UNIVERSITY OF CALIFORNIA
RIVERSIDE

Unsupervised Data-Driven Event Analysis of Smart Grid Time-Series

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Armin Aligholian

June 2022

The Dissertation of Armin Aligholian is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I appreciate all of the help and guidance that I received from my advisor, Professor Hamed Mohsenian-Rad, who gave me the opportunity to grow, to be more organized and to think more critically.

I would like to thank Dr. Samet Oymak and Dr. Nael B. Abu-Ghazaleh for their support and being a part of my dissertation defense and qualifying exam committee. Also, I would like to thank all of my colleagues and friends in the Smart Grid Research Lab which made this journey precious and joyful.

Dedicated to my parents, my brother and my partner for all the support.

ABSTRACT OF THE DISSERTATION

Unsupervised Data-Driven Event Analysis of Smart Grid Time-Series

by

Armin Aligholian

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2022
Dr. Hamed Mohsenian-Rad, Chairperson

There have been major advancements in recent years to enhance situational awareness in power distribution systems by using advanced sensor technologies. Smart meters and distribution-level phasor measurement units (D-PMUs) are among the most common sensors that have been deployed recently in power distribution networks. Included in the captured time-series of the measurements from these sensors, there are "events", that are generally unscheduled, infrequent, and often unknown in their type and nature. Therefore, in practice, we often do not have any prior knowledge about the events until they occur. In this regard, such sensor measurements can be seen as time series that are in form of unlabeled data. Accordingly, in this thesis, we address the analysis of events in the selected types of smart grid time series by using unsupervised machine learning.

We start by the analysis of time series in smart meter data to extract events and abnormalities. Our analysis also includes extracting proper choices of features and methods. Next, we move to the analysis of the time series data from D-PMUs that have a much higher resolution and carry more information than the measurements from smart meters as they also measure phase angles. Accordingly, three versions of unsupervised event detection methods are developed, which work based on generative adversarial networks and deep recurrent neural networks. These methods, specifically focused on high frequency and small time series windows of one D-PMU data. Results based on real-world sensor data show that by learning normal behaviour of the system via the proposed methods, we can extract the events more accurately compared to the prevalent methods. Subsequently, a two-step unsupervised clustering method is also proposed, which works based on a linear mixed integer programming formulation to cluster events in time series from D-PMUs. Finally, to address the task of unsupervised event clustering for a situation within a low observable distribution system with only a handful of available D-PMUs, a novel unsupervised graph representation learning model is developed. The developed unsupervised clustering model, extracts the time domain features from the time series in fundamental and harmonics phasor measurements, and then it takes advantage of the system topology by using graph learning models to separate, characterize, and classify the events.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Power Distribution Networks

Electric power systems are large interconnected networks of different electric components, which transfer the generated electricity to the consumer. A traditional power system constitute of four sectors which are generation, transmission, distribution, and consumption. Typically, electricity is generated by large power plants, then, a meshed network of transmission lines transfer high voltage electricity to all across the network with different distances. Afterwards, many regional networks, a.k.a. power distribution systems, deliver the electricity to the final consumer. Common consumers are residential, commercial and industrial loads. In this thesis, we focus on the distribution system, which can contain many substations connected to the transmission system. An example of power distribution system is shown in Fig. 1.1, which is a 8-bus system and it includes multiple measurement units such as smart meters and distribution-level phasor measurement units (D-PMUs).

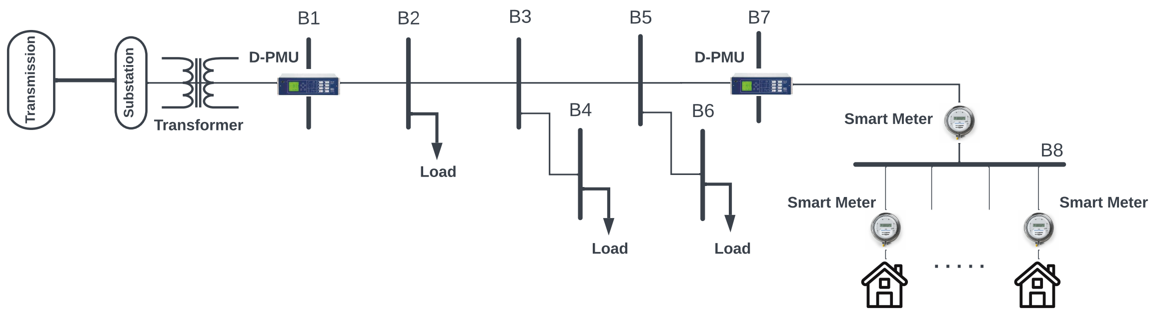


Figure 1.1: An example of power distribution system.

1.2 Sensor Measurements in Power Distribution

Power distribution systems are becoming more active and dynamic due to the increasing penetration of distributed energy sources, electric vehicles, dynamic loads, etc. This gives rise to various monitoring and control issues. Distribution service providers implement different sensors in the power system network to have proper system awareness and visibility in the electric grid for better decision making. There are many different types of sensors used in the power distribution systems, which can measure variety of quantities such as voltage and current RMS values, power and energy consumption, voltage and current phasors and voltage and current wave forms for power quality monitoring [1]. The reporting rate of these sensors can be vary from hours to micro seconds. Among these sensors, smart meters and distribution-level phasor measurement units, a.k.a, micro-PMUs are the most common ones and we analyze them in this thesis.

1.3 Time Series from Smart Meters

Smart meter sensors, measure and record the electricity consumption data that can communicate remotely with utilities. The deployment of smart meters has provided system operators with an unprecedented level of visibility over the distribution networks and customer loads, with a multitude of applications [2]. Depend on the type of the smart meters the reporting rate for these devices can be vary from seconds to hours. Based on the EIA annual electric power industry report in 2020 [3], U.S. electric utilities had about 102.9 million advanced smart metering infrastructure installations. About 88% of the smart meter installations were in the residential sector.

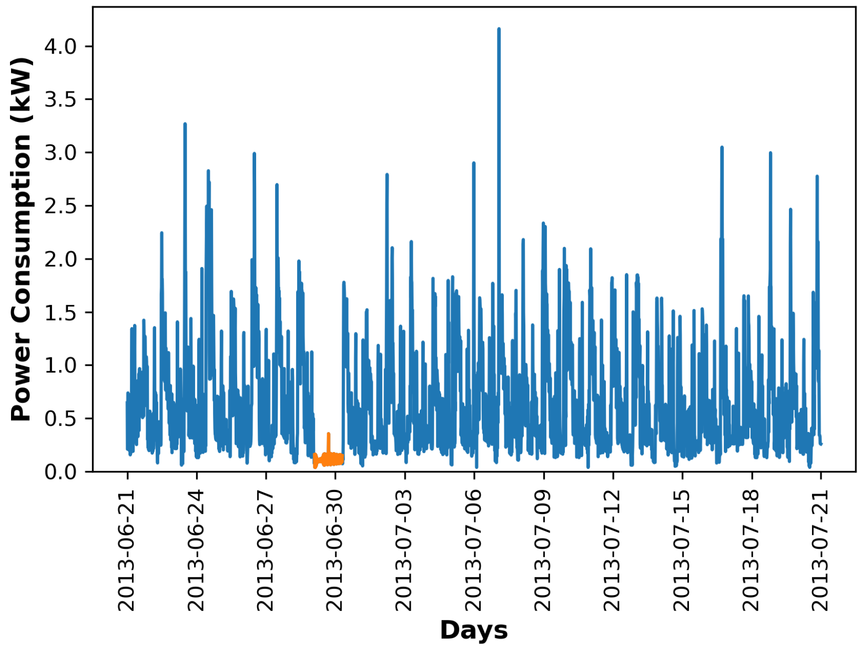


Figure 1.2: One month of power consumption for a household measured by a smart meter sensor in the Pecan street project, with an abnormality on June 30, 2013.

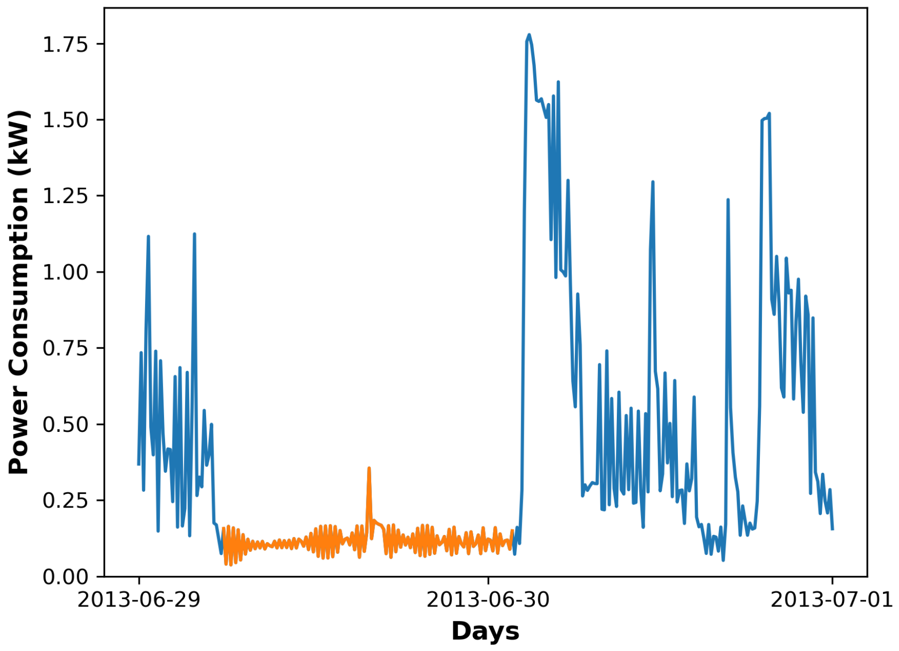


Figure 1.3: The abnormality in the power consumption for the household shown in Fig. 1.2 between June 29 and 30, 2013.

However, it is now a challenge to handle the tremendous growth in the volume and velocity of the data that is generated by the smart meters in the load sector. Therefore, it is necessary to find ways to extract the most useful parts of the data and transform them to actionable information. One of the essential data analysis is to find abnormalities among the huge stream of data. Abnormality in the context of this thesis is broadly defined as any unusual electricity consumption *instance* or *trend* that falls outside of the normal power consumption patterns for a load, whether in terms of magnitude, time, duration, etc.

The analysis of abnormalities in smart meter data has applications in load forecasting, cyber security, fault detection, electricity theft detection, demand response, etc. Abnormality detection in smart meter data is challenging due to high volume of data, substantially different load profile of residential houses, they are unknown and there is no prior knowledge about them. There are other effective factors that are effective on the result of abnormality detection in the smart meters such as weather condition and contextual features. To address these issues, this thesis evaluates many unsupervised data driven approaches, alongside with many different and prevalent features to find a trade off between using appropriate features and type of data driven model for each category of abnormalities, i.e. instance and trend. The test cases in this study are based on the smart meter data from Pecan Street project in Austin, TX [4]. The collected data is for 92 consecutive days for five households with resolution of 15 minutes. For instance, Fig 1.2 shows one month power consumption in a household which captured by smart meter sensors. Also, the highlighted orange curve is an abnormality in the data on June 30, 2013, which the magnified version of this abnormality is shown in Fig. 1.3.

1.4 Time Series from Micro-PMUs

In order to capture the real time dynamics of the system for better situational awareness, distribution system operators can not use smart meter data due to its limited quantity measurement as well as low resolution data. For better power system visibility and reliability, distribution system operators use distribution-level phasor measurement units, a.k.a., micro-PMU [5]. Micro-PMU can be connected to single- or three-phase distribution circuits to measure magnitudes and phase angles of the voltage and current phasors normally at 120 readings per second and it is GPS time-referenced. This is 108,000 times faster than the reporting rate of a typical smart meter or 124,416,000 readings per micro-PMU per day. One of the emerging applications of micro-PMUs is to study “events” in power distribution systems. Here, an event is defined rather broadly and may refer to 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. [6, 7, 8].

For instance, Fig. 1.4 shows magnitude of voltage phasor time series in three phases that are captured with an actual micro-PMU in Riverside, CA. The reporting rate of this micro-PMU is 120 Hz and Fig. 1.4 shows 15000 sample points or 125 seconds in each time series. As we can see there is a voltage dip event in all three phases around sample number 7025 which is magnified in Fig. 1.5.

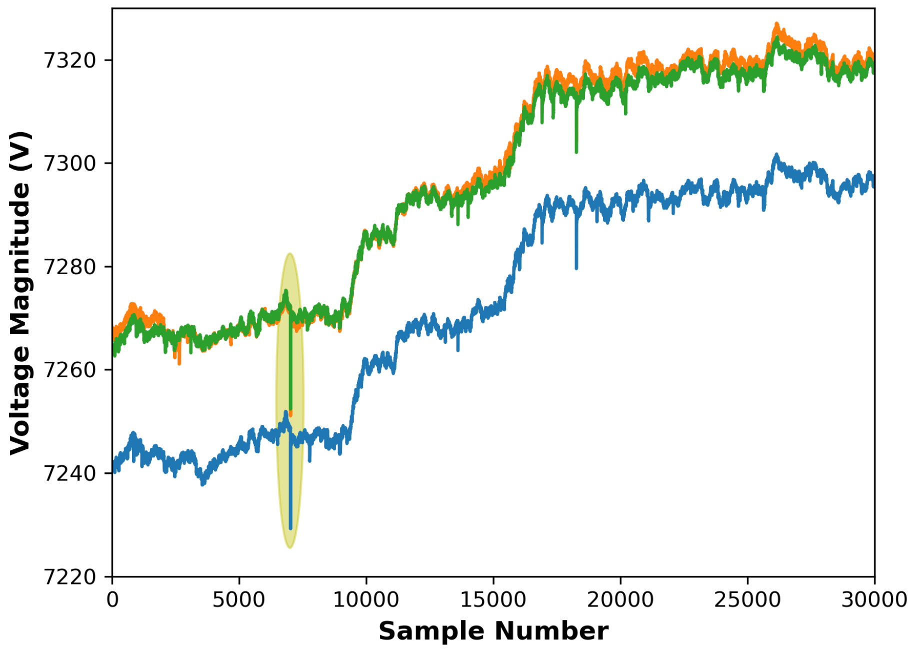


Figure 1.4: Three phase voltage magnitude measurement that captured by a micro-PMU for 125 seconds. A voltage dip event in all three phases highlighted in this figure which occur around sample number 7025.

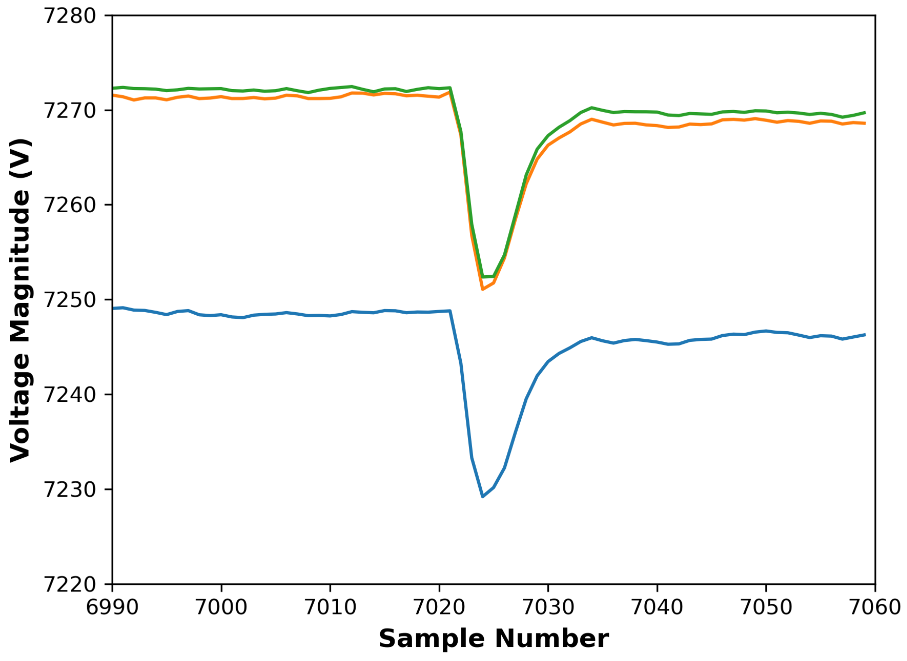


Figure 1.5: The voltage event highlighted in Fig. 1.2

Hence, in a common high resolution time series captured by each micro-PMU, events are considered as anomalies as their frequency of happening is very low. It should be mention that using anomaly to describe events in micro-PMU time series does not refer to something extraordinary or destructive, yet any rare behaviour compare to the normal state of the system. Event-based studies of micro-PMU measurements have a wide range of use cases, such as in situational awareness [8], equipment health diagnostics, such as for inverters [9], capacitor banks [10], transformers [11], distribution-level oscillation detection and analysis [12], fault analysis and fault location [13]. Before one can do any event-based analysis, including for the above use cases in [8, 13, 10, 11, 12, 9], 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.

1.5 Research Questions

Given the enormous size of measurement data that is generated by micro-PMUs and smart meter data, the challenges that we listed above call for developing effective data-driven techniques that are automated and require minimal prior knowledge. In this thesis we want to answer the questions below:

1. By having many residential load profiles for a given period of time, how we can extract different type abnormalities and what is the optimal combination of model and features for this goal?

2. Considering one available micro-PMU time series data, how we can extract the events without having any prior knowledge about the events?

3. After detecting the events from the micro-PMU time series, how we can cluster them such that it can be useful for the system operator analysis and what are the use cases?

4. Given a distribution system with a locationally-scarce data availability (few installed micro-PMUs in the distribution system), how we can cluster all of the detected events, considering topology of the system and harmonics data?

1.6 Literature Review

This section presents the related studies in three main areas, which are: related work in abnormality detection for smart meter time series, event detection for micro-PMU time series and event clustering for micro-PMU time series.

1.6.1 Abnormality Detection in Smart Meter Time Series

The analysis of abnormalities in smart meter data streams is of great interest to several applications, such as load forecasting [14], cyber attack detection [15], fault and outage detection [16], electricity theft detection [17], demand response [14], etc.

Our approach in this thesis is based on machine learning. Accordingly, this study is in its broad sense comparable with those in [18, 19, 20, 21]. A deep semi-supervised convolutional neural network with confidence sampling is proposed in [18]. Also, a supervised ensemble-based method with sliding window is proposed in [19]. However, when it comes to abnormality detection, we must deal with an inherently unsupervised learning problem because abnormalities do *not* have a known paradigm; they are rather determined in comparison with the history of data. Therefore, the usage of unsupervised learning methods are more practical than supervised and semi-supervised methods. In [20], a prediction-based unsupervised abnormality detection method is proposed that comprises a dynamic regression model and an adaptive abnormality threshold. In [21], an unsupervised clustering-based algorithm on the low-dimensional dissimilarity matrix is used to detect irregular power consumption. However, these two studies do not consider the role of feature selection in training their model. They also only consider specific types of abnormality. Hence, in this thesis in order to extract events or abnormalities from smart meter data we have evaluated combinations of different unsupervised machine learning models alongside different categories of features, to show the importance of specific combinations for detecting various types of abnormalities.

1.6.2 Event Detection in Micro-PMU Time Series

The literature on event detection in micro-PMU data can be divided into two broad classes; namely *statistical methods*, such as in [22, 8, 23], and *machine learning methods*, such as in [24, 25, 26, 27]. For example, in [8], 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. In [23], the analysis of the inverse power flow problem is combined with the turning point test method to detect events. In [22], the physical equations of the power distribution circuits are combined with techniques from statistical quality control in order to develop a hierarchical anomaly detection architecture that uses data from optimally placed micro-PMUs.

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 [28] and fault detection [29], IoT-based occupancy sensor unusual behaviour [30]. In [24], a machine learning method, called ensembles of bundle classifiers, is used to train multiple classifiers based on multiple instances of the same predetermined event, so that the patterns of that event are learned in order to detect more instances of that event in the micro-PMU data. In [25], a hidden structure semi-supervised machine learning model is established to combine micro-PMU data for both labeled and unlabeled events. A parametric dual optimization procedure is used to tackle the non-convex learning objective function.

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 [26, 25]. The event detection method in [24] is based on *supervised* machine learning. Also, the method in [25] is based on *semi-supervised* machine learning. In both cases, full or partial *expert knowledge* is needed in order to establish the event detection tool. 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 [26, 25]. 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 [31], significant known events such as three phase fault or cap bank switching [32] or cyber attacks [33].

In contrast, the event detection method in this thesis 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. [34, 7, 8]. In [33], 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 thesis. Similarly, in [32] 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 [35], are based on determining the graph similarity between the sample windows of the micro-PMU data. We used the method in [35] as a benchmark to evaluate the performance of our event detection approach.

Generative Adversarial Networks (GANs) are broadly studied in areas such as image generation [36], high-dimensional likelihood-free inference [37], medical time-series generation [38], 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 [39] to detect bogus samples, cyber attacks, or general time-series anomalies [40]. Here, the ultimate goal of the discriminator is to distinguish normal from abnormal samples; Thus, in this thesis, 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 by generator.

1.6.3 Event Clustering in Micro-PMU Time Series

The literature on data-driven event types analysis in distribution-level phasor measurements can be generally divided into two categories. First, there are studies that use supervised learning, i.e. event classification, such as those in [41, 42, 8]. They require prior labeling of the events in the training data set, which may not be doable in practice.

The second group are the studies that use unsupervised learning, i.e., they attempt to cluster the events by grouping their distinctive characteristics. In [43], k-means clustering and Ward's clustering with focus on clustering voltage sag events is proposed. In [44], the authors used an unsupervised clustering method, with focus on some specific faults; such as single-line-to-ground versus line-to-line faults.

None of the unsupervised learning methods in [43, 44] takes into account the information on the topology of the power distribution network. Furthermore, none of them takes into account the availability of harmonic phasor measurements in addition to the fundamental phasor measurements.

In order to consider the network topology into the process of event clustering, one plausible way is to implement graph-based knowledge. Graph theory and more generally graph-based analysis have been considered in power systems, such as for event detection [45], event location identification [46, 47], data recovery and prediction [48], and the analysis of power system dynamics [49].

