaluation of the Detectors

TP, TN, FP, and FN define the true positives, true negatives, false positives, and false negatives, respectively. TP refers to a sample that is malicious and detected as malicious. TN indicates a benign sample that is detected as benign. FP means that the sample is benign but detected as malicious. FN represents the malicious sample that is detected as benign. To evaluate the performance of the developed detectors, we use (a) Detection Rate (DR = TP/(TP+FN)) that determines the number of malicious readings that were correctly detected as malicious by the detector. (b) False Alarm (FA = FP/(TN+FP)) refers to the number of benign samples that were incorrectly detected as malicious. (c) Highest Difference (HD = DR - FA), which refers to the measured difference between DR and FA.

The calculated label $\boldsymbol{Y}_{\text{CAL}}$ is compared against $\boldsymbol{Y}_{\text{TST}}$ to produce a confusion matrix in order to calculate the performance evaluation metrics. To compute $\boldsymbol{Y}_{\text{CAL}}$, a threshold is computed based on the median of the interquartile range (IQR) of the receiver operating characteristic (ROC) curve. If a score is less than the threshold value, it represents a benign sample. However, if a score is less than the threshold value, it represents a malicious sample.

Algorithm 2: Training of LSTM-VAE

```
Input Data: $\boldsymbol{X}_{\text{TR}}$
Initialization: Weights $\boldsymbol{U}^l_{(\cdot)}$, $\boldsymbol{W}^l_{(\cdot)}$, $\boldsymbol{V}^l_{(\cdot)}$, and bias $\boldsymbol{b}^l_{(\cdot)}$ $\forall l$
while not converged do
    for each training sample $\boldsymbol{x}$ do
        Feed Forward
        Encoder:
        for each hidden layer $l = 1, \dots, L/2$ do
            for each time step $t$ do
                $\boldsymbol{i}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_i \boldsymbol{x}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_i)$,
                $\boldsymbol{f}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_f \boldsymbol{x}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{E},t-1} + \boldsymbol{b}^l_f)$,
                $\boldsymbol{c}^l_{\text{E},t} = f^l_{\text{E},t} c^l_{\text{E},t-1} + i^l_{\text{E},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t +$
                  $\boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{b}^l_c)$,
                $\boldsymbol{o}^l_{\text{E},t} = \varphi(\boldsymbol{W}^l_o \boldsymbol{x}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{E},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{E},t} + \boldsymbol{b}^l_o)$,
                $\boldsymbol{h}^l_{\text{E},t} = \boldsymbol{o}^l_{\text{E},t} \tanh(\boldsymbol{c}^l_{\text{E},t})$,
            end
            $\boldsymbol{h}'^l = \boldsymbol{h}^l_{\text{E},t}$,
            $\boldsymbol{c}'^l = \boldsymbol{c}^l_{\text{E},t}$.
        end
        Generate $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma^2_x}$:
        $\boldsymbol{\mu_x} = \varphi(\boldsymbol{V}_\mu \boldsymbol{a}^{l-1}(x)) + \boldsymbol{b}^l_\mu$
        $\boldsymbol{\sigma^2_x} = \varphi(\boldsymbol{V}_\sigma \boldsymbol{a}^{l-1}(x)) + \boldsymbol{b}^l_\sigma$
        Sample data $\breve{\boldsymbol{x}}$ from $\mathcal{G}(\boldsymbol{\mu_x}, \boldsymbol{\sigma^2_x})$
        Decoder:
        The decoder hidden and cell states at initial time step are
          equal to $\boldsymbol{h}'$ and $\boldsymbol{c}'$
        for each hidden layer $l = L/2 + 1, \dots, L$ do
            for each time step $t$ do
                $\boldsymbol{i}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_i \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_i \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_i \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_i)$,
                $\boldsymbol{f}^l_{\text{D},t} =$
                  $\varphi(\boldsymbol{W}^l_f \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_f \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_f \boldsymbol{c}^l_{\text{D},t-1} + \boldsymbol{b}^l_f)$,
                $\boldsymbol{c}^l_{\text{D},t} = f^l_{\text{D},t} c^l_{\text{D},t-1} + i^l_{\text{D},t} \tanh(\boldsymbol{W}^l_c \boldsymbol{x}^l_t +$
                  $\boldsymbol{U}^l_c \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{b}^l_c)$,
                $\boldsymbol{o}^l_{\text{D},t} = \varphi(\boldsymbol{W}^l_o \breve{\boldsymbol{x}}^l_t + \boldsymbol{U}^l_o \boldsymbol{h}^l_{\text{D},t-1} + \boldsymbol{V}^l_o \boldsymbol{c}^l_{\text{D},t} + \boldsymbol{b}^l_o)$,
                $\boldsymbol{h}^l_{\text{D},t} = \boldsymbol{o}^l_{\text{D},t} \tanh(\boldsymbol{c}^l_{\text{D},t})$,
            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 $\boldsymbol{U}^l_{(\cdot)}$, $\boldsymbol{W}^l_{(\cdot)}$, $\boldsymbol{V}^l_{(\cdot)}$, and $\boldsymbol{b}^l_{(\cdot)}$ $\forall l$.
```

