Unsupervised Event Detection, Clustering, and Use Case Exposition in Micro-PMU Measurements

Armin Aligholian, *Student Member, IEEE*, Alireza Shahsavari, *Member, IEEE*, Emma Stewart, *Senior Member, IEEE*, Ed Cortez, Hamed Mohsenian-Rad, *Fellow, IEEE*

***Abstract*—Distribution-level phasor measurement units, a.k.a, micro-PMUs, report a large volume of high resolution phasor measurements which constitute a variety of *event signatures* of different phenomena that occur all across power distribution feeders. In order to implement an event-based analysis that has useful applications for the utility operator, one needs to extract these events from a large volume of micro-PMU data. However, due to the *infrequent*, *unscheduled*, and *unknown* nature of the events, it is often a challenge to even figure out what kind of events are out there to capture and scrutinize. In this paper, we seek to address this open problem by developing an *unsupervised* approach, which requires *minimal prior human knowledge*. First, we develop an unsupervised event detection method based on the concept of Generative Adversarial Networks (GAN). It works by training deep neural networks that learn the characteristics of the normal trends in micro-PMU measurements; and accordingly detect an event when there is any abnormality. We also propose a two-step unsupervised clustering method, based on a novel linear mixed integer programming formulation. It helps us categorize events based on their origin in the first step and their similarity in the second step. The active nature of the proposed clustering method makes it capable of identifying new clusters of events on *an ongoing basis*. The proposed unsupervised event detection and clustering methods are applied to real-world micro-PMU data. Results show that they can outperform the prevalent methods in the literature. These methods also facilitate our further analysis to identify important clusters of events that lead to unmasking several use cases that could be of value to the utility operator.**

Keywords: **Micro-PMU, distribution synchrophasors, unsupervised data-driven analysis, event detection, event clustering, deep learning, generative adversarial network, unmasking use cases.**

I. Introduction

A. Background and Motivation

Power distribution systems are becoming more active and dynamic due to the increasing penetration of distributed energy sources, electric vehicles, dynamic loads, and etc. This gives rise to various monitoring and control issues. Many of these issues can be addressed by the use of distribution-level phasor measurement units, a.k.a., micro-PMU [1].

One of the emerging applications of micro-PMUs is to study "events" in power distribution systems. Event-based studies of micro-PMU measurements have a wide range of use cases, such as in situational awareness [2], equipment health diagnostics, such as for inverters [3], capacitor banks [4], transformers [5], distribution-level oscillation detection and analysis [6], fault analysis and fault location [7].

A. Aligholian, A. Shahsavari and H. Mohsenian-Rad are with the University of California, Riverside, CA, USA. E. Stewart is with the Lawrence Livermore National Laboratory, Livermore, CA, USA. E. Cortez is with the Riverside Public Utilities, Riverside, CA, USA. This work is supported in part by UCOP grant LFR-18-548175. The corresponding author is H. Mohsenian-Rad.

Before one can do any event-based analysis, including for the above use cases in [2]–[7], one needs to first detect and identify the *events that are of value*. However, this is a challenging task due to at least the following three reasons: 1) most events are *infrequent*; 2) most events are inherently *unscheduled*; and 3) it is often *not known* ahead of time, what kind of events we should seek to find and identify; i.e., we often do not have a prior knowledge about what to look for.

Given the enormous size of measurement data that is generated by micro-PMUs, such as 124,416,000 readings per micro-PMU per day [1], the challenges that we listed above call for developing effective data-driven techniques that are automated and require minimal prior knowledge. Addressing this open problem is the focus of this paper.

B. Summary of Technical Contributions

Given the unknown, infrequent and unscheduled nature of events in micro-PMU measurements, we propose an interconnected unsupervised event detection and unsupervised event clustering method; which followed by a comprehensive analysis of the engineering implications for the events in each key clusters that we identify from real-world micro-PMU measurements. The main contributions in this paper are listed as follows:

- A novel unsupervised event detection method is developed based on the concept of Generative Adversarial Networks (GAN) by training deep neural networks. Given the infrequent nature of events in micro-PMU data, the central idea is to train the GAN models to learn the normal behavior and trends, which according to the prior experimental results account for 99.6% of the samples in micro-PMU data. Accordingly, any pattern and signatures that deviates from the captured normal characteristics of the micro-PMU data is marked as an event by the trained discriminator. A set of extracted events by expert knowledge from the real-world data set is used for evaluation. The results show the effectiveness of the proposed event detection method compared to multiple state-of-the-art methods in the literature. The proposed event detection relies solely on micro-PMU measurements and it does *not* need the network model or prior labeling of the events.
- A two-step unsupervised event clustering method is proposed. In a pre-processing step, events are categorized based on their origin (i.e., the features that are affected by the event), which is obtained from the proposed event detection method. In the second step, in each pre-processed category, a new clustering model is formulated and solved in form of a mixed-integer linear programming

(MILP). A rolling based similarity measure, maximum correlation coefficient (MaxCorr), is used to compare any two events. The proposed clustering method is *active*, i.e., it is capable of identifying new clusters of events on *an ongoing basis*. New clusters are optimally extracted as needed; in order to account for any unknown upcoming events. The experimental results confirm the effectiveness of the proposed clustering model compared to the prevalent clustering methods. The performance of the proposed clustering method is evaluated and verified also in comparison with a reference set of clustered events that are obtained by the expert knowledge.

- The events in each identified cluster are scrutinized in order to unmask their engineering implications and use cases. The origin and the cause of the events are identified to determine their value to the system operator. By implementing the proposed unsupervised approach one can identify the frequency of happening and other statistical characteristics of different event types, extract specific events by combining the event clusters' characteristics and time of occurrence; find rare and unusual events, such as faults and incipient failures and new major loads. It can even identify deficiency in micro-PMU data reporting.

C. Literature Review

The *event detection* component in this paper can be broadly compared with the other data-driven studies such as in [2], [3], [8]–[16]. Some methods are based on principles in statistics. For example, in [2], which we consider as one of the benchmarks for performance comparison in this study, a data-driven statistical event detection method is proposed that is based on absolute deviation around median, combined with dynamic window sizes. On the other hand, machine learning models, including deep learning models, are getting significant attention in different research areas due to immense increase in the amount of measurement data. Power system is not an exception with massive data collection by measurement units such as smart meters, micro-PMUs and smart inverters. Thus, these large data sets make researchers capable for implementing deep learning model to address issues that are mainly data dependent and complex to solve them with common models. One of the promising applications of deep learning models are anomaly detection which has been implemented in vast scale in smart grid such as, outage detection in the network by using GAN models [17] and fault detection [18], IoT-based occupancy sensor unusual behaviour [19] and smart meter anomaly detection [20].

Some of which are either supervised or semi-supervised. That means, they require either full labeling or partial labeling of the events, e.g., in [8], [10]. On the other hand, few event detection methods in the literature that are unsupervised; they are focused on some specific types of events, such as frequency events [12], significant known events such as three phase fault or cap bank switching [15] or cyber attacks [21]. In contrast, the event detection method in this paper covers a wide range of event types which here, an event is defined rather broadly and may refer to balanced/unbalanced load switching, capacitor bank switching, connection or disconnection of distributed energy resources (DERs), inverter malfunction, a minor fault, a signature for an incipient fault, etc. [2], [22], [23]. In [21], the authors used symbolic dynamic filters to extract features and dynamic Bayesian networks to learn the system behaviour to detect false data injection. Although the method is unsupervised, this method requires prior knowledge about the dynamics of the system; which is typically not available in practice; such as in the case of the field study and the real-world data analysis in this paper. Similarly, in [15] physical aspect of the system needs to be available in order to detect, classify and localize the abnormalities in the system. Some other unsupervised anomaly detection methods, such as the Generalized Graph Laplacian (GGL) method in [16], are based on determining the graph similarity between the sample windows of the micro-PMU data. We used the method in [16] as a benchmark to evaluate the performance of our event detection approach. Other methods that we used as benchmarks include the unsupervised statistical model in [2] and the unsupervised learning method introduced in the preliminary conference version of this paper in [11].

Generative Adversarial Networks (GANs) are broadly studied in areas such as image generation [24], high-dimensional likelihood-free inference [25], medical time-series generation [26], and so on. These models typically focus on the sample generation capability of the GAN model, i.e., the desirable features of the *"generator"* sub-system in the GAN model. However, recent studies have shown that the GAN model can offer other applications also through the desirable features of a *"discriminator"* sub-system. For instance, the GAN model has been used in the recent study in [27] to detect bogus samples, cyber attacks, or general time-series anomalies [28]. Here, the ultimate goal of the discriminator is to distinguish normal from abnormal samples; Thus, in this paper, the proposed model is focused on carefully adjusting the GAN models for our specific purpose; which is detecting events by discriminator, through learning the normal behaviour of the system.