Recent literature also includes the use of GNNs, to address some prevalent power system issues, including the analysis of events. In [50], a *supervised* GNN-based method is proposed for event classification in power transmission systems. No knowledge about the topology of the power transmission network is assumed to be available; therefore, full connectivity is assumed in the graph-level analysis. The events are labeled based on the data on voltage and frequency. In [51], the authors used Graph Convolutional Network (GCN) for short term voltage stability assessment. Importantly, neither of the studies in [51] or [50] consider the issue of locational scarcity among the sensors within a known network topology. None of them also considers using harmonic phasor measurements.

Finally, there is a rich literature on the analysis of power quality events using measurements related to harmonics. The focus is usually on the analysis of waveform measurements, such as in [52, 53, 54]. For example, in [54], the authors proposed an AED to extract the features for clustering the daily variations in steady-state voltage harmonics. Interestingly, while we *do* use H-PMU measurements, our focus is *not* on the typical analysis of steady-state harmonics. Instead, we use the harmonic phasor measurements in addition to the fundamental phasor measurements to better capture the *distinctive transient signatures* in various events in power distribution systems under locationally-sparse phasor measurements. Furthermore, the prior studies in this field, including those in [52, 53, 54], do *not* consider using the information about the network topology.

1.7 Summary of Contributions

The research questions in Section 1.5, are expounded in the following chapters and a summary of contributions for each topic is mentioned below.

1.7.1 Smart Meter Abnormality Detection

The main contributions of Chapter 2 are as follows:

1. This thesis provides a systematic comparative study of four different *unsupervised* machine learning methods to understand how different methods can best serve to detect different types of abnormalities in real-world smart meter data. Specifically, we examine load prediction regression-based, load prediction neural-network-based, clustered-based, and projection-based methods for abnormality detection and compare their performance.

2. Different features are investigated for different methods to obtain the *best combination of features* for each method. An important conclusion is that, when it comes to historical load features, they are useful in prediction-based methods for the purpose of finding abnormal load *trends*; while they are also useful in cluster-based methods for the purpose of finding abnormal load *instances*. In addition, cluster-based methods can use a proper combination of historical load features, contextual features, and environmental features to simultaneously identify both abnormal load trends and abnormal load instances.

3. To speed up detection, all methods are implemented in *online* mode, where the models are updated upon the arrival of new data. To the best of our knowledge, this is the first study to address the application of Isolation Forest (IF) and Lightweight On-line Detector of Anomalies (LODA), as two computationally efficient online methods, to do abnormality detection in smart meter data.

1.7.2 A Single Micro-PMU Event Detection and Event Clustering

The main contributions of Chapter 3 are as follows:

4. Three novel unsupervised event detection methods are 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 methods compared to multiple state-of-the-art methods in the literature.

5. All methods are *unsupervised* deep learning methods, which require no or minimal human knowledge; which makes them suitable for automated and scalable operation. Furthermore, they can detect both *point-signatures* and *group-signatures* in micro-PMU data. This is an important capability because of the unbalanced nature of power distribution circuits; where many events may affect only a subset of the features on only one or two phases.

6. Real-world micro-PMU data is used to evaluate the proposed event detection methods. In order to create a reference, first, more than 1000 events of different kinds are extracted manually from the micro-PMU data within a given period of time. It is

observed that both the basic and the enhanced methods highly outperform a prevalent statistical method. The advantage is particularly major for the events that cause *small changes in magnitude.*

7. 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.

8. 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.

1.7.3 Multiple Micro-PMU Event Clustering

The main contributions of Chapter 4 are as follows:

9. A new unsupervised two-step *spatio-temporal* feature learning method is developed based on Graph Neural Networks (GNN) and Auto Encoder Decoder (AED) to capture the locational and temporal information of the sensors on the distribution network. The time series of the measurements at each bus are transferred to a lower dimension latent space. Accordingly, a graph learning method is implemented to the obtained embedding vectors to extract the *topology-related features* for event clustering. To the best of our knowledge, this is the first time that a physics-aware graph learning method is used for utilizing D-PMU (or H-PMU) data in event clustering.

10. A graph-level representation learning is developed which uses *local-global mutual information maximization* to learn the structural connection of the event data with its node-level representation. To extract the shared information between graph-level and node-level embedding that is sensitive to the graph topology, a *negative graph sampling* based on a random network tree structure is proposed. This makes the proposed GNN more aware of the system topology, by encoding aspects of the data that are shared across different local nodes, and proper adversarial sampling for mutual information estimation.

11. Incorporating the 3rd and the 5th harmonic phasor measurements along with the fundamental phasor measurements into the above aforementioned design. This is done by training a separate AED module is trained for each individual harmonic order. Then, the new aggregated vectors are used as additional input to the proposed graph learning process in order to capture the underlying locational patterns for each event, by taking into account both fundamental and harmonic phasor measurements.

Chapter 2

Unsupervised Abnormality Detection in Smart Meter Time Series

2.1 Abstract

In this section, we describe the unsupervised online abnormality detection approach for smart meter data. This section aims to evaluate the performance of four unsupervised machine learning methods for abnormality detection on real-world smart meter data, namely prediction-based regression, prediction-based neural network, clustered-based, and projection-based methods. Different types of features, such as load-based, contextual, and environmental, are investigated to construct the data-driven models. It is shown that different abnormality detection methods have different ability for detecting different types

of abnormalities; and their performance depends on the set of features used for training the method. Accordingly, different types of features are scrutinized for each abnormality detection method.

2.2 Feature Selection

A key component of any data-driven study is feature selection. Electricity power consumption depends on diverse types of features which can be used to study its characteristics. Broadly speaking, the features of electricity power consumption data can be categorized into three generic groups: load-based features, contextual features, and environmental features.

2.2.1 Load Based Features

These features account for the power consumption of residential household in different time steps. They are obtained from historical power consumption data with different time lags. The followings are the set of load based features that we consider in this study:

$$\begin{aligned} L^t &= \{P^t, P_Y^t, P_W^t, P_M^t\}, \\ L^w &= \{P^{t-24}, P^{t-23}, \cdots, P^{t-1}\}, \end{aligned} \tag{2.1}$$

where L^t is the set of historical load data at time t. In this set, P^t is power consumption at time t, P_Y^t is power consumption yesterday at time t, P_W^t is power consumption in the last two weeks at time t, and P_M^t is the mean of power consumption at time t. As for L^w, it is the set of previous 24 hours.

2.2.2 Contextual Features

These features are not specific to power consumption, but they do have indirect impact on power consumption. Time of the day, day of the week, weekends versus weekdays flag, holidays, and season of the year are instances of contextual information, as listed below:

$$C = \{T_d^t, D_w^t, W_s^t, H^t, S^t\}. \tag{2.2}$$

2.2.3 Environmental Features

Electricity consumption in some appliances such as heating ventilation and air conditioning (HVAC) systems depend on some environmental features such as temperature. Therefore, total power consumption of household are affected by these features which in this study considered as set E and comprises of temperature ($Temp^t$) and humidity (Hum^t) factors as illustrated below:

$$E = \{Temp^t, Hum^t\}. \tag{2.3}$$

These three features can be correlated. This may affect the quality of the learning process. Thus, we must study the effect of different feature combinations on each detection method, in order to customize features with respect to each model.

2.3 Abnormality Detection Techniques

Importantly, the abnormality detection problem does *not* have a known paradigm; therefore, it is inherently an unsupervised learning problem. This is more so when it comes to smart meter data, because we must *explore* the type of abnormalities that may arise in such data streams, along with the potential applications of detecting such different abnormalities. It should be added that, for analyzing “unusual” load patterns of costumers, we do not have specific pre-determined labels.

Online unsupervised learning methods are updated as soon as they see new data; thus, *they can learn new patterns and the changes in trends*, such as due to seasonal changes. Moreover, these methods can be implemented in the real time in order to detect abnormalities quickly. In this study, we implement four unsupervised online abnormality detection methods:

2.3.1 Load Prediction with Regression (LPBSVR)

This method works based on the comparison between the predicted and the actual power consumption. Accordingly, it is required to be built upon a prediction method. In this method, the prediction of power consumption is done using Support Vector Regression (SVR). SVR is a regression model which tries to minimize errors associated with the support vectors, so the prediction model is trained based on the outliers [55]. Therefore, this method is suitable for abnormality detection, where the abnormal patterns are treated as outliers. The SVR optimization problem can be formulated as follows [55]:

$$\begin{aligned} &\underset{W,b,\epsilon_i,\epsilon_i^*\geq 0}{\textbf{minimize}} \;\; \frac{1}{2}W^2 + C\sum_{i=1}^{N}(\epsilon_i+\epsilon_i^*) \\ &\textbf{subject to} \;\; y_i - W.x_i - b \leq \epsilon + \epsilon_i \\ &\qquad\qquad\quad W.x_i + b - y_i \leq \epsilon + \epsilon_i^*, \qquad \epsilon_i, \epsilon_i^* \geq 0; \end{aligned} \tag{2.4}$$

where W and b are the coefficient vector and intercept of the regression line which are

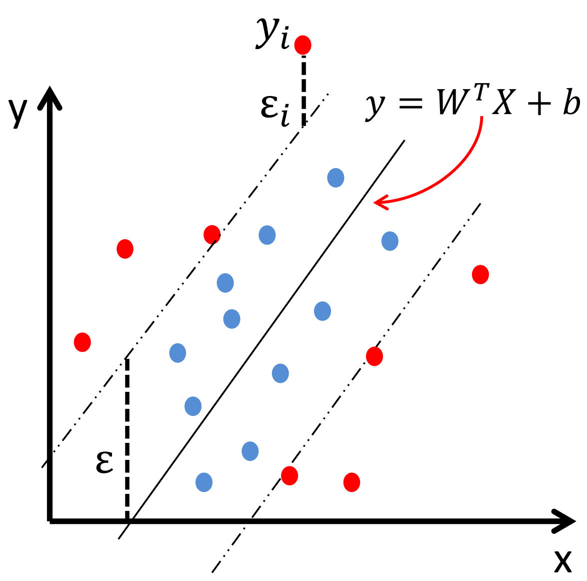


Figure 2.1: The soft margin loss setting of SVR.

unknown and act as optimization variables. Parameter C is the penalty cost, and ϵ_i as well as ϵ_i^* are the distances of support vectors and the two marginal lines. Also, y_i is the data output and ϵ is the marginal distance.

Based on the obtained regression model using SVR, the residual can be calculated as the difference between real electricity consumption of each data and the predicted data in each time slots. These residuals are characterized with a probability distribution function (PDF), which can be used to detect the outliers data. For example, given that the PDF of residuals is a normal distribution with mean μ and variance σ, the data that falls outside of the $[\mu - 3\sigma, \mu + 3\sigma]$ span can be considered as outliers, i.e., abnormality data and outliers.

2.3.2 Load Prediction with Neural Network (LPBNN)

As far as sole load prediction is concerned, neural network (NN) is proven to be a powerful data-driven tool, e.g., see [56]. Therefore, they can be used to develop a load prediction-based abnormality detection method. Other than the method of prediction, LPBNN is similar to LPBSVR. but the prediction is now done using a neural network [57]. We examined different neural network configurations, with 1 to seven hidden layers, three to thirteen nodes in each hidden layer, and both Relu and sigmoid as activation functions. The best results are then used in terms of prediction accuracy, with respect to MSE.

For each new reading from the smart meters, it first passes to the trained NN model to predict power consumption. Next, the residual is obtained; and if it is out of the mentioned span, then it is labeled as abnormal. The model is updated after making decision for each new data: if the new data is labeled as normal, then it is used as a new training data to update the NN model and residual PDF; otherwise, i.e., if the new data is abnormal; then it is reported and is not used to update the NN model. This process exactly implement for the regression based model which in that case SVR model are updated.

2.3.3 Clustered Based Method

In this method, the whole set of available data is clustered into two sets of "abnormal data" and "normal data". Some examples of cluster base methods are K Nearest Neighborhood (KNN) [58] and Local Outlier Factor (LOF) [59]. However, in this section, we use the isolated forest (IF) method [60]. The basic idea of IF is to isolate instances without calculating any type of distance among measurements. This helps enhance computational time and online detection. IF utilizes two main characteristics when it comes to abnormalities: a) abnormal data is very rare; b) certain features of abnormal data are very different from those of normal data. Clustering is done using binary tree clustering. Because of susceptibility to isolation, anomalies tend to be isolated closer to the root of the binary tree. There are two reasons for that: first, instances with obvious feature value are tend to be divided in the early partitioning process; Second, in different parts which contains anomalies, less anomalies creates fewer partitions which causes shorter paths in the tree. It is worth mentioning that path length of each terminal node in the tree is obtained by adding together all edges in the tree from root to the terminal node which an instance traverse in a specific tree.

2.3.4 Projection Based Method

Input space is projected to a subspace by using projection vectors mainly for dimension reduction. Principle Component Analysis (PCA) and LODA [61] are two common projection based methods for abnormality detection. In this study, we use LODA, due to its computational efficiency for random sparse projection. It is based on ensemble of some random sparse projection of feature vector. It works by first generating k sparse projection vectors with $\sqrt{d}$ non-zero element where, d is the number of features for the input data. After that, for each input and projection vectors we obtain their projected value by their inner product or $z_i = x_j^T w_i$, where z_i is the projected value of input vector x_j^T on projection vector w_i.

The projected values of a training data with respect to a projection vector w_i, give us a one-dimensional set which is used to obtain a histogram for each set. Hence, we have k one-dimensional histogram from training data. The LODA output can be defined as negative log-likelihood of the sample data:

$$f(x) = -\frac{1}{k}\sum_{i=1}^{k} \log \hat{p}_i(x^T w_i). \tag{2.5}$$

A higher value of $f(x)$ indicates a lower probability of the sample being abnormal. Also, $\hat{p}_i$ denotes the respective probability of projected value of vector w_i and input x. For online abnormality detection of each input data, the projection value in each w_i is calculated and the respective $\hat{p}_i$ is found by it's histogram. Therefore, the LODA output of input data can be found based on trained histograms. By comparing this value with a certain threshold, the input data is labeled as normal or abnormal data. If the input data is recognized as abnormal, then it is used to update the histograms.

The number of bins in LODA for each histogram is calculated through the following optimization problem [61]:

$$\begin{aligned} &\underset{W,b}{\textbf{maximize}} \sum_{i=1}^{b} n_i \log \frac{bn_i}{N} - \left[b - 1 + (\log b)^{2.5}\right] \\ &\textbf{subject to} \;\; N = \sum_{i=1}^{b} n_i, \end{aligned} \tag{2.6}$$

where N is the total number of samples, n_i is the number of samples in the i^{th} bin, and b is the total number of bins. This optimization problem is solved for each histogram individually. As another important parameter in LODA, the number of sparse projection vectors is calculated as:

$$\hat{\sigma}_k = \frac{1}{N}\sum_{i=1}^{N} f_{k+1}(x_i) - f_k(x_i). \tag{2.7}$$

where k is the number of histograms and the optimum value of this parameter can be determined by equation as arg $\underset{k}{min} \frac{\hat{\sigma}_k}{\hat{\sigma}_1}$.

2.4 Case Studies

The test cases in this section are based on the smart meter data from Pecan Street project in Austin, TX [4]. The collected data is for 92 consecutive days for five households with resolution of 15 minutes. After pre-processing and cleansing, the data is divided into 70% train data and 30% test data, respectively. As we mentioned before, abnormality detection methods in this thesis are inherently an unsupervised learning process. Therefore, since our data has no per-determined labels for abnormality, we need to obtain a benchmark to define unusual electricity consumption.

2.4.1 Defining Abnormality in Electricity Consumption

Unusual electricity consumption can have different signatures and different duration of time. In order to capture abnormalities of different lengths, we use *moving windows of different sizes* on recent data and compare the data in the most recent window with those in the previous windows. By examining various experimental data in the database, we figured out that the power consumption data over the last *three-weeks* could efficiently show the trend of historical data. Therefore, by taking the average of each three-weeks period of data, we can construct a model to capture the trends of data over time. By conducting such comparison, we obtain a set of residual data for any window size and then fit a normal distribution function to the residuals with mean μ_p and standard deviation σ_p.

For any residual that deviates from $3\sigma_p$, we consider it as an unusual trend or abnormality data. For the points which are determined in several moving windows, we consider the *largest consecutive window.* Such window can be obtained by analyzing residual curves related to each window size. After that, we select the curve with one peak in the detected points. Essentially, the peak point among all detected abnormalities is the size of the largest window that detected the abnormality.

Fig. 2.2 shows two consecutive abnormalities on one day which are captured by *two different window size.* This figure also depicts electricity consumption of the day, mean of the same day for the last three-weeks, and the residuals for the two detection window sizes. Note that, the window with size 3 detect the whole span of time slots $44 - 57$ but as the window with size 13 is a larger detector, and we have one peak (rather than several peaks in the window with size 3) in the detected span, we choose the later window as the period of abnormality.

2.4.2 Comparing Four Methods

All the proposed methods are applied on the available test data. In the "base case" all the three categories of features, i.e., load related, contextual and environmental, are applied to the models to detect abnormalities. Fig. 2.3 shows the performance of the four methods for part of the test data. The benchmark abnormalities are marked on the data curves using triangles. The detected abnormalities time slots for each methods have been illustrated with different shapes below the curves.

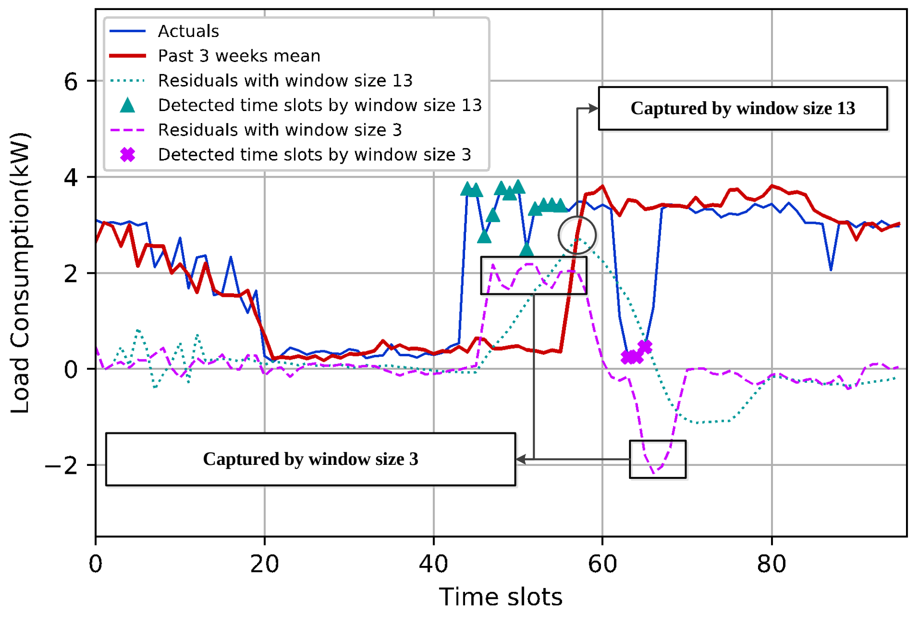


Figure 2.2: Abnormality benchmark based on statistical model.

To compare the four methods, we use the Matthews Correlation Coefficient (MCC) which is defined as [62]:

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP+TF)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.8}$$

Here, TP, TN, FP, FN are true positive (correctly identified), true negative (correctly rejected), false positive (incorrectly identified) and false negative (incorrectly rejected), respectively. MCC is broadly used as a measure of accuracy in binary classification, which essentially includes abnormality detection as a special case. MCC score is between -1 (the worst performance) and 1 (the best performance).

Based on the above results, IF has the best performance with MCC equal to 0.81; while MCC for LODA, LPBSVR and LPBNN is 0.54, 0.49 and 0.47, respectively. In the base case, all features are used which gives the best prediction result, i.e. the lowest MSE.

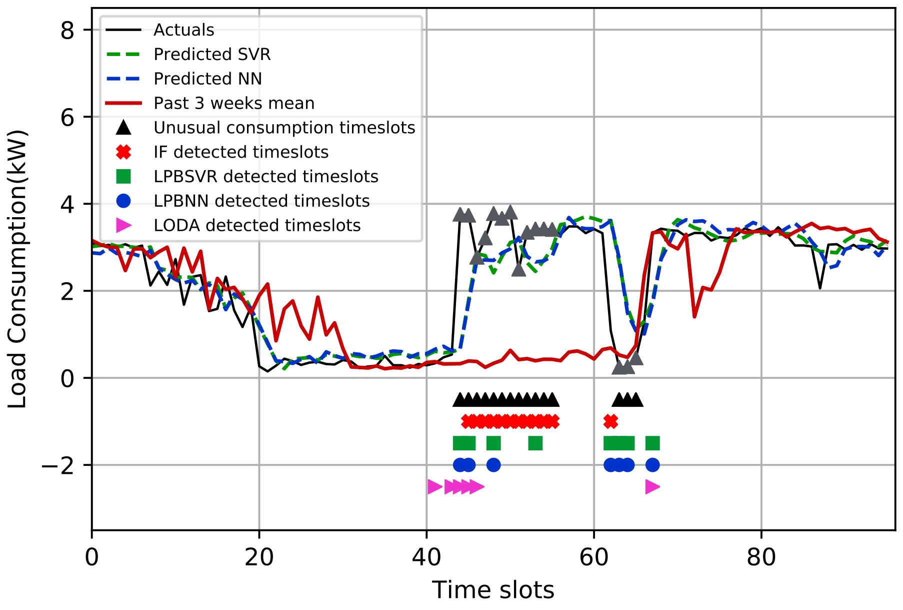


Figure 2.3: Comparison of different abnormality detection methods with benchmark in the base case when all features are utilized.

However, despite having good prediction performance, LPBSVR and LPBNN have poor performance in detecting abnormality. This is due to comparing the consumption with its prediction, not with the previous consumption trends, which are different at the unusual benchmark points.

On the other hand, IF and LODA consider all features to detect abnormalities rather than conducting prediction. IF detects many points as abnormalities even more than benchmark. This may derive the fact that these points are different in certain feature from the usual trends, such as humidity, temperature, higher consumption in these time slots or even holiday flag.

