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

Armin Aligholian, *Student Member, IEEE*, Hamed Mohsenian-Rad, *Fellow, IEEE*

***Abstract*—This paper 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 and infrequent, and have unknown characteristics. 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.**

***Keywords:* D-PMU, H-PMU, locationally scarce measurements, feeder topology, graph representation learning, event clustering.**

I. Introduction

A. Motivations and Challenges

Modern power distribution systems consist of various elements, including the utility equipment and devices, such as capacitor banks, transformer tap changers, and protection devices, as well as customer devices, such as different types of loads, plugged-in electric vehicles, renewable generation units, and other distributed energy resources. The typical and benign operation (i.e., switching on and switching off) as well as mis-operation of these elements can create different types of *events* in the system. Capturing and understanding these events is crucial to achieve situational awareness about the operation of the power distribution systems and their components [1].

Accordingly, a growing sub-area in the field of smart grids has emerged recently that focuses on the analysis of various events in power distribution systems. Great attention has been devoted to using data from distribution-level phasor measurement units (D-PMU), a.k.a micro-PMUs [2]. D-PMUs provide synchronized three-phase measurements of the *fundamental* phasors of voltage and current on power distribution systems. The very high reporting rate of D-PMUs, makes it possible to capture and analyze a wide range of informative events.

Meanwhile, the field of smart grid sensors has continued to grow also in the area of measuring *harmonic* synchrophasors, which can be obtained from another new class of smart grid sensors, namely harmonic phasor measurement units (H-PMUs) [3]–[6]. In fact, D-PMUs and H-PMUs may soon converge into one sensor, where one device can provide both fundamental and harmonic synchrophasors, as needed [1], [5].

The high reporting rate of D-PMUs allow us to capture additional details about the signatures of various events that cannot be captured by the typical SCADA meters. Therefore, adding D-PMUs certainly improves data availability. However, when we use data from D-PMUs to analyze events, there are at least two major challenges that we need to address:

1) *Scarcity in Measurement Locations*: Due to the high cost of D-PMUs, including purchase, installation, and data streaming, it is typical that a power distribution feeder is equipped with only a small number of D-PMUs. Hence, although the available D-PMUs provide synchronized phasor measurements at high resolution, such high resolution data availability is *limited* to only a few locations on the power distribution feeder.
2) *Limited Prior Knowledge about Event Signatures*: Analysis of events in power distribution systems is difficult, because such events are diverse and many of them are infrequent. Furthermore, the cause of many events, and hence the characteristics of their signatures in the measurements, are not known in advance [7].

The key to address the second issue is to conduct *unsupervised* learning to identify the *type* and *cause* of the events. This can be done through *event clustering*. The goal in event clustering is to identify the type of the events with minimal prior knowledge. It should be clarified that, the term "unsupervised" learning in this paper is used as the opposite of "supervised" learning in the machine learning context. In the latter, we need the expert knowledge to obtain prior labeling of the events. Of course, if prior labeling of the events *is* available, then we can apply supervised or semi-supervised learning methods, such as in our work in [8]. However, it is *not* common in practice that we have access to prior labeling of the events. Therefore, we need to achieve event clustering without the availability of the prior labeling of the events.

The key to address the first issue is to make use of any available *contextual information*, to enhance our ability to do event clustering when we face locational scarcity in data availability. In this paper, the primary contextual information is the graph of the topology of the power distribution network.

B. Summary of Technical Contributions

In this paper, we seek to significantly improve the performance in event clustering under locationally-scarce data availability by taking the following two new directions:

A. Aligholian and H. Mohsenian-Rad are with the University of California, Riverside, CA, USA. The corresponding author is H. Mohsenian-Rad.

1) Using the information about the *relative location* of the phasor measurement units on the network topology, notwithstanding the fact that such locations are scarce.
2) Using not only the *fundamental* phasor measurements, that are commonly used in the literature in this field, but also the *harmonic* phasor measurements in the process of analyzing the signatures of various events.

Both of the above directions are ways of incorporating new contextual information to the task of event clustering. The first one takes into account the location of sensors. The second one takes into account the changes in the harmonics that are caused by each event, as seen at the *same* existing sensor locations.

In addition to taking the above new approaches, the methodologies that are developed in this paper to implement these new approaches carry new technical contributions, as listed below:

- 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.
- 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.
- 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, which 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.

