Machine Learning Based Anomaly Detection for Load Forecasting Under Cyberattacks

Mingjian Cui, *Senior Member, IEEE*, Jianhui Wang, *Senior Member, IEEE*, and Meng Yue, *Member, IEEE*

***Abstract*—Accurate load forecasting can make both economic and reliability benefits for power system operators. However, the cyberattack on load forecasting may mislead operators to make unsuitable operational decisions for the electricity delivery. To effectively and accurately detect these cyberattacks, this paper develops a machine learning based anomaly detection (MLAD) methodology. First, load forecasts provided by neural networks are used to reconstruct the benchmark and scaling data by using the *k*-means clustering. Second, the cyberattack template is estimated by the naive Bayes classification based on the cumulative distribution function and statistical features of the scaling data. Finally, the dynamic programming is utilized to calculate both the occurrence and parameter of one cyberattack on load forecasting data. A widely-used Symbolic Aggregation approXimation (SAX) method is compared with the developed MLAD method. Numerical simulations on the publicly load data show that the MLAD method can effectively detect cyberattacks for load forecasting data with a relatively high accuracy. Also, the robustness of MLAD is verified by thousands of attack scenarios based on Monte Carlo simulation.**

***Index Terms*—Anomaly detection, cyberattack, dynamic programming, load forecasting, machine learning.**

I. Introduction

ACCURACY and correctness of load forecasting data can benefit both the economic and reliable operations of power systems. Power system operators highly depend on the load forecasting information to make operational decisions and plans under various power grid conditions. However, with the rapid adoption of modern technologies and the increasing capability of the attackers in recent years, more and more cyberattacks have been found and severely affect the reliability and security of power systems [1]. A recent representative blackout occurred against the Ukrainian power grid in December 2015 and caused serious power outages [2]. The malware is used by attackers to tamper the computer system of a power company and arbitrarily open breakers [3]. The restoration efforts are also delayed by attackers so as to induce this severe blackout which received worldwide attention [4].

Though current techniques have significantly improved the accuracy of load forecasting data, the operators may still make fallacious decisions when the forecasting data is tampered by highly skilled adversaries in a coordinated manner. From the prospective of time series data, there are three types of cyberattacks for load forecasts, namely point attacks, contextual attacks, and collective attacks [5]. From the prospective of the attackers' capability, there are five types of cyberattacks, namely pulse attack, scaling attack, ramping attack, random attack [6], and smooth-curve attack [7], which are described in the following statements.

The current detection methods mainly rely on identifying anomalies caused by cyberattacks for load forecasting data. Mohammadpourfard *et al*. [8] developed an unsupervised anomaly detection algorithm to identify cyberattacks in power systems that are affected by sustainable energy sources or system reconfigurations. Moghaddass and Wang [9] developed a real-time anomaly detection framework to detect the occurrence of anomalous events and abnormal conditions at both lateral and customer levels. Zhao *et al*. [10] proposed a false data injection detection method based on short-term state forecasting considering the temporal correlation. Chen *et al*. [11] presented a two-stage identification and restoration method for the inaccurate measurement and abnormal disturbance to improve the load forecasting accuracy.

Machine learning techniques have been widely used in the anomaly detection community. Buczak and Guven [12] described a focused literature survey of machine learning methods for cyber security anomaly detection. Ghafoori *et al*. [13] developed a semisupervised machine learning technique to clean suspected anomalies from unlabeled training sets including applications to the datasets of shuttle, breast cancer, human activity recognition, etc. In terms of the power system area, Wang *et al*. [14] trained a machine learning model to detect PMU data manipulation anomalies. Esmalifalak *et al*. [15] developed two machine-learning-based techniques for stealthy attack detection in the smart grid. However, there are very few applications to load forecasting by using machine learning techniques as detection methods.

As a widely-used anomaly detection method, a heuristically ordered time series based Symbolic Aggregation approXimation (SAX) [16] intends to identify the most unusual discords or sub-sequences in a sequence of given load forecasting data [7]. Though the SAX method performs consistently well to detect anomalies, it can raise more false alarm which may still confuse the practitioners for the application. In addition, since SAX detects an anomaly as the sub-sequence of load forecasts, it cannot provide any detailed information into this identified sub-sequence which is more helpful for practitioners, such as the specific occurrence and parameter of one cyberattack.

To bridge the gap between the SAX detection method and an informed method, this paper seeks to address two critical questions for load forecasting cyberattacks. Is it possible to determine the accurate start- and end-time information of one attack? Can the practitioners estimate the attack parameter if it is a parametric attack? To this end, this paper develops a novel machine learning based anomaly detection (MLAD) method to effectively identify cyberattacks for load forecasting data. This developed methods aims to improve the probability

M. Cui and J. Wang are with the Department of Electrical Engineering at Southern Methodist University, Dallas, TX, 75275 USA (email: {mingjiancui, jianhui}@smu.edu).

M. Yue is with the Department of Sustainable Energy Technologies, Brookhaven National Laboratory, Upton, NY, 11973 USA (email: yuemeng@bnl.gov).

Manuscript received, 2018.

and success ratio of detection. The main contributions of this paper include: (i) determining the attack template by using a supervised machine learning method based on the reconstructed scaling data and (ii) estimating the specific occurrence and parameter information of one cyberattack.

The organization of this paper is as follows. In Section II, the templates of cyberattacks and load forecasts are briefly introduced. Section III presents the detailed methodology of MLAD, including data reconstruction in Section III-A, template determination in Section III-B, and dynamic programming in Section III-C. Section IV describes the evaluation metrics to validate the effectiveness of MLAD. Case studies and result analysis performed on the publicly load data are discussed in Section V. Concluding remarks are summarized in Section VII.

II. Templates of Cyberattacks of Load Forecasts

A. Research Motivations

Load forecasting results are highly needed by power system operators and/or market participants to project upcoming grid conditions and make informed operational decisions. However, in recent years, there is a lack of understanding of how adversaries perform cyberattacks on load forecasting data and impact evaluations accordingly. As the cyber adversaries are increasingly skillful and sophisticated, it is more challenging to detect and mitigate those attacks that can do serious harm to power system operations. Thus, it is important to identify load forecasts tampered with by cyber adversaries before mitigation can be done [17].

B. Cyberattack Templates

In this section, we are not aiming to develop new adversary models of cyberattack templates. The adversary models used in this paper are motivated by the particular adversary models referred in [6] and [7]. Inspired by existing adversary models for attacking automatic generation control (AGC), we assume that smart attackers could migrate these adversary models to those on load forecasting data in this paper. The cyber attacks for load forecasting are divided into five categories: pulse, scaling, ramping, random, and smooth-curve [6], [7], which are briefly described as follows. Note that this paper does not aim to develop new attack templates.

1) Pulse Attack: Load forecasts are modified to higher/lower values at a specific point during the entire duration of an attack. The attack parameter is set as λ_{P}.

$$\dot{p}_t^{\mathrm{F}} = (1+\lambda_{\mathrm{P}}) \times p_t^{\mathrm{F}}, \quad \text{for} \quad t = t_{\mathrm{P}} \tag{1}$$

where t_{P} is the occurrence time of one pulse attack. p_t^{F} is the original load forecast that is not tampered with any cyber attack. $\dot{p}_t^{\mathrm{F}}$ is the load forecast tampered with cyber attacks.

2) Scaling Attack: Scaling attacks involve modifying the values in a specified duration multiplied by a scaling attack parameter λ_{S}.

$$\dot{p}_t^{\mathrm{F}} = (1+\lambda_{\mathrm{S}}) \times p_t^{\mathrm{F}}, \quad \text{for } t_s < t < t_e \tag{2}$$

where t_s and t_e represent the start- and end-time of one cyber attack, respectively.