The *event clustering* component in this paper can be broadly compared with studies such as in [2], [13], [29], [30]. In [29], auto encoder-decoder is used for feature extraction; and the latent space of the auto encoder-decoder is used for supervised event classification. In [2], supervised support vector classification is used to classify the events based on their source location. In [13], different types of voltage sag events are detected based on a threshold, which is defined by voltage magnitude slope per cycle, then k-means and Ward-method clustering are used to identify the characteristics of the voltage sag events. The main limitation in [2], [29] is that they both require prior event labeling. As for the method in [13], it is focused on voltage sag events. In [30], the authors used an unsupervised clustering method; however, the focus is on specific faults; such as single-line to ground or line-to-line faults. In contrast, the event clustering method in this paper deals with a wide range of events, including benign yet informative events about the operation status of the power distribution system, its equipment, and its loads. Importantly, no prior knowledge is used in the proposed method. Therefore, various events can be detected, classified, and characterized;

all in unsupervised fashion. In fact, it is designed to explore new events even if they do not match any of the existing clusters through actively searching for new clusters. Thus, the proposed method is well-suited to unmask meaningful use cases based on the outcome of the proposed unsupervised event detection and unsupervised event clustering methods.

This study is fundamentally different from all the previous studies that we have done based on the micro-PMU measurements in our pilot project cite in Riverside, CA. The work in [2], is about the analysis of events by using *supervised* learning methods. This is in sharp contrast compared to the current paper that is about *unsupervised* learning method. Unlike in [2], where we had to use our field knowledge to do an extensive and time-consuming task of manual event labeling, the method in this paper requires minimal prior knowledge about the power distribution feeder. The work in [4], [6], [31] is about scrutinizing some specific types of events, such as capacitor bank switching or certain faults. The assumption in all these three papers is that the events of interests are *already detected and classified*. In this regard, the studies in [4], [6], [31] can actually benefit from the methodologies that we propose in this paper;The work in [7], [32] is *inherently model-based*; where the focus is on identifying the location of an event that is *already detected and classified*. The location identification methods in these two papers require access to the physical models of the power distribution feeder. The work in [33]–[35] is about distribution system state estimation; therefore the analysis is unrelated to this paper; and the results are based on computer simulations.Finally, the work in [36], [37] are about cyber-security attacks against micro-PMUs.

Common event detection schemes, such as moving average filters, statistical change detection methods and PCA, have shown low accuracy due to the variety of event signatures in power distribution feeders. Also, methods such as PCA, are meant to represent high-dimensional data with much lower dimensional vectors; which is a suitable approach only if the data lies near a linear manifold in the high dimensional space. However, even with kernel PCA, the non-linearity of the data cannot be modeled appropriately. In general, by increasing the number of training samples, deep learning models usually show higher accuracy compared to the other aforementioned models. Also, GAN models, due to their specific design with a min-max game between a generator and a discriminator, are suitable for learning the essentials of a data distribution. Hence, by a proper problem definition and appropriate generator design, we can articulate real data distribution. Consequently, the discriminator can distinguish between real data and fake data (not part of the normal real data distribution). In this regard, using GAN model, alongside with deep learning, is a promising combination for event detection in micro-PMU measurements, which is the approach that we take in this study.

Compared to the conference version of this work in [11], which was *solely about event detection*, this paper has several new contributions. First, the model architecture and the features are different and result in better performance. Second, to identify the type of detected events, a two-step unsupervised clustering model is proposed. Third, together, the proposed unsupervised event detection and clustering methods enable us to expose use cases and applications of the key event clusters.

II. Unsupervised Detection Method

The proposed GAN-based event detection method is developed by training two deep neural networks. In short, the first deep neural network, a.k.a., generator, tries to generate data points that follow the distribution of the real-world data. The second deep neural network, a.k.a., discriminator, tries to distinguish between the generated data points and the real-world data. The architecture and process of the GAN models is as follow:

A. Features: Checking the *magnitude* of voltage and current in micro-PMU measurements is a common option to detect and identify events, e.g., see [2], [11], [38]. However, due to the fluctuations in the frequency of the power system, the phase angles of voltage and current are often not used directly. Instead, active power and reactive power are usually used as the two features that involve voltage and current phase angle measurements, besides the magnitude of voltage and current, to detect events in micro-PMU measurements. In this paper, we propose to use power factor as the feature that involves the voltage and current phase angle measurements. Thus, the features across the three phases that we use in this paper are

$$|V_\phi|, |I_\phi|, \cos(\theta_\phi), \quad \phi = A, B, C. \tag{1}$$

which denote the voltage magnitude, current magnitude, and power factor in each phase ϕ, respectively. For notation simplicity, in the rest of paper, we refer to the features in (1) for *all the three phases*, without specifying subscript ϕ. Also, batch normalization have been implemented in order to prevent internal covariate shift.

B. Generator: It is a deep neural network that comprises Long Short-Term Memory (LSTM) modules [39] as well as dense layers similar to [11]. Given a noise vector z from a distribution function $p_z(z)$, such as $z \sim \mathcal{N}(\mu_z, \sigma_z^2)$, the generator aims to produce samples that follow the distribution of the real-world data. Thus, a neural network $G(z, \theta_g)$ is trained to minimize the following objective function, where θ_g denotes the weights of the generator network [40]:

$$\frac{1}{N}\sum_{i=1}^{N}\big[log(1 - D(G(z_i)))\big]. \tag{2}$$

Here, N denotes the number of samples in a batch of training data set. Also, D and G denote the discriminator function and the generator function, respectively.

C. Discriminator: It aims to distinguish between the generated samples by the generator and the actual measurements. It contains LSTM modules and dense layers similar to [11]. Neural network $D(x, \theta_d)$ is trained to report a single value as output. Here, x and θ_d are the vectors of measurements and the weights of the discriminator network, respectively. The discriminator maximizes the probability of distinguishing the measurement from the data generated by the generator, as:

$$\frac{1}{N}\sum_{i=1}^{N}\big[log(D(x_i)) + log(1 - D(G(z_i)))\big], \tag{3}$$

where x_i is the i^{th} real sample. On one hand, the generator tries to minimize (2). On the other hand, the discriminator tries to maximize (3). Thus, the generator and the discriminator play a *min-max* game over the following function:

$$V(G,D) = \mathbb{E}_x \sim p_{data}(x)[log(D(x))] + \mathbb{E}_z \sim p_z(z)[log(1 - D(G(z)))]. \quad (4)$$

D. Training and Convergence: Prior studies have shown that only about 0.04% of micro-PMU measurements contain events [11]. Therefore, we train all the nine constructed GAN models, one model for each feature, so as to *learn* the characteristics of the *normal* trends in micro-PMU measurements. We detect an event when there is an abnormality. For each GAN model, the solution of the min-max game over $V(G,D)$ in (4) must satisfy the following two conditions:

- **C1:** For any fixed G, the optimal discriminator D^* is:

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}. \quad (5)$$

- **C2:** There exists a global solution such that:

$$\min_G (\max_D(V(G,D))) \iff p_g(x) = p_{data}(x). \quad (6)$$

The training of the GAN model and proofs are explained in details in [40]. However, training of GANs is known to be unstable and sensitive to the choices of hyper-parameters. Hence, obtaining compelling results such as achieving global optimum and creating a sample distribution close enough to the real data distribution is challenging and requires an assumption that the discriminator is optimal at each step [41]. Experimental results in our case with different micro-PMU data set show that local optima and mode collapse situation almost never happen due to non-sharp gradients of the discriminator function around real data points [41].

The choice of the hyper-parameters of the GAN model is critical in achieving an equilibrium. In particular, based on the two criteria in (5) and (6) and *convergence constant* $\epsilon > 0$ the equilibrium should satisfy the following conditions [40]:

$$|\max_D(V(G,D)) - (-log4)| < \epsilon, \quad |D_g(x) - \frac{1}{2}| < \epsilon. \quad (7)$$

E. Event Scoring: Once all the nine GAN models are trained, they provide us with nine distinct event detectors; one per each feature. Each discriminator gives us a *score* as its output, which indicates how close a given window of measurements is to the global optimum that is obtained from (5) and (6). If, for any GAN model, the score is not close enough to the global optimum, then it means that the given window of measurements does *not* match the normal behavior that is learned by the GAN model; therefore, it is deemed to contain an event. In this process, the D'Agostino's K-squared test [42], with a significant level of 0.05, is applied to the discriminator output from the training set; and the results show strong evidence of normality. Thus, a normal probability distribution function (PDF) is fit to the obtained scores for training set, to have $\zeta \sim \mathcal{N}(\mu, \sigma^2)$, where μ is almost equal to the global optimum and σ is small.