Table 2.1: The Methods Accuracy in Features Selection Scenarios

Feature scenarios	IF	LODA	LPBSVR	LPBNN
Whole features	0.8132	0.5447	0.493	0.4751
L^t	0.79156	0.7313	0.9276	0.7288
$L^t + L^w$	0.7666	0.7313	0.3206	0.2273
$L^t + C$	0.7924	0.3467	0.9276	0.7662
$L^t + C + L^w$	0.8783	0.7976	0.43102	0.5204
$L^t + C + E$	0.9196	0.43427	0.8852	0.6874

Table 2.2: MSE of prediction based methods with respect to different feature combination

Feature scenarios	LPBSVR	LPBNN
Whole features	0.2741	0.3016
L^t	0.6516	0.6375
$L^t + L^w$	0.2723	0.3893
$L^t + C$	0.7501	0.6770
$L^t + C + L^w$	1.3376	0.3778
$L^t + C + E$	0.7839	0.6547

2.4.3 Feature Selection and Sensitivity Analysis

In this section, we examined the impact of different features on the performance of each method. The result of the simulation are summarized in Table 2.1. It is worth mentioning that all methods have been examined with different thresholds and the best MCC result is reported for each method. Also, the MSE of the prediction based methods is given in the Table 2.2.

Recall from Section 2.4.2 that, while the use of all features can improve prediction in prediction-based methods, it does not necessarily improve abnormality detection. This issue is better understood in Fig. 2.4, where features L^t, C and L^w are used. Despite having higher MSE, LPBSVR and LPBNN have lower MCC. In prediction-based methods, we need

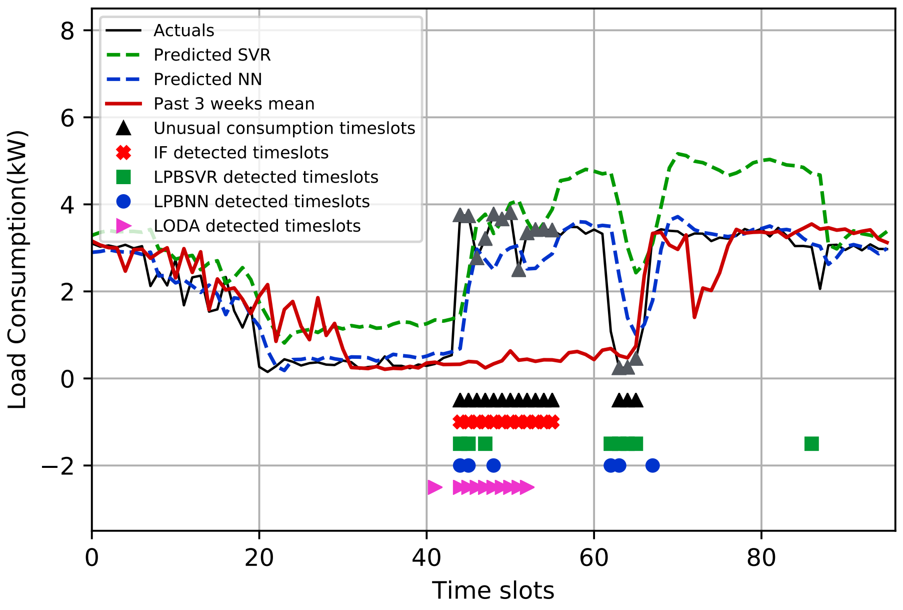


Figure 2.4: Comparison with benchmark based on $L^t + C + L^w$ features

a prediction (dashed green curve) close to the previous consumption trend (red curve) and not necessarily close to the real power consumption (blue curve). In fact, for prediction-based methods, those features that simulate the previous consumption trends serve better for abnormality detection. Fig. 2.5 shows the results for the case where only the L^t features are used. Here, prediction-based methods appropriately simulate the previous consumption trend (red curve) as its expected value (dashed green curve) for the actual load. Conversely, for more accurate predictions, the better descriptive and complementary features should be used. It is worth mentioning that, NN works better than SVR in most cases in predicting the electricity consumption. However, LPBSVR has better performance than LPBNN in detecting abnormality in most cases, except when L^t, L^w and C are all utilized. This is because SVR is trained based on the outliers, which helps in abnormality detection.

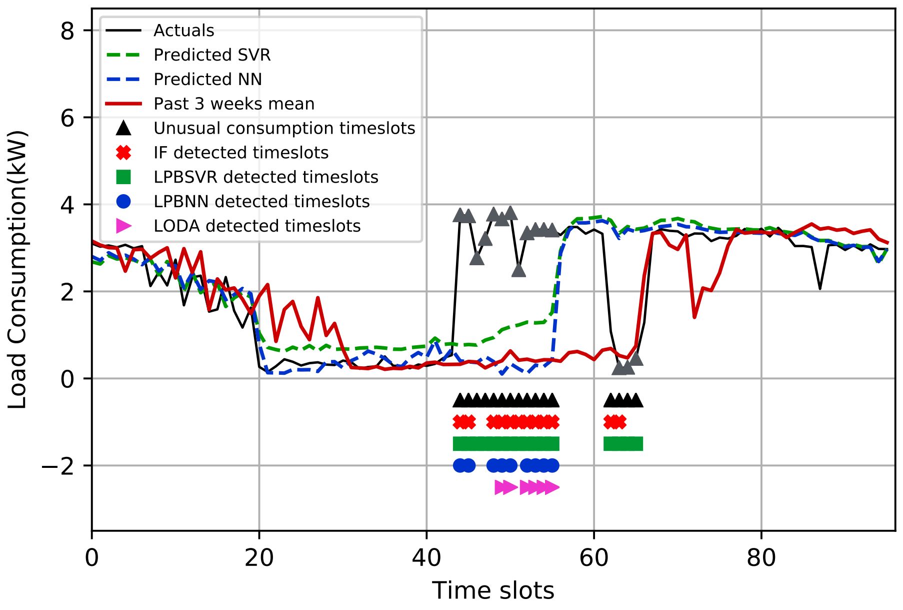


Figure 2.5: Comparison with benchmark based on L^t features

Another observation is that the performance of IF is *not* sensitive to L^w, i.e., window features. This is due to the cohesion of these features which are related and close to each other. As a result, IF cannot divide them suitably in the trained model (trained trees). In contrast, if we substitute the window features with the environmental features, the best performance for IF is happened. This shows that the tree nodes in IF are sensitive to the environmental features. Understanding the cause of unusual consumption can be derived by examining diverse features which is out of the scope of this study. Analyzing simulation results shown that prediction based method generally is more accurate for the abnormalities with very high or very low magnitude. Conversely, other two methods, specially IF, are more accurate for unusual trends which may does not have a high peak power consumption.

Since LODA depends on random projection, many repetition must be considered to test its performance. The result with L^t, C and E as features in different Houses and time slots shows that LODA tends to detect power consumption which are really close to zero. In other words, LODA is sensitive to the very low power consumption periods. However, it has an acceptable performance compared to other methods when L^t is used or it is accompanied with L^t features.

Chapter 3

Unsupervised Event Detection and Clustering for a Single Micro-PMU Time Series

3.1 Abstract

In this section we introduce the proposed methods which are developed by constructing *unsupervised deep learning* anomaly detection models; thus, providing event detection algorithms that require *no or minimal human knowledge*. First, we develop the core components of our approach based on a Generative Adversarial Network (GAN) model. 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 refer to this method as the *basic* method. It uses the same features that are often used

in the literature to detect events in micro-PMU data. Next, we propose a second method, which we refer to as the *enhanced* method, which is enforced with additional feature analysis. Both methods can detect *point signatures* on single features and also *group signatures* on multiple features. This capability can address the unbalanced nature of power distribution circuits. The proposed methods are evaluated using *real-world* micro-PMU data. We show that both methods highly outperform a state-of-the-art statistical method in terms of the event detection accuracy. The enhanced method also outperforms the basic method.

Moreover, we develop a complete interconnected event detection and event clustering method. The unsupervised event detection method constitute of as the same number of GAN models as the feature numbers, such that we can identify any event regarding to each individual feature. Then we 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 also 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.

It should be mentioned that the main elements of GAN base event detection model are introduced in the Basic model and for the other two event detection model we explain the new elements and essential differences.

3.2 Basic Event Detection Method

In its core, the proposed basic event detection method uses a GAN model which has two components, a *generator* and a *discriminator*. The generator is a deep neural network that tends to *produce* data samples that follow the distribution of the historical training data. The discriminator is a deep neural network that tends to *distinguish* between the data samples generated by the generator and the true historical data. By training the generator and the discriminator subsequently and iteratively, the GAN model can achieve an *equilibrium*, at which the discriminator can no longer distinguish between the distribution of the generated samples and the historical data.

3.2.1 Features

As in [8, 22], we use the following time-series as the features to train the GAN model in our basic method: 1) magnitude of voltage, i.e., V; 2) magnitude of current, i.e., I; 3) active power, i.e., P; and 4) reactive power, i.e., Q. All these features are defined separately for each three phases. Therefore, in total, the GAN model is trained with 12 time-series. Note that, while micro-PMUs measure V and I directly, P and Q are obtained rather indirectly by combining V and I with the measurements on voltage phase angle and current phase angle, which are both provided by micro-PMUs.

3.2.2 Generator

This element is common between all of the developed event detection models. It is a deep neural network that comprises Long Short-Term Memory (LSTM) modules [63] as well as dense layers. It takes a noise vector z from a distribution function $p_z(z)$, such as $z \sim \mathcal{N}(\mu_z, \sigma_z^2)$, and tries to produce samples similar to the ones from the true sample distribution. We seek to train a neural network $G(z, \theta_g)$ to generate samples which follow the distribution of the historical data. Here, θ_g denotes weights of the generator network. Mathematically, we seek to minimize the following objective function [64]:

$$\frac{1}{N}\sum_{i=1}^{N}\big[log(1 - D(G(z_i)))\big], \tag{3.1}$$

where N is the number of samples in each training batch, D is the discriminator function, G is the generator function, and z_i is the random vector for ith generated sample. In order to train the generator, after forward propagation, we need to update the generator parameters by calculating gradient and using a proper optimizer, such as Adam optimizer [65].

3.2.3 Discriminator

This element also is common between all of the developed event detection models. It is meant to distinguish between the fake data samples generated by the generator and the real measurements and it contains LSTM modules and dense layers. Our goal is to train a neural network $D(x, \theta_d)$, which creates a single scalar value as its output. Here, x is the vector of the actual measurement data and θ_d is the weights of the discriminator network. The primary objective of the discriminator is to maximize the probability of distinguishing between the true measurement data and the data generated by the generator. Therefore, we seek to minimize [64]:

$$\frac{1}{N}\sum_{i=1}^{N}\big[log(D(x_i)) + log(1 - D(G(z_i)))\big], \tag{3.2}$$

where x_i is the ith real sample and the second term is the same as the term in (3.1). Together, the generator and the discriminator play a *min-max* game with the following value function [64]:

$$\begin{aligned} V(G, D) = \mathrm{E}_x \sim p_{data}(x)[log(D(x))] + \\ \mathrm{E}_x \sim p_z(z)[log(1 - D(G(z)))]. \end{aligned} \tag{3.3}$$

3.2.4 Training

Both the generator and discriminator are formed with Long Short-Term Memory (LSTM) modules, which are connected back-to-back to capture the relationship between different features and their time dependencies. The micro-PMU data is normalized and segregated into sequences of training blocks.

The value of $V(G, D)$ can attain its global optimum by satisfying the following two conditions:

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

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}. \tag{3.4}$$

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

$$\min(\max_D(V(G, D))) \Longleftrightarrow p_g(x) = p_{data}(x). \tag{3.5}$$

If these conditions are not satisfied at the equilibrium, then the training is repeated with new random initial points.

3.2.5 Event Scoring

After training the basic model, the blocks of micro-PMU data stream are passed to the discriminator and the output is a scalar number which is defined as *score*. We pass the whole training set to the discriminator and calculate the scores. A normal probability distribution function (pdf) is fitted to the obtained scores, i.e., $scores \sim \mathcal{N}(\mu, \sigma^2)$, due to the fact that these scores must be very close to the global optimum, see (3.4) and (3.5). This is because of the infrequent nature of the events in power distribution systems.

3.2.6 Algorithm

The proposed basic event detection method is summarized in Algorithm 1. It works based on the fact that events in micro-PMU data are infrequent. In fact, our analysis of the real-world micro-PMU data shows that events occur at about 0.04% of the times. Thus, the default for the trained model must be the normal operation of the power distribution system. As a result, the discriminator is essentially trained to distinguish between the absence and the presence of the events, which is exactly what is needed in order to detect the events. 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 [66].

Algorithm 1 Event Detection - Basic Method

```
Input: Training data and test data: $V$, $I$, $P$ and $Q$.
Output: Event Detection Flag $F$.
// Learning Phase
Train the $GAN$ model.
Use the Discriminator as scoring function $D^*(\cdot)$.
Calculate the scores for the training data.
Fit a Normal PDF $\mathcal{N}(\mu, \sigma^2)$ to the obtained scores.
// Detection Phase
For each new micro-PMU test data Do
    Calculate the score $s$ using $D^*(\cdot)$.
    If $s \notin (\mu - z_p\delta, \mu + z_p\delta)$ Then
        $F = 1$ // Event
    Else
        $F = 0$ // No Event
    End
End
```

3.3 Enhanced Event Detection Method

The basic method in Section 3.2 requires training a *single* GAN model, where the features are V, I, P, and Q. However, given the characteristics of the micro-PMU data, in this section, we propose to develop and train *two* separate GAN models, one for the voltage measurements V, and another one for the rest of the measurements, i.e., I, P, and Q.

3.3.1 Feature Analysis

After applying the basic method to real-world micro-PMU data, we observed that Algorithm 1 sometimes fails to detect events that demonstrate signatures only in voltage magnitude. Such event cannot trigger the score to exceed the threshold. Further investigation revealed that this is because, in power distribution systems, voltage measurements are much less volatile than current measurements. Therefore, the GAN model sometimes cannot properly extract the characteristics of the voltage measurements. The correlation between four elements of micro-PMU data in 3 phase illustrate low dependency of P and Q to V compare to I. Since, with a small variation in current, P and Q totally follow the trend, however, in many events which related mostly to V the other variables do not show any significant change.

3.3.2 Training Multiple GAN Models

To remedy the above issue, we propose to construct two separate GAN models that are trained in parallel. One GAN model, denoted by GAN_V, has 3 features as its input, which are the voltage magnitude measurements across the three phases. The other GAN model, denoted by $GAN_{I,P,Q}$, has 9 features as its input, which are current magnitude, active power, and reactive power measurements across the three phases. Importantly, it is observed that I has high correlations with P and even Q, which makes it desirable to combine I, P, and Q into one GAN model; as opposed to having four GAN models for V, I, P, and Q.

3.3.3 Event Scoring

Once each of the two GAN models is trained, the resulting Discriminator function is used to generate its own scores. An example for the scores that are generated by the two GAN models are shown in Fig. 3.1. The *blue dots* represent *normal data*. The red

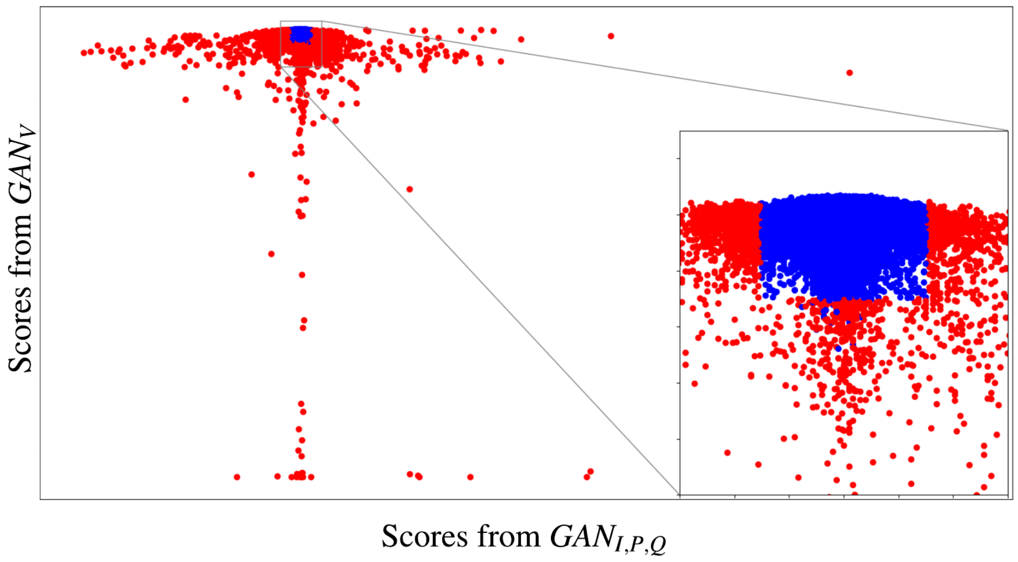


Figure 3.1: The importance of using two GAN models in the enhanced method: while the scores from the $GAN_{I,P,Q}$ model can detect most events; there are events that are detected only if the scores from the GAN_V model are also considered. Blue dots denote normal data while red dots denote events.

dots represent *events*. We can see that each of the two GAN models detects only a sub-set of events. The events that are scattered across x-axis are the ones that are detected by $GAN_{I,P,Q}$. They include the majority of the events. The events that are scattered across y-axis are the ones that are detected by GAN_V. Thus, both GAN models are both needed to enhance accuracy of event detection.

3.3.4 Algorithm

The proposed enhanced event detection method is summarized in Algorithm 2. It works by examining the scores of the two separate GANs; thus having a dedicated deep learning architecture to detect the events in voltage magnitude and another deep learning architecture to detect the events that involve the current, active power, and reactive power. The rest of the algorithm is similar to Algorithm 1.

Algorithm 2 Event Detection - Enhanced Method

```
Input: Training data and test data: $V$, $I$, $P$ and $Q$.
Output: Event Detection Flag $F$.
// Learning Phase
Train the $GAN_{I,P,Q}$ model.
Use the Discriminator as scoring function $D^*_{I,P,Q}(\cdot)$.
Calculate the scores for the training data.
Fit a Normal PDF $\mathcal{N}(\mu, \sigma^2)$ to the obtained scores.
Train the $GAN_V$ model.
Use the Discriminator as scoring function $D^*_V(\cdot)$.
Calculate the scores for the training data.
Fit a Normal PDF $\mathcal{N}(\phi, \varphi^2)$ to the obtained scores.
// Detection Phase
For each new micro-PMU test data Do
    Calculate the score $s_1$ using $D^*_{I,P,Q}(\cdot)$.
    Calculate the score $s_2$ using $D^*_V(\cdot)$.
    If $s_1 \notin (\mu - z_p\delta, \mu + z_p\delta)$ or
      $s_2 \notin (\phi - z_p\varphi, \phi + z_p\varphi)$ Then
      $F = 1$ // Event
    Else
      $F = 0$ // No Event
    End
End
```

3.4 Proposed Event Detection Method

Same as the other two developed event detection model the proposed model in this section is based on GAN, however, it is slightly different in terms of features and process. The architecture and algorithm of the GAN models is as follow:

3.4.1 Features

Checking the *magnitude* of voltage and current in micro-PMU measurements is a common option to detect and identify events, e.g., see [8, 22]. 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 section, 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 section are

$$|V_\phi|, |I_\phi|, \cos(\theta_\phi), \quad \phi = A, B, C. \tag{3.6}$$

which denote the voltage magnitude, current magnitude, and power factor in each phase ϕ, respectively. For notation simplicity, in the rest of section, we refer to the features in (3.6) for *all the three phases*, without specifying subscript ϕ. Also, batch normalization have been implemented in order to prevent internal covariate shift.

3.4.2 Training and Convergence

In this model we train all nine separate GAN models, one model for each feature, so as to *learn* the characteristics of the *normal* trends in micro-PMU measurements for each feature and each phase. 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 (3.3) must satisfy the conditions 3.4 and 3.5.

The training of the GAN model and proofs are explained in details in [64]. 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 [67]. 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 [67].

The choice of the hyper-parameters of the GAN model is critical in achieving an equilibrium. In particular, based on the two criteria in (3.4) and (3.5) and *convergence constant* $\epsilon > 0$ the equilibrium should satisfy the following conditions [64]:

$$\begin{aligned} &|\max_{D}(V(G,D)) - (-log4)| < \epsilon, \\ &|D_g(x) - \frac{1}{2}| < \epsilon. \end{aligned} \tag{3.7}$$

3.4.3 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 (3.4) and (3.5). 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 [68], 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.

3.4.4 Algorithm

The proposed event detection method is summarized in Algorithm 3. 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:

$$\boldsymbol{E}^{w}_{9\times 1} = [e^{w}_{1}, \cdots, e^{w}_{9}] \tag{3.8}$$

The detection vector is a 9×1 *binary* vector, where 9 is the number of features as in (3.6). 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 [66].

The detection vectors show us the existence of event as well as providing us with the inputs that we need for our clustering algorithm; which we will explain in Section 3.5.

Algorithm 3 Event Detection - Proposed Method

```
Input: Training and test data based on the features in (3.6).
Output: Event detection vector $E^w_{9\times1}$ for the $w^{th}$ data.
// Learning Phase
Foreach feature $f$ in (3.6):
    Train the $GAN_f$ model
    Use discriminator as scoring function $D^*_f(\cdot)$.
    Calculate the scores for the training data.
    Fit a Normal PDF $\mathcal{N}(\mu_f, \sigma_f^2)$ to the obtained scores.
End
// Detection Phase
Foreach new micro-PMU test data ($w$):
    Foreach feature $f$ in (3.6):
        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
```

3.4.5 Evaluation metric

We use the Matthews correlation coefficient (MCC) [62] as the metric to assess accuracy; for both detection and clustering. As explained in [69], 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:

$$MCC = \frac{N_s Tr(\psi) - \sum_{k=1}^{K}\sum_{l=1}^{K}\psi_k\psi_l}{\sqrt{N_s^2 - \sum_{k=1}^{K}\sum_{l=1}^{K}\psi_k\psi_l^T}\sqrt{N_s^2 - \sum_{k=1}^{K}\sum_{l=1}^{K}\psi_k^T\psi_l}}, \tag{3.9}$$

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.

3.5 Unsupervised Clustering Method

Given the detection vectors in Section 3.4, in this section, we develop a two-step event clustering method so that we can later study different types of events in details.