On one hand, our approach is applicable to the most common scenario, in which *only* the fundamental phasor measurements are available, as in the conventional case of D-PMUs. On the other hand, our approach is applicable also to the case where *not only* the fundamental phasor measurements *but also* the harmonic phasor measurements are available, as in the emerging case of H-PMUs.

We refer to the proposed new design as GraphPMU. The results from various case studies show that GraphPMU leads to significant improvements in the accuracy of event clustering, despite the locational scarcity in the available phasor measurements. Importantly, even if we do *not* use the harmonic phasor measurements, the mere use of topological information in the proposed graph learning paradigm can highly improve the performance in event clustering, specifically for the case of small events. The case studies in this paper include comparison with the state of the art clustering methods as well as with a variety of the GraphPMU implementation options, with different structures to show the importance of system topology, local-global feature learning, and number of sensors.

C. Literature Review

The literature on data-driven and machine learning event types analysis in distribution-level phasor measurements can be generally divided into two categories. First, there are studies that use supervised or semi-supervised learning, i.e. event classification, such as those in [8]–[16]. They require prior labeling of the events in the training data set, which may *not* be available in all of the cases in practice, whether for *all* or a *subset* of the events in the training data set. This line of work also includes the cases where the labeled data has a low rate or a low quality. In [9] and [14], the authors proposed implementing and analyzing imperfect labeling for event classification. In [13], the authors proposed a hybrid semi-supervised learning model for event identification, where there is a limited number of labeled events available. In [15] a transfer learning techniques has been used to increase the reliability of the event classifier. It should be noted that, among the above studies, only the ones in [8] and [12] are related to *distribution-level* phasor measurements.

The second group are the studies that use unsupervised learning. They attempt to cluster the events by grouping their distinctive characteristics, *without* the need for any prior labeling. In [17], k-means clustering and Ward's clustering are proposed to cluster voltage sag events. In [18], an unsupervised clustering method is proposed for some specific faults; such as single-line-to-ground versus line-to-line faults. In [19] the authors proposed a three stage unsupervised learning method for event detection, identification, and localization. In [7], an unsupervised event detection and clustering method is proposed, which requires solving a mixed integer program. Importantly, the study in [7] is limited to the analysis of phasor measurements from only one D-PMU. It is inherently unrelated to the idea of taking into account the topological information of D-PMUs in the clustering task.

None of the unsupervised learning methods in [7], [17]–[19] take 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.

To consider the network topology in the process of event clustering, one plausible way is to do graph-based analysis. Graph theory and more generally graph-based analysis have been used in power systems, such as for event detection [20], event location identification [21], data recovery and prediction [22], and to study power system dynamics [23].

Recent literature also includes the use of GNNs, to address some prevalent power system issues, including the analysis of

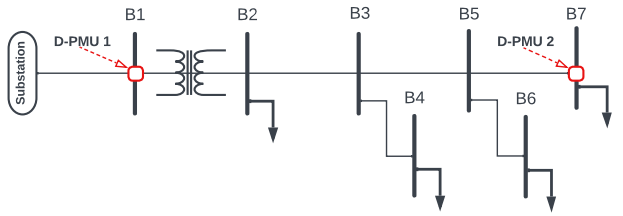


Fig. 1. An example power distribution network with $N = 7$ buses. Two PMUs are installed are installed at buses in B1 and B5. We have $\mathcal{M} = \{B1, B7\}$.

events. In [24], 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 of voltage and frequency. In [25], authors have proposed a Graph Convolutional Network (GCN) based model for event classification and region identification in a transmission network. In [26], the authors used Graph Convolutional Network (GCN) for short term voltage stability assessment. Importantly, neither of the studies in [24]–[26] 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 [27]–[29]. For example, in [29], 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-scarce phasor measurements. Furthermore, the prior studies in this field, including those in [27]–[29], do *not* consider using the information about the network topology.

II. Topology-Based Representation Learning

Consider a power distribution system, such as the one in Fig. 1. Let $\mathcal{B}$ denote the set of all buses, such as $\{B_1, \ldots, B_7\}$ in Fig. 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. 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 V.

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 are captured during event i by the D-PMU at bus j, where $j \in \mathcal{M}$. Similar to [7], 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 [1, 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 [30] and Synergy [31].