3) Ramping Attack: There are two types of ramping attacks. Type I ramping attack only considers up-ramping anomaly. The values in the specified range are multiplied by a ramping function $\lambda_{\mathrm{R}} t$.

$$\dot{p}_t^{\mathrm{F}} = \lambda_{\mathrm{R}} \times (t - t_s) \times p_t^{\mathrm{F}}, \quad \text{for } t_s < t < t_e \tag{3}$$

Type II ramping attack considers both up- and down-ramping anomalies. This attack is more challenging to detect for operators.

$$\dot{p}_t^{\mathrm{F}} = [1 + \lambda_{\mathrm{R}} \times (t - t_s)] \times p_t^{\mathrm{F}}, \quad \text{for } t_s < t < \lfloor \frac{t_s + t_e}{2} \rfloor \tag{4}$$

$$\dot{p}_t^{\mathrm{F}} = [1 + \lambda_{\mathrm{R}} \times (t_e - t)] \times p_t^{\mathrm{F}}, \quad \text{for } \lfloor \frac{t_s + t_e}{2} \rfloor < t < t_e \tag{5}$$

where $\lfloor \cdot \rfloor$ indicates the floored value which is used to present the approximate intermediate point between t_s an t_e.

4) Random Attack: This attack involves the addition of positive values returned by a uniform random function to load forecasts.

$$\dot{p}_t^{\mathrm{F}} = p_t^{\mathrm{F}} + \lambda_{\mathrm{RA}} \times \mathsf{rand}\,(t)\,, \quad \text{for } t_s < t < t_e \tag{6}$$

where rand is a uniformly distributed random number generator that can be achieved by a built-in function in MATLAB. λ_{RA} is a scale factor and defined as half of the maximum of load forecast value, i.e., $\lambda_{\mathrm{RA}} = \max\left(p_t^{\mathrm{F}}\right)/2$. The start- and end-time of one random attack is assumed to be randomly set by attackers.

5) Smooth-Curve Attack: Smooth-curve attacks are implemented by replacing the set of contiguous start and end points in the original forecasting data. In this paper, a polynomial fitting is used to generate a smooth curve and replace the original forecasting data with neighboring points.

C. Load Forecasts

There have been large amounts of load forecasting methods published in the current literature. However, it is very challenging to take advantage of each of them. Alternatively, if we can find a representative forecasting method that can be easily implemented by users and validate the effectiveness of this generic method for anomaly detection, advanced load forecasting methods must also be applicative. Based on this motivation, we choose a widely-used neural network (NN) method to perform load forecasting [18]. Note that this paper does not aim at developing new methods of load forecasting, which has been done in most papers. The input data sets of NN include: temperature and dew point (measuring humidity) forecast at time t+1, the hour number of time t+1 in the day (i.e., hour 1, 2, $\cdots$, 24), days of the week (Monday through Sunday are presented by number 1, 2, $\cdots$, 7), working days (yes or no), the load at the same hour of the previous day, the load at the same hour of the previous week, and the average load in the 24 hours prior to time t+1. When cyber attackers tamper the essential input and output data of load forecasting, different attack templates may present depending on the capability of attackers, which has been described in Section II-B.

III. Machine Learning Based Anomaly Detection

The developed MLAD method mainly consists of three steps. The first step is to reconstruct the benchmark data corresponding to load forecasts. The second step is to determine the exact attack template based on the series of scaling data. The third step is to identify the specific occurrence and parameter information of one cyberattack.

A. Unsupervised Machine Learning Based Load Data Preprocessing

After generating the load forecasts mentioned in Section II-C, it is still challenging for operators to know whether these forecasts are attacked since the real load data is still unknown for the coming days (weeks or months). Thus, a benchmark series of load data is first reconstructed corresponding to the predicted load data. Since load data cannot be labeled with natural groupings and patterns, the data reconstruction is attributed to an unsupervised machine learning problem. There are several types of unsupervised machine learning algorithms, such as *k*-means, mixture models, and hierarchical clustering. Compared with mixture models and hierarchical clustering, the *k*-means clustering method is easy to implement [19]. With a large number of real load data, the *k*-means clustering method is computationally faster than the hierarchical clustering and mixture models. In addition, it can also produce tighter clusters than other unsupervised algorithms. Hence, the *k*-means clustering method is chosen as the unsupervised machine learning algorithm in this section and used to reconstruct the benchmark load data.

Given the training set of 24-dimensional daily load $(\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_j, \cdots, \mathbf{p}_n)$ in n days, the *k*-means clustering can partition n daily load into k load cluster sets, i.e., $\mathbb{S} = (\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_i, \cdots, \mathbf{s}_k)$. The objective is to minimize the within-cluster sum of squares [20], given by:

$$\min_{\mathbb{S}} \sum_{i=1}^{k} \sum_{\mathbf{p}_j \in \mathbf{s}_i} \|\mathbf{p}_j - \mu_i\|^2 \tag{7}$$

where μ_i is the mean of load data in the ith load cluster set $\mathbf{s}_i$ and $i = 1, 2, \cdots, k$. k is the total number of load cluster sets, which is predefined by operators. $\mathbf{p}_j$ is the load data vector in the jth day and $j = 1, 2, \cdots, n$. n is the total number of training days. The denotation of $\mathbf{p}_j \in \mathbf{s}_i$ means that the jth day's load data vector $\mathbf{p}_j$ is the element of the ith load cluster set $\mathbf{s}_i$.

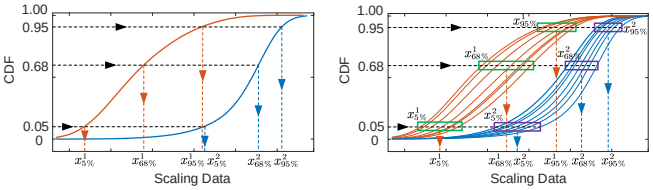
Based on k clusters, the closest cluster is found for each load forecast scenario by minimizing the Euclidean distance, given by:

$$\mathbf{p}_j^{\mathrm{B}} = \min\left(\|\mathbf{p}_j^{\mathrm{F}} - \mathbf{s}_i\|\right) \tag{8}$$

where $\begin{bmatrix}\mathbf{p}_1^{\mathrm{B}} & \mathbf{p}_2^{\mathrm{B}} & \cdots & \mathbf{p}_n^{\mathrm{B}}\end{bmatrix}^{\mathrm{T}}$ is the reconstructed benchmark series, and $p_t^{\mathrm{B}} \in \begin{bmatrix}\mathbf{p}_1^{\mathrm{B}} & \mathbf{p}_2^{\mathrm{B}} & \cdots & \mathbf{p}_n^{\mathrm{B}}\end{bmatrix}^{\mathrm{T}}$. Based on the load forecasts and the reconstructed benchmark, the scaling data x_t at time t can be calculated as:

$$x_t = p_t^{\mathrm{F}} / p_t^{\mathrm{B}} \tag{9}$$

where p_t^{F} is generated by the NN method in Section II-C.



(a) Single CDF curves (b) CDF clusters

Fig. 1. An example of three statistical features ($x_{5\%}$, $x_{68\%}$, and $x_{95\%}$) of two CDFs of scaling data. Orange curves: data with scaling attacks; blue curves: data with ramping attacks.

The scaling data is used to determine the specific attack template in the second step, as shown in Section III-B, and detect the accurate information of one attack in the third step, as shown in Section III-C.

B. Supervised Machine Learning Based Cyber Attack Templates Classification

Since cyber attacks are classified into five templates (or classes) with clear labels, this is taken as a supervised machine learning (classification) problem. There are several types of supervised machine learning algorithms, such as linear discriminant, logistic regression, perceptron, etc. The naive Bayes classifier is easy to implement with a low model size [21]. It requires a small amount of training load data with cyber attacks to estimate its parameters. A naive Bayes classifier can converge more quickly than discriminative models, which significantly shortens the training time. Also, it is not sensitive to irrelevant features. Thus, a naive Bayes classifier is chosen as the supervised machine learning algorithm in this section and trained by the measured statistical characteristics of the scaling data. Based on the widely used 68–95–99.7 rule (three-sigma rule) [22], five statistical features are adapted as predictors of the multi-class naive Bayes model, i.e., $x_{0.3\%}$, $x_{5\%}$, $x_{68\%}$, $x_{95\%}$, and $x_{99.7\%}$. The cumulative distribution function (CDF) of the scaling data is used to generate main statistical features due to its monotonicity. Fig. 1 shows an example of three statistical features ($x_{5\%}$, $x_{68\%}$, and $x_{95\%}$) of two CDFs of the scaling data. Fig. 1a shows how the scaling values in the x-axis are generated from single CDF curves for scaling attack (orange curve) and ramping attack (blue curve). Fig. 1b shows how the training samples of three representative statistical features are generated from CDF clusters, where CDFs with scaling attacks (orange curves) and ramping attacks (blue curves) are taken as an example. Essentially, the discrete statistical features are used to approximately represent the continuous CDF curves. Thus, the naive Bayes classifier only has five input variables (i.e., five statistical features of CDF curves: $x_{0.3\%}$, $x_{5\%}$, $x_{68\%}$, $x_{95\%}$, and $x_{99.7\%}$), which makes it easy to implement. To quantitatively present the difference of CDF curves with different attacks, some samples of statistical features for five types of cyber attacks are shown in Table I. As can be seen, the difference of statistical features with the same type of attacks is significantly small, while it differs between different types of attacks, such as pulse and ramping attacks. Thus, this observation can be used to train the naive Bayes classifier and thereby determine the specific template used to launch a cyber attack on the testing data. As can be seen, these