Algorithm 1 Unsupervised Event Detection

Input: Training and test data based on the features in (1).
Output: Event detection vector $E^w_{9\times1}$ for the w^{th} data.
// Learning Phase
Foreach feature f in (1):
  Train the GAN_f model
  Use discriminator as scoring function $D^*_f(\cdot)$.
  Calculate the scores for the training data.
  Fit a Normal PDF $\mathcal{N}(\mu_f, \sigma^2_f)$ to the obtained scores.
End
// Detection Phase
Foreach new micro-PMU test data (w):
  Foreach feature f in (1):
    Calculate score s^w_f using $D^*_f(\cdot)$.
    If $s^w_f \notin (\mu_f - z_p\delta_f, \mu_f + z_p\delta_f)$ **Then**
      $e^w_f = 1$ // Event
    Else
      $e^w_f = 0$ // No Event
    End
    Append e^w_f to E^w
  End
End

F. Algorithm: The proposed event detection method is summarized in Algorithm 1. The algorithm has two phases. First, a learning phase, in which the GAN models are trained for each feature; and their associated normal PDF are constructed. Second, an event detection phase, in which, for each window w of sample data, the scores are calculated by all the nine GAN models and accordingly the *detection vector* is obtained:

$$\boldsymbol{E}^w_{9\times1} = [e^w_1, \cdots, e^w_9] \quad (8)$$

The detection vector is a 9×1 *binary* vector, where 9 is the number of features as in (1). Entry e^w_f is 1 if an event is detected in w^{th} window and f^{th} feature, otherwise zero. Vector $\boldsymbol{E}_T$ is the set of all detection vectors. It should be noted that, a common choice for z_p in the threshold $\mu \pm z_p\sigma$ is 3, known as the three-sigma rule [43].

The detection vectors not only show us the existence of event; they also provide us with the inputs that we need for our clustering algorithm; which we will explain in Section III.

G. Evaluation metric: We use the Matthews correlation coefficient (MCC) [44] as the metric to assess accuracy; for both detection and clustering. As explained in [45], a common evaluation criteria, such as F1-score, can sometimes be misleading and show over-optimistic inflated results, especially on imbalanced data-sets; such as anomalies which inherently have low frequency compared to normal samples. The MCC, instead, is a more reliable statistical metric which produces a high score only if the obtained prediction results are adequate in *all* of the four categories of the confusion matrix, i.e., true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), proportionally both to the size of the positive elements and the size of the negative elements in the data-set. On the other hand, for multi-class clustering/classification problems, the general format of MCC is implemented. Therefore, for both event detection and

clustering, we use MCC as the evaluation metric:

$$MCC = \frac{N_s Tr(\psi) - \sum_{k=1}^{K}\sum_{l=1}^{K} \psi_k \psi_l}{\sqrt{N_s^2 - \sum_{k=1}^{K}\sum_{l=1}^{K} \psi_k \psi_l^T} \sqrt{N_s^2 - \sum_{k=1}^{K}\sum_{l=1}^{K} \psi_k^T \psi_l}}, \tag{9}$$

where N_s is the number of samples, K is the number of clusters, ψ is the confusion matrix which is $K \times K$, ψ_k and ψ_l are the k^{th} row and l^{th} column of ψ, respectively. It should be mentioned that, for event detection, a special case of general MCC with $K = 2$ is used in this study. MCC is a number between –1 and 1; where 1 represents a perfect prediction. MCC is used for the evaluation sets that are extracted by expert knowledge for both event detection and clustering.

III. Unsupervised Clustering Method

Given the detection vectors in Section II, in this section, we develop a two-step event clustering method so that we can later study different types of events in details.

A. Step I: Pre-Processing

An obvious choice for clustering is to group the events based on their detection vector. For each measurement window w that contains an event, vector $\boldsymbol{E}^w_{9\times 1}$ has at least one entry that is one. Accordingly, we can put all the events with the same detection vector in the same category; based on the nine features in (1). For example, we put all the events with $\boldsymbol{E}^w_{9\times 1} = [111\ 000\ 000]$ in the same category because they similarly causes abnormalities only in voltage magnitude on all phases.

In theory the detection vector can result in $2^9 - 1 = 511$ possible combinations; when an event is detected. However, based on the physics of the power system; only some of these combinations can actually happen in practice. In fact, our analysis of the real-world micro-PMU data resulted in only a handful of such combinations across thousands of detected events; as we will discuss in details in Section IV-C.

Thus, in practice, the above clustering mainly serves as a *pre-processing* in the clustering problem. We often need to further break down a category into several clusters to expose the use case of the events in that category. This is done through a comprehensive clustering optimization in Section III-B.

B. Step II: Clustering Optimization

In this section, we explain the similarity measure, the proposed clustering optimization problem formulation, its solution based on exact linearization, the cluster representatives, and the optimum cluster numbers in each category.

1) Rolling-Based Similarity Measure: The key to proper clustering is to accurately measure how similar (or dissimilar) different event signatures are within each pre-processed category. However, this is a challenging task because similar events may not have exact same duration. Events need to be aligned with respect to their shape and their corresponding measurement windows for appropriate similarity assessment.

To address the above two challenges, we propose to first expand the measurement window size for each captured event to make sure that the entire event is included in the measurement window. Once this is done, for each event i, we define:

$$\boldsymbol{P}_i = \begin{bmatrix} \alpha_i^{1,1} & \cdots & \alpha_i^{1,\tau} \\ \vdots & \ddots & \vdots \\ \alpha_i^{9,1} & \cdots & \alpha_i^{9,\tau} \end{bmatrix}. \tag{10}$$

There are nine rows in $\boldsymbol{P}_i$ corresponding to the nine features in (1). The columns correspond to the measurement time instances, where τ is the maximum expanded window size of the two events that are compared with each other.

To determine the similarity between two events i and j, we need to align matrices $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$, because we do *not* know *where exactly* the event is located within each measurement window. Therefore, we propose to take matrix $\boldsymbol{P}_i$ as fixed, and *roll* matrix $\boldsymbol{P}_j$ in the time axis, one time slot at a time. In other words, in each rolling step, the last column is removed from $\boldsymbol{P}_j$ and appended before the first column in $\boldsymbol{P}_j$; thus, we have τ rolling steps for each two event comparison.

For each rolling step k, where $k = 1, \ldots, \tau$, let us define $c^k_{i,j}$ as the average of the 9 correlation coefficients that can be calculated between each of the 9 rows in $\boldsymbol{P}_i$ and its corresponding row in $\boldsymbol{P}_j$; where $\boldsymbol{P}_j$ is rolled for k steps. We define MaxCorr as the rolling-based measure of similarity as:

$$MaxCorr_{i,j} = \underset{k=1,\ldots,\tau}{\text{maximum}}\ c^k_{i,j}; \tag{11}$$

to be used as the similarity measure between events i and j.

2) Optimization Problem Formulation: Consider a given category of events based on the pre-processing step in Section III-A. Suppose there are I detected events in this category and we want to break them down into C clusters. We propose to solve the following clustering optimization problem:

$$\underset{u}{\text{minimize}} \quad \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{c=1}^{C} u_{i,c}u_{j,c}(1 - MaxCorr_{i,j}) \tag{12a}$$

$$\text{subject to} \quad u_{i,c} \in \{0,1\}, \tag{12b}$$

$$\sum_{c=1}^{C} u_{i,c} = 1 \quad \forall i. \tag{12c}$$

where $u_{i,c}$ is a binary variable. It is one if event i is in cluster c; otherwise it is zero. Problem (12) minimizes the sum of the distances between the events, measured as 1-$MaxCorr_{i,j}$, across different clusters. The constraint in (12c) assures that each event is assigned to only one cluster. Problem (12) is a MINLP.