3.5.1 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 (3.6). 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. 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 3.5.2.

3.5.2 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.

A. 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:

$$\boldsymbol{P}_i = \begin{bmatrix} \alpha_i^{1,1} & \cdots & \alpha_i^{1,\tau} \\ \vdots & \ddots & \vdots \\ \alpha_i^{9,1} & \cdots & \alpha_i^{9,\tau} \end{bmatrix}. \tag{3.10}$$

There are nine rows in $\boldsymbol{P}_i$ corresponding to the nine features in (3.6). 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 $c_{i,j}^k$ 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:

$$MaxCorr_{i,j} = \underset{k=1,\ldots,\tau}{\text{maximum}}\ c_{i,j}^k; \tag{3.11}$$

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

B. Optimization Problem Formulation

Consider a given category of events based on the pre-processing step in Section 3.5.1. 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:

$$\underset{u}{\text{minimize}} \quad \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{c=1}^{C} u_{i,c}u_{j,c}(1 - MaxCorr_{i,j}) \tag{3.12a}$$

$$\text{subject to} \quad u_{i,c} \in \{0,1\}, \tag{3.12b}$$

$$\sum_{c=1}^{C} u_{i,c} = 1 \quad \forall i. \tag{3.12c}$$

where $u_{i,c}$ is a binary variable. It is one if event i is in cluster c; otherwise it is zero. Problem (3.12) minimizes the sum of the distances between the events, measured as 1-$MaxCorr_{i,j}$, across different clusters. The constraint in (3.12c) assures that each event is assigned to only one cluster. Problem (3.12) is a MINLP.

C. Exact Linearization

To enhance computational performance, the MINLP in (3.12) is replaced with an *exact equivalent* Mixed Integer Linear Programming (MILP), as follows:

$$\underset{u,\, t}{\text{minimize}} \quad \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{c=1}^{C} t_{i,j,c}(1 - MaxCorr_{i,j}) \tag{3.13a}$$

$$\text{subject to} \quad u_{i,c}, t_{i,j,c} \in \{0,1\}, \tag{3.13b}$$

$$\sum_{c=1}^{C} u_{i,c} = 1 \quad \forall i, \tag{3.13c}$$

$$u_{i,c} + u_{j,c} - t_{i,j,c} \leq 1 \quad \forall i,j,c, \tag{3.13d}$$

$$-u_{i,c} - u_{j,c} + 2t_{i,j,c} \leq 0 \quad \forall i,j,c. \tag{3.13e}$$

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 (3.13d) and (3.13e) are used to make sure that $t_{i,j,c}$ is indeed equal to such product in order to assure an *exact* linearization. Problem (3.13) can be solved using any MILP solver for a set of detected events in a given time period as training set.

D. Cluster Representatives

Once the clusters are obtained by using the training data and solving the MILP problem in (3.13), 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 (3.13). To determine the optimum representative for each cluster, we solve the following optimization problem:

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{c=1}^{C} u_{i,c}v_{j,c}(1 - MaxCorr_{i,j}) && (3.14a) \\ \text{subject to} \quad & v_{i,c} \in \{0,1\}, && (3.14b) \\ & \sum_{i=1}^{I} v_{i,c} = 1 \quad \forall c && (3.14c) \end{aligned}$$

Variable $v_{j,c}$ is binary. It is one, if event j is the representative event for cluster c, and zero otherwise. Constraint (3.14c) 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 (3.14) is an MILP by construction.

E. 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 (3.13) 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 (3.14).

3.5.3 Active Clustering

Given the large number of events that are detected in micro-PMU measurements, 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 for each cluster (Algorithm 4).

Algorithm 4 Unsupervised Event Clustering

```
Input: Event detection vectors $\boldsymbol{E_T}$ from (3.8) ; Event time-series data, Number of clusters ($N_c$), Similarity threshold ($\varphi$).
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 (3.13).
    Determine the cluster representative based on (3.14).
  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 $\varphi$ 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
```

3.6 Result Comparing of Basic and Enhanced Methods

In this section we compare the event detection results for basic and enhanced methods. The proposed event detection methods are applied to the *real-world* data from a distribution feeder in Riverside, CA [8]. The resolution of the data is 120 readings per second. In total, 1.8 billion measurement points are analyzed. In particular, two weeks of data are used to train the GAN models. One day of data is used to test the event detection methods. Event detection is applied on windows of size 40 data points. Each window has an overlap of size 20 data points with the next window in order to assure not missing any event.

3.6.1 Performance Comparison

The effectiveness of the event detection methods is investigated over 1000 reference events in micro-PMU data, that are visually extracted within a specific period of time. The summary of the results are shown in Table 3.1. We can see that the basic method significantly outperforms the benchmark statistical event detection method in [8]. Furthermore, the enhanced method considerably outperforms the basic method. Next, we explain the underlying causes for these differences by going through several examples of the events that are detected.

Table 3.1: Event Detection Accuracy

	Benchmark [8]	Basic Method	Enhanced Method
Accuracy	0.3640	0.6943	0.8805
F1-score	0.3614	0.7676	0.9023

3.6.2 Assessment of the Basic Method

Figs. 3.2 to 3.6 show five examples of the events that are detected by the basic method. Importantly, the prevalent statistical method in [8] detected only the first two of such events. Regarding the events in Figs. 3.4 and 3.5, they are not detected by the method in [8] because the changes in the magnitudes are relatively small and do not significantly affect the statistical measures, such as the absolute deviation around median. As for the event in Fig. 3.6, all the pieces of this long event are detected by the basic method at several subsequent windows of the data. However, the statistical method in [8] only captures the step change the beginning of this event; because the statistical characteristics remain the same afterwards.

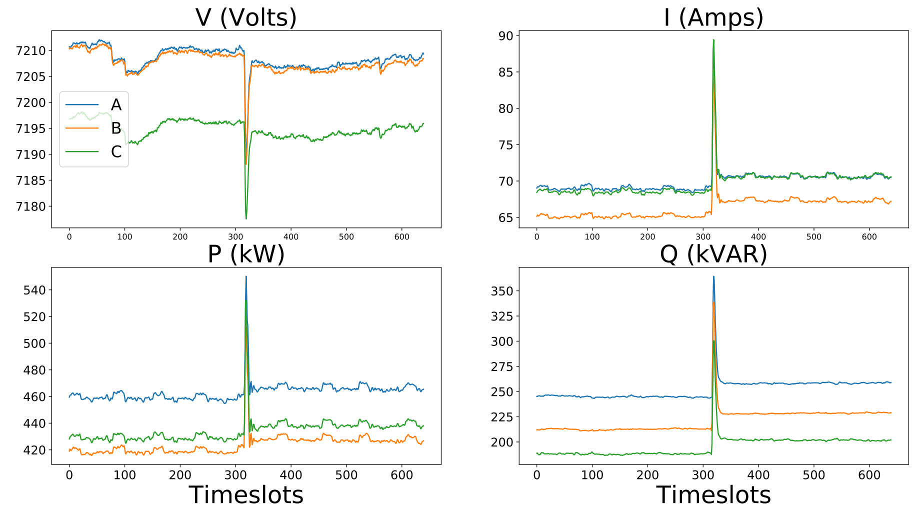


Figure 3.2: Inrush current with impact on all features. This event is detected by all the three methods: statistical, basic, and enhanced.

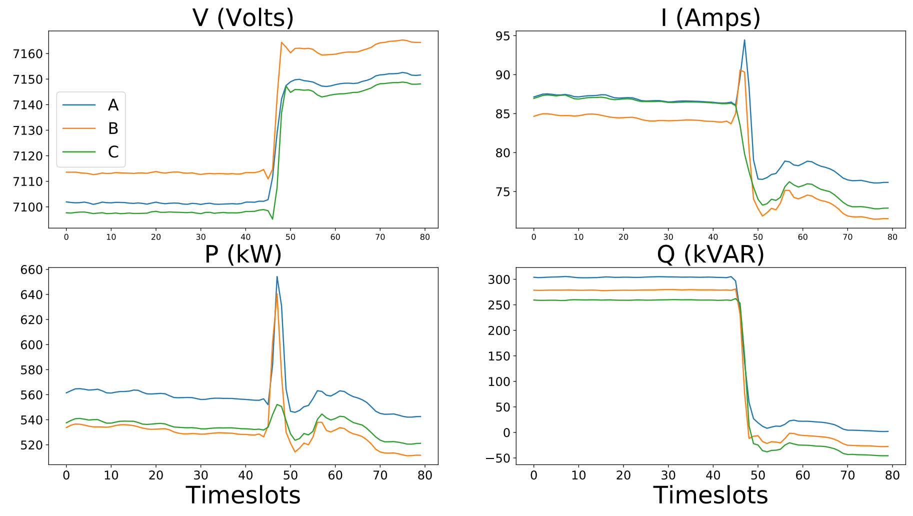


Figure 3.3: Capacitor bank switching with impact on all features. This event is detected by all the three methods: statistical, basic, and enhanced.

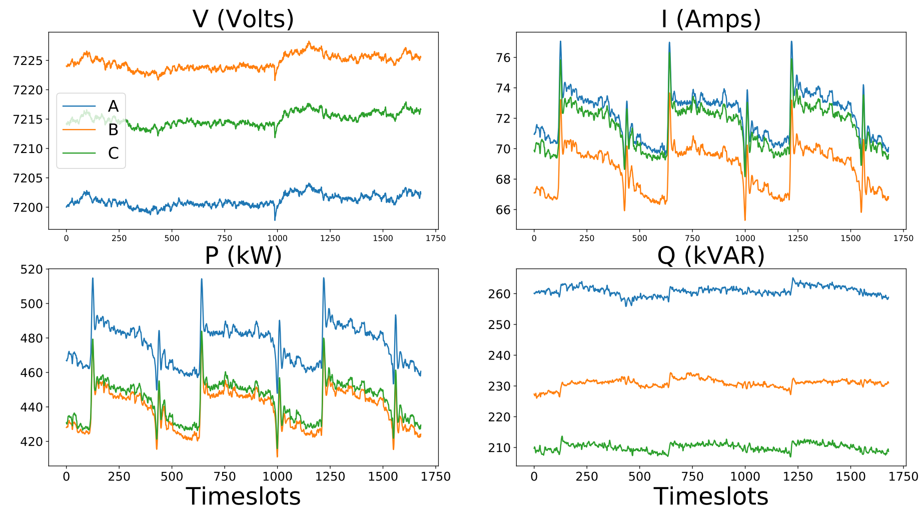


Figure 3.4: An event with major impact only on current and active power. This event is detected by the basic method, but not by the statistical method.

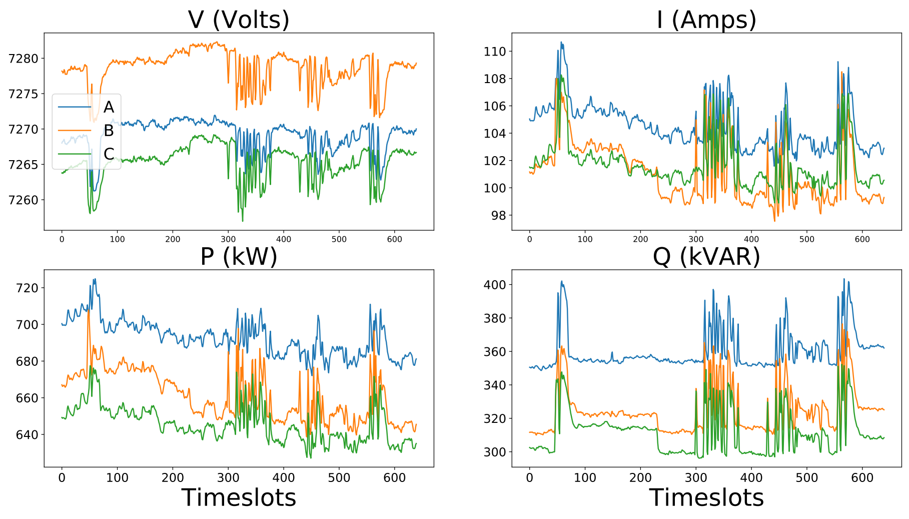


Figure 3.5: An event involving oscillations. This event is detected by the basic method, but it is not detected by the statistical method.

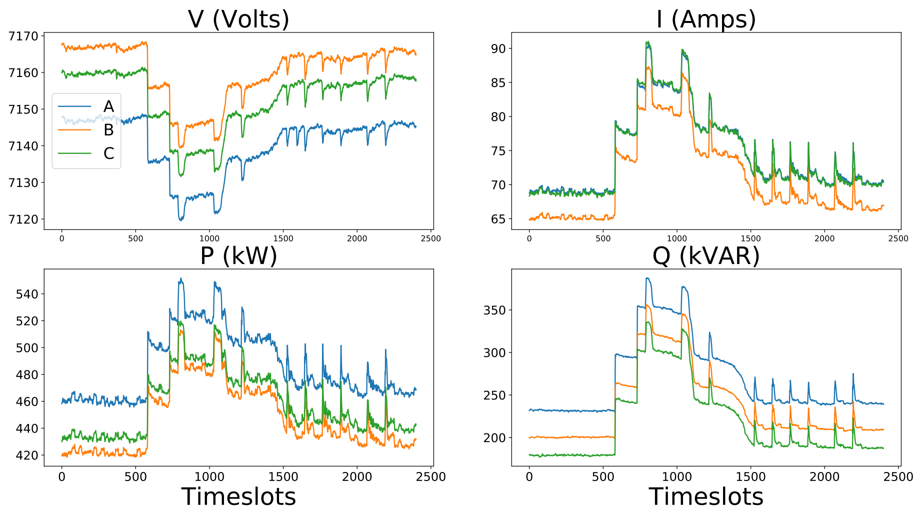


Figure 3.6: A rare and long event with 20 seconds of transient signature. All pieces of this long event are detected and captured by the basic method. The statistical method only detects a step change at the beginning of this event.

3.6.3 Assessment of the Enhanced Method

Figures 3.7 and 3.8 show two events that are detected by the enhanced method. But they are not detected by either the prevalent statistical method in [8] or even the basic method. The basic method fails to detect these two events because the main signatures are in voltage and they are relatively small in magnitude. Therefore, only the additional GAN model in the enhanced method can capture these events. This demonstrates the importance of the change in the model that was proposed in the enhanced method. Regarding the event in Fig. 3.8, it demonstrates momentary oscillations that started only after some sort of actions, possibly a tap changing event, where the oscillations damped after a short period of time. Events like this are important, for example, for asset monitoring. However, only the enhanced method was able to detect such event.

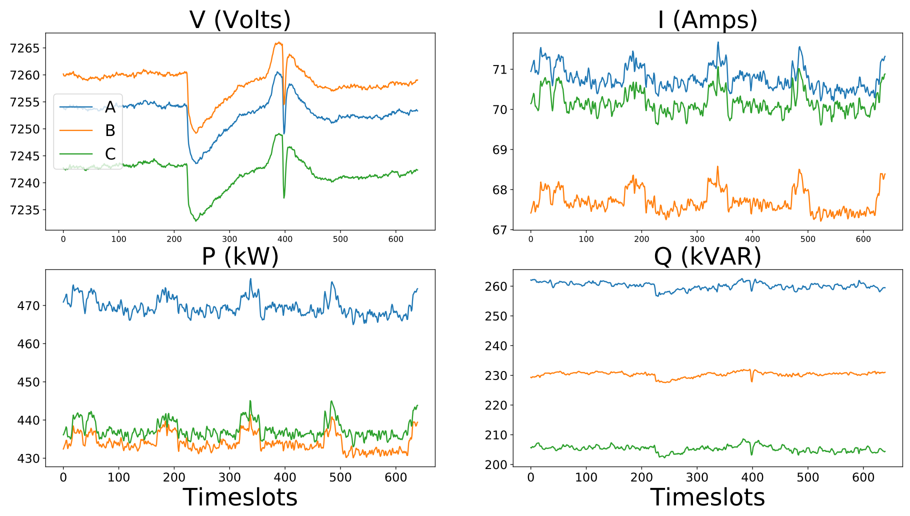


Figure 3.7: An event with impact mainly on voltage. It is detected by the enhanced method. But it is not detected by the basic method or the statistical method.

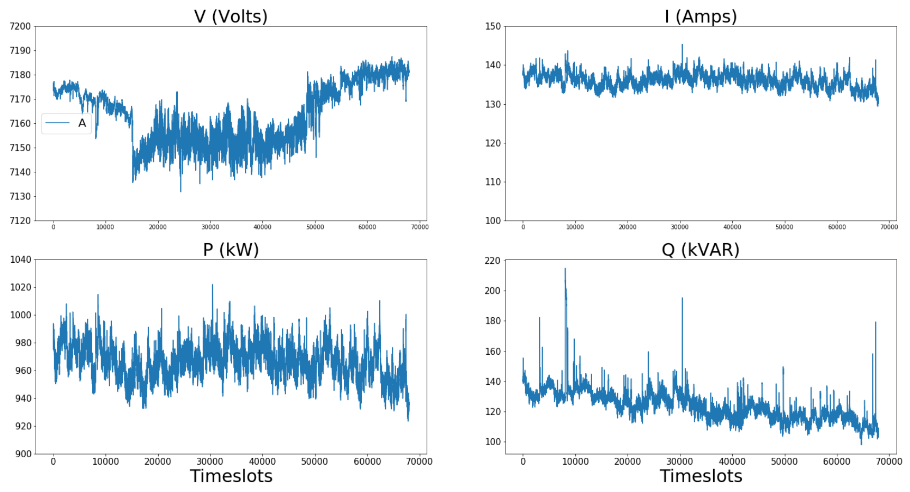


Figure 3.8: An event with momentary and damping oscillations in voltage, shown on one phase only. This event is detected by the enhanced method. But it is not detected by the basic method or the statistical method.

3.7 Event Detection 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.

3.7.1 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.

3.7.2 Accuracy Analysis

Table 3.2 shows the MCC for the proposed event detection method, in comparison with the benchmark methods in [8, 35] and enhanced method. A total of 1200 reference events are visually extracted by expert knowledge within 6 hours (64800 window samples

Table 3.2: Event Detection Information for Confusion Matrix, Precision, Recall and MCC

Metric	Statistical [8]	GGL [35]	Enhanced Method	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

of 40 time-slots) to evaluate the performance of event detection. The proposed method outperforms the methods in [8], [35] and enhanced method.

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 [8] and enhanced method 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 [35], 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 [35]. Another point about the [35] is that, due to the use of similarity graph, the method in [35] 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 enhanced method is that, the choice of the independent features in (3.6), 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 [35] 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 enhanced method 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. 3.20, 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 [8] has reasonable accuracy in detecting most of the three phase events that are *balanced*; due to the fact that the method in [8] was not designed to particularly detect unbalanced events. The method in [8] 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.

3.8 Event Clustering Results

The proposed event clustering method is applied to the captured events in Section 3.7.2; and its performance is compared with the following prevalent clustering methods in the literature: kNN [70], k-Medoids [71], and fuzzy-k-Medoids [72]. Different similarity measures are also considered: euclidean, DTW [73], soft-DTW [74], 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 [75]. 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 4.2 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, 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,

Table 3.3: 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

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.

3.8.1 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 3.4. As part of the pre-processing step in Section 3.5.1, these detection vectors result in five categories, denoted by Category I to Category V, as shown on the last column in Table 3.4. Categories I, II, and III include *balanced* events; while Categories IV and V include *unbalanced* events.

The optimization-based clustering in Section 3.5.2 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

Table 3.4: 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]		
$\boldsymbol{E}_5$	[000	110	110]	13956	IV
$\boldsymbol{E}_6$	[000	011	011]		
$\boldsymbol{E}_7$	[000	000	111]		
$\boldsymbol{E}_8$	[000	000	110]	52	V
$\boldsymbol{E}_9$	[000	000	011]		

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.

3.8.2 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.

A. Identifying Different Load Types

Fig. 3.9(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. 3.10 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. 3.9(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. 3.11 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.

B. Capacitor Bank State of Health Monitoring

Figs. 3.12(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 health. For example, consider the capacitor bank switching off event in Fig. 3.12(b). We can see that there is a relatively long *overshoot* on Phase A current and a

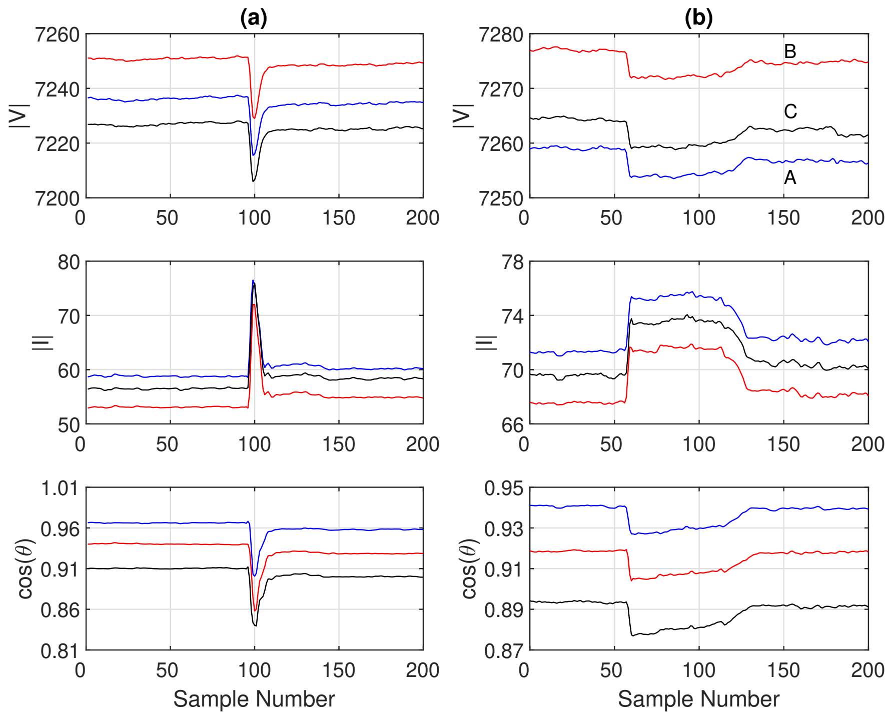


Figure 3.9: Examples of load switching events: (a) inrush current in Cluster #1; (b) long transient with a plateau in Cluster #2.