C. Hyper-parameter Optimization

Optimal choices of the detector hyper-parameters lead to improved detection performance. The hyper-parameters that we optimized are: the number of hidden layers (Dense or LSTM) (L), which is the same for the encoder and decoder layers, the optimal number of neurons in those layers (N_l), the optimizer (O), the dropout rate (D), and the hidden and output activation functions (A_{H} and A_{O}, respectively).

As shown in Algorithm 3, hyper-parameter optimization is carried out through four main sequential steps. Due to the large number of hyper-parameters that we aim to optimize, an exhaustive grid search presents a high computational complexity. Thus, we carried out a sequential grid search by optimizing one hyper-parameter at a time [8]. The motivation behind such an approach is to reduce the computational complexity and improve the overall detection performance. For the selection of the hyper-parameters, we implement a cross-validation over $\boldsymbol{X}_{\text{TR}}$ to decrease the chance of sub-optimality. Let P^* denote the hyper-parameter optimal setting that results in improved detection accuracy against the validation set. A given combination of hyper-parameters leads to a specific model (MD).

Algorithm 3: Hyper-parameter Optimization

```
Initialization: Optimizer = SGD, dropout rate = 0, hidden activation
  = Relu, output activation = Softmax
Output: A combination of optimized hyper-parameters
Input: Training set $\boldsymbol{X}_{\text{TR}}$
for $L \in \mathcal{L}$ do
    for $N_l \in \mathcal{N}$ do
        Algorithms 1 and 2 are applied with $L$ and $N_l$ along with
          other initial hyper-parameters ;
        DR and FA are recorded;
    end
end
The optimal $L^*$ and $N^*_l$ along with initial other hyper-parameters
  introduce model MD1
for $O \in \mathcal{O}$ do
    Algorithms 1 and 2 are applied with MD1's hyper-parameters
      and $o$;
    DR and FA are recorded;
end
$L^*$, $N^*_l$ and $O^*$ along with initial other hyper-parameters introduce
  model MD2
for $D \in \mathcal{D}$ do
    Algorithms 1 and 2 are applied with MD2's hyper-parameters
      and $D$;
    DR and FA are recorded;
end
$L^*$, $N^*_l$, $O^*$, and $D^*$ along with initial other hyper-parameters
  introduce model MD3
for $A_{\text{H} \in \mathcal{A}_{\text{H}}}$ do
    for $A_{\text{O} \in \mathcal{A}_{\text{O}}}$ do
        Algorithms 1 and 2 are applied with MD3's
          hyper-parameters and $A_{\text{H}}$ and $A_{\text{O}}$;
        DR and FA are recorded;
    end
end
$L^*$, $N^*_l$, $O^*$, $D^*$, $A^*_{\text{H}}$, $A^*_{\text{O}}$ are the optimal parameters.
```

IV. Experimental Results

A. Threshold Value

After dividing the ROC curves into three quartiles and computing the IQR's median, the optimal threshold values for the FC-VAE and LSTM-VAE turned out to be 0.43 and 0.47, respectively.

B. Hyper-parameter Optimization

Table I summarizes the optimized hyper-parameter values that are selected from the following sets: number of layers $\mathcal{L} = \{2, 3, 4, 5\}$, number of neurons $\mathcal{N} = \{100, 200, 300, 400, 500\}$, optimizer $\mathcal{O} = \{$SGD, Adam, Adamax, and Rmsprop$\}$, dropout rate $\mathcal{D} = \{0, 0.2, 0.4, 0.5\}$, hidden activation functions $\mathcal{A}_{\text{H}} = \{$ Relu, Sigmoid, Linear, Tanh$\}$, and output activation layer $\mathcal{A}_{\text{O}} = \{$ Softmax, Sigmoid$\}$.