3) Exact Linearization: To enhance computational performance, the MINLP in (12) is replaced with an *exact equivalent* Mixed Integer Linear Programming (MILP), as follows:

$$\underset{u,\,t}{\text{minimize}} \quad \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{c=1}^{C} t_{i,j,c}(1 - MaxCorr_{i,j}) \tag{13a}$$

$$\text{subject to} \quad u_{i,c}, t_{i,j,c} \in \{0,1\}, \tag{13b}$$

$$\sum_{c=1}^{C} u_{i,c} = 1 \quad \forall i, \tag{13c}$$

$$u_{i,c} + u_{j,c} - t_{i,j,c} \le 1 \quad \forall i,j,c, \tag{13d}$$

$$-u_{i,c} - u_{j,c} + 2t_{i,j,c} \le 0 \quad \forall i,j,c. \tag{13e}$$

where the nonlinear product of $u_{i,c}$ and $u_{j,c}$ in the objective function is replace with linear term $t_{i,j,c}$. The linear constraints in (13d) and (13e) are used to make sure that $t_{i,j,c}$ is indeed equal to such product in order to assure an *exact* linearization. Problem (13) can be solved using any MILP solver for a set of detected events in a given time period as training set.

4) Cluster Representatives: Once the clusters are obtained by using the training data and solving the MILP problem in (13), we define a representative for each cluster to speed up the process of clustering incoming events. Thus, the new events are compared to a few cluster representatives rather than to all events through (13). To determine the optimum representative for each cluster, we solve the following optimization problem:

$$\underset{v}{\text{minimize}} \quad \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{c=1}^{C} u_{i,c}v_{j,c}(1 - MaxCorr_{i,j}) \quad (14a)$$

$$\text{subject to} \quad v_{i,c} \in \{0,1\}, \quad (14b)$$

$$\sum_{i=1}^{I} v_{i,c} = 1 \quad \forall c \quad (14c)$$

Variable $v_{j,c}$ is binary. It is one, if event j is the representative event for cluster c, and zero otherwise. Constraint (14c) is used to make sure that there is only one representative for each cluster. Notice that $u_{i,c}$ is parameter, not a variable, in this optimization problem; because the clusters are already formed. Therefore, problem (14) is an MILP by construction.

5) Number of Clusters (N_c): So far, we have assumed that the number of clusters, i.e., parameter c is fixed. However, we do obtain the optimal number of clusters in our proposed method. This is done by solving the optimization problem in (13) with respect to different number of clusters. Then, the optimal number of clusters is determined based on the silhouette values of the clusters. Subsequently, cluster representatives is identified for the optimally obtained cluster by using (14).

C. Active Clustering

Since the proposed event clustering method is active, it can create new clusters. Given the large number of events that are detected in micro-PMU measurements for our data set, it is computationally prohibitive to cluster all of 15 days events at the same time. On the other hand, by training a subset of the detected events and assign the new events to the trained cluster, those new types of events would be assigned to a wrong cluster. To address these issues, we solved the clustering optimization problem only on the first day in our data set to set up a base for the event clusters. The newly detected events would be compared to each base cluster representatives and they will be assigned to the closest cluster. Unless, if MaxCorrs of a new event is less than a threshold (φ) for every existing cluster representative, then a new cluster is created. In practice, such new cluster is added only occasionally, which shows the common events are almost appear in every day. However, the newly added clusters are usually those weakly or rare events. Furthermore, the clusters can be updated using the complete optimization-based approach periodically once every few days in order to pick the optimal representative s for each cluster.

Algorithm 2 Unsupervised Event Clustering

Input: Event detection vectors $\boldsymbol{E}_T$ from (8)
Event time-series data,
Number of clusters (N_c),
Similarity threshold (φ).
Output: Clusters and their representatives,
Silhouette value.
// Learning Phase (Offline)
Create categories based on unique sets in $\boldsymbol{E_T}$.
Assign each event to its category.
Foreach n from 1 to N_c:
  Foreach category in $\boldsymbol{E_T}$:
    Cluster the events based on (13).
    Determine the cluster representative based on (14).
  End
End
Calculate Silhouette value for all possible combinations.
Set the number of clusters and their representatives
//Active clustering Phase (Online)
Foreach new event in test data:
  If the detection vector of new event is in $\boldsymbol{E_T}$ **Then**
    Calculate MaxCorr with all representatives.
    If all calculated MaxCorrs are less than φ **Then**
      Make a new cluster in the related category.
      Set new event as the cluster representative.
    Else:
      Assign new event to the closet cluster.
    End
  Else:
    Create a new category.
    Create a cluster with new event as representative
  End
End

IV. Experimental Results

The proposed event detection and clustering methods are applied to 1.2 billion measurements over 15 days of real-world micro-PMU data. Fourteen days of data are used for training the event detection method and one day of data is used to test it. One day of data is used for cluster optimization; and active clustering is done for the rest of the data.

A. Parameters Detail

The architecture of the GAN model has two parts. The generator starts with a dense layer of size 40, three layers of LSTM with 32, 64 and 128 modules, and a dense layer of size 256. The discriminator is in reverse order; the only difference is that the last layer in the discriminator is a dense layer with size 1. All activation functions are LeakyReLU except the last layer in the discriminator; which is sigmoid. In the LeakyReLU functions, the slope of the leak is set to 0.2 in all models. For tuning the hyper-parameters, we used the *coarse-to-fine* method. In this method, we first randomly chose a set of values for each parameter. Then we narrowed down the choices to a smaller subset based on the obtained results. This procedure was repeated until we achieved the desired value for each parameter. It should be mentioned that,

depending on the hyper-parameter, scaling can be helpful, such as log scale for learning rate. This can help fasten the search for suitable values of choice. The learning rate α is set to 0.0002 for Adam optimizer and β_1 is set to 0.5 for better stability in training. A critical parameter when it comes to capturing the events appropriately is window size, which first it should be wide enough to capture the essential signatures of an event and second it should be small enough to prevent event synchronicity and high computational time. By analyzing different window sizes for different set of micro-PMU data, the best result in terms of accuracy and reasonable computational time, is 40 data points. Also, in order to assure that events are not overlooked, we consider that each window has 20 data points overlap with the previous window. All GAN models are developed with Tensorflow in Python by using Nvidia GTX 1050 ti GPU and a core i-7 2.2GHz CPU with 32 GB RAM.

B. Event Detection Results

Table I shows the MCC for the proposed event detection method, in comparison with the benchmark methods in [2], [11], [16]. A total of 1200 reference events are visually extracted by expert knowledge within 6 hours (64800 window samples of 40 time-slots) to evaluate the performance of event detection. The proposed method outperforms the methods in [2], [11] and [16]. The combined training time of all 9 GAN models is 2 hours. Once the initial training is done, it takes less than 4 milliseconds to determine a new incoming sample as normal or event; i.e., the detection time is 4 milliseconds. The detection time for [2] and [11] is 3 and 10 milliseconds, respectively. Thus, the proposed method maintains the same level of computational complexity; but it achieves much better accuracy. It should be added that, to have a fair comparison with the model in [16], we analyzed different window sizes for the aforementioned method; and as we increase the sample numbers, the accuracy is improved, however, the *rate* of the improvement in accuracy was decreasing, in other words, accuracy does not change significantly by extending the window at a certain point; Also, the training time is ascending as well. Thus, the best performance with the same training time as GAN models are considered for the method in [16]. Another point about the method in [16] is that, due to the use of similarity graph, the method in [16] needs to be re-trained every time which makes it impractical for detecting events with new upcoming data in an online mode, however, for offline mode this method has the lowest False Positive (FP).

An interesting observation when we compare the proposed event detection model with the model in [11] is that, the choice of the independent features in (1), in particular the use of $\cos(\theta)$ instead of active power and reactive power, improves the accuracy of event detection. It also improves the independence in the outputs of the trained GAN models. This makes the resulting detection vectors to even enhance the performance of the subsequent clustering method. One of the main advantages of the proposed model compared to the GGL method in [16] is the aspect of learning the normal operation of the system. Although the GGL method has a very low false positive rate, the number of its true positives is lower than the method in [11] as well as the proposed method in this paper. The reason is that, when several events happen continuously, i.e., they happen back to back, such as the events in Fig. 12, the GGL method would train its similarity matrix based on the considered window sample. In this case most of the samples are events, thus, events are *not* anomaly anymore for the GGL method. As a result, there would be higher score value of similarity among such event-containing back-to-back window samples. This leads to a lower true positive rate for the GGL method. On the contrary, the proposed model in this paper considers each sample *individually* and it compares each sample with learned signature of the normal samples by the GAN models. This improves true positive rate. It should be noted that, the statistical method in [2] has reasonable accuracy in detecting most of the three phase events that are *balanced*; due to the fact that the method in [2] was not designed to particularly detect unbalanced events. The method in [2] performs poorly also to detect events with low magnitude. Both of these issues are resolved in this paper. Given the fact that it is common to have unbalanced events in power distribution systems, this particular advantage of the proposed method is of importance in real-world applications.