TABLE I
SOME SAMPLES OF STATISTICAL FEATURES FOR FIVE CYBER ATTACKS

Examples of Cyber Attacks	Statistical Features as Predictors of the Naive Bayes Classifier				
	$x_{0.3\%}$	$x_{5\%}$	$x_{68\%}$	$x_{95\%}$	$x_{99.7\%}$
Pulse	0.9088	0.9435	1.0098	1.0524	1.1556
Pulse	0.9089	0.9434	1.0099	1.0525	1.1555
Ramping	0.9312	0.9572	1.0183	1.6486	2.2378
Ramping	0.9313	0.9573	1.0189	1.6413	2.2574
Scaling	0.9119	0.9446	1.0153	1.1962	1.2347
Scaling	0.9117	0.9442	1.0169	1.1938	1.2165
Random	0.9276	0.9541	1.0188	1.4024	1.6326
Random	0.9285	0.9598	1.0173	1.2362	1.7018
Smooth Curve	0.9291	0.9503	1.0145	1.0679	1.4309
Smooth Curve	0.9279	0.9474	1.0148	1.0586	1.4107

statistical features have different values in the x-axis and can be used as predictors of the naive Bayes model. To accurately characterize the irregular and multimodal distribution of the scaling data, Gaussian mixture model (GMM) is used to fit the distributions [23], and generates both the CDFs and statistical features. Based on predictors generated by the CDF of GMM, the naive Bayes classifier is constructed from the probability model with the objective of the maximum posterior probability $\hat{P}(\cdot)$, given by:

$$\begin{aligned}&\arg\max_{a\in\mathbf{\Lambda}} \hat{P}\left(A=a|x_{0.3\%},x_{5\%},x_{68\%},x_{95\%},x_{99.7\%}\right)\\&=\frac{\pi\left(A=a\right)\prod_{j\in\mathbf{\Phi}}P\left(X=x_j|A=a\right)}{\sum_{a\in\mathbf{\Lambda}}\pi\left(A=a\right)\prod_{j\in\mathbf{\Phi}}P\left(X=x_j|A=a\right)} \qquad (10)\\&\Rightarrow \arg\max_{a\in\mathbf{\Lambda}}\pi\left(A=a\right)\prod_{j\in\mathbf{\Phi}}P\left(X=x_j|A=a\right) \qquad (11)\end{aligned}$$

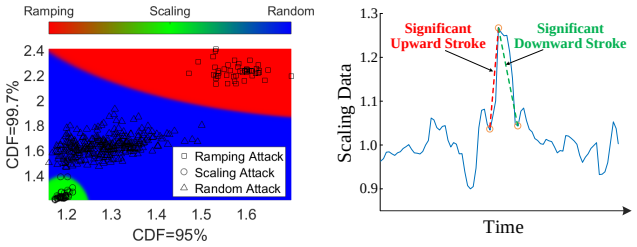
$$\mathbf{\Lambda}=\{Pulse, Scaling, Ramping, Random, Smooth\} \quad (12a)$$

$$\mathbf{\Phi}=\{0.3\%, 5\%, 68\%, 95\%, 99.7\%\} \quad (12b)$$

where $\mathbf{\Phi}$ is the set of statistical features and $\mathbf{\Lambda}$ is the set of attack templates. $\pi(A=a)$ is the prior probability of the attack template a. $P\left(X=x_j|A=a\right)$ is the conditional probability of the jth statistical feature x_j given attack template a. Since values of the feature x_j and labels of the attack template a are all given, the denominator in (10) is effectively constant. Hence, the objective function is equivalent to only maximizing the numerator in (10), which is the joint probability model in (11). Fig. 2a shows an example of the posterior probability regions for three cyberattacks using the naive Bayes classifier. As can be seen, the rectangles (ramping attacks), triangles (random attacks), and circles (scaling attacks) can be distinctly classified by using the naive Bayes classifier.

C. Dynamic Programming

1) Methodology Description: Dynamic programming method is used to detect the exact occurrence (start and end points) of one cyber attack and its specific parameters by solving an objective function. The simultaneous detection of such information is much challenging by using other methods. For example, the signal processing methods, such as wavelet decomposition [24] and empirical model decomposition [25], can only roughly detect the occurrence of a disturbance. The parameters of cyber attacks, such as the scaling attack parameter λ_{S} and the ramping attack parameter λ_{R}, remain unidentified by practitioners. Essentially, these methods transform the original data to a predefined metric, such as the



(a) (b)

Fig. 2. Examples for illustration. (a) An example of three statistical features ($x_{5\%}$, $x_{68\%}$, and $x_{95\%}$) of two CDFs of scaling data. Orange curves: data with scaling attacks; blue curves: data with ramping attacks. (b) An example of the scaling data for the ramping attack template.

normalized wavelet energy (NWE) [26], based on a relatively narrow sliding window. During this transformation process, the detailed information of the original data may be lost. Thus, to detect both the occurrence and parameters of a cyber attack, dynamic programming method is chosen and briefly introduced in this section. Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems [27]. The time intervals must comply with the anomaly rules described as follows.

The anomaly rule under cyberattacks is predefined by practitioners and required for the developed MLAD method based on the scaling data. Fig. 2b shows an example of the scaling data for the ramping attack template. Generally, one cyber attack consists of one upward stroke (the red dash line) and one downward stroke (the green dash line). For pulse, scaling, and ramping (Type II) attacks, there are one significant upward stroke (SUS) and one significant downward stroke (SDS). For random and smooth-curve attacks, there may exist multiple upward and downward strokes. Given that the total number of SUS and SDS is M, the key of the MLAD method is to detect the initial SUS (ST_1) and the terminal SDS (ST_M). Assuming that the set of strokes is $\mathbb{S}=\{ST_1,\cdots,ST_m,\cdots,ST_M\}$, where $ST_m=(s_m,e_m)$ represents the mth significant stroke with the corresponding start point (s_m) and end point (e_m), the magnitude rule R_{mag} checks whether the scaling data has increased (or decreased) by a specified threshold Tr_{mag}, and defined as:

$$R_{\mathrm{mag}}=1, \quad if \quad |x_{s_m}-x_{e_m}|>Tr_{\mathrm{mag}} \qquad (13)$$

where x_{s_m} and x_{e_m} indicate the scaling data at time s_m and e_m, respectively.