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. [10]. 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.

C. Two-step Events

Fig. 3.13 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

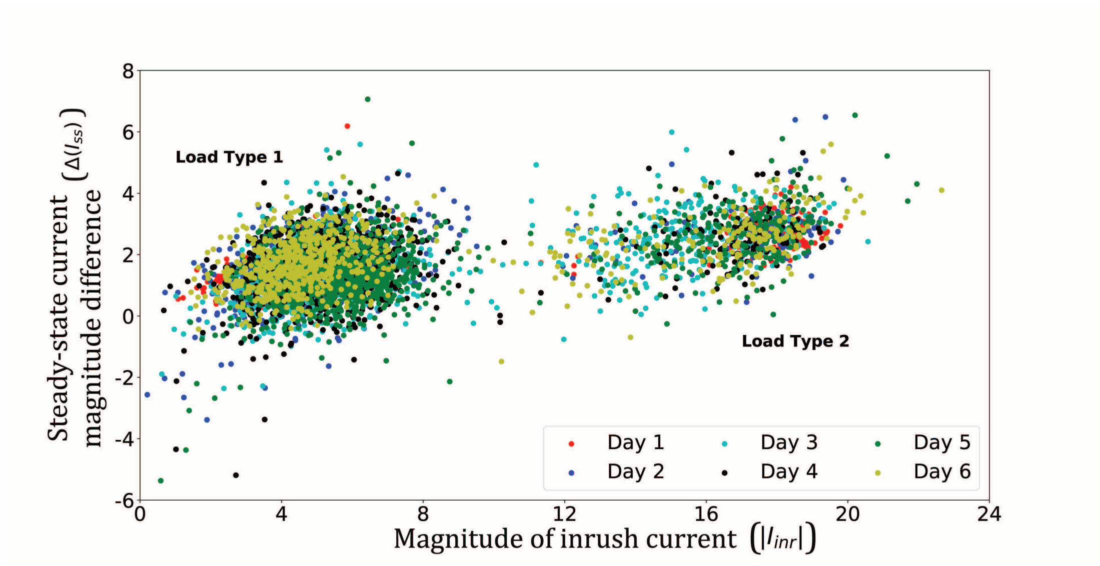


Figure 3.10: Identifying two major load types based on Cluster #1.

its unique switching pattern that is repeated every time this event occurs.

Oscillations in Current Induced by Step Changes

Fig. 3.14 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. 3.14(b)) that also is followed by an oscillation event which is magnified in Fig. 3.14(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 best remedial action; as deemed necessary

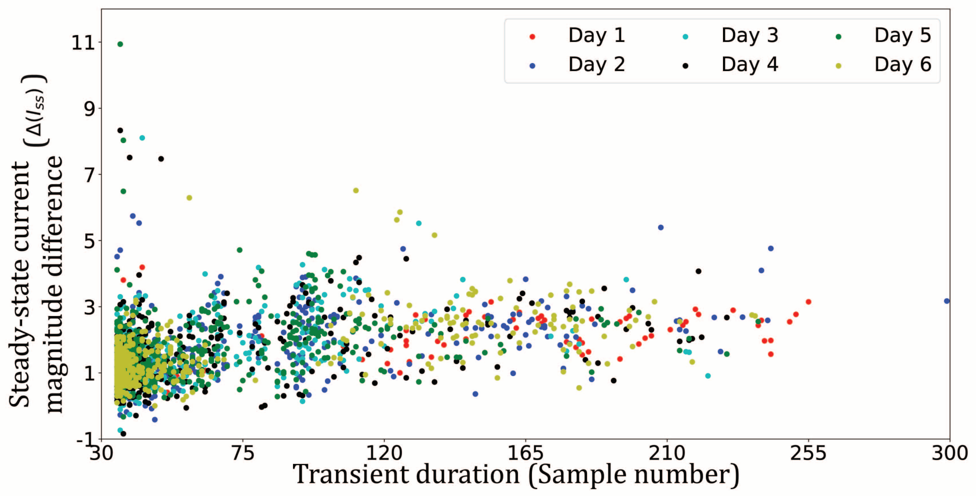


Figure 3.11: Scatter plot for the events in Cluster #2 over 6 days.

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.

3.8.3 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.

A. Voltage Events

Fig. 3.15(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. 3.15(b) shows an example event in Cluster #8, which is a

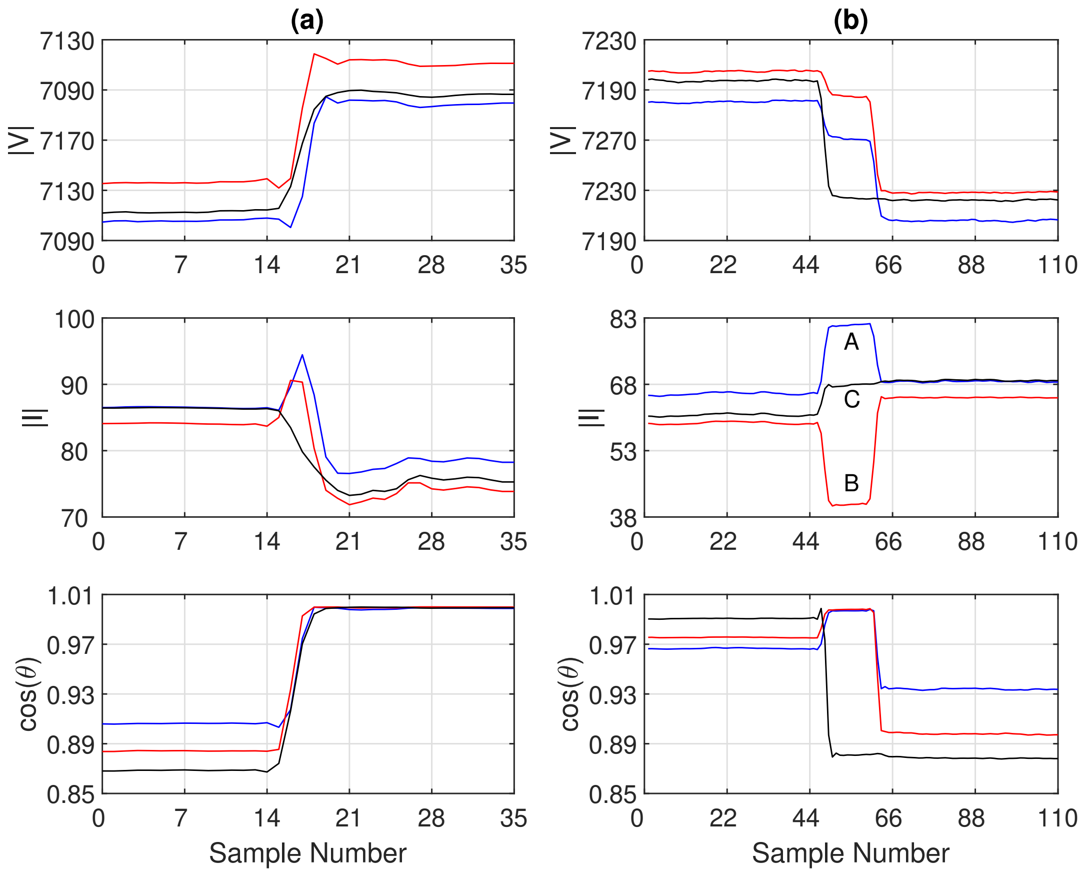


Figure 3.12: Monitoring the operation and health of a capacitor bank based on Clusters #3 and #4: (a) switch on; (b) switch off.

voltage event with a *plateau*. The transient shape of the voltage in Cluster #8 is similar to voltage changes in Cluster #2, see Fig. 3.9(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.

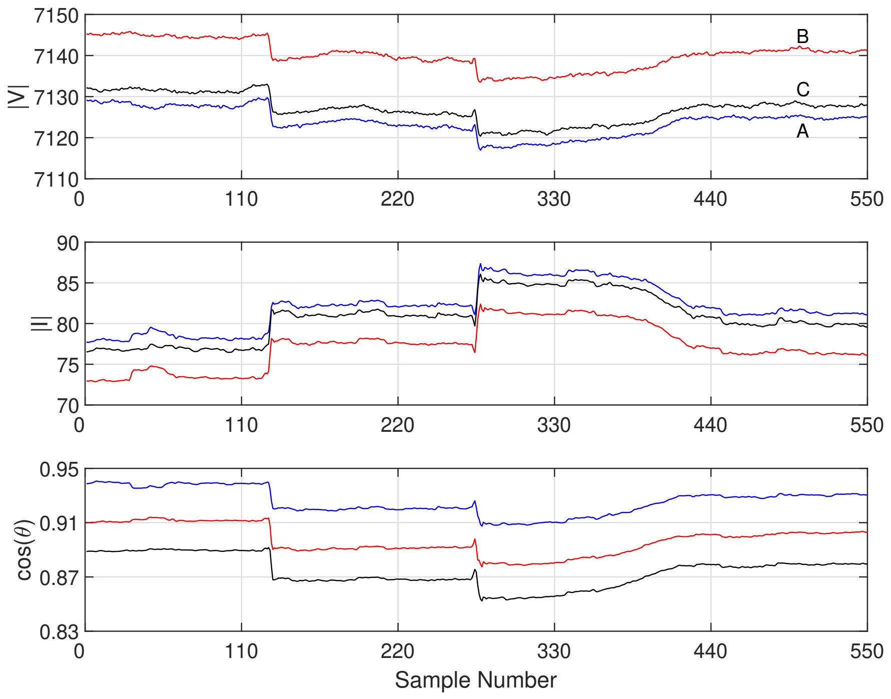


Figure 3.13: An example for the two-step event in Cluster #5.

B. Voltage Oscillation Events

Fig. 3.16 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 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.

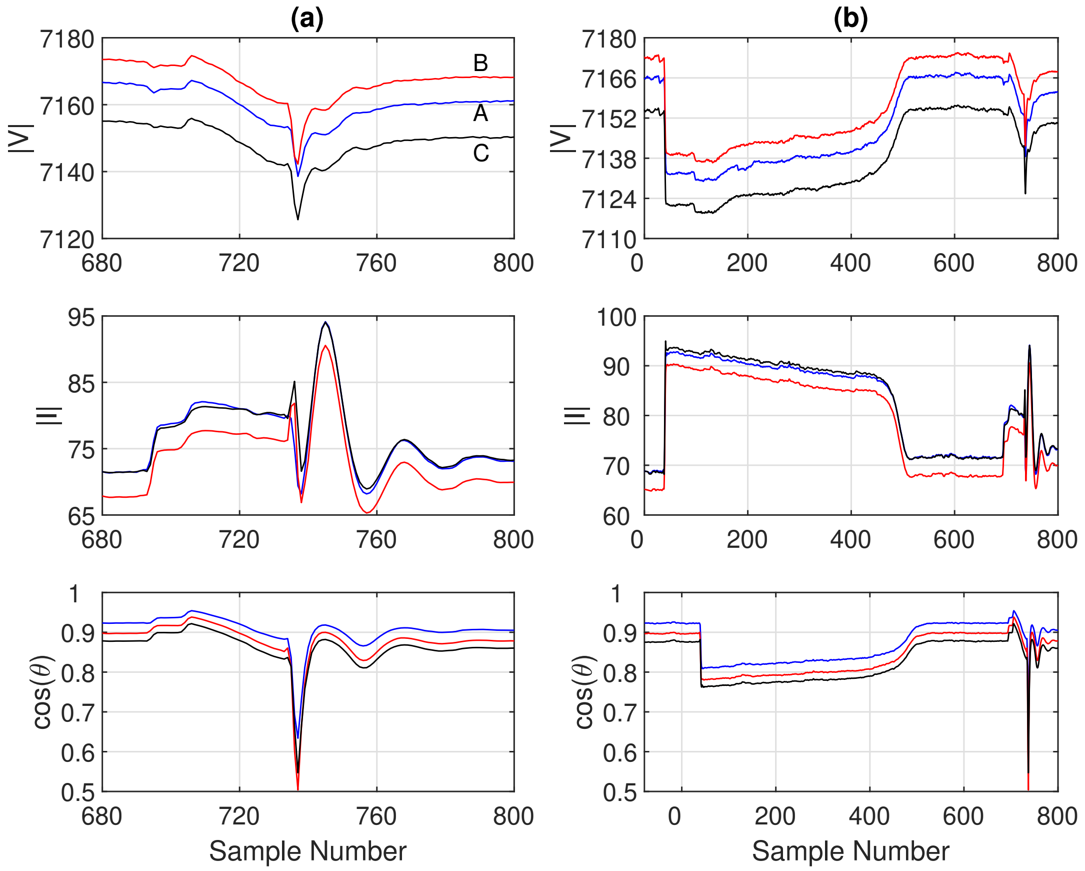


Figure 3.14: An example for the oscillation event in Cluster #6: (a) oscillation in current; (b) the step change prior the oscillations.

3.8.4 Use Case Exposition: One Cluster in Category III

One cluster is identified in Category III; denoted by Cluster #10. Fig. 3.17 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.

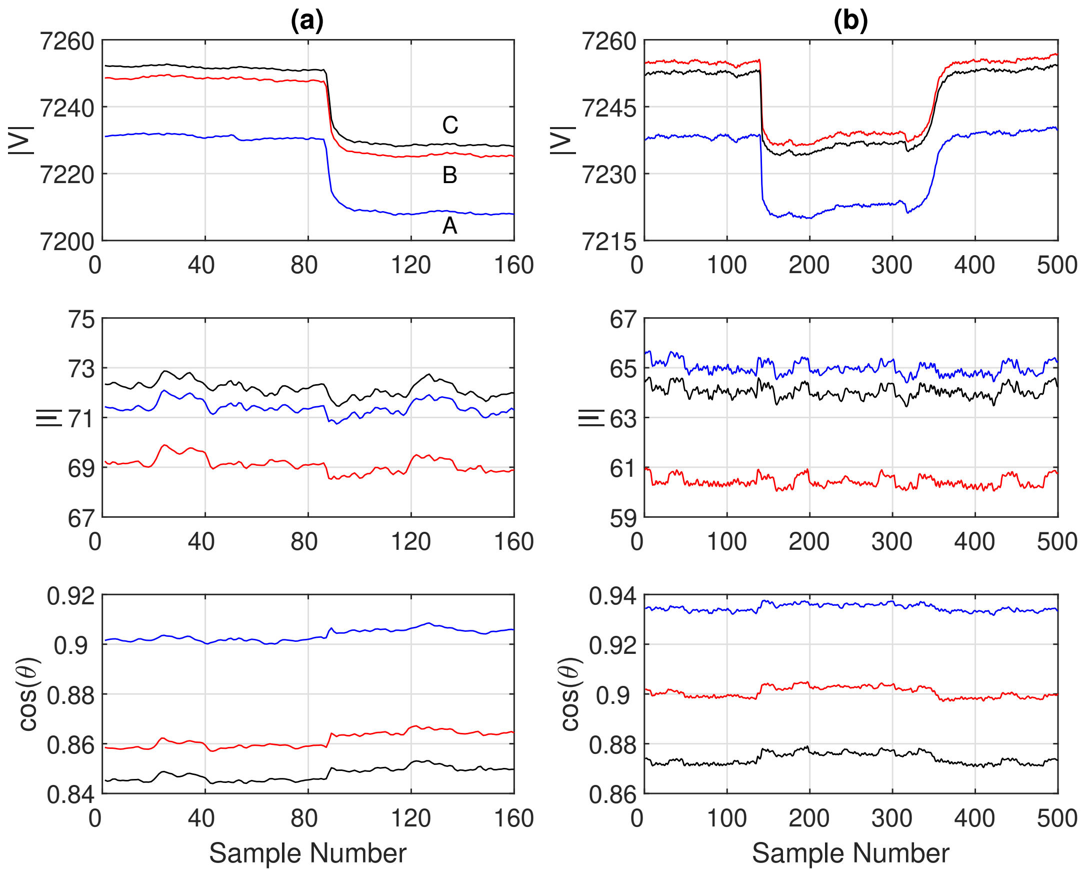


Figure 3.15: Examples of voltage events: (a) transformer tap-changer in Cluster #7; (b) voltage plateau in Cluster #8.

3.8.5 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. 3.18 shows an example of the event in Cluster #11. This event is *not* detected by the enhanced method; 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.

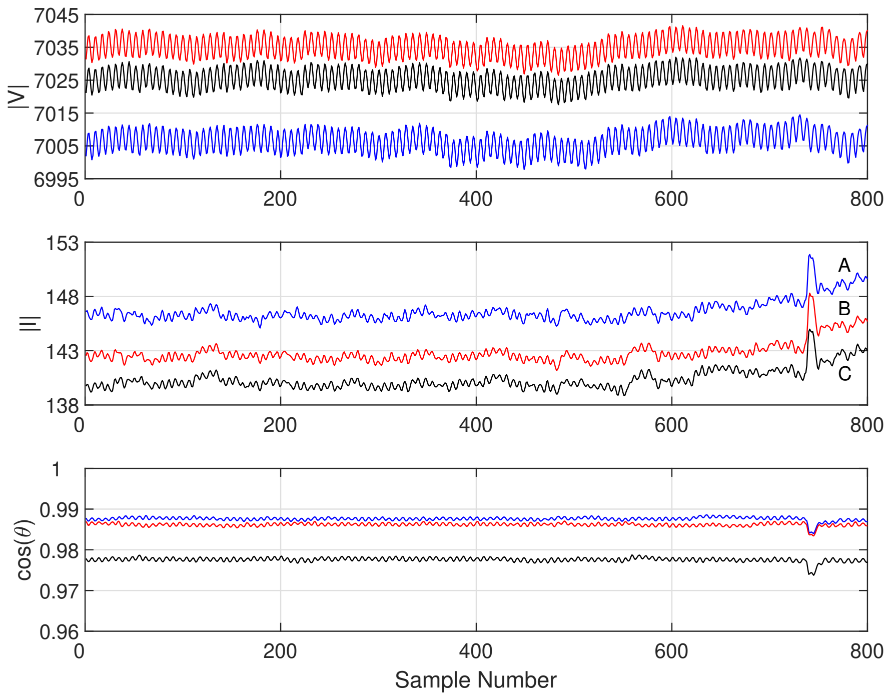


Figure 3.16: An example for voltage oscillation event in Cluster #9.

3.8.6 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. 3.19. 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.

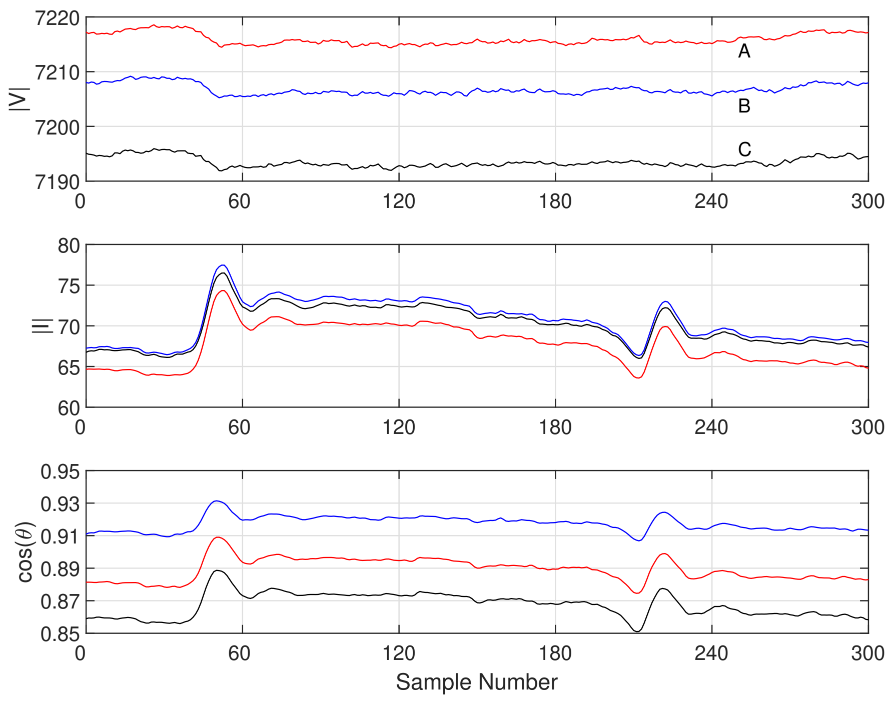


Figure 3.17: An example for current oscillation event in Cluster #10.

3.8.7 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. 3.20. 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. 3.14. Then, after about 60 seconds, a series of over 100 events occur that all belong to Cluster #10.The exact same sequence of events occurred on the same day and around the same time each week.

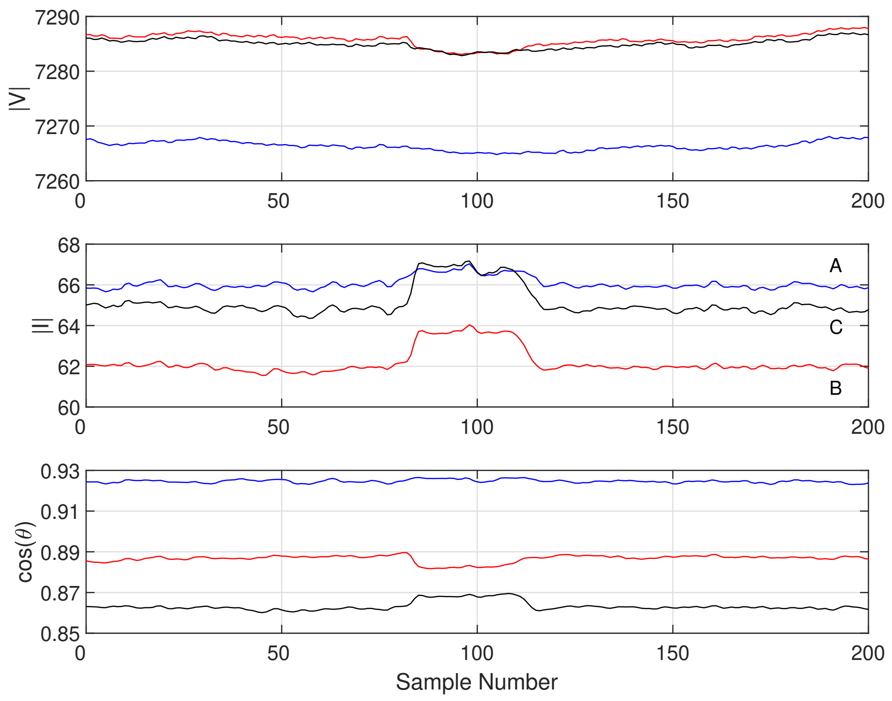


Figure 3.18: An example for the unbalanced events in Cluster #11. The event affects the current magnitude of phases B and C.