TABLE I
EVENT DETECTION INFORMATION FOR CONFUSION MATRIX, PRECISION, RECALL AND MCC

Metric	Statistical [2]	GGL [16]	Enhanced Method [11]	Proposed Method
TP	311	990	1033	1132
FN	889	210	167	68
FP	210	1	56	36
TN	63390	63599	63544	63546
Precision	0.596	0.998	0.948	0.947
Recall	0.259	0.825	0.861	0.943
MCC	0.386	0.906	0.901	0.955

C. Event Clustering Results

The proposed event clustering method is applied to the captured events in Section IV-B; and its performance is compared with the following prevalent clustering methods in the literature: kNN [46], k-Medoids [47], and fuzzy-k-Medoids [48]. Different similarity measures are also considered: euclidean, DTW [49], soft-DTW [50], and MaxCorr. In order to compare clustering results, different indices are implemented in the literature such as Jaccard Index, Adjusted Rand Index, Fowlkes Mallows Index, Normalized Mutual Information and Silhouette index; where the last one, i.e., the silhouette index is generally known to show better results within variety of data sets [51]. However, if a labeled evaluation set is available, the analysis and assessment of the clustering model is more intuitive and informative. Thus, in this paper the comparison is conducted over 4000 reference events that are visually clustered with expert knowledge. These events are clustered after being detected by the proposed event detection method.

Table II shows the MCC for different clustering methods. Two observations can be made based on the results in this table. First, the clustering methods are almost always more accurate when MaxCorr is used as similarity measure. Second,

TABLE II
MCC IN EVENT CLUSTERING FOR DIFFERENT METHODS AND DIFFERENT DISTANCE CRITERIA

Distance	KNN	k-medoids	Fuzzy k-medoids	Proposed Method
Euclidean	0.451	0.543	0.522	0.447
DTW	0.584	0.863	0.871	0.911
soft-DTW	0.579	0.861	0.871	0.888
MaxCorr	0.645	0.887	0.882	0.938

TABLE III
CLUSTER CATEGORIES FROM PRE-PROCESSING

Detection Vector	Features $\|V\|$	$\|I\|$	$cos(\theta)$	Number of Events	Pre-Processing Category
$\boldsymbol{E}_1$	[111	111	111]	34242	I
$\boldsymbol{E}_2$	[111	000	000]	12270	II
$\boldsymbol{E}_3$	[000	111	111]	809	III
$\boldsymbol{E}_4$	[000	100	100]	13956	IV
$\boldsymbol{E}_5$	[000	110	110]		
$\boldsymbol{E}_6$	[000	011	011]		
$\boldsymbol{E}_7$	[000	000	111]	52	V
$\boldsymbol{E}_8$	[000	000	110]		
$\boldsymbol{E}_9$	[000	000	011]		

our proposed clustering method outperforms kNN, k-Medoids, and fuzzy-k-Medoids for any similarity measure. The computational time to *train* the KNN, k-medoids, Fuzzy k-medoids, and the proposed models (when MaxCorr is considered as similarity measure) are 5 minutes, 7 minutes, 15 minutes, and 65 minutes, respectively. Note that, training is done *offline*. Therefore, the higher accuracy of the proposed model does *not* cause higher computational time during the operation. Importantly, recall that the proposed clustering method is *active*. In fact, when it comes to clustering new upcoming events that require creating new clusters, which is done *online* and during operation, all of the above methods have almost the same computational time; which are less than 4 milliseconds.

D. Analysis of Identified Clusters

A total of nine detection vectors were observed among all the events which they are denoted by $\mathbf{E}_1$ to $\mathbf{E}_9$, as shown in Table III. As part of the pre-processing step in Section III-A, these detection vectors result in five categories, denoted by Category I to Category V, as shown on the last column in Table III. Categories I, II, and III include *balanced* events; while Categories IV and V include *unbalanced* events.

The optimization-based clustering in Section III-B is then applied to the above five categories. It resulted in identifying a total of **16 final clusters**. In this regard, Category I is divided into six clusters; Category II is divided into three clusters; Category III is one cluster by itself; Category IV is divided into three clusters; and Category V is divided into three clusters.

Next, we use the above clustering results to scrutinize and expose the use cases for the events within each cluster.

E. Use Case Exposition: Six Clusters in Category I

Six clusters are identified in Category I; denoted by Clusters #1 to #6. Clusters #1 and #2 can help identify different load types. Clusters #3 and #4 can reveal malfunctions in the operation of capacitors. Cluster #5 can help identify a specific two-step transient events. Cluster #6 can identify oscillations.

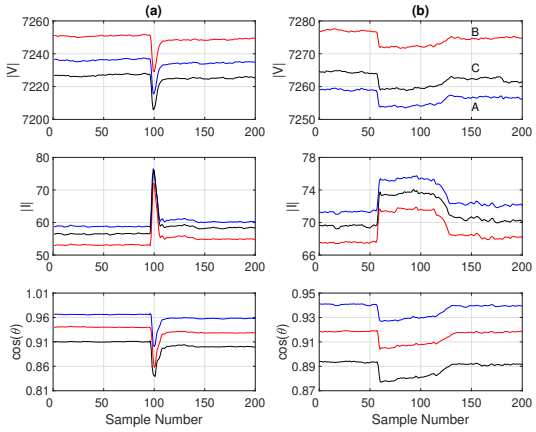

Fig. 1. Examples of load switching events: (a) inrush current in Cluster #1; (b) long transient with a plateau in Cluster #2.

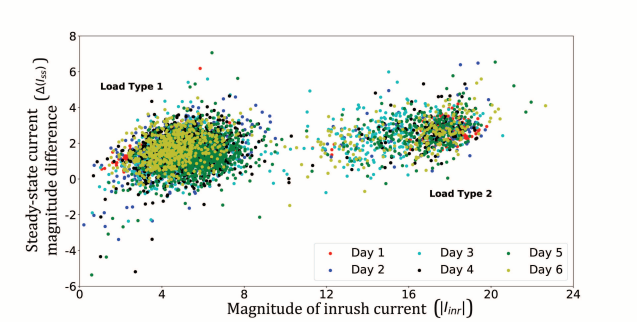

Fig. 2. Identifying two major load types based on Cluster #1.

1) Identifying Different Load Types: Fig. 1(a) shows an example for Cluster #1, which is the most frequent event in this system. It is the inrush current from load switching. The transient time of these events is less than 10 time slots, i.e., 83.3 msec, and one pinnacle which illustrates the magnitude of inrush current. Fig. 2 shows the scatter plot for the change in the steady-state current, i.e., $\Delta(I_{ss})$, versus the magnitude of inrush current, i.e., $|I_{inr}|$ during 6 different days. As it can be seen, Cluster #1 can it self be divided into two main sub-clusters which show two major types of loads in this cluster.

Fig. 1(b) shows an example for Cluster #2. It is for the load types that create much longer transient period to switch and creates a *plateau*; which is very different from the inrush current in Cluster #1 with a *pinnacle*. Fig. 3 shows a scatter plot for the events in Cluster #2. On the y-axis it shows the change in steady-state current, *before* and *after* the event, which is denoted by $\Delta(I_{ss})$. The x-axis is the length of the transient period of the event. There is a dense concentration area, where $\Delta(I_{ss})$ fluctuates at around 1.5 A. This observation empowers the system operator to more readily detect any abnormalities in this cluster, with regard to $\Delta(I_{ss})$ and transient duration, such as multiple simultaneous load switching.

2) Capacitor Bank State of Health Monitoring: Figs. 4(a) and (b) show examples of clusters #3 and #4, which are related to capacitor bank switching 'on' and switching 'off' events, respectively. Capacitor bank switching occurs on a daily basis.

Monitoring the switching actions of capacitors can not only keep the utility operator informed of switching status of the capacitor banks; it can also help to evaluate their state of

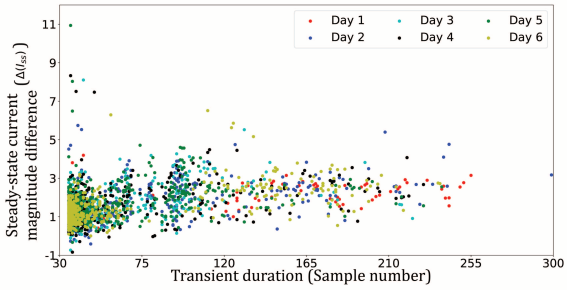

Fig. 3. Scatter plot for the events in Cluster #2 over 6 days.