Based on the predefined anomaly rule, time intervals that satisfy the magnitude rule are rewarded by a score function; otherwise, their score is set to zero. An increasing length score function S is designed based on the length of time intervals. Given a time interval (i, j) of discrete time points of scaling data and a time point k located into this interval (i.e., $i<k<j$), the score function should conforms to a super-additivity property, given by:

$$S(i,j)>S(i,k)+S(k,j), \quad \forall k: i<k<j \qquad (14)$$

There are a family of score functions that can satisfy this property. In this paper, the score function presented in [28], [29] is adopted and given by:

$$S(i,j)=(i-j)^2\times R_{\mathrm{mag}}(i,j) \qquad (15)$$

where $R(i,j)$ represents the magnitude rule in (13). Then, an objective function J is constituted according to the dynamic programming, given by:

$$J(i,j) = \max_{i<k<j} [S(i,k) + J(k+1,j)] \tag{16}$$

Based on (14)–(16), the process of solving the optimization problem can proceed recursively as follows. Time intervals of the scaling data under normal operations without any significant strokes are $\overline{\mathbb{S}} = \{\overline{ST_1}, \cdots, \overline{ST_m}, \cdots, \overline{ST_M}\}$, where $\overline{ST_m}$ indicate the mth non-stroke and $\overline{ST_m} = (\overline{s_m}, \overline{e_m})$. For the mth non-strokes, the magnitude rule, score function, and objective function of the dynamic programming can respectively be calculated as:

$$R_{\text{mag}}(i,j) = 0, \quad \forall i,j: \ \overline{s_m} < i < j < \overline{e_m} \tag{17}$$

$$S(i,j) = 0, \quad \forall i,j: \ \overline{s_m} < i < j < \overline{e_m} \tag{18}$$

$$J^*(\overline{s_m}, \overline{e_m}) = 0, \quad \forall m: \ 1 \le m < M \tag{19}$$

For the mth time interval with SUS or SDS, i.e., $ST_m = (s_m, e_m)$, the magnitude rule, score function, and objective function of the dynamic programming can respectively be calculated by:

$$R_{\text{mag}}(i,j) = 1, \quad \forall i,j: \ s_m < i < j < e_m \tag{20}$$

$$S(i,j) = (i-j)^2, \quad \forall i,j: \ s_m < i < j < e_m \tag{21}$$

$$\begin{aligned} J^*(s_m, e_m) &= \max_{s_m<k_1<e_m} S(s_m,k_1) + J(k_1+1, e_m) \\ &= \max_{s_m<k_1<e_m} S(s_m,k_1) + \max_{k_1+1<k_2<e_m} S(k_1+1,k_2) \\ &\quad + \cdots + \max_{k_{i-1}+1<k_i<e_m} S(k_{i-1}+1,k_i) + J(k_i+1,e_m) \\ &= \max_{s_m<k_1<k_2<\cdots<k_{i-1}<k_i<e_m} S(s_m,k_1) + S(k_1+1,k_2) \\ &\quad + \cdots + S(k_{i-1}+1,k_i) + S(k_i,e_m) \end{aligned} \tag{22}$$

Assuming that a given scaling data series $\{x_{\overline{s_1}}, \cdots, x_{\overline{s_m}}, \cdots, x_{s_1}, \cdots, x_{s_m}, \cdots, x_{\overline{e_M}}\}$ starts without strokes at the beginning and can be presented as $\boldsymbol{\Theta} = \{\overline{ST_1}, \cdots, \overline{ST_m}, ST_1, \cdots, ST_m, \cdots, \overline{ST_M}\}$, the solution to (16), $J^*(\overline{s_m}, \overline{e_M})$, for the mth compression interval without strokes is obtained by the recursive process using the dynamic programming until $k_i = e_M - 1$. Considering (18) and (19), the objective $J^*(\overline{s_m}, \overline{e_M})$ can be transformed to the objective $J^*(s_{m+1}, \overline{e_M})$ of the $(m+1)$th compression interval with strokes, given by:

$$\begin{aligned} J^*(\overline{s_m}, \overline{e_M}) &= \max_{\overline{s_m}<k_1<k_2<\cdots<k_{i-1}<k_i<\overline{e_M}} J(k_i, \overline{e_M}) \\ &= J^*(s_{m+1}, \overline{e_M}) \end{aligned} \tag{23}$$

Considering the super-additivity in (14), the final detected anomalies of scaling data $\{x_{\overline{s_1}}, \cdots, x_{\overline{s_m}}, \cdots, x_{s_1}, \cdots, x_{s_m}, \cdots, x_{\overline{e_M}}\}$ is solved as:

$$J^*(\overline{s_1}, \overline{e_M}) = \sum_{m=1}^{M} S(s_m, e_m) \tag{24}$$

Finally, the application of dynamic programming will yield the set of SUS and SDS of cyberattacks for load forecasting data, i.e., $\mathbb{S} = \{ST_1, \cdots, ST_m, \cdots, ST_M\}$.

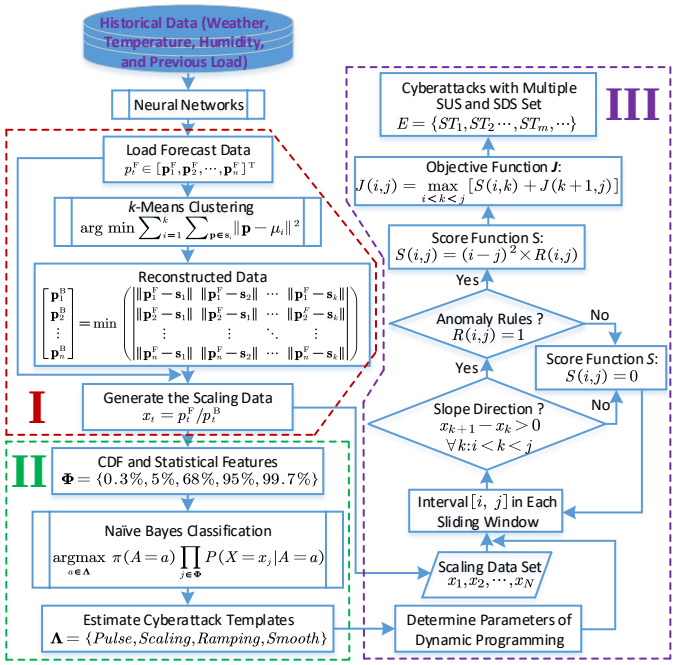


Fig. 3. Flowchart of the developed MLAD methodology. Part I: unsupervised machine learning (k-means clustering); Part II: supervised machine learning (naive Bayes classification); and Part III: dynamic programming.

2) Discussion of Computational Complexity Reduction: To efficiently solve the dynamic programming based problem, the predefined magnitude rule R_{mag} in (13) can significantly reduce the computational complexity of dynamic programming. This is because both the score function and the objective function of time intervals (or sub-intervals) that cannot conform to the magnitude rule R_{mag} are calculated as zero, which are formulated in (17)–(19). That is to say, only these time intervals that can conform to the magnitude rule R_{mag} can be assigned with a specific nonzero score by the score function that is formulated in (20)–(22). Finally, only very few intervals with strokes (usually one or two strokes) are assigned with a score, namely the cyber attacks to be detected. Though the number of steps k may be relatively large, the number of cyber attacks is significantly small. It means that most of time intervals without cyber attacks are assigned with zero values. Thus, during this recursive process of dynamic programming, its computational complexity can be significantly reduced.

3) Determination of Magnitude Threshold: The magnitude threshold Tr_{mag} in (13) can be automatically determined from the historical load dataset under the normal condition. First, practitioners can readily gather the maximum magnitude of increment $M_{\max}^{\text{Norm}}$ from the normal scaling dataset. However, the maximum magnitude $M_{\max}^{\text{Norm}}$ may change along with the corresponding normal scaling dataset that practitioners choose. Thus, a tolerance value is added and defined as a small proportion of the maximum magnitude $M_{\max}^{\text{Norm}}$ of the normal scaling dataset. Finally, we can get the formulation of the magnitude threshold, given by:

$$Tr_{\text{mag}} = M_{\max}^{\text{Norm}} + \underbrace{\phi_{\text{mag}} \times M_{\max}^{\text{Norm}}}_{Tolerance\ Value} \tag{25}$$

where ϕ_{mag} is the tolerance coefficient of the magnitude threshold. Based on the experimental experience, ϕ_{mag} can

be chosen in the range of 10%–20%. This formulation can guarantee that the normal increment of load scaling data is not contained by the defined magnitude threshold Tr_{mag}.

D. Procedure of the Developed MLAD Method

To accurately detect the occurrence of cyberattacks, the flowchart of the developed MLAD methodology is shown in Fig. 3. Three major steps are briefly summarized, including:

- ***Step 1***: The load forecast data p_t^{F} is generated by NN and put into the *k*-means clustering model, which has been trained by the training set of load data, to reconstruct the benchmark data p_t^{B} and the scaling data x_t.
- ***Step 2***: Statistical features of the scaling data are used as inputs of the naive Bayes classifier to determine the specific template of one attack. For each estimated attack template, parameters of the dynamic programming have been predefined by practitioners.
- ***Step 3***: Dynamic programming is used to estimate the final occurrence and parameter of one cyber attack by maximizing the objective function. Evaluation metrics introduced in the following section are calculated to evaluate the detection performance.