A. 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* [32].

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:

$$\{G_1, G_2, \ldots, G_M\}. \tag{1}$$

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:

$$H_i^{k+1} = \sigma(H_i^k W^k), \tag{2}$$

where σ is the activation function, e.g., $\text{ReLU}(x) = \max(0, x)$, 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 (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 II-B, we will build an unsupervised GNN-based graph encoder to transform each G_i to a single vector. In Section II-C 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 II-D we will develop a discriminator module to calculate the aforementioned mutual information.

B. 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 [33] with the following updating formulation in its hidden layers[1]:

$$H_i^{k+1} = \sigma(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}H_i^k\omega^k). \tag{3}$$

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 (3) and the one in (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. [33].

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:

$$\mathbf{h}_i^\omega(j) = [\mathbf{h}_i^1(j), \mathbf{h}_i^2(j), \ldots, \mathbf{h}_i^K(j)]. \tag{4}$$

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

$$\mathbf{h}_i^{\omega,g} = S(\{\mathbf{h}_i^\omega(1), \ldots, \mathbf{h}_i^\omega(N)\}) \tag{5}$$

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 [32]. The outputs of graph encoder are obtained as:

$$\{\mathbf{h}_i^\omega(j), \mathbf{h}_i^{\omega,g}\} = \mathcal{E}(G_i), \quad \forall j = 1, \ldots, N, \tag{6}$$

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 (6).

Before we end this section, we shall point out that the node-level representation inherently includes the location-dependent features at each node during each event, which is influenced by the type of the event and the location of the event. These factors create different transient signatures in the time-series of the phasor measurements at the locations of the sensors. By using these node-level representations *in combination* with the graph-level representations, we take into account these location-dependent features at each node, as well as their connections to the other nodes through the topology structure (which we use in the graph encoder). Therefore, we force the model to learn and embed the informative substructure data representation into the graph-level representation, in order to have more separable features for the purpose of event clustering.

C. Mutual Information

Similar to the unsupervised learning methods in [36] and [37], the objective for the proposed graph encoder is to maximize the average mutual information (MI) [38] 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:

$$\text{Maximize: } \mathcal{I} = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{i=1}^{N}\text{MI}(\mathbf{h}_i^\omega(j); \mathbf{h}_i^{\omega,g}). \tag{7}$$

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 [37]. 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 [33].

Calculating $\mathcal{I}$, in a continuous and high-dimensional settings is difficult. A solution is suggested in [39], 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.

D. 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 [37]. First, we construct the *negative graphs*. They have the same input node data as in the positive graphs, i.e., X_i^j for event i and node j. But they have a different graph

[1]Other similar modules, such as those in [34] and [35], can also be used.

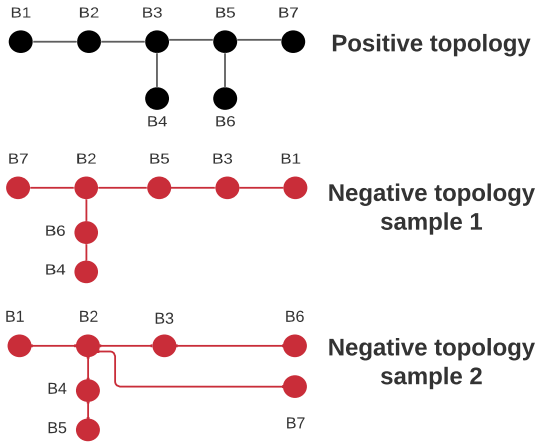


Fig. 2. The actual (i.e., *positive*) topology of the power distribution network in Fig. 1 is shown in black. Two arbitrary alternative (i.e., *negative*) topologies with the *same* number of nodes/buses and edges/lines are shown in red.