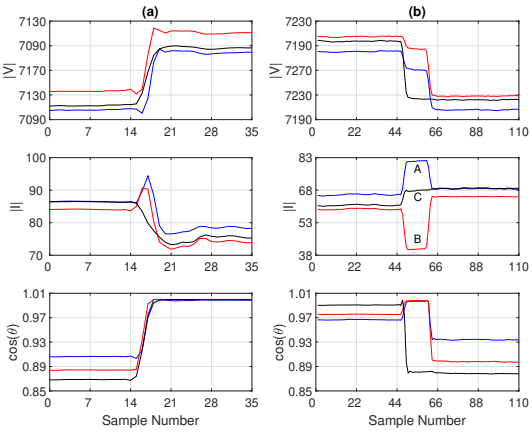

Fig. 4. Monitoring the operation and health of a capacitor bank based on Clusters #3 and #4: (a) switch on; (b) switch off.

health. For example, consider the capacitor bank switching off event in Fig. 4(b). We can see that there is a relatively long *overshoot* on Phase A current and a relatively long *undershoot* on Phase B current before the capacitor is de-energized. This is likely due to a *malfunction* in the switching control mechanism at the capacitor bank, c.f. [4]. By clustering all the capacitor switching events, we can conduct statistical analysis on the characteristics of such transient switching responses and dispatch the field crew to examine the capacitor bank switching controller and perform repairs.

3) Two-step Events: Fig. 5 shows an example of the special load in Cluster #5. This special type of load has two separate but subsequent steps. By using the proposed unsupervised event detection and unsupervised event clustering method we were able to capture it and identify its unique switching pattern that is repeated every time this event occurs.

4) Oscillations in Current Induced by Step Changes: Fig. 6 shows an example of an oscillation event in Cluster #6. These events always occur immediately after a particular pattern of a step up change event in the current magnitude (as we can see at the beginning of the Fig. 6(b)) that also is followed by an oscillation event which is magnified in Fig. 6(a). For this particular class of oscillatory events, we have observed that the median for the frequency of the oscillations is 5.17 Hz; while the median for the damping ratio of the oscillations is 2.64%. This information is valuable to the utility. In particular, such information that is obtained in an unsupervised fashion by our proposed algorithms, when combined with a subsequent field inspection by the utility crew members, can quickly lead to the

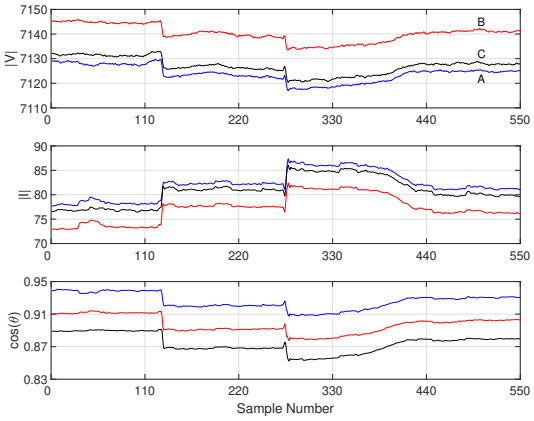

Fig. 5. An example for the two-step event in Cluster #5.

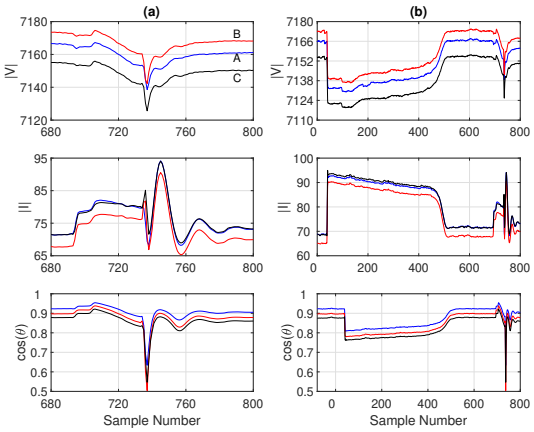

Fig. 6. An example for the oscillation event in Cluster #6: (a) oscillation in current; (b) the step change prior the oscillations.

best remedial action; as deemed necessary by the utility. This type of event causes the *highest transient power factor* change at this distribution feeder, when compared with all kinds of events that we have captured in this study. The amount of the transient change in power factor is 0.4.

F. Use Case Exposition: Three Clusters in Category II

Three clusters are identified in Category II; denoted by Cluster #7 to Cluster #9. Clusters #7 and #8 can help identify voltage events. Cluster #9 can identify voltage oscillations.

1) Voltage Events: Fig. 7(a) shows an example of Cluster #7, which is a transformer tap changing event. The events in this cluster inform the utility about voltage regulation status and the operation of tap-changers. Fig. 7(b) shows an example event in Cluster #8, which is a voltage event with a *plateau*. The transient shape of the voltage in Cluster #8 is similar to voltage changes in Cluster #2, see Fig. 1(b); however, these two events are different because there is no change in current phasors ($|I|$ and $cos(\theta)$) in the events in Cluster#8. The events in Clusters #7 and #8 are often initiated at transmission level.

2) Voltage Oscillation Events: Fig. 8 shows an example for an event in Cluster #9, which is a high frequency low magnitude event in $|V|$. Since there is no major change in current, this event can be due to two possible phenomena: 1) voltage oscillation from the upstream system; 2) temporary

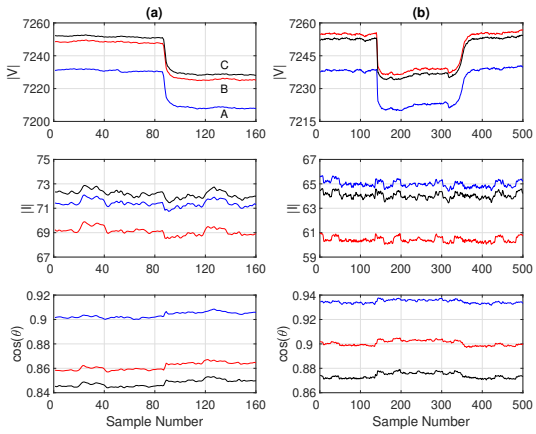

Fig. 7. Examples of voltage events: (a) transformer tap-changer in Cluster #7; (b) voltage plateau in Cluster #8.

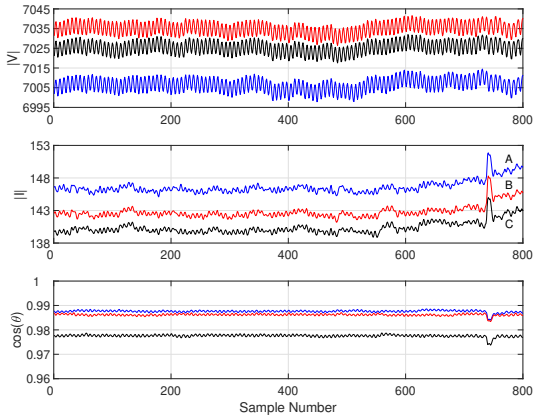

Fig. 8. An example for voltage oscillation event in Cluster #9.

malfunction in micro-PMU data reporting. The later can be considered as a possibility if it persists and if other micro-PMUs do not report a similar behavior. In that case, this can be used as an indicator to request micro-PMU diagnostics. Importantly, this cluster is a new cluster that is added by the active clustering method; which resulted from the significant difference between the samples in this cluster and the samples that were used during the offline training process.

G. Use Case Exposition: One Cluster in Category III

One cluster is identified in Category III; denoted by Cluster #10. Fig. 9 shows an example for this cluster. The events in this cluster affect only the current magnitude and power factor, rather than the voltage magnitude. It should be noted that, the pre-processing step in the proposed two-step clustering method helps to distinguish the events in Cluster #10 from the events in Clusters #2 and #5, despite their relatively high MMC.

H. Use Case Exposition: Three Clusters in Category IV

Three clusters are identified in Category IV; denoted by Clusters #11 to #13. The events in these clusters are *unbalanced*. Fig. 10 shows an example of the event in Cluster #11. This event is *not* detected by the event detection method in [11]; because that method fails to notice small changes in just one feature, i.e. in $|I_B|$. However, in our method, by using one GAN model for each feature, even small events are detected.