IV. Evaluation Metrics of Detection Performance

A. Metrics I: Numerical Detection Errors

To numerically evaluate the performance of different detection methods for cyberattacks, two metrics are used for comparison, namely mean absolute percentage error (MAPE) and root mean square error (RMSE), and given by:

$$MAPE = \frac{1}{N_{\text{S}}} \sum_{i=1}^{N_{\text{S}}} \left| \frac{A_i - D_i}{A_i} \right| \times 100\% \tag{26}$$

$$RMSE = \frac{1}{D_{\max}} \sqrt{\frac{1}{N_{\text{S}}} \sum_{i=1}^{N_{\text{S}}} (A_i - D_i)^2} \times 100\% \tag{27}$$

where N_{S} is total number of possible attack scenarios. A_i and D_i is the actual and detected information (start-time, end-time, and attack parameters) of the ith attack scenario, respectively. Smaller MAPE and RMSE indicate that the corresponding method produces more accurate detected information of cyberattacks.

B. Metrics II: Visualized Detection Performance

Table II provides a measure of skill based on a contingency table [30] for comparison. Assuming that sets of Ω_1, Ω_2, Ω_3, Ψ_1, Ψ_2, and Ψ_3 are shown in Fig. 4, where $\Omega_1 : t \in t_s \pm \phi_1$; $\Omega_2 : t \in (t_s - \phi_2,\ t_s - \phi_1)$; $\Omega_3 : t \in (t_s + \phi_1, t_s + \phi_2)$; $\Psi_1 : t \in t_e \pm \phi_1$; $\Psi_2 : t \in (t_e + \phi_1,\ t_e + \phi_2)$; $\Psi_3 : t \in (t_e - \phi_2,\ t_e - \phi_1)$. t_s and t_e are the real start- and end-time of one attack respectively, and $t_s \in \Omega$, $t_e \in \Phi$. True positive (TP) is the number of detected attacks of which both the start- and end-time are within a smaller tolerance ϕ_1, and $TP \in \Omega_1 \cap \Psi_1$, which means the start-time must be located into Ω_1 and the end-time must be located into Ψ_1; false positive (FP) is the number of detected attacks of which the start-time is pre-detected or the end-time is post-detected within tolerance values, and $FP \in (\Omega_1 \cap \Psi_2) \cup (\Omega_2 \cap \Psi_1) \cup (\Omega_2 \cap \Psi_2)$; false negative (FN) is the number of detected attacks of which the start-time is post-detected or the end-time is pre-detected within tolerance values, and $FN \in (\Omega_1 \cap \Psi_3) \cup (\Omega_3 \cap \Psi_1) \cup (\Omega_3 \cap \Psi_3)$; and true negative (TN) is the number of attacks that are inaccurately detected in excess of a larger tolerance ϕ_2, and $TN = N_{\text{S}} - TP - FP - FN$. FP attacks can cause false alarms for users with redundant operations. FN attacks are missed by the detection method with insufficient operations for users. The performance diagram is visualized by metrics including the probability of detection (POD), critical success index (CSI), frequency bias score (FBIAS), and success ratio (SR), given by:

TABLE II
Contingency Table for Real and Detected Attacks

	Real (YES)	Real (NO)	Total
Detected (YES)	TP(hit)	FP(false alarm)	TP+FP
Detected (NO)	FN(miss)	TN(inaccurate)	FN+TN
Total	TP+FN	FP+TN	N_{S}=TP+FP+FN+TN

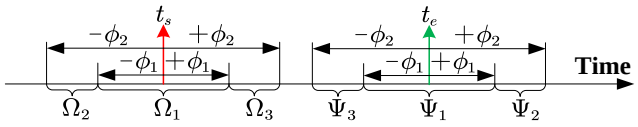


Fig. 4. Start- and end-time of one real attack with different tolerances.

$$POD = TP/\left(TP + FN\right) \tag{28}$$

$$CSI = TP/\left(TP + FN + FP\right) \tag{29}$$

$$FBIAS = \left(TP + FP\right)/\left(TP + FN\right) \tag{30}$$

$$SR = TP/\left(FP + TP\right) \tag{31}$$

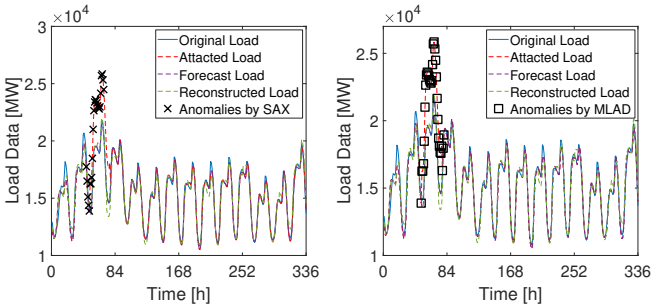
Detailed information about the performance diagram and metrics can be found in [31].

V. Case Studies and Results

The raw load data are obtained directly from ISO New England [32]. Hourly load data is sampled on the NEPOOL region (courtesy ISO New England) from 2004 to 2007 and tested on out-of-sample data from 2008. The pulse, scaling, ramping (Type II only), random, and smooth-curve attack templates are used to tamper the load data on the sliding window of 14 days (336 points) in the test data. The SAX method developed in [16] is chosen to compare the performance of the developed MLAD method. Different sliding window sizes (n_s) of SAX are set from 24 to 34 hours. The number of symbols of SAX is set as 4. The cluster number k is set as 100. For the simplicity of comparison, three representative versions of SAX, i.e., SAX-I, SAX-II, and SAX-III, are defined as benchmark methods to validate the effectiveness of the developed MLAD method. Parameters of three benchmark SAX methods are shown in Table III.

TABLE III
PARAMETERS OF THREE BENCHMARK SAX METHODS

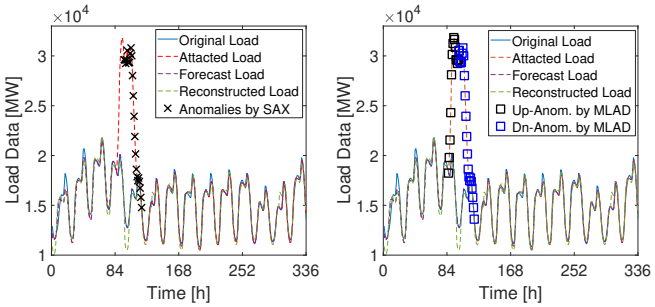
Benchmark SAX Methods	SAX-I	SAX-II	SAX-III
Sliding Window Size n_s [h]	30	32	34



(a) Detected by SAX (n_s=24)

(b) Detected by MLAD

Fig. 5. Scaling attack for comparing SAX (a) with MLAD (b).



(a) Detected by SAX (n_s=24)

(b) Detected by MLAD

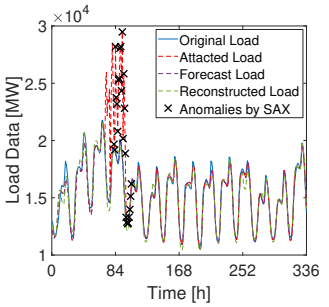
Fig. 6. Ramping attack for comparing SAX (a) with MLAD (b).

A. Performance Evaluation of The Proposed Method

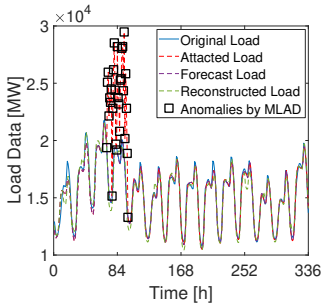
Fig. 5–Fig. 7 compare the detection results of multiple representative cyberattacks using SAX (left column) and MLAD (right column). Fig. 5 compares the detection results of one scaling attack. The latter part of the scaling attack cannot be detected by SAX. Fig. 6 compares the detection results of one Type II ramping attack. The front part of the scaling attack cannot be detected by SAX. Fig. 7 compares the detection results of one random attack. Similar with the scaling attack, SAX cannot detect the front part of the random attack. Overall, SAX cannot accurately detect a complete cyberattack. This is because the SAX method only identifies a sub-sequence of the sliding window size (n_s) as anomalies, which may always include some normal data points and neglect some anomaly data points. The MLAD method can accurately detect cyberattacks for load forecasting data. Another interesting finding is that MLAD is also capable of detecting both the up- and down-ramping anomalies for the Typer II ramping attack as seen in Fig. 6b. The up-ramping anomalies are marked with black rectangles, and down-ramping anomalies are marked with blue rectangles.