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. 2. The positive topology represents the actual topology of the power distribution system that we saw in Fig. 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 [39]. 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 [36], and the method in [39], we *simultaneously estimate and maximize* the objective function $\mathcal{I}$ in (7) as shown below:

$$\hat{\mathcal{I}}_{\omega,\psi} = \frac{1}{2MN}\sum_{i=1}^{M}\Big(\sum_{j=1}^{N}\mathbb{E}_{\mathbb{P}}[-\sigma(-\Psi(\mathbf{h}_i^{\omega}(j),\mathbf{h}_i^{\omega,g})] - \sum_{j=1}^{N}\mathbb{E}_{\mathbb{P}\times\mathbb{P}'}[\sigma(\Psi(\mathbf{h}_{i'}^{\omega}(j),\mathbf{h}_i^{\omega,g})]\Big), \tag{8}$$

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 $\mathbb{P}$, is identical to the probability distribution of the negative events, which is denoted by $\mathbb{P}^{'}$. Also, $\mathbb{E}_{\mathbb{P}}$ and $\mathbb{E}_{\mathbb{P}\times\mathbb{P}'}$ 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 (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 1 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 (8).
15: Update the ω and ψ by conducting back propagation and using Adam optimizer [40].
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.

E. 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.

F. Algorithm: Topology-Based Representation Learning

Algorithm 1 shows the summary of the steps that we took in Sections II-B to II-E. 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.

III. Temporal Representation Learning

The design that we presented in Section II 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 II *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.

A. 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) [41] model is proposed which includes Long Short Term Modules (LSTM) [42] 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.

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:

$$Em_i^j = E(X_i^j), \forall i = 1, ..., M, \forall j = 1, ..., N, \tag{9}$$

$$\theta_E, \theta_D = \underset{\theta_E, \theta_D}{\operatorname{argmin}} \frac{1}{M} \sum_{j=1}^{M} \frac{1}{N} \sum_{i=1}^{N} \text{MSE}(X_i^j, D(Em_i^j)), \tag{10}$$

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. This means that during the training phase, for any event in any bus, the AED parameters are being updated. 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 (9).

B. Algorithm: Temporal Representation Learning

Algorithm 2 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 2 Temporal Representation Learning

1: **Input:** Normalized event time series X_i^j.
2: **Output:** Embedding vectors Em_i^j as in (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 (10).
7: Update θ_E and θ_D through back propagation [40].
8: **End**
9: **// Embedding Extraction**
10: **For** each event i and node j **Do**
11: $Em_i^j = E(X_i^j)$.
12: **End**

IV. 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 II with the temporal representation learning design in Section III. Fig. 3 shows how these two design components are integrated in order to achieve GraphPMU.

The architecture in Fig. 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 II-D. 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 II-E.

V. 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 I, 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.

A. 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

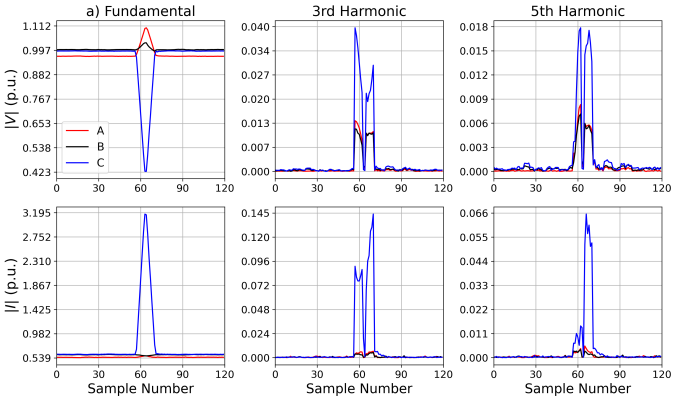


Fig. 4. Comparing the signatures during the *same* event at the fundamental phasor measurements vs. at the 3rd and 5th harmonic phasor measurements.

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 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(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(b) and at the 5th harmonic in Fig. 4(c) are considerably more distinctive.

B. 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 2.

We concatenate all the embedding vectors to form:

$$Em_i^j = [Em_i^{j,1}\ Em_i^{j,3}\ Em_i^{j,5}]^T. \tag{11}$$

C. Extended Topology-Based Learning

Next, we feed the new vectors Em_i^j that are derived in (11) as the input to the GNN using Algorithm 1 to complete the process for the event clustering task. Since the size and nature of the input vector is different from those in Section II. 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 III). This is because the default steady state values in unobserved locations are assumed not to have any harmonics.