3.8.8 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.

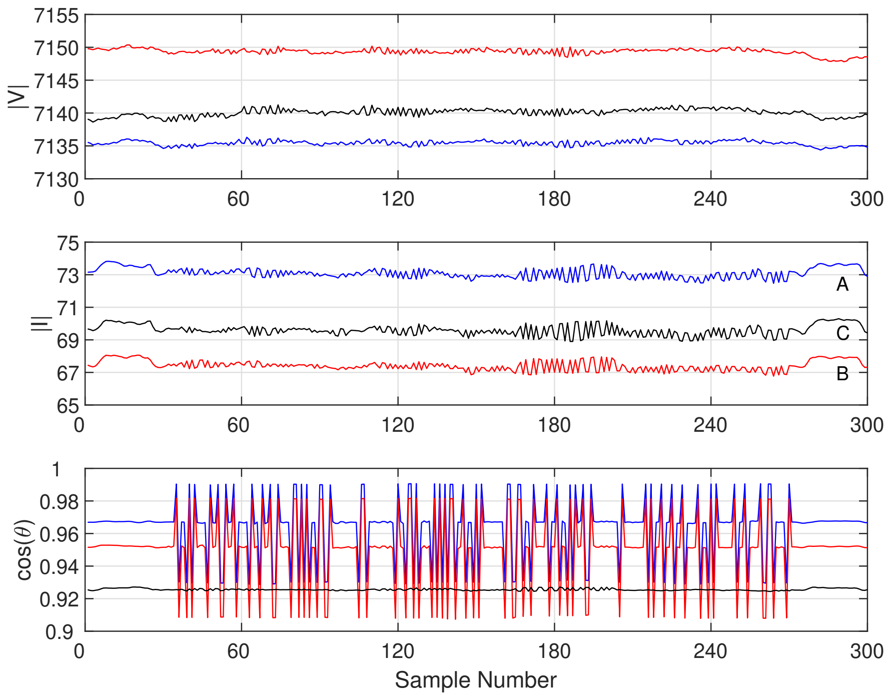


Figure 3.19: An example for power factor event in Cluster #14.

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 [76]. 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 [76].

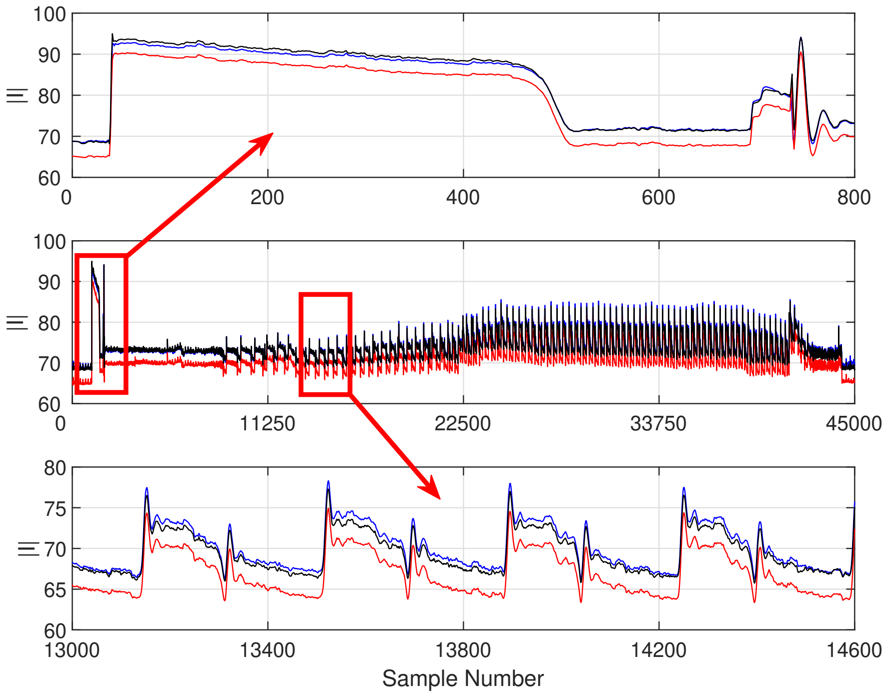


Figure 3.20: An example for the special sequence of the events in the current magnitude that are repeated occasionally. It was captured based on the collaboration of Clusters #6 and #10.

Chapter 4

GraphPMU: Event Clustering via Graph Representation Learning Using Locationally-Scarce Distribution-Level Fundamental and Harmonic PMU Measurements

4.1 Abstract

This section is concerned with the complex task of identifying the *type* and *cause* of the events that are captured by distribution-level phasor measurement units (D-PMUs) in order to enhance situational awareness in power distribution systems. Our goal is to

address two fundamental challenges in this field: a) *scarcity in measurement locations* due to the high cost of purchasing, installing, and streaming data from D-PMUs; b) *limited prior knowledge about the event signatures* due to the fact that the events are diverse, infrequent, and inherently unscheduled. To tackle these challenges, we propose an *unsupervised graph-representation learning* method, called GraphPMU, to significantly improve the performance in event clustering under *locationally-scarce data availability* by proposing the following two new directions: 1) using the *topological information* about the *relative location* of the few available phasor measurement units on the graph of the power distribution network; 2) utilizing not only the commonly used *fundamental* phasor measurements, bus also the less explored *harmonic* phasor measurements in the process of analyzing the signatures of various events. Through a detailed analysis of several case studies, we show that GraphPMU can highly outperform the prevalent methods in the literature.

4.2 Topology-Based Representation Learning

Consider a power distribution system, such as the one in Fig. 4.1. Let $\mathcal{B}$ denote the set of all buses, such as $\{B_1, \ldots, B_7\}$ in Fig. 4.1 and $N = \mathcal{B}$ denotes the number of buses. Also let $\mathcal{M}$ denote the set of those buses that are equipped with D-PMUs, such as $\{B_1, B_7\}$ in Fig. 4.1. For now, suppose the D-PMUs only provide the measurements for the fundamental phasors. The case where D-PMUs also act as H-PMUs to measure harmonic phasors will be discussed later in Section 4.5.

When an event occurs, its impact is *simultanously* captured by *all* the D-PMUs at the buses in set $\mathcal{M}$. Let X_i^j denote the *time series* of the phasor measurements that

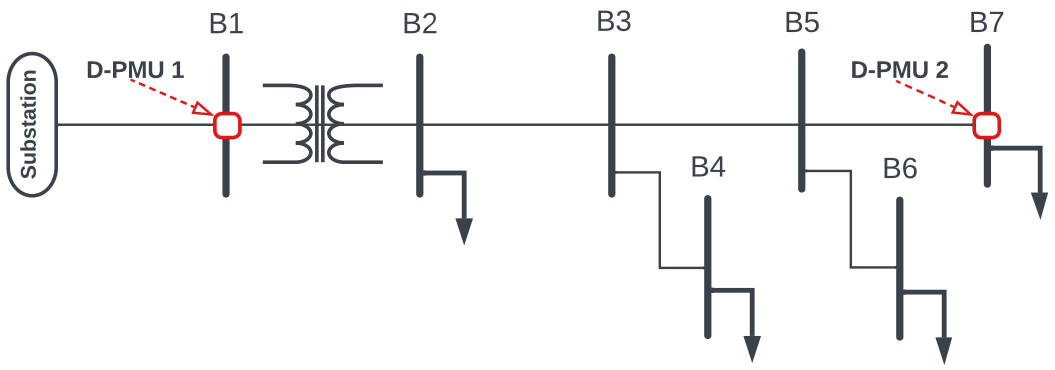


Figure 4.1: An example power distribution network with $N = 7$ buses. Two PMUs are installed are installed at buses in B1 and B5. We have $\mathcal{M} = \{B1, B7\}$.

are captured during event i by the D-PMU at bus j, where $j \in \mathcal{M}$. Similar to [77], such time series is assumed to be a *window* of the following measurements at a given D-PMU: the per-phase magnitude of voltage V_ϕ, the per-phase magnitude of current I_ϕ, and the per-phase power factor PF_ϕ, where ϕ is the given phase, i.e., $\phi \in \{A, B, C\}$. The reason for using these measurements is to remove the impact of off-nominal frequencies in the phase angle measurements, see [78, pp. 113].

As for the buses in set $\mathcal{B} \backslash \mathcal{M}$, we do *not* have any phasor measurement available at these buses. Therefore, we inevitably assume a *constant* value, i.e., a flat time series, for V_ϕ, I_ϕ, and PF_ϕ at these buses during the event. We obtain such constants by running a simple steady-state power flow analysis based on the *nominal load* (NL) at each bus. Such analysis is readily available in practice by using the utility's models of its feeders in standard software, such as CYME [79] and Synergy [80].

4.2.1 Graph Learning Approach

In this paper, we use GNN to conduct *topology-based representation learning.* GNN is the general framework for defining deep neural networks based on *graph data* [81].

To benefit from the GNN attributes, we need to translate the power system topology and the measurements into *graph-structured data.* Suppose A is *the adjacency matrix* for the graph of the power distribution network, where each node in the graph is a bus and each link in the graph is a distribution line. For each event i, we define an *event graph*, denoted by G_i, which has the same adjacency matrix A. For each node j in graph G_i, we define X_i^j as the input matrix. In this regard, if we have the measurements for M events in the data set, then the set of graph-structured data can be shown as:

$$\{G_1, G_2, \ldots, G_M\}. \tag{4.1}$$

The main reason for using GNN is to encode the graph-structured data G_i for each event i to a single graph-level representation vector with low dimension, which incorporates both the measurements at event i and the system topology. Such low-dimension representation helps to achieve a more accurate, interpretive and distinctive event clustering outcome.

Similar to the neural networks (NNs), GNNs can include multiple hidden layers with trainable weights. However, GNNs also take into account the graph topology or the adjacency matrix. This means that, each nodal vector data at any hidden layer in a GNN is updated based on not only its own trainable weights, but also its *neighboring nodes*' nodal vector data.

To see the importance of differences and similarities between NNs and GNNs in the context of the analysis in this paper, let us define X_i as the input matrix for graph G_i, such that row j of matrix X_i is the stacked vector of time series in X_i^j. Also, let us define H_i^k as the hidden matrix data for graph G_i at hidden layer k. We note that $H_i^0 = X_i$. In a common NN model we obtain the input matrix on the next layer by conducting forward propagation such as:

$$H_i^{k+1} = \sigma(H_i^k W^k), \tag{4.2}$$

where σ is the activation function, e.g., $\mathrm{ReLU}(x) = \max(0, x)$, and W^k is the trainable weight matrix of layer k.

In the NN framework, the input matrix X_i and the hidden layer matrix H_i^k contain different samples in their rows, which are often independent and identically distributed random variables. However, when it comes to a GNN, these samples (rows of data) are *related to each other*. In this paper, these samples are the nodal data at each bus, which are simultaneously captured during the same event, but from the viewpoint of the sensors at different buses on the power distribution system. These samples are related to each other through the physics of the distribution system and the network topology. In the GNN framework, each nodal hidden layer data is updated based on its own NN output as well as its neighbours' NN output by using the adjacency matrix A. The revision of the equation in (4.2) for the case of GNN will be given in Section II-B.

In the next three sub-sections, we will explain how to implement our proposed graph learning method. In Section 4.2.2, we will build an unsupervised GNN-based graph encoder to transform each G_i to a single vector. In Section 4.2.3 we will set the objective of the graph encoder to maximize the mutual information between its node-level data and its graph-level data. Finally, in Section 4.2.4 we will develop a discriminator module to calculate the aforementioned mutual information.

4.2.2 Graph Encoder

In this section, we develop a GNN-based graph encoder, denoted by $\mathcal{E}$, in order to learn a single vector that summarizes the time series for each graph-structured data G_i. Such vector will ultimately serve as the graph-level representation for each event. It is obtained by encoding the underlying shared properties of the data based on the topology of the system. The encoding process is based on maximizing mutual information between the node-level representation at each bus and the graph-level representation, which involves all of the buses.

We construct the graph encoder by using GCN [82] with the following updating formulation in its hidden layers[1]:

$$H_i^{k+1} = \sigma(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}H_i^k\omega^k). \tag{4.3}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix with added self-connections, D is the *degree matrix*, where $D_{aa} = \Sigma_b\tilde{A}_{ab}$, and ω^k is the set of trainable weights in the k^{th} layer of the GNN. The main difference between the formulation in (4.3) and the one in (4.2) is the use of $D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}$, which aggregates the nodal data from neighbouring nodes. This element also symmetrically normalizes the rows of matrix H_i^{k+1} cf. [82].

[1]Other similar modules, such as those in [83] and [84], can also be used.

For each graph G_i and each hidden layer k, let $\mathbf{h}_i^k(j)$ denote row j of matrix H_i^k. We refer to $\mathbf{h}_i^k(j)$ as the node-level (or *local*) representation of the event. Accordingly, for each node let us put together all such node-level representations at all the layers $k = 1, \ldots, K$ as follows:

$$\mathbf{h}_i^{\omega}(j) = [\mathbf{h}_i^1(j), \mathbf{h}_i^2(j), \ldots, \mathbf{h}_i^K(j)]. \tag{4.4}$$

Superscript ω in $\mathbf{h}_i^{\omega}(j)$ indicates the set of parameters for graph encoder $\mathcal{E}$. Furthermore, let us define:

$$\mathbf{h}_i^{\omega,g} = S(\{\mathbf{h}_i^{\omega}(1), \ldots, \mathbf{h}_i^{\omega}(N)\}) \tag{4.5}$$

as the graph-level (or *global*) representation of event i, where S is a permutation invariant function that summarizes the node-level representation vectors to a single graph-level representation vector, such as via element-wise mean or max functions [81]. The outputs of graph encoder are obtained as:

$$\{\mathbf{h}_i^{\omega}(j), \mathbf{h}_i^{\omega,g}\} = \mathcal{E}(G_i), \quad \forall j = 1, \ldots, N, \tag{4.6}$$

which include all the N node-level representations and a single graph-level representation for each event i. The objective in the design of the graph encoder is to find the structural dependencies among the vectors from that are listed in (4.6).

4.2.3 Mutual Information

Similar to the unsupervised learning methods in [85] and [86], the objective for the proposed graph encoder is to maximize the average mutual information (MI) [87] of the graph-level representation $\mathbf{h}_i^{\omega,g}$ and all of the node-level representations $\mathbf{h}_i^{\omega}(j)$ for any

$j = 1, \ldots, N$ for each event i as:

$$\text{Maximize: } \mathcal{I} = \frac{1}{M}\sum_{j=1}^{M} \frac{1}{N}\sum_{i=1}^{N} \text{MI}(\mathbf{h}_i^{\omega}(j); \mathbf{h}_i^{\omega,g}). \tag{4.7}$$

The above maximization enforces the GNN graph-level representation to carry the type of information that is present in all of the nodes in the network and all the layers [86]. It should be mentioned that, in this paper, we focus on graph-level representation learning, rather than on substructure representation learning. The latter strictly focuses on node-level tasks, such as for the node classification in [82].

Calculating $\mathcal{I}$, in a continuous and high-dimensional settings is difficult. A solution is suggested in [88], in which we use a mutual information *estimator* between the input and the output of the deep neural networks. This method, is based on training a classifier (a discriminator) that separates samples from the joint distribution and their product of marginals.

4.2.4 Discriminator: Positive and Negative Graphs.

The first step in *estimating* the mutual information is to define the joint and marginal distributions. The joint distribution, i.e., positive samples in this paper, are defined as node-level/graph-level representation pairs $(\mathbf{h}_i^{\omega}(j), \mathbf{h}_i^{\omega,g})$, for each *actual event* G_i. Also, we refer to G_i as *positive graphs*.

We also need to construct negative samples in order to define the product of marginals. Note that, the choice of the negative samples has impact on the type of structural information that is desirable to be captured as a byproduct of estimating MI [86]. First, we construct the *negative graphs*. They have the same input node data as in the positive

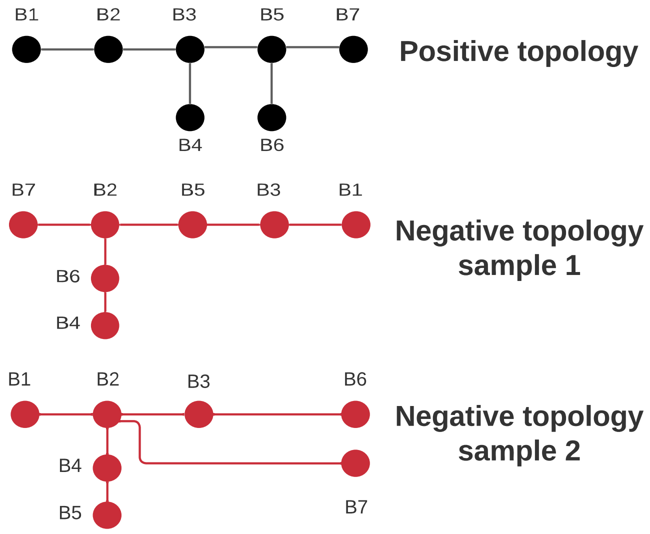


Figure 4.2: The actual (i.e., *positive*) topology of the power distribution network in Fig. 1 is shown in black. Two arbitrary alternative (i.e., *negative*) topologies with the *same* number of nodes/buses and edges/lines are shown in red.

graphs, i.e., X_i^j for event i and node j. But they have a different graph topology, which are *random trees* with the same number of nodes and links. Next, we consider any pair of a graph-level representation from an actual event graph (a positive graph) with any node-level representation of a negative graph, that are obtained from graph encoder, as a negative sample.

The concept of positive graphs and negative graphs is illustrated in an example in Fig. 4.2. The positive topology represents the actual topology of the power distribution system that we saw in Fig. 4.1. The other two arbitrary negative topologies are used to shape two samples of negative graphs. These positive and negative samples are used to train the discriminator-based method that is proposed in [88]. This approach enforces the encoder to learn the structural dependency of the data, which leads to an overall MI maximization in an average sense.

In this study, discriminator Ψ is a neural network with a set of parameters ψ. We set the discriminator to output 1 for a positive sample and a 0 for negative sample. Accordingly, based on the Jensen-Shannon MI estimator that is proposed in [85], and the method in [88], we *simultaneously estimate and maximize* the objective function $\mathcal{I}$ in (4.7) as shown below:

$$\begin{aligned}\hat{\mathcal{I}}_{\omega,\psi} = \frac{1}{2MN}\sum_{i=1}^{M}\Big(&\sum_{j=1}^{N} E_P[-\sigma(-\Psi(\mathbf{h}_i^{\omega}(j),\mathbf{h}_i^{\omega,g})] \\ &- \sum_{j=1}^{N} E_{P\times P'}[\sigma(\Psi(\mathbf{h}_{i'}^{\omega}(j),\mathbf{h}_i^{\omega,g})]\Big),\end{aligned} \tag{4.8}$$

where $\hat{\mathcal{I}}_{\omega,\psi}$ is the Jensen-Shannon MI estimator; and $\sigma(x) = log(1 + e^x)$ is the softplus function. In this study, for each physical (i.e., positive) event i, we make a corresponding random negative event $i^{'}$. Accordingly, the probability distribution of the positive events, which is denoted by P, is identical to the probability distribution of the negative events, which is denoted by $P^{'}$. Also, E_P and $E_{P\times P'}$ are the expected value for the discriminator output related to the positive samples (or the joint distribution) and the negative samples (or the product of marginals), respectively. Due to the summation, for both negative and positive samples, we include the coefficient 1/2 in (4.8). Notations $\mathbf{h}_i^{\omega}(j), \mathbf{h}_{i'}^{\omega}(j)$ and $\mathbf{h}_i^{\omega,g}$ indicate the outputs of the graph encoder $\mathcal{E}$, and represent the node-level representation of the positive event i in node j, the node-level representation of the negative event $i^{'}$ in node j, and the graph-level representation of the positive event i, respectively.

Algorithm 5 Topology-Based Representation Learning

```
1:  Input: Event time series $X_i^j$ and network topology $A$.
2:  Output: Graph-level representation vectors clusters.
3:  // Positive and Negative Graphs
4:  For each training event $i$ Do
5:    Construct the positive graphs and assign $X_i^j$ to all nodes.
6:    Construct the negative graphs and assign $X_i^j$ to all nodes.
7:  End
8:  // Training Graph Encoder and Discriminator
9:  For each epoch of training data Do
10:   Obtain $\{\mathbf{h}_{g_b}^{\omega}(j), \mathbf{h}_{g_b}^{\omega,g}\} = \mathcal{E}(g_b)$ for all positive graphs.
11:   Obtain $\{\mathbf{h}_{g_b'}^{\omega}(j)\} = \mathcal{E}(g_b')$ for all negative graphs.
12:   Pair each $\mathbf{h}_{g_b}^{\omega}(j)$ with its relative $\mathbf{h}_{g_b}^{\omega,g}$ as positive
      sample; and label the discriminator's output as 1.
13:   Pair each $\mathbf{h}_{g_b'}^{\omega}(j)$ with its relative $\mathbf{h}_{g_b}^{\omega,g}$ as negative
      sample; and label the discriminator's output as 0.
14:   Calculate the loss function in (4.8).
15:   Update the $\omega$ and $\psi$ by conducting back propagation
      and using Adam optimizer [65].
16: End
17: // Graph-level Representation
18: For each graph $G_i$ Do
19:     Obtain $\{\mathbf{h}_i^{\omega,g}\} = \mathcal{E}(G_i)$.
20: End
21: // Clustering
22: Cluster the event vectors $\{\mathbf{h}_i^{\omega,g}\}$ using GMM.
```

4.2.5 Clustering

After training the graph encoder, the graph-level representations $\mathbf{h}_i^{\omega,g}$, are obtained for all events $i = 1, \ldots, M$, and they are clustered by using the Gaussian Mixture Model (GMM). The GMM uses expectation maximization algorithm for fitting a mixture of Gaussian models to the training data set, considering a pre-defined number of clusters. Then, each $\mathbf{h}_i^{\omega,g}$ is assigned to the most probable cluster.