TABLE I
OPTIMAL HYPER-PARAMETER VALUES

Hyper-parameter	**FC-VAE**	**LSTM-VAE**
L^*	8	4
O^*	Adam	SGD
D^*	0.4	0
A^*_{H}	Relu	Tanh
A^*_{O}	Softmax	Sigmoid

TABLE II
PERFORMANCE EVALUATION

Model	**DR**	**FA**	**HD**
FC-VAE	88	11	77
LSTM-VAE	**91**	**7**	**84**
SVM	76	29	47
ARIMA	77	20	57

C. Performance Evaluation

Table II summarizes the performance of the developed detectors. An improvement of 3% in DR, 4% in FA, and 7% in HD was observed when the LSTM-VAE model was used compared to the FC-VAE model. Such an improvement is due to the fact that the LSTM-based model captures better the time-series nature of the energy consumption data [14].

We also compare the performance of the developed deep auto-encoder-based anomaly detectors against the current state-of-the-art anomaly detectors, that are trained on benign data only, including (a) single-class SVM and (b) ARIMA-based anomaly detector that predicts future consumption with minimum prediction MSE. During testing, whenever the MSE is above a threshold, the detector announces a malicious sample. The SVM-based detector represents a static classifier that does not capture the time-series nature of the data. Although the ARIMA model captures the time-series nature of the data, it still represents a shallow architecture that does not capture well the complex patterns within the electricity readings. On the other hand, since the proposed VAEs represent a deep structure that captures the complex patterns and temporal correlations within the electricity consumption data, it outperforms the existing state-of-the-art anomaly detectors. As summarized in Table II, the developed VAE detector improves the DR, FA, and HD by $11 - 15\%$, $9 - 22\%$, and $27 - 37\%$, respectively.

V. CONCLUSION

This paper proposed novel anomaly detectors for electricity theft detection based on variational auto-encoders. The developed anomaly detectors are trained only on benign energy consumption samples, a strategy that overcomes the limitation caused by the reduced number of malicious energy consumption profiles. This paper investigated whether deep architectures offer better detection performance compared to shallow detectors and whether recurrent LSTM-based architectures offer better detection performance compared to static fully connected feed forward-based detectors. Our study revealed a significant improvement when deep and recurrent anomaly detectors are employed compared to shallow and static structures. The best detection performance is achieved by LSTM-VAE with 91% detection rate, 7% false alarm, and 84% highest difference offering an improvement of up to 15%, 22%, and 37% in detection rate, false alarm, and highest difference, respectively, compared to shallow detectors.

REFERENCES

[1] P. Jokar, N. Arianpoo, and V. Leung, "Electricity theft detection in AMI using customers' consumption patterns," *IEEE Trans. Smart Grid*, vol. 7, no. 1, Jan. 2016.

[2] S. McIntyre, Termineter, Jan. 2018. [Online]. Available: https://github.com/securestate/termineter/blob/master/README.md

[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 J. Sel. Topics in Signal Processing*, vol. 12, no. 4, Aug. 2018.

[4] Electric Sector Failure Scenarios and Impact Analyses Version 3.0, National Electric Sector Cybersecurity Organization Resource, Dec. 2015. [Online]. Available: http://smartgrid.epri.com/doc/NESCOR-15.pdf

[5] J. Nagi, K. Yap, S. Tiong, S. Ahmed, and F. Nagi, "Improving SVM-based nontechnical loss detection in power utility using the fuzzy inference system," *IEEE Trans. Power Delivery*, vol. 26, no. 2, Apr. 2011.

[6] V. Krishna, R. Iyer, and W. Sanders, "ARIMA-based modeling and validation of consumption readings in power grids," *10th International Conference on Critical Information Infrastructures Security (CRITIS 2015)*, pp. 199-210, May 2016.

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

[8] I. Goodfellow, Y. Bengio, and A. Courville, 'Deep Learning, *MIT*, 2016.

[9] *Irish Social Science Data Archive*, Online: http://www.ucd.ie/issda/data/commissionforenergyregulationcer/.

[10] H. He, Y. Bai, E. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *IEEE World Congress on Computational Intelligence*, pp. 1322-1328, June 2008.

[11] D. Kingma and M. Welling, "Auto-encoding variational bayes," *2nd International Conf. Learning Representations (ICLR)*, pp. 1-14, 2014.

[12] M. S. Kim, J. P. Yun, S. Lee and P. Park, "Unsupervised Anomaly detection of LM Guide Using Variational Autoencoder," *019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pp. 1-5, 2019.

[13] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *20th Conf. Computational Natural Language Learning (CoNLL)*, pp. 10-21, 2016.

[14] 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," *24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018.

[15] E. Serpedin, T. Chen, and R. Dinesh, *Mathematical Foundations for Signal Processing, Communications and Networks*, CRC Press/Francis & Taylor, 2012,