VI. 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. 5. The network simulation model is developed in PSCAD to assure capturing the transient signatures of the events [43]. 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

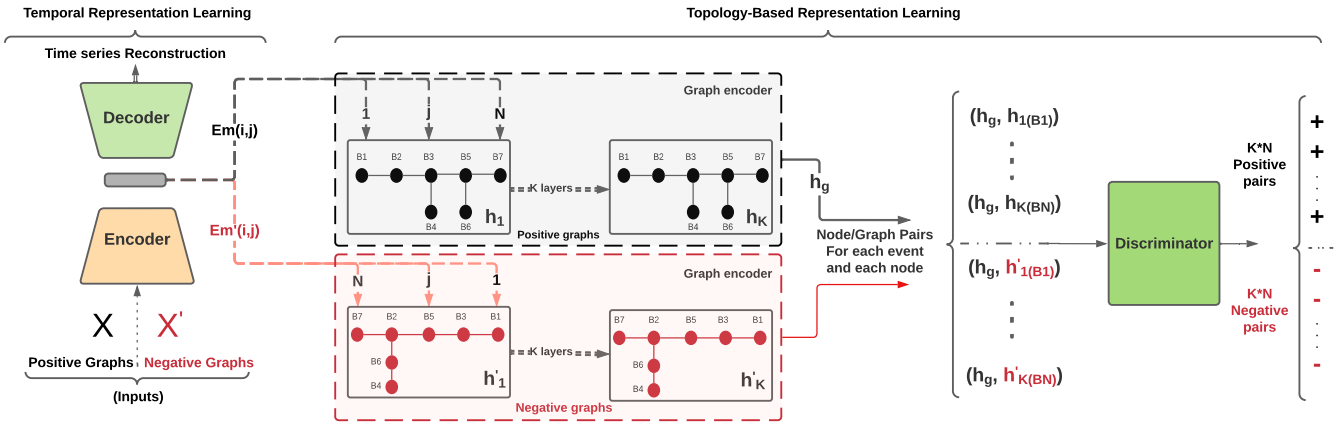


Fig. 3. GraphPMU: AED learns the optimal representation vectors for nodal data for all events. Then these vectors alongside the positive and negative graphs are used as input for the proposed graph encoder. The output of the graph encoder, node-level and graph-level representations are paired to shape the positive and negative samples. Then, discriminator learns to discriminate between these samples for MI maximization.

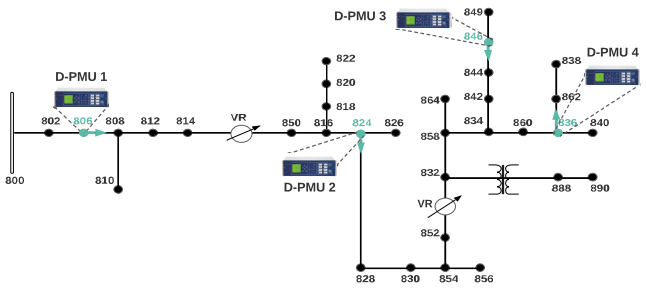


Fig. 5. The IEEE 34-bus test system with locationally-scarce phasor measurements. There are only four D-PMUs (H-PMUs) available on this network, as marked on the figure. However, the events can happen at *any* location.

distribution network. The location of the D-PMUs (H-PMUs) are shown on Fig. 5. Note that, we have:

$$\mathcal{M} = \{806, 824, 836, 846\}. \tag{12}$$

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 [7]; 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.

The number of epochs for each model training is considered to be 100, and the batch size is 1000. During the training phase, the average learning time for each epoch is about 28.8 seconds, which is around 48 minutes for all the 100 epochs. Importantly, we do *not* have any learning process during the testing phase when we label the test events. All that is needed during the testing phase is to obtain and pass the features to the trained model to have it label/cluster each event. The average calculation time (considering all the different scenarios of the analysis with different number of D-PMUs, different locations for sensitivity analysis, and different hyper-parameters choices) is 7.8 seconds for 5000 test events, which is 1.56 milliseconds for each test event. It should be noted that, the AED model is trained separately. The training time of the AED model is about 13.6 minutes for 100 epoch. The average testing time of the AED is 0.73 milliseconds for each event, for all the time-series features from all the buses. Therefore, in total, the entire testing phase takes $1.56 + 0.73 = 2.29$ milliseconds for each event.

A. Parameters of GraphPMU

The graph encoder has two layers of GCN [33], 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 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.

TABLE I
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**

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 [7]. 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 [44], 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 [45] is used to assess accuracy in a clustering task. ARI is a number between 0 and 1. A higher ARI means a better clustering.