B. Robustness Analysis Using Monte Carlo Simulation

The Monte Carlo simulation coupled with Latin hypercube sampling (LHS) [33] is used for the robustness analysis of the developed MLAD method. LHS is often used to construct computer experiments for Monte Carlo integration. First, LHS generates five positive integer $\lceil lhs(5) \rceil$, where $\lceil \cdot \rceil$ indicates rounding towards plus infinity. Each integer represents



(a) Detected by SAX (n_s=24)



(b) Detected by MLAD

Fig. 7. Random attack for comparing SAX (a) with MLAD (b).

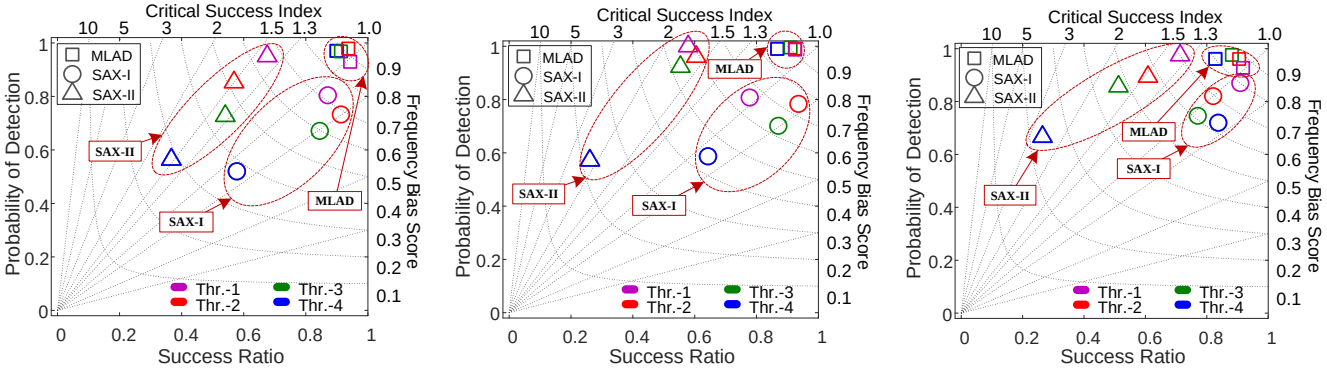
a cyberattack template: 1→pulse, 2→scaling, 3→ramping, 4→random, and 5→smooth-curve. A cyberattack is chosen by randomly sampling a positive integer. Second, for each attack scenario, an attack randomly tampers the load forecasting data at any time in 14 days (336 points). The total number of attack scenarios is set as 3,000. Table IV shows the accuracy rate of detected attack templates using naive Bayes classification. For pulse attacks, 98.92% of real pulse attacks are accurately detected and 1.08% of those are confused with smooth curve attacks. For scaling attacks, 93.46% of real scaling attacks are accurately detected and 6.54% of those are confused with random attacks. For ramping attacks, 100% of real ramping attacks are accurately detected. For random attacks, 96.28% of real random attacks are accurately detected; 0.53% of those are confused with scaling attacks; and 3.19% of those are confused with ramping attacks. For smooth-curve attacks, 76.31% of real smooth-cure attacks are accurately detected and 23.69% of those are confused with random attacks. As can be seen, smooth-curve attacks are the most challenging to detect. This is mainly because this attack template is very secretive and presents a smooth curve together with neighboring data points at the beginning and end of the load forecasting data.

Table V compares the MAPE values of the SAX and MLAD methods for cyberattacks' occurrence. Three types of SAX methods with the best performance are used for comparison: SAX-I (n_s=30), SAX-II (n_s=32), and SAX-III (n_s=34). As can be seen, SAX cannot be used to detect any pulse attacks as their ultrashort discretized duration (1 point) does not significantly affect the relatively long sliding window (28∼32 points). This similar phenomenon has also been verified in [7]. However, MLAD can accurately detect the occurrence of pulse attacks with the smallest MAPE value of 0.005%, which is mainly due to the confusion of smooth-curve attacks shown in Table IV. For scaling, ramping, and random attacks, MLAD can detect the start- and end-time of cyberattacks with the smallest MAPE (1%∼3%), compared with three types of SAX methods (8%∼32%). For the smooth-curve attack, MLAD can still work better than SAX methods, though the MAPE value of MLAD is relatively larger than that for other attacks. Also, this observation verifies that smooth-curve attacks are the most challenging to detect since the start- and end-time of these attacks are disguised very well.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSG.2018.2890809, IEEE Transactions on Smart Grid

TABLE IV
DETECTED ATTACK TEMPLATES IN 3,000 SCENARIOS USING NAIVE BAYES CLASSIFICATION

Real Attack Templates	Detected Attack Templates				
	Pulse Attack	Scaling Attack	Ramping Attack	Random Attack	Smooth Curve Attack
Pulse Attack	98.92%	0	0	0	1.08%
Scaling Attack	0	93.46%	0	6.54%	0
Ramping Attack	0	0	100%	0	0
Random Attack	0	0.53%	3.19%	96.28%	0
Smooth Curve Attack	0	0	0	23.69%	76.31%



(a) NRMSE=1.68% (b) NRMSE=3.36% (c) NRMSE=5.04%

Fig. 8. Performance diagram for comparison of SAX-I (n_s=30), SAX-II (n_s=32), and MLAD with three load forecasting errors (i.e., three NRMSEs).

TABLE V
MAPE VALUES OF DIFFERENT DETECTION METHODS FOR CYBERATTACKS' OCCURRENCE

Attack Templates	Detection Methods			
	SAX-I	SAX-II	SAX-III	MLAD
Pulse Attack	-	-	-	***0.005%***
Scaling Attack	16.49%	20.83%	21.29%	***1.43%***
Ramping Attack	8.42%	11.86%	14.76%	***3.12%***
Random Attack	28.41%	30.29%	32.66%	***1.44%***
Smooth Curve Attack	15.28%	20.69%	24.12%	***10.45%***

Note: SAX-I: n_s=30; SAX-II: n_s=32; and SAX-III: n_s=34.

C. Impacts of Load Forecasting Accuracy on Anomaly Detection Performance

To analyze the impacts of load forecasting accuracy on the anomaly detection performance of different methods, the normalized root mean square errors (NRMSE) is used to represent the load forecasting accuracy. A smaller NRMSE value indicates a higher load forecasting accuracy. Three NRMSEs, i.e., 1.68%, 3.36%, and 5.04%, are used for comparison and analysis. An interesting advantage of the developed MLAD method is that it can be used to estimate parameters of both scaling and ramping attacks. Though SAX methods have been widely used to detect collective anomalies, it intends to only identify a sub-sequence from the attacked load forecasting data. However, the detailed information among this identified sub-sequence, especially for the attack parameter, is still unknown for practitioners using SAX. When using MLAD, as the initial SUS ($ST_1 = (s_1, e_1)$) and the terminal SDS ($ST_M = (s_M, e_M)$) have been accurately detected by dynamic programming, the estimated attack parameter $\tilde{\lambda}$ can be calculated by:

$$\widetilde{\lambda} = \frac{1}{2N_{\mathrm{S}}} \sum_{i=1}^{N_{\mathrm{S}}} \left[\frac{x_{e_1,i} - x_{s_1,i}}{e_{1,i} - s_{1,i}} + \frac{x_{e_M,i} - x_{s_M,i}}{e_{M,i} - s_{M,i}} \right] \tag{32}$$

Table VI illustrates the sensitivity of estimated parameters of both scaling and ramping attacks under three NRMSE conditions. As can be seen, estimated parameters of both scaling attack ($\widetilde{\lambda}_{\mathrm{S}}$) and ramping attack ($\widetilde{\lambda}_{\mathrm{R}}$) present relatively high accuracies. The MAPE metrics are in the range of 9.26%∼17.92% and 9.59%∼14.45%, respectively. The RMSE metrics are in the range of 6.62%∼13.36% and 9.23%∼10.80%, respectively. In addition, the accuracy of estimated parameters increases with the decrease of load forecasting errors, which means the estimation accuracy is sensitive to the load forecasting performance. In other words, the improvement of load forecasts can significantly enhance the accuracy of estimated attack parameters.