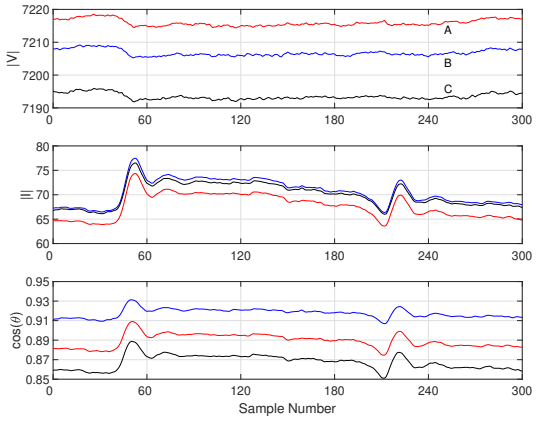

Fig. 9. An example for current oscillation event in Cluster #10.

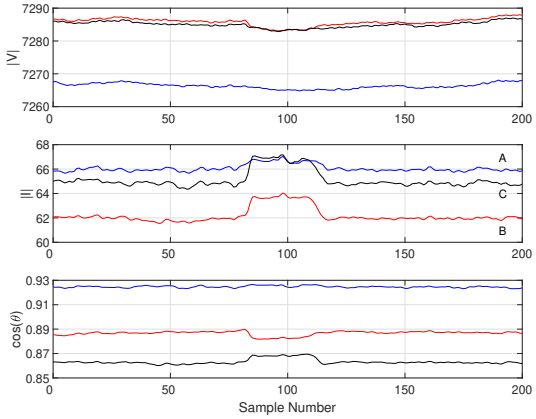

Fig. 10. An example for the unbalanced events in Cluster #11. The event affects the current magnitude of phases B and C.

I. Use Case Exposition: Three Clusters in Category V

Three clusters are identified in Category V; denoted by Clusters #14 to #16. They are all related to power factor events. An example for an event in Cluster #14 is shown in Fig. 11. It shows oscillations in power factor. There are also some minor oscillations, in the magnitudes of current and voltage during the same period. Other types of power factor events are also captured by the clusters in this category; not shown here. It should be mentioned that the clusters in this category were added by the active clustering; i.e., they were not among the initial clusters that we had obtained during the offline training process. The creation of these new clusters was triggered mainly because of their different detection vectors.

J. Special Sequence of Events

One of the applications of the proposed unsupervised methods is to analyze the shape, occurrence time and sequence of the detected and clustered events. Our analysis shows that certain events come in sequence. This is an important observation to enhance the predictability of the system, its dynamics, and its events. An example is shown in Fig. 12. It is a *super event* which consists of a sequence of several smaller events that belong to Clusters #6 and #10. This super event is first triggered by an event that belongs to Cluster #6, which we previously saw in Fig. 6. Then, after about 60 seconds, a series of over 100 events occur that all belong to Cluster

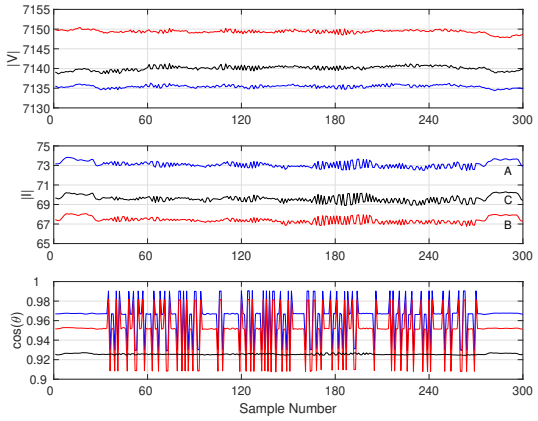

Fig. 11. An example for power factor event in Cluster #14.

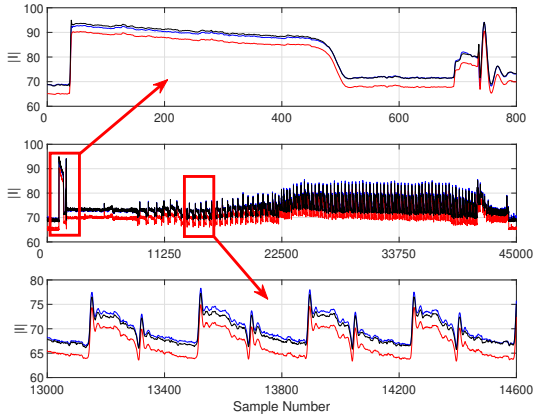

Fig. 12. An example for the special sequence of the events in the current magnitude that are repeated occasionally. It was captured based on the collaboration of Clusters #6 and #10.

#10. This sequence continues with a growing amplitude until it goes away. The exact same sequence of events occurred on the same day and around the same time each week.

K. Versatility of the Proposed Model

In order to show the versatility of the proposed event detection model, the developed model is applied to two other micro-PMU data set; which were not used to developed the existing model. First, a new data set that was from the *neighboring* feeder is used for training but during a different time of the year. In this case, all we needed to do was to slightly fine tune the original model, i.e., we only needed to re-train the last layer in the existing GAN models, based on the new training batches, which lead to faster model training by using pre-trained model. Thus, the training process in terms of computational time to achieve the equilibrium is around 14 minuets. It should be noted that, just like in the procedure mentioned in Section IV-B, a total of 1200 events were extracted manually within 6 hours. The MCC of the fine tuned model for this new data set is 0.9018.

Second, we used a micro-PMU data set from a *completely different* type of feeder. This time we used real-world micro-PMU data from solar distribution feeder in a solar farm; based on the data in [52]. The nature of the power distribution feeder in this second case is drastically different from the nature of the original power distribution feeder that serves loads; which has been the focus throughout this paper. For the case of this second data set, we were able to keep the proposed architecture of our model; but we had to re-train the model with the new data set. It should be mentioned that the structure and the hyper-parameters (except for epoch and batch size numbers) remained the same as in our original model. Nevertheless, the result was promising. The results and other details about the analysis of the events at this solar farm are available in [52].

V. Conclusions

A set of new unsupervised methods are proposed to detect and cluster different types of events in micro-PMU measurements. The test results based on real-world micro-PMU data confirm that the proposed event detection method, which works based on training a novel GAN model, outperforms the existing methods , in particular when it comes to detecting the events that may impact only a subset of the features or only a subset of the phases. Test results also show the effectiveness of the proposed two-step clustering method, compared to the other prevalent methods, due to the proposed choice of the similarity measure and also the proposed architecture that improves clustering accuracy. Moreover, the active nature of the proposed clustering method makes it capable of identifying new clusters of events on *an ongoing basis*. Once the events are detected and clustered, the results are used in various use case analysis. Statistical analysis as well as human expert knowledge are used to scrutinize the events in each cluster; to unmask different applications for the utility operator.

References

[1] H. Mohsenian-Rad, E. Stewart, and E. Cortez, "Distribution synchrophasors: Pairing big data with analytics to create actionable information," *IEEE Power and Energy Magazine*, vol. 16, no. 3, pp. 26–34, May 2018.

[2] A. Shahsavari, M. Farajollahi, E. Stewart, E. Cortez, and H. Mohsenian-Rad, "Situational awareness in distribution grid using micro-pmu data: A machine learning approach," *IEEE Trans. on Smart Grid*, vol. 10, pp. 6167–6177, Nov. 2019.

[3] Y. Seyedi, H. Karimi, and S. Grijalva, "Irregularity detection in output power of distributed energy resources using pmu data analytics in smart grids," *IEEE Trans. on Industrial Informatics*, vol. 15, no. 4, pp. 2222–2232, 2018.

[4] A. Shahsavari, M. Farajollahi, E. Stewart, A. von Meier, L. Alvarez, E. Cortez, and H. Mohsenian-Rad, "A data-driven analysis of capacitor bank operation at a distribution feeder using micro-pmu data," in *Proc. of the IEEE PES ISGT*, Washington, DC, Feb. 2017.

[5] E. M. Stewart, V. Hendrix, M. Chertkov, and D. Deka, "Integrated multi-scale data analytics and machine learning for the distribution grid and building-to-grid interface," Lawrence Livermore Lab, Tech. Rep., 2017.

[6] A. Shahsavari, M. Farajollahi, E. Stewart, E. Cortez, and H. Mohsenian-Rad, "A machine learning approach to event analysis in distribution feeders using distribution synchrophasors," in *IEEE PES SGSMA*, 2019.