B. Comparison with Temporal-Based Benchmarks

Table I 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 I, 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 [41], DEC [46], Kernel k-means, k-Shape clustering and k-means clustering methods [47]. Here, Time Series (TS) means that the method uses the raw time series data, without any encoding.

From Table I, 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 values of nominal loading information at the buses without D-PMUs for all the ten methods in Table I.

C. Comparison with Topology-Based Benchmarks

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

1) Advantage of Using Data Compression: If we compare TS+N/G+NL versus GraphPMU in Table I, 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 I that do *not* use any graph model.

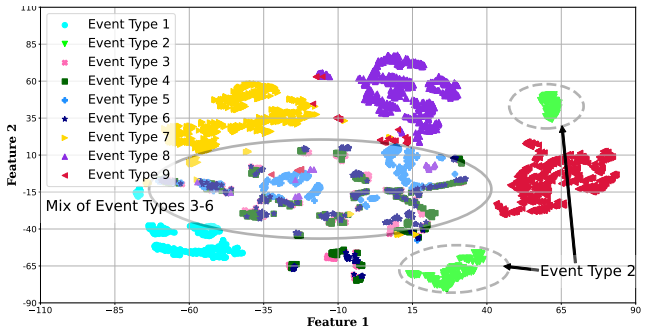
2) Advantage of Pairing Node-Level and Graph-Level Vectors: If we compare AED+G+NL versus GraphPMU in Table I, we can see that their difference is only in terms of using G versus N/G. The method with AED+G+LN considers only the last layer of the graph learning model for the positive and the negative 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.

3) Advantage of Using Nominal Load Data: If we compare AED+N/G versus GraphPMU in Table I, 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.

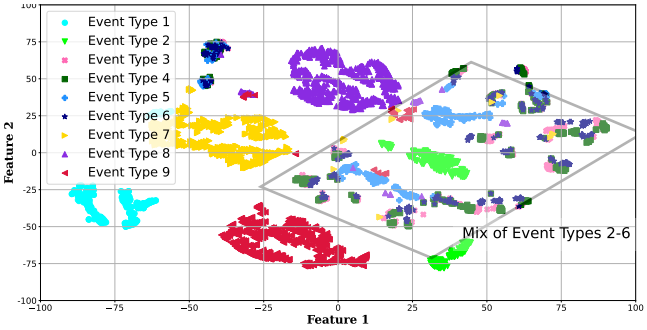
It is worth clarifying that for all the methods under "without Graph Model" in the top segment of Table I, they too make use of the nominal loading information for the buses without D-PMUs. However, for the methods under "with Graph Model" in the bottom segment of Table I, we separately examined the cases where the nominal loading information is used versus the cases where such information is not used, such that we can explicitly evaluate the importance of using such information in the design of GraphPMU.

D. Analysis Based on Different Types of Events

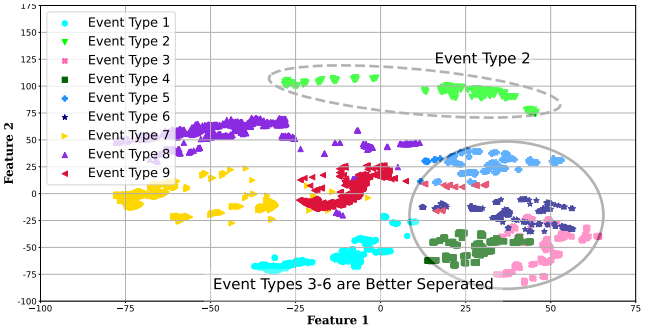
Fig. 6 shows the t-distributed stochastic neighbor embedding (t-SNE) scatter plot of all *test* events for three methods: a) DEC; b) AED+G+NL; c) GraphPMU. Each point indicates one event. The shapes and colors indicate the *true labels* of the nine different event types. One of the major weaknesses of the methods that do *not* use graph models is their inability



(a) DEC, ARI = 0.524



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



(c) GraphPMU, ARI = 0.720

Fig. 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.

to properly cluster the "smaller" events, such as events types 3, 4, 5, and 6. For example, see the area in Fig. 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. 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. 6(b).

GraphPMU addresses all of these shortcomings, as we can see in Fig. 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. 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.