Fig. 8 shows the visualized performance diagram for comparison of SAX and MLAD with different NRMSEs. Two types of SAX with the best performance are chosen for comparison, i.e., SAX-I (n_s=30) and SAX-II (n_s=32). For a performance diagram shown in Fig. 8, 1) the left axis represents the value of POD; 2) the bottom axis represents SR; 3) the diagonal dashed lines represent FBIAS; and 4) the dashed curves represent CSI. For a better performance, the points should be close toward the top right corner of the performance diagram. Four sets of thresholds are predefined based on different values of the smaller tolerance ϕ_1 and the larger tolerance ϕ_2, i.e., Thr.-1: ϕ_1=5, ϕ_2=10; Thr.-2: ϕ_1=4, ϕ_2=9; Thr.-3: ϕ_1=3, ϕ_2=8; and Thr.-4: ϕ_1=2, ϕ_2=7.

Fig. 8 demonstrates the good detection performance of the developed MLAD method compared with SAX methods.

TABLE VI
PARAMETER ESTIMATION OF SCALING AND RAMPING ATTACKS WITH THREE LOAD FORECASTING ERRORS USING MLAD

Real Parameters	Metrics	Forecast Errors (NRMSE)		
		1.68%	3.36%	5.04%
Scaling Attack (λ_S=0.2)	$\widetilde{\lambda}_S$	0.2035	0.2198	0.2238
	MAPE	9.26%	16.43%	17.92%
	RMSE	6.62%	12.74%	13.36%
Ramping Attack (λ_R=0.1)	$\widetilde{\lambda}_R$	0.1036	0.1042	0.1061
	MAPE	9.59%	11.91%	14.45%
	RMSE	9.23%	9.70%	10.80%

TABLE VII
PARAMETER ESTIMATION OF SCALING AND RAMPING ATTACKS WITH TWO LOAD FORECASTING METHODS USING MLAD

Real Parameters	Metrics	Methods	
		Conventional NN	SIWNN
Scaling Attack (λ_S = 0.2)	$\widetilde{\lambda}_S$	0.2137	0.2065
	MAPE	15.41%	10.22%
	RMSE	10.68%	8.54%
Ramping Attack (λ_R=0.1)	$\widetilde{\lambda}_R$	0.1049	0.1023
	MAPE	12.24%	8.96%
	RMSE	11.23%	8.78%

The rectangles (detected by MLAD) are closer to the top right corner than circles (detected by SAX-I) and triangles (detected by SAX-II). For different load forecasting errors, the developed MLAD method performs better than any SAX method. Specifically, when using MLAD, the mean SR value (bottom x-axis) with three NRMSEs is 0.88, whereas the mean SR values using SAX-I and SAX-II are 0.72 and 0.58, respectively. Moreover, when using MLAD, the mean POD value (left y-axis) with three NRMSEs is 0.96, whereas the mean POD values using SAX-I and SAX-II are 0.79 and 0.76, respectively. This phenomenon shows that the developed MLAD method is robust to the load forecasting accuracy which can be easily achieved by most current load forecasting techniques.

In addition, for most cases in Fig. 8, the pink points with the largest threshold (Thr.-1) are closer to the top right corner, whereas the blue points with the smallest threshold (Thr.-4) are closer to the bottom left corner. As larger thresholds can generate wider ranges of sets of Ω_1, Ω_2, Ω_3, Ψ_1, Ψ_2, and Ψ_3 in Fig. 4, more detected points are counted as true positive points. This observation can verify the effectiveness of the performance diagram.

D. Impacts of Advanced Load Forecasting Method

To guarantee the universality and generalization of the developed MLAD, the widely used conventional NN is taken as a representative example of load forecasting methods since it can be readily implemented by practitioners who are not experts in the load forecasting research area. That is to say, if the developed MLAD could be applicable to the conventional NN forecasting method with a relatively larger RMSE, it should be readily applicable to the advanced load forecasting method with a relatively smaller RMSE. To validate this, an advanced similar day-based wavelet neural network (SIWNN) method is used to improve the load forecasting performance. Detailed information about SIWNN can be found in [34]. The information of training and testing data is the same as that of the conventional NN method.

Table VII illustrates the performance of two load forecasting methods (conventional NN and SIWNN) based on estimated parameters of both scaling and ramping attacks. As can be seen, when using the improved SIWNN forecasting method, estimated parameters of both scaling attack ($\widetilde{\lambda}_S$) and ramping attack ($\widetilde{\lambda}_R$) present relatively high accuracies with smaller MAPE and RMSE values. While using the conventional NN forecasting method, estimated parameters present relatively low accuracies. For the scaling attack, the MAPE metric is reduced by 33.68% [=(15.41%-10.22%)/15.41%] after using SIWNN. For the ramping attack, the MAPE metric is reduced by 26.8% [=(12.24%-8.96%)/12.24%] after using SIWNN. This observation shows that the improved SIWNN method can enhance the accuracy of estimated attack parameters.

VI. DISCUSSION

The aforementioned research mainly considers cyber attacks performed on load forecasting results through the anomaly detection model. However, it is possible that adversaries may tamper with the anomaly detection model and subsequently the detection result itself. Under this circumstance, a layered defense model can be added into the developed architecture of assessing the impact and evaluate the response to cybersecurity issues, as shown in Fig. 9. In the layered defense step, the anomaly detection output is compared with the forecast output. If there is any evident difference, it means the anomaly detection model is manipulated and remedial actions should be taken on procedures of load forecasts, anomaly detection, and grid operations. Then, an alternative anomaly detection model is enabled and used to replace the manipulated one. Otherwise, if there is no difference, the anomaly detection model is cyber-secure while remedial actions should be taken on procedures of load forecasts and grid operations. It should be recognized that any detection means is breakable, no matter how well designed, and the attacked data may be passed to and used in the grid operation. It is important to have additional layers of defenses to minimize the impacts in the operation stage if this happens, e.g., the attacked data may be crosschecked using state estimation information or PMU measurements.

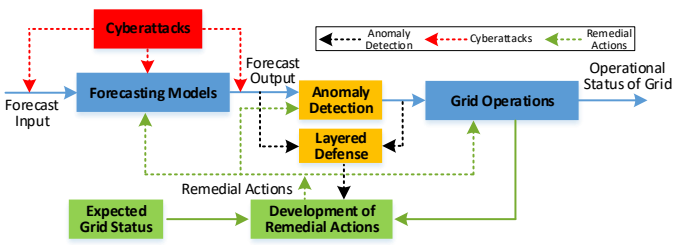


Fig. 9. An architecture of assessing the impact and evaluate the response to cybersecurity issues.

VII. Conclusion

This paper develops a machine learning based anomaly detection (MLAD) method for load forecasting under cyber-attacks. The predicted load data is first used to reconstruct the benchmark and scaling data by using the *k*-means clustering. The naive Bayes classification is then used to determine the specific attack template. Finally, the dynamic programming is utilized to calculate both the occurrence and the parameter of one cyberattack on load forecasting data. Compared with the widely-used SAX method, the effectiveness and robustness of the developed MLAD method is verified by numerical simulations on the publicly load data. Some universal and common lessons are shown as follows:

(i) The detailed occurrence and parameter information of the cyberattacks could be detected with a considerably high accuracy by using MLAD.
(ii) The developed MLAD method is robust to the load forecasting errors with a relatively high success ratio.
(iii) The improvement of load forecasts can significantly enhance the accuracy of estimated attack parameters.

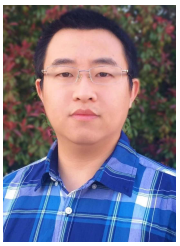
The developed MLAD method is not able to find the specific derivations of the cyberattacks, e.g., the broken topology of the forecasting model, falsified parameters of forecasting methods, or attacked pre-processing techniques. In the future, this research can also be further improved by analyzing the sensitivities of different load forecasting methods with the load data tampered with cyber attacks.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive suggestions to this research.