[7] M. Farajollahi, A. Shahsavari, E. M. Stewart, and H. Mohsenian-Rad, "Locating the source of events in power distribution systems using micro-pmu data," *IEEE Trans. on Power Systems*, vol. 33, no. 6, pp. 6343–6354, 2018.

[8] Q. Cui and Y. Weng, "Enhance high impedance fault detection and location accuracy via μ-pmus," *IEEE Trans. on Smart Grid*, vol. 11, no. 1, pp. 797–809, 2019.

[9] S. Liu, Y. Zhao, Z. Lin, Y. Liu, Y. Ding, L. Yang, and S. Yi, "Data-driven event detection of power systems based on unequal-interval reduction of pmu data and local outlier factor," *IEEE Trans. on Smart Grid*, vol. 11, no. 2, pp. 1630–1643, 2020.

[10] Y. Zhou, R. Arghandeh, and C. J. Spanos, "Partial knowledge data-driven event detection for power distribution networks," *IEEE Trans. on Smart Grid*, vol. 9, no. 5, pp. 5152–5162, Sep. 2018.

[11] A. Aligholian, A. Shahsavari, E. Cortez, E. Stewart, and H. Mohsenian-Rad, "Event detection in micro-pmu data: A generative adversarial network scoring method," in *IEEE PES General Meeting*, August 2020.

[12] N. Duan and E. M. Stewart, "Frequency event categorization in power distribution systems using micro pmu measurements," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3043–3053, 2020.

[13] T. J. Swenson, E. Vrettos, J. Müller, and C. Gehbauer, "Open μpmu event dataset: Detection and characterization at lbnl campus," in *IEEE PES General Meeting (PESGM)*, 2019.

[14] C. Hannon, D. Deka, D. Jin, M. Vuffray, and A. Y. Lokhov, "Real-time anomaly detection and classification in streaming pmu data," *arXiv preprint arXiv:1911.06316*, 2019.

[15] S. Pandey, A. Srivastava, and B. Amidan, "A real time event detection, classification and localization using synchrophasor data," *IEEE Transactions on Power Systems*, 2020.

[16] M. Cui, J. Wang, A. R. Florita, and Y. Zhang, "Generalized graph laplacian based anomaly detection for spatiotemporal micropmu data," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 3960–3963, 2019.

[17] Y. Yuan, K. Dehghanpour, F. Bu, and Z. Wang, "Outage detection in partially observable distribution systems using smart meters and generative adversarial networks," *arXiv preprint arXiv:1912.04992*, 2019.

[18] P. Tehrani and M. Levorato, "Frequency-based multi task learning with attention mechanism for fault detection in power systems," *arXiv preprint arXiv:2009.06825*, 2020.

[19] E. Samani, P. Khaledian, A. Aligholian, E. Papalexakis, S. Cun, M. H. Nazari, and H. Mohsenian-Rad, "Anomaly detection in iot-based pir occupancy sensors to improve building energy efficiency," in *Proc. of IEEE PES ISGT*, February 2020.

[20] A. Aligholian, M. Farajollahi, and H. Mohsenian-Rad, "Unsupervised learning for online abnormality detection in smart meter data," in *IEEE PES General Meeting*, August 2019.

[21] H. Karimipour, A. Dehghantanha, R. M. Parizi, K.-K. R. Choo, and H. Leung, "A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids," *IEEE Access*, vol. 7, pp. 80 778–80 788, 2019.

[22] A. von Meier, E. Stewart, A. McEachern, M. Andersen, and L. Mehrmanesh, "Precision micro-synchrophasors for distribution systems: A summary of applications," *IEEE Trans. on Smart Grid*, vol. 8, no. 6, pp. 2926–2936, Nov 2017.

[23] M. F. McGranaghan and S. Santoso, "Challenges and trends in analyses of electric power quality measurement data," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, p. 057985, Dec 2007.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7354–7363.

[25] Vinay Jethava and D. P. Dubhashi, "Gans for LIFE: generative adversarial networks for likelihood free inference," *CoRR*, vol. abs/1711.11139, 2017.

[26] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[27] Z. Zhang, M. Li, and J. Yu, "On the convergence and mode collapse of gan," in *SIGGRAPH Asia 2018 Technical Briefs*, 2018, pp. 1–4.

[28] Dan Li, D. Chen, J. Goh, and S. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *CoRR*, vol. abs/1809.04758, 2018.

[29] Y. Tang and J. Yang, "Dynamic event monitoring using unsupervised feature learning towards smart grid big data," in *IJCNN*, May 2017.

[30] E. Klinginsmith, R. Barella, X. Zhao, and S. Wallace, "Unsupervised clustering on pmu data for event characterization on smart grid," in *2016 5th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*. IEEE, 2016, pp. 1–8.

[31] A. Shahsavari, M. Farajollahi, E. Stewart, C. Roberts, F. Megala, L. Alvarez, E. Cortez, and H. Mohsenian-Rad, "Autopsy on active distribution networks: A data-driven fault analysis using micro-pmu data," in *2017 North American Power Symposium (NAPS)*. IEEE, 2017, pp. 1–7.

[32] M. Farajollahi, A. Shahsavari, and H. Mohsenian-Rad, "Location identification of high impedance faults using synchronized harmonic phasors," in *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2017, pp. 1–5.

[33] M. Farajollahi, A. Shahsavari, and H. Mohsenian-Rad, "Tracking state estimation in distribution networks using distribution-level synchrophasor data," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2018, pp. 1–5.

[34] M. Farajollahi, A. Shahsavari, and H. Mohsenian-Rad, "Linear distribution system state estimation using synchrophasor data and pseudo-measurement," in *2019 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*. IEEE, 2019, pp. 1–6.

[35] A. Akrami, M. S. Asif, and H. Mohsenian-Rad, "Sparse distribution system state estimation: An approximate solution against low observability," in *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2020, pp. 1–5.

[36] M. Kamal, M. Farajollahi, H. Nazaripouya, and H. Mohsenian-Rad, "Cyberattacks against event-based analysis in micro-pmus: Attack models and counter measures," *IEEE Transactions on Smart Grid*, 2020.

[37] M. Kamal, M. Farajollahi, and H. Mohsenian-Rad, "Analysis of cyber attacks against micro-pmus: The case of event source location identification," in *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2020, pp. 1–5.

[38] M. Jamei, A. Scaglione, C. Roberts, E. Stewart, S. Peisert, C. McParland, and A. McEachern, "Anomaly detection using optimally placedμPMUsensors in distribution grids," *IEEE Trans. on Power Systems*, vol. 33, no. 4, pp. 3611–3623, July 2018.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NIPS*, Montreal, Canada, Dec. 2014.

[41] Naveen Kodali, J. D. Abernethy, J. Hays, and Z. Kira, "How to train your DRAGAN," *CoRR*, vol. abs/1705.07215, 2017.

[42] Ralph B. D'Agostino and A. Belanger, "A suggestion for using powerful and informative tests of normality," *The American Statistician*, vol. 44, no. 4, pp. 316–321, 1990.

[43] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.

[44] J. Gorodkin, "Comparing two k-category assignments by a k-category correlation coefficient," *Computational biology and chemistry*, vol. 28, no. 5-6, pp. 367–374, 2004.

[45] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, p. 6, 2020.

[46] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *proc. of the Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, CA, 1967.

[47] L. Kaufman and P. Rousseeuw, *Clustering by Means of Medoids*, ser. Delft University of Technology : reports of the Faculty of Technical Mathematics and Informatics, 1987.

[48] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi, "Low-complexity fuzzy relational clustering algorithms for web mining," *IEEE Trans. on Fuzzy Systems*, vol. 9, no. 4, pp. 595–607, Aug 2001.

[49] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *KDD Workshop*, 1994.

[50] M. Cuturi and M. Blondel, "Soft-DTW: a differentiable loss function for time-series," in *Proc. of International Conference on Machine Learning*, Sydney, Australia, Aug 2017.

[51] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, "Clustering algorithms: A comparative approach," *PLOS ONE*, vol. 14, no. 1, pp. 1–34, 01 2019.

[52] P. Khaledian, A. Aligholian, and H. Mohsenian-Rad, "Event-based analysis of solar power distribution feeder using micro-pmu measurements," in *Proc. of the IEEE PES Conference on Innovative Smart Grid Technologies (ISGT)*, Washington, DC, Feb. 2021, pp. 1–5. [Online]. Available: https://intra.ece.ucr.edu/ hamed/KhAMRcISGT2020.pdf