Note that, the purpose of our proposed method is to properly incorporate the topological information from the sensor measurements to learn the most distinctive representation for the type of each event, such that we can enhance the clustering accuracy with the already existing clustering methods. We shall note that, we did examine other clustering methods, such as K-means and DBSCAN; however, GMM demonstrated the highest average clustering performance.

4.2.6 Algorithm: Topology-Based Representation Learning

Algorithm 5 shows the summary of the steps that we took in Sections 4.2.2 to 4.2.5. It is divided into four segments. First, we generate the positive and negative graphs; see lines 4 to 7. Next, we train the graph encoder and the discriminator; see lines 9 to 16. Third, we obtain the graph-level representations for all the events; see lines 18 to 20. Finally, we do the clustering task using GMM and based on the obtained graph-level representations of the events; see line 22.

4.3 Temporal Representation Learning

The design that we presented in Section 4.2 can fully incorporate the knowledge about the topology of the network and the relative location of the measurements into the task of event clustering. However, if we use the method in Section 4.2 *as is*, then it may *not* result in a significant improvement compared to some benchmark methods in the literature. The main issue here is the *high dimentionality* in the time series that needs to be placed at each node of the graph in this field.

4.3.1 Tackling High Dimensionality

To address the above issue, we propose to compress the data in the time series by learning the temporal-dependent features of the events. By compressing the event data in time domain, we can lower the dimension of the feature space. This leads to achieving a higher computational and clustering efficiency with less numerical challenges.

Accordingly, an Auto-Encoder-Decoder (AED) [89] model is proposed which includes Long Short Term Modules (LSTM) [63] for proper temporal-based representation learning of each node time series for each event. AED constitutes of two main parts. The first part is the temporal encoder (E), which tries to summarize and transfer each event time series matrix X_i^j into a single embedding vector (Em_i^j). The second part is the temporal decoder (D), which tries to reconstructs the actual time series with the mentioned embedding vectors.

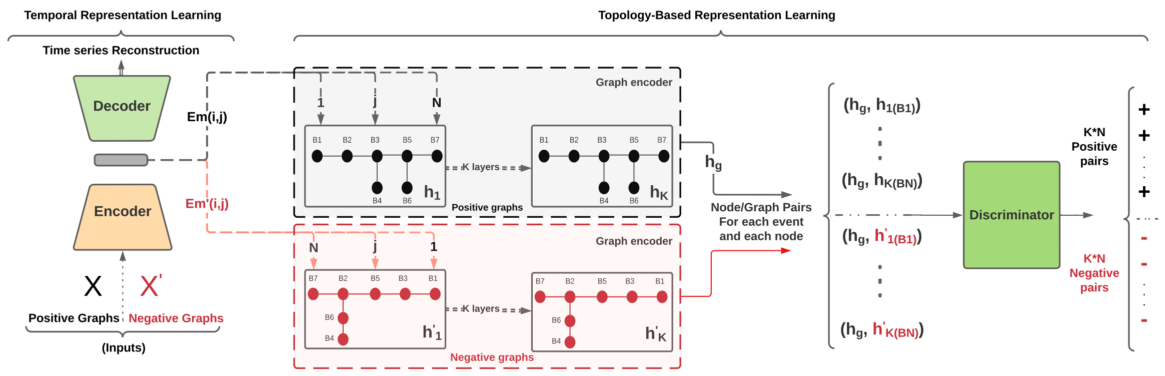


Figure 4.3: GraphPMU: AED learns the optimal representation vectors for nodal data for all events. Then these vectors alongside the positive and negative graphs are used as input for the proposed graph encoder. The output of the graph encoder, node-level and graph-level representations are paired to shape the positive and negative samples. Then, discriminator learns to discriminate between these samples for MI maximization.

The objective function of AED is to minimize the Mean Square Error (MSE) of the time series input to E and the time series output of D. Here are the details of the AED model:

$$Em_i^j = E(X_i^j), \forall i = 1, ..., M, \forall j = 1, ..., N, \tag{4.9}$$

$$\theta_E, \theta_D = \underset{\theta_E, \theta_D}{\operatorname{argmin}} \frac{1}{M} \sum_{j=1}^{M} \frac{1}{N} \sum_{i=1}^{N} \operatorname{MSE}(X_i^j, D(Em_i^j)), \tag{4.10}$$

which θ_E, θ_D are the encoder deep neural network and decoder deep neural network parameters, respectively.

In this study, the parameters of AED are shared between all the buses. In other words, instead of considering multiple AEDs for each bus, we rather implement a global AED. This makes the training process faster. It also allows the AED model to take advantage of the learned features from different locational time series. This prevents an over-fitting over a single bus data. After training the AED model, all event time series data for each node X_i^j can be encoded to their Em_i^j by using (4.9).

4.3.2 Algorithm: Temporal Representation Learning

Algorithm 6 shows the steps for temporal representation learning. First, we train the temporal encoder and decoder; see lines 4 to 8. After that, we obtain the compressed embedding vector for all events and nodes time series; see lines 10 to 12.

Algorithm 6 Temporal Representation Learning

```
1:  Input: Normalized event time series $X_i^j$.
2:  Output: Embedding vectors $Em_i^j$ as in (4.9).
3:  // Training Phase
4:  For each epoch Do
5:      Pass $X_b$ to the temporal AED ($E$ and $D$).
6:      Calculate loss function from (4.10).
7:      Update $\theta_E$ and $\theta_D$ through back propagation [65].
8:  End
9:  // Embedding Extraction
10: For each event $i$ and node $j$ Do
11:     $Em_i^j = E(X_i^j)$.
12: End
```

4.4 GraphPMU: Combining Topology-Based and Temporal Representations

We are now ready to introduce our ultimate GraphPMU method by combining the topology-based representation learning design in Section 4.2 with the temporal representation learning design in Section 4.3. Fig. 4.3 shows how these two design components are integrated in order to achieve GraphPMU.

The architecture in Fig. 4.3 can be explained by going through its parts from left to right. The process starts with training the time domain AED with matrices X_i^j. This step is independent from the network topology; hence, it is the same for positive and negative graphs, as they have the same input time series.

After the AED is trained, the obtained embedding vectors Em_i^j from the temporal encoder E, are used to train the GNN model. Subsequently, the positive and negative graphs are shaped based on the embedding vectors and the defined topologies in Section 4.2.4. These positive and negative graphs are passed to the graph encoder $\mathcal{E}$ to construct the node-level and graph-level representations. Then the obtained positive and negative samples are used to train the discriminator ψ.

Given the models for (E and D) and ($\mathcal{E}$ and Ψ), the graph-level representation vectors are obtained as $\mathbf{h}_i^{\omega,g} = \mathcal{E}(E(X_i^j))$ for all buses $j \in \mathcal{B}$. These graph-level representations are then clustered by the GMM method as we explained in Section 4.2.5.

4.5 Extension to Incorporate Harmonic Synchro-Phasors

So far, we have assumed that all the phasor measurements are obtained at the fundamental frequency. This is indeed the state of practice in this field for a typical PMU. However, as we mentioned in Section 1, it is envisioned that standard D-PMUs may in the future also act as H-PMUs to provide the phasor measurements not only at the fundamental frequency but also at selected harmonic frequencies. Accordingly, in this section, we will expand the GraphPMU model to incorporate such emerging advancement in data availability in this field.

4.5.1 More Distinctive Event Signatures

Without loss of generality, we assume that each D-PMU provides the synchronized phasor measurements for the 3rd and 5th harmonics, in addition to the fundamental frequency. Expanding the analysis to include higher harmonic orders would be similar, although it may not be necessary; because most events manifest themselves properly in either the 3rd or the 5th harmonics, or in both. At each bus j in $\mathcal{M}$, we collect V_ϕ, I_ϕ, and PF_ϕ; however, this is done not only for the fundamental frequency but also for the 3rd and the 5th harmonics. In this regard, taking into account the harmonic phasors can be highly beneficial as they can demonstrate *more distinctive signatures* for the purpose of clustering the events. This can help compensate for some of the challenges in having locationally-scarce measurements; thus, contributing to the overall success in the proposed GraphPMU method.

As an example, Fig. 4.4 shows the event signatures in different types of phasor measurements during a single-line-to-ground fault. The event signature in the fundamental frequency in Fig. 4.4(a) is a simple voltage sag and a simple inrush current. However, the event signatures in the harmonic phasor measurements at the 3rd harmonic in Fig. 4.4(b) and at the 5th harmonic in Fig. 4.4(c) are considerably more distinctive.

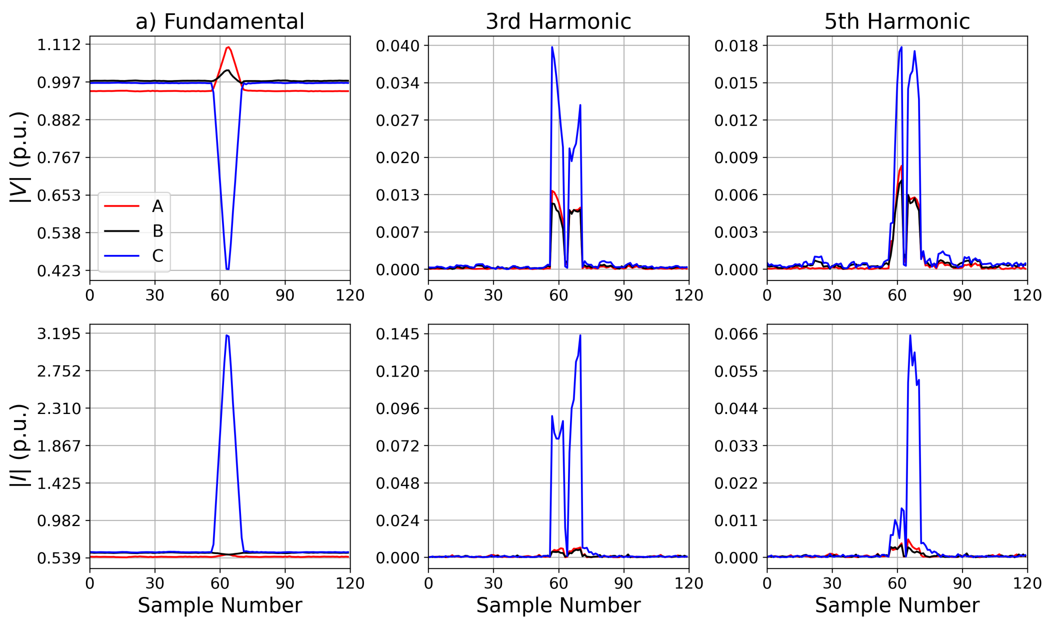


Figure 4.4: Comparing the signatures during the *same* event at the fundamental phasor measurements vs. at the 3rd and 5th harmonic phasor measurements.

4.5.2 Extended Temporal-Based Learning

Same as in the case for the fundamental phasors data, we use AED to learn time domain representations for the time series of the harmonic phasor measurements. Accordingly, we obtain the embedding vectors to use them in the clustering process. Importantly, since the strength and the overall nature of the time series of the harmonic phasors are different from those of the fundamental phasors, we must train *different* AEDs for *each* fundamental or harmonic order. This makes the time domain representation learning more reliable and more accurate than using a single AED for these different time series. The training is done by using Algorithm 6. We concatenate all the embedding vectors to form:

$$Em_i^j = [Em_i^{j,1}\ Em_i^{j,3}\ Em_i^{j,5}]^T. \tag{4.11}$$

4.5.3 Extended Topology-Based Learning

Next, we feed the new vectors Em_i^j that are derived in (4.11) as the input to the GNN using Algorithm 5 to complete the process for the event clustering task. Since the size and nature of the input vector is different from those in Section 4.2. We need to re-train the GraphPMU based on the new vector of features. Last but not least, for each bus $j \in \mathcal{B}\backslash\mathcal{M}$, which does not have a sensor, we use zero padding concatenation to the fundamental embedding vectors that the previously obtained in section 4.3). This is because the default steady state values in unobserved locations are assumed not to have any harmonics.

4.6 Case Studies

In this section, we conduct various case studies based on the IEEE 34-bus three-phase power distribution test system, which is shown in Fig. 4.5. The network simulation model is developed in PSCAD to assure capturing the transient signatures of the events [90]. Nine different types of events are simulated:

1. Three-Phase Capacitor Bank switching at bus 840
2. Three-Phase Capacitor Bank switching at bus 849
3. Single-phase load switching at bus 858
4. Three-phase load switching at bus 836
5. Three-phase motor-load switching at bus 812

6. Three-phase motor load switching at bus 828
7. Single-phase-to-ground fault at bus 852
8. Two-phase-to-ground fault at bus 862
9. Three-phase-to-ground fault at bus 816.

Unless stated otherwise, we assume that there are only four phasor measurement units are available on the power distribution network. The location of the D-PMUs (H-PMUs) are shown on Fig. 4.5. Note that, we have:

$$\mathcal{M} = \{806, 824, 836, 846\}. \tag{4.12}$$

Depending on the case study, we assume that each D-PMU either provides the phasor measurements only for the fundamental component, or for the fundamental component together with the 3rd and the 5th harmonics. As in practice, we assume that events occur rarely [77]; therefore, we assume only a small number of each type of event are available to train GraphPMU. We augmented the data from the few available events by conducting time shifting and adding noises to the raw data. This is done for each type of event and for all sensors. In total, we considered 50,000 events of various types for training, 5000 events for evaluation, and 5000 events for testing.

4.6.1 Parameters of GraphPMU

The graph encoder has two layers of GCN [82], where the sizes of the vectors for the hidden layers are 128 and 64, respectively. The discriminator contains two fully-connected layers with 192 and 32 neurons. We concatenate the hidden layer features and

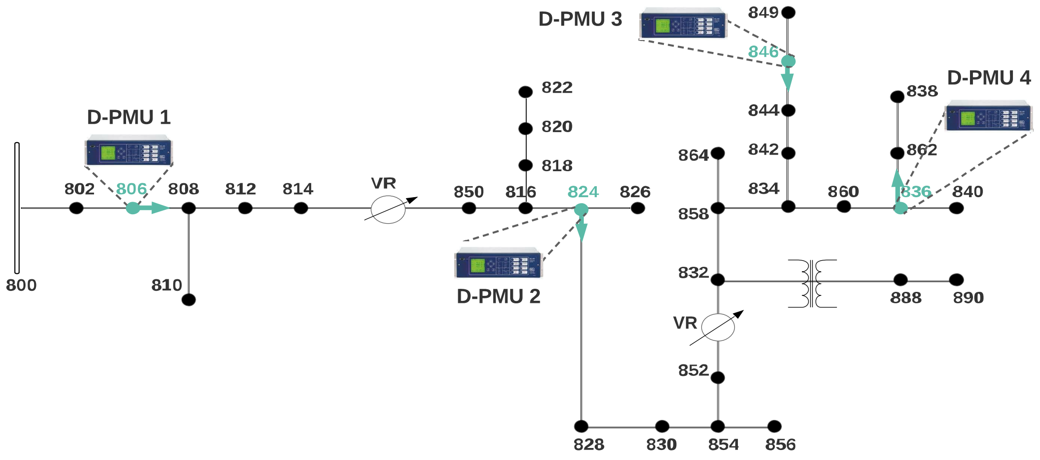


Figure 4.5: The IEEE 34-bus test system with locationally-scarce phasor measurements. There are only four D-PMUs (H-PMUs) available on this network, as marked on the figure. However, the events can happen at *any* location.

the global graph features together; thus, the input size of discriminator is 128+64 = 192. By intentionally choosing a naive discriminator with only two fully-connected NNs, we enforce the GNN encoder to learn more discriminative features. This can help with event clustering.

The encoder portion of the AED has two layers of LSTM modules with 32, and 64 units, following with a 32 units fully- connected layer. The decoder portion is an almost reverse version of the encoder, with a fully-connected 64×125 layer, followed by two LSTM layers with 64 and 32 units.

All activation functions are LeakyReLU, where the slope of the leak is 0.2. For tuning the hyper-parameters, we used the *coarse-to-fine* method [77]. The learning rate α is $1e^{-3}$ for Adam optimizer, and β_1 is 0.5 for better stability in training. All models are developed with Pytorch. The GNN models are built with the Deep Graph Library [91], by using Nvidia GTX 1050 ti GPU and a core i-7 2.2GHz CPU with 32 GB RAM.

The MSE for the training and testing stages in the fundamental phasor are 0.04425 and 0.04522, respectively. This shows that the encoder is able to compress high-resolution data to a low dimension such that the decoder can reconstruct the time series with high accuracy. This confirms the performance of the AED sub-system for the rest of our analysis.

In this paper, Adjusted Rand Index (ARI) score [92] is used to assess accuracy in a clustering task. ARI is a number between 0 and 1. A higher ARI means a better clustering.

4.6.2 Comparison with Temporal-Based Benchmarks

Table 4.1 shows the ARI for the proposed event clustering method (in the last row), in comparison with several benchmark methods (in the first nine rows). In this section, our focus is on the *top segment* in Table 4.1, i.e., the first five methods. These are the methods that do *not* use any information about the network topology. These five methods are AED [89], DEC [93], Kernel k-means, k-Shape clustering and k-means clustering methods [94]. Here, Time Series (TS) means that the method uses the raw time series data, without any encoding.

From Table 4.1, among the five methods that do *not* use graph models, DEC and AED have the highest accuracy. To have a fair comparison, we assume the same steady-state constants at the buses without sensors for all the ten methods in Table 4.1.

4.6.3 Comparison with Topology-Based Benchmarks

The next five methods in the *bottom segment* of Table 4.1 *do* use the information about the network topology. All of these combinations could have been used for our purpose.

Table 4.1: ARI Score for Different Methods Under Locationally-Scare Phasor Measurements at Four Buses

	Method	ARI
Without Graph Model	AED	0.473
	DEC	0.520
	Kernel TS	0.237
	k-shape TS	0.418
	k-means TS	0.343
With Graph Model	TS + N/G + NL	0.487
	AED + N/G	0.423
	AED + N/G + RL	0.533
	AED + G + NL	0.585
	GraphPMU = AED + N/G + NL	**0.720**

However, only the last row shows our ultimate design for GraphPMU. The rest of the methods serve as benchmarks. Regarding the new abbreviations in Table 4.1, G means using only the graph-level representation in the GNN, N/G means using both the node-level and the graph-level representations, NL means using the nominal load flow model to obtain the constants at the buses with no sensors, RL means using random loading data instead of using nominal loading data.

A) Advantage of Using Data Compression

If we compare TS+N/G+NL versus GraphPMU in Table 4.1, we can see that their difference is only in the use of AED instead of TS. Importantly, since the input to the GNN is more compressed in GraphPMU, it becomes more distinctive for the GNN, as opposed to using the raw time series in TS+N/G+NL. Thus, the overall performance in event clustering is much better for the GraphPMU. Nevertheless, the use of topology information in TS+N/G+NL can still outperform most of the benchmark methods in the top segment of Table 4.1 that do *not* use any graph model.

B) Advantage of Pairing Node-Level and Graph-Level Vectors

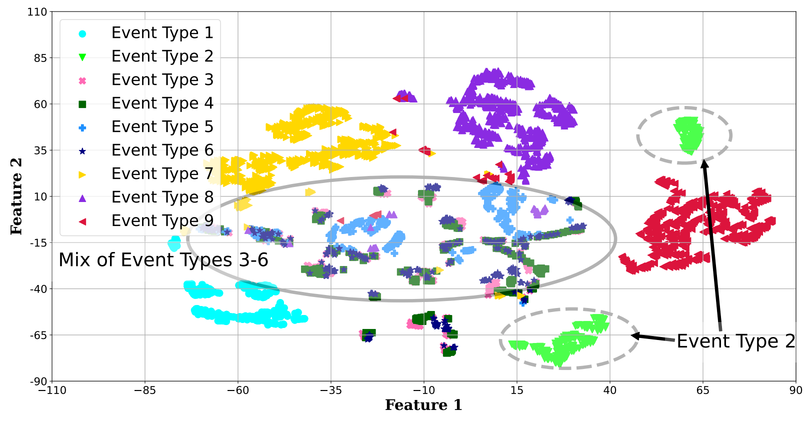
If we compare AED+G+NL versus GraphPMU in Table 4.1, we can see that their difference is only in terms of using G versus N/G. The method with AED+G+LN considers only the last layer of the graph learning model for the positive and the negative graphs for the MI maximization, rather than using the node-level/graph-level pairs. However, the use of such pair in GraphPMU is necessary to properly extract the shared structure between the node-level and the graph-level representations, in order to have more distinctive clusters.

C) Advantage of Using Nominal Load Data

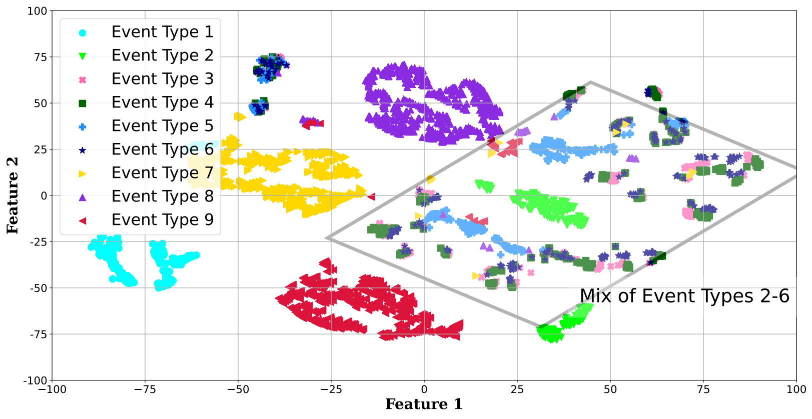
If we compare AED+N/G versus GraphPMU in Table 4.1, we can see that their difference is only in terms of using N/L. In AED+N/G, we do *not* include the buses with no sensors in graph-based learning. As a result, the accuracy of the method drops significantly. The reason is that there are only four nodes on the graph, i.e., the four buses with sensors. This is due to the locational scarcity of the sensors. Such a small graph does *not* give much room to benefit from topology-based learning. As for AED+N/G+RL, this method too suffers a considerable drop in performance. These results confirm that we *do* benefit from conducting a simple power flow analysis based on the nominal loading data.