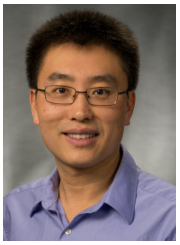
References

[1] G. Liang, S. R. Weller, F. Luo, J. Zhao, and Z. Y. Dong, "Generalized FDIA-based cyber topology attack with application to the Australian electricity market trading mechanism," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3820–3829, 2018.
[2] C.-W. Ten, K. Yamashita, Z. Yang, A. Vasilakos, and A. Ginter, "Impact assessment of hypothesized cyberattacks on interconnected bulk power systems," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4405–4425, 2018.
[3] Y. Liu, S. Hu, and A. Y. Zomaya, "The hierarchical smart home cyberattack detection considering power overloading and frequency disturbance," *IEEE Trans. Ind. Inform.*, vol. 12, no. 5, pp. 1973–1983, 2016.
[4] Y. Xiang, L. Wang, and N. Liu, "A robustness-oriented power grid operation strategy considering attacks," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4248–4261, 2018.
[5] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2250–2267, 2014.
[6] S. Sridhar and M. Govindarasu, "Model-based attack detection and mitigation for automatic generation control," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 580–591, 2014.
[7] M. Yue, "An integrated anomaly detection method for load forecasting data under cyberattacks," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Chicago, IL, USA, 2017, pp. 1–5.
[8] M. Mohammadpourfard, A. Sami, and Y. Weng, "Identification of false data injection attacks with considering the impact of wind generation and topology reconfigurations," *IEEE Trans. Sustain. Energy*, vol. 9, no. 3, pp. 1349–1364, 2018.
[9] R. Moghaddass and J. Wang, "A hierarchical framework for smart grid anomaly detection using large-scale smart meter data," *IEEE Trans. Smart Grid*, vol. 9, no. 6, pp. 5820–5830, 2018.
[10] J. Zhao, G. Zhang, M. L. Scala, Z. Y. Dong, C. Chen, and J. Wang, "Short-term state forecasting-aided method for detection of smart grid general false data injection attacks," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1580–1590, Jul. 2017.
[11] X. Chen, C. Kang, X. Tong, Q. Xia, and J. Yang, "Improving the accuracy of bus load forecasting by a two-stage bad data identification method," *IEEE Trans. Power Syst.*, vol. 29, no. 4, pp. 1634–1641, 2014.
[12] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 2, pp. 1153–1176, 2016.
[13] Z. Ghafoori, S. M. Erfani, S. Rajasegarar, J. C. Bezdek, S. Karunasekera, and C. Leckie, "Efficient unsupervised parameter estimation for one-class support vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 5057–5070, 2018.
[14] J. Wang, D. Shi, Y. Li, J. Chen, H. Ding, and X. Duan, "Distributed framework for detecting PMU data manipulation attacks with deep autoencoders," *IEEE Trans. Smart Grid*, 2018, in press.
[15] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, sep 2017.
[16] E. Keogh, J. Lin, and A. Fu, "Hot SAX: Efficiently finding the most unusual time series subsequence," in *Proc. IEEE Int. Conf. Data Mining*, Houston, TX, USA, 2005, pp. 226–233.
[17] Assess the impact and evaluate the response to cybersecurity issues (AIERCI). [Online]. Available: https://www.energy.gov/sites/prod/files/2017/04/f34/BNL_AIERCI_FactSheet.pdf
[18] The Mathworks, Inc. Load Forecasting. [Online]. Available: https://www.mathworks.com/discovery/load-forecasting.html.
[19] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient *k*-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, 2002.
[20] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.
[21] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002, pp. 841–848.
[22] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.
[23] M. Cui, C. Feng, Z. Wang, and J. Zhang, "Statistical representation of wind power ramps using a generalized Gaussian mixture model," *IEEE Trans. Sustain. Energy*, vol. 9, no. 1, pp. 261–272, Jan. 2018.
[24] D.-I. Kim, T. Y. Chun, S.-H. Yoon, G. Lee, and Y.-J. Shin, "Wavelet-based event detection method using PMU data," *IEEE Trans. Smart Grid*, vol. 8, no. 3, pp. 1154–1162, 2017.
[25] M. K. Jena, B. K. Panigrahi, and S. R. Samantaray, "A new approach to power system disturbance assessment using wide-area postdisturbance records," *IEEE Trans. Ind. Inform.*, vol. 14, no. 3, pp. 1253–1261, 2018.
[26] M. Cui, J. Wang, J. Tan, A. Florita, and Y. Zhang, "A novel event detection method using PMU data with high precision," *IEEE Trans. Power Syst.*, vol. 34, no. 1, pp. 454–466, Jan. 2019.
[27] N. G. Boulaxis and M. P. Papadopoulos, "Optimal feeder routing in distribution system planning using dynamic programming technique and GIS facilities," *IEEE Trans. Power Deliv.*, vol. 17, no. 1, pp. 242–247, 2002.
[28] R. Sevlian and R. Rajagopal, "Detection and statistics of wind power ramps," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 3610–3620, Nov. 2013.
[29] M. Cui, D. Ke, Y. Sun, D. Gan, J. Zhang, and B.-M. Hodge, "Wind power ramp event forecasting using a stochastic scenario generation method," *IEEE Trans. Sustain. Energy*, vol. 6, no. 2, pp. 422–433, Apr. 2015.
[30] M. Cui, J. Zhang, A. R. Florita, B.-M. Hodge, D. Ke, and Y. Sun, "An optimized swinging door algorithm for identifying wind ramping events," *IEEE Trans. Sustain. Energy*, vol. 7, no. 1, pp. 150–162, Jan. 2016.
[31] P. J. Roebber, "Visualizing multiple measures of forecast quality," *Weather and Forecasting*, vol. 24, no. 2, pp. 601–608, 2009.
[32] ISO New England. [Online]. Available: https://www.iso-ne.com/
[33] The Mathworks, Inc. Latin Hypercube Sample. [Online]. Available: https://www.mathworks.com/help/stats/lhsdesign.html
[34] Y. Chen, P. B. Luh, C. Guan, Y. Zhao, L. D. Michel, M. A. Coolbeth, P. B. Friedland, and S. J. Rourke, "Short-term load forecasting: similar day-based wavelet neural networks," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 322–330, 2010.



Mingjian Cui (S'12–M'16–SM'18) received the B.S. and Ph.D. degrees from Wuhan University, Wuhan, Hubei, China, all in Electrical Engineering and Automation, in 2010 and 2015, respectively.

Currently, he is a Postdoctoral Research Associate at Southern Methodist University, Dallas, Texas, USA. He was also a Visiting Scholar from 2014 to 2015 in the Transmission and Grid Integration Group at the National Renewable Energy Laboratory (NREL), Golden, Colorado, USA. His research interests include power system operation, wind and solar forecasts, machine learning, data analytics, and statistics. He has authored/coauthored over 50 peer-reviewed publications. Dr. Cui serves as an Associate Editor for the journal of IET Smart Grid. He is also the Best Reviewer of the IEEE Transactions on Smart Grid for 2018.



Jianhui Wang (M'07–SM'12) received the Ph.D. degree in electrical engineering from Illinois Institute of Technology, Chicago, Illinois, USA, in 2007.

Presently, he is an Associate Professor with the Department of Electrical Engineering at Southern Methodist University, Dallas, Texas, USA. Prior to joining SMU, Dr. Wang had an eleven-year stint at Argonne National Laboratory with the last appointment as Section Lead – Advanced Grid Modeling. Dr. Wang is the secretary of the IEEE Power & Energy Society (PES) Power System Operations, Planning & Economics Committee. He has held visiting positions in Europe, Australia and Hong Kong including a VELUX Visiting Professorship at the Technical University of Denmark (DTU). Dr. Wang is the Editor-in-Chief of the IEEE Transactions on Smart Grid and an IEEE PES Distinguished Lecturer. He is also a Clarivate Analytics highly cited researcher for 2018.

Meng Yue (M'03) received the B.S. degree in electrical engineering from Xi'an Technological University, Xi'an, China, the M.S. degree in electrical engineering from Tianjin University, Tianjin, China, and the Ph.D. degree in electrical engineering from Michigan State University, East Lansing, MI, USA, in 1990, 1995, and 2002, respectively.

Currently, he is with the Department of Sustainable Energy Technologies, Brookhaven National Laboratory, Upton, NY, USA. His research interests include power system stability and dynamic performance analysis, preventive, corrective, and stabilizing control, renewable energy modeling and integration, and high performance computing and probabilistic risk assessment applications in power systems.