4.6.4 Analysis Based on Different Types of Events

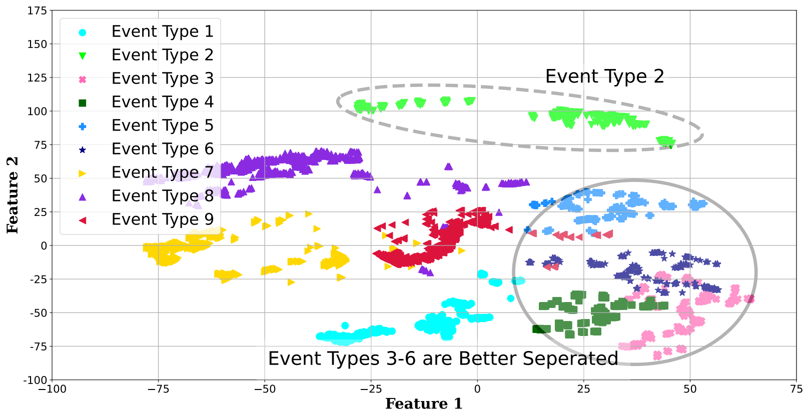
Fig. 4.6 shows the t-distributed stochastic neighbor embedding (t-SNE) scatter plot of all *test* events for three methods: a) DEC; b) AED+G+NL; c) GraphPMU. Each point indicates one event. The shapes and colors indicate the *true labels* of the nine different



(a) DEC, ARI = 0.524



(b) AED+G+NL, ARI = 0.585



(c) GraphPMU, ARI = 0.720

Figure 4.6: The t-SNE scatter plots for the test events for three different methods based on two main features. Only four D-PMUs are available.

event types. One of the major weaknesses of the methods that do *not* use graph models is their inability to properly cluster the “smaller” events, such as events types 3, 4, 5, and 6. For example, see the area in Fig. 4.6(a) that is marked with an oval with a solid line. These four different event types are all mixed up in this area. Accordingly, the DEC method is not able to distinguish event types 3-6.

Next, consider the results in Fig. 4.6(b), which are for AED+G+NL. The area that is marked with a diamond shows that AED+G+NL too is incapable of separating the “small” events. However, its ARI is slightly higher than that of DEC due to the more distinctive clusters for the “major” event types 1, 7, 8 and 9. However, the DEC method has incorrectly split event type 2 into two separate groups of points, as we see in the two separate circles with dashed lines in Fig. 4.6(b).

GraphPMU addresses all of these shortcomings, as we can see in Fig. 4.6(c). On one hand, GraphPMU tends to separate the “major” event types as far as possible. For example, in the dashed oval area in Fig. 4.6(c), all the points for event type 2 are close to each other and away from the rest of the events. This highly improves the accuracy in clustering event type 2.

On the other hand, GraphPMU also maintains the “smaller” event types reasonably away from each other. For example, in the circle area that is marked with a solid line in Fig. 4.6(c), the points for event types 3, 4, 5, and 6 are separated from each other much better compared to the other figures.

Table 4.2: ARI Score for GraphPMU and the Top Two Methods without Graph Models when Adding Harmonic Phasor Measurements

Method	ARI
AED (Fundamental + Harmonics)	0.666
DEC (Fundamental + Harmonics)	0.694
GraphPMU (Fundamental + Harmonics)	**0.814**

4.6.5 Impact of Adding Harmonic Phasor Measurements

Table 4.2 shows the event clustering results for AED, DEC and GraphPMU when we use not only the fundamental phasor measurements but also the harmonic phasor measurements. By comparing Table 4.2 with Table 4.1, we can see that the performance in event clustering has highly improved in all three methods. This is due to the more distinctive transient signatures for different event types, as we saw in Section 4.5.

Among the nine event types, *unbalanced events* i.e., event types 3, 7 and 8, have the highest accuracy improvements. Based on Tables I and II, GraphPMU significantly outperforms the rest of the methods, whether we only use the fundamental phasor measurements as in Table I, or we use both the fundamental and harmonic phasor measurements as in Table II. An ARI of 0.814 is very high, given that we have sensors in 4 of the 34 buses, i.e., only in 12% of the buses.

4.6.6 Impact of the Number of D-PMUs

Fig. 4.7 shows the ARI scores for GraphPMU in comparison with two other methods versus different number of available sensors. We can identify *three patterns* in these figures. First, GraphPMU always outperforms the rest of the methods. Its relative superior performance is the highest when we have fewer sensors, i.e., under the *locational-scarcity*

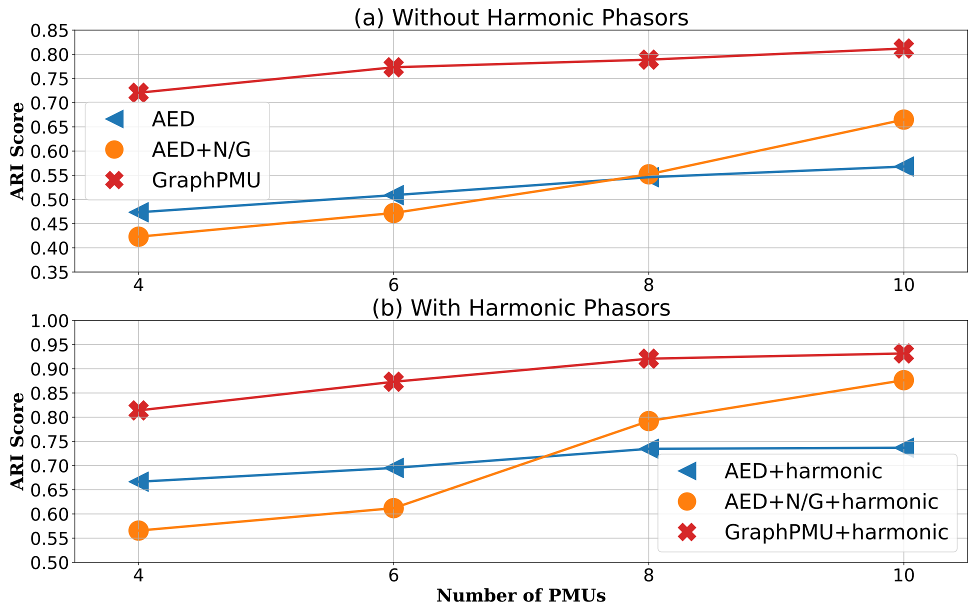


Figure 4.7: ARI scores for AED, AED+N/G and GraphPMU methods vs. the number of D-PMUs, *with* and *without* using harmonic synchrophasors.

conditions. Second, as we increase the number of available sensors, the overall clustering accuracy improves for *all* these methods. Third, AED+N/G always has a worse performance than AED under severe locational-scarcity, but it surpasses AED as we increase the number of sensors. This is due to the fact that, AED+N/G is capable of taking advantages of the information about the network topology only when we have several sensors available. This shortcoming is addressed by GraphPMU.

In Fig. 4.7(b), GraphPMU achieves a very high ARI score of 0.92 with only 8 H-PMUs in a network with 34 buses. Fig. 4.8 shows the performance of GraphPMU in clustering different types of events, when there are 10 sensors available. If we compare Fig. 4.8 with Fig. 4.6(c), we see that having more D-PMUs helps GraphPMU to put almost all events in correct separated clusters, for both “major” or “small” event types.

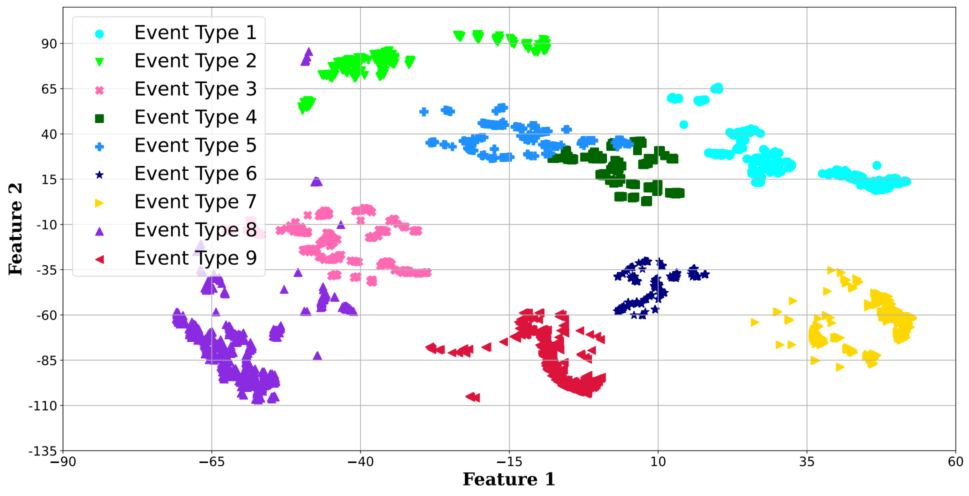


Figure 4.8: The t-SNE for GraphPMU for all test events when using 10 D-PMUs.

Chapter 5

Conclusions and Future Work

5.1 Summary of the Conclusions

In this dissertation, we conduct a comprehensive event analysis in smart meter and micro-PMU time series data and propose multiple unsupervised event detection and event clustering methods.

In Chapter 2, we examine the performance of four online unsupervised machine learning abnormality detection methods to detect abnormalities in smart meter data. The real-world data traces are used for this purpose. Four key conclusions are made. First, it is observed that, in general, i.e., when all available features are considered, clustering-based methods, such as IF, have a better performance that prediction-based and projection-based methods. Second, prediction-based methods gain their best performance when the prediction model simulates the previous consumption trend accurately rather than following the upcoming real-time electricity consumption. Third, simulation results show that prediction based methods generally are more accurate for the abnormalities with very high or very low magnitude. Forth, projection-based methods, such as LODA, do not show promising performance for abnormality detection in smart meter data; however, LODA can demonstrate a slightly better performance through a better feature selection when only a certain subset of available features are utilized. In this regard, when it comes to the detection of abnormalities in smart meter data, it is better to customize the features for each method individually, despite the fact that the common practice in the previous literature is to consider the same set of features for all methods.

In Chapter 3, A set novel unsupervised deep learning methods are proposed to detect an cluster events in micro-PMU data streams. 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. They are capable of extracting the characteristics of a wide verity of events in large volumes of micro-PMU data. All methods are capable of detecting events with point-signatures and group-signatures. They are particularly well-suited to detect the events in distribution systems where the event may impact only a subset of the features and only or two phases; in addition to the cases that all three phases are affected. 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.

In Chapter 4, a novel unsupervised graph-representation learning method, called GraphPMU, was proposed to cluster different types of events in power distribution systems. The proposed method does not require any prior knowledge about the events. It is solely based on the event signatures in D-PMU (and H-PMU) measurements, as well as the information about the network topology. Importantly, GraphPMU is meant to address a challenging scenario, where the phasor measurements are *locationally scarce*. By conducting a comprehensive data-driven analysis, it was shown that the proper combination of *topology-based* and *temporal-based* representation learnings of phasor measurements can result in very high clustering accuracy. The results of different case studies confirmed that the proposed method outperforms the existing methods in the literature. By using the measurements from not only fundamental but also the harmonic phasors, we further improved the clustering accuracy, particularly for unbalanced event types.

5.2 Future Works

In the future, the results of Chapter 2 can be used for detecting households with less abnormalities for demand response purposes as their loads are more predictable and reliable for system operators. The analysis in Chapter 3 can be used in different ways. First, system operators can extract unknown yet informative events from the high resolution micro-PMU data without any supervision. This will enable them to get notify by the system which can not only save them energy and time but also it will prevent human error. Second, the proposed active clustering method can group the events of a single micro-PMU based on their time series shape. This gives the system operator the capability to: 1) identify different types of event and their occurrence frequency in different time periods. 2) identify which quantities are impacted by a certain event, for instance is it a single line fault in phase A or a three phase event that affected all features in all phases. 3) identify enormous and unusual events by observing the time of detected events as well as the type of event in a sequence. The analysis in Chapter 4, future work may include: 1) extending the analysis to also achieve unsupervised event location identification; 2) applying the proposed method to other sensor measurements such as synchronized waveform measurements; and 3) incorporating some additional physical information to the graph-based analysis, such as the impedance of the distribution lines.

Bibliography

[1] H. Mohsenian-Rad, *Smart Grid Sensors: Principles and Applications.* Cambridge University Press, UK, April 2022.

[2] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Trans. on Smart Grid*, accepted for publication, March 2018.

[3] "EIA annual electric power industry report in 2020," https://www.eia.gov/electricity/data/eia861/.

[4] "Pecan street inc. (2018). pecan street dataport. austin, tx, usa. retrieved from," http://www.pecanstreet.org/energy/.

[5] 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.

[6] 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.

[7] 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. [Online]. Available: https://doi.org/10.1155/2007/57985

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] A. Tascikaraoglu and B. M. Sanandaji, "Short-term residential electric load forecasting: A compressive spatio-temporal approach," *Energy and Buildings*, vol. 111, pp. 380 – 392, 2016.

[15] E. Knorr and R. T. Ng, "Finding intentional knowledge of distance-based outliers." in *Proc. of the international conference on very large data bases*, San Francisco, CA, Jun 1999, pp. 211–222.

[16] Y. Liao, Y. Weng, C. w. Tan, and R. Rajagopal, "Urban distribution grid line outage identification," in *Proc. of Probabilistic Methods Applied to Power Systems (PMAPS), 2016 International Conference on*, Beijing, China, 2016, pp. 1 – 8.

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

[18] N. L. Tasfi, W. A. Higashino, K. Grolinger, and M. A. M. Capretz, "Deep neural networks with confidence sampling for electrical anomaly detection," in *Proc. of 2017 IEEE International Conference on Internet of Things*, Exeter, UK, Jun 2017, pp. 1038–1045.

[19] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak, "An ensemble learning framework for anomaly detection in building energy consumption," *Energy and Buildings*, vol. 144, pp. 191 – 206, 2017.

[20] J. LUO, T. HONG, and M. YUE", "Real-time anomaly detection for very short-term load forecasting," *Journal of Modern Power Systems and Clean Energy*, vol. 6, no. 2, pp. 235 – 243, Mar 2018.

[21] A. Saad and N. Sisworahardjo, "Data analytics-based anomaly detection in smart distribution network," in *Proc. of International Conference on High Voltage Engineering and Power Systems*, Sanur, Indonesia, oct 2017, pp. 1 – 5.

[22] 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.

[23] O. Ardakanian, Y. Yuan, R. Dobbe, A. von Meier, S. Low, and C. Tomlin, "Event detection and localization in distribution grids with phasor measurement units," in *IEEE PES General Meeting*, July 2017.

[24] Y. Zhou, R. Arghandeh, and C. Spanos, "Distribution network event detection with ensembles of bundle classifiers," in *IEEE PES General Meeting*, 06 2016.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

[39] 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.

[40] 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.

[41] Y. Yuan, Y. Guo, K. Dehghanpour, Z. Wang, and Y. Wang, "Learning-based real-time event identification using rich real pmu data," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5044–5055, 2021.

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

[43] 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.

[44] 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.

[45] D. Ma, X. Hu, H. Zhang, Q. Sun, and X. Xie, "A hierarchical event detection method based on spectral theory of multidimensional matrix for power system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2173–2186, 2019.

[46] S. Pandey, A. K. Srivastava, and B. G. Amidan, "A real time event detection, classification and localization using synchrophasor data," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4421–4431, 2020.

[47] W. Li and D. Deka, "Physics-informed graph learning for robust fault location in distribution systems," *arXiv e-prints*, pp. arXiv–2107, 2021.

[48] J. James, D. J. Hill, V. O. Li, and Y. Hou, “Synchrophasor recovery and prediction: A graph-based deep learning approach,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7348–7359, 2019.

[49] T. Zhao, M. Yue, and J. Wang, “Structure-informed graph learning of networked dependencies for online prediction of power system transient dynamics,” *IEEE Transactions on Power Systems*, 2022.

[50] Y. Yuan, Z. Wang, and Y. Wang, “Learning latent interactions for event identification via graph neural networks and pmu data,” *IEEE Transactions on Power Systems*, 2022.

[51] Y. Luo, C. Lu, L. Zhu, and J. Song, “Data-driven short-term voltage stability assessment based on spatial-temporal graph convolutional network,” *International Journal of Electrical Power & Energy Systems*, vol. 130, p. 106753, 2021.

[52] M. R. Alam, F. Bai, R. Yan, and T. K. Saha, “Classification and visualization of power quality disturbance-events using space vector ellipse in complex plane,” *IEEE Transactions on Power Delivery*, vol. 36, no. 3, pp. 1380–1389, 2020.

[53] A. J. Wilson, D. R. Reising, R. W. Hay, R. C. Johnson, A. A. Karrar, and T. D. Loveless, “Automated identification of electrical disturbance waveforms within an operational smart power grid,” *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4380–4389, 2020.

[54] C. Ge, R. A. de Oliveira, I. Y.-H. Gu, and M. H. Bollen, “Deep feature clustering for seeking patterns in daily harmonic variations,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–10, 2020.

[55] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, “Support vector regression machines,” in *Advances in Neural Information Processing Systems 9*, may 1997, pp. 155 – 161.

[56] Y. Liang, D. Niu, and W. C. Hong, “Short term load forecasting based on feature extraction and improved general regression neural network model,” *Energy*, vol. 166, pp. 653 – 663, 2019.

[57] S. S. Haykin,, *Kalman Filtering and Neural Networks.* New York, NY, USA: John Wiley & Sons, Inc., 2001.

[58] E. Knorr and R. T. Ng, “Finding intentional knowledge of distance-based outliers.” in *Proc. of the international conference on very large data bases*, Jun 1999, pp. 211–222.

[59] M. Breunig, a. R. T. N. H. P. Kriegel, and v. n. p. y. o. a. J. Sander, booktitle=in Proc. of the 2000 ACM SIGMOD international conference on Management of data, “LOF: identifying density-based local outliers.”

[60] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation-based anomaly detection,” *ACM Trans. on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.

[61] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, no. 2, pp. 275 – 304, Feb 2016.

[62] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37 – 63, 2011.

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

[64] 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.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[66] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994. [Online]. Available: http://www.jstor.org/stable/2684253

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

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

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

[74] 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.

[75] 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.

[76] 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

[77] A. Aligholian, A. Shahsavari, E. Stewart, E. Cortez, and H. Mohsenian-Rad, “Unsupervised event detection, clustering, and use case exposition in micro-pmu measurements,” *IEEE Trans. on Smart Grid*, 2021.

[78] H. Mohsenian-Rad, *Smart Grid Sensors: Principles and Applications.* Cambridge University Press, UK, April 2022.

[79] https://www.cyme.com/software/.

[80] https://www.dnv.com/software/products/synergi-products.html.

[81] W. L. Hamilton, “Graph representation learning,” *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[82] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[83] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[84] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.

[85] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” *arXiv preprint arXiv:1808.06670*, 2018.

[86] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax.” *ICLR (Poster)*, vol. 2, no. 3, p. 4, 2019.

[87] T. M. Cover and J. A. Thomas, “Entropy, relative entropy and mutual information,” *Elements of information theory*, vol. 2, pp. 12–13, 1991.

[88] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, “Mutual information neural estimation,” in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 531–540.

[89] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, “Adversarial autoencoders,” *arXiv preprint arXiv:1511.05644*, 2015.

[90] https://www.pscad.com/.

[91] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[92] A. J. Gates and Y.-Y. Ahn, "The impact of random models on clustering similarity," *arXiv preprint arXiv:1701.06508*, 2017.

[93] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning.* PMLR, 2016, pp. 478–487.

[94] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-091.html

[95] J. Shi, B. Foggo, and N. Yu, "Power system event identification based on deep neural network with information loading," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5622–5632, 2021.

[96] M. Khodayar and J. Wang, "Deep generative graph learning for power grid synthesis," in *2021 International Conference on Smart Energy Systems and Technologies (SEST).* IEEE, 2021, pp. 1–6.

[97] F. N. Shimim, H. Nehrir, M. Bahramipanah, and Z. Shahooei, "A graph-theory-based clustering method for improving resiliency of distribution systems," in *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe).* IEEE, 2020, pp. 1–6.

[98] A. Tahabilder, P. K. Ghosh, S. Chatterjee, and N. Rahman, "Distribution system monitoring by using micro-pmu in graph-theoretic way," in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE).* IEEE, 2017, pp. 159–163.

[99] G. Frigo, A. Derviškadić, P. A. Pegoraro, C. Muscas, and M. Paolone, "Harmonic phasor measurements in real-world pmu-based acquisitions," in *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).* IEEE, 2019, pp. 1–6.

[100] 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.

[101] S. Jain, P. Jain, and S. N. Singh, "A fast harmonic phasor measurement method for smart grid applications," *IEEE Trans. on Smart Grid*, vol. 8, no. 1, pp. 493–502, Jan 2017.

[102] L. Chen, W. Zhao, F. Wang, and S. Huang, "Harmonic phasor estimator for P class phasor measurement units," *IEEE Trans. on Instrumentation and Measurement*, vol. 58, no. 10, pp. 1–10, May 2019.

[103] A. Carta, N. Locci, and C. Muscas, "A PMU for the measurement of synchronized harmonic phasors in three-piece distribution networks," *IEEE Trans. on Instrumentation and Measurement*, vol. 58, no. 10, pp. 3723–3730, Oct 2009.

[104] B. Zeng, Z. Teng, Y. Cai, S. Guo, and B. Qing, "Harmonic phasor analysis based on improved FFT algorithm," *IEEE Trans. on Smart Grid*, vol. 2, no. 1, pp. 51–59, Mar 2011.

[105] L. Chen, M. Farajollahi, M. Ghamkhari, W. Zhao, S. Huang, and H. Mohsenian-Rad, "Switch status identification in distribution networks using harmonic synchrophasor measurements," *IEEE Trans. on Smart Grid*, vol. 12, no. 3, pp. 2413–2424, May 2021.

[106] R. Kankale, S. Paraskar, and S. Jadhao, "Classification of power quality disturbances in emerging power system with distributed generation using space phasor model and normalized cross correlation," in *2021 8th International Conference on Smart Computing and Communications (ICSCC)*. IEEE, 2021, pp. 340–345.

[107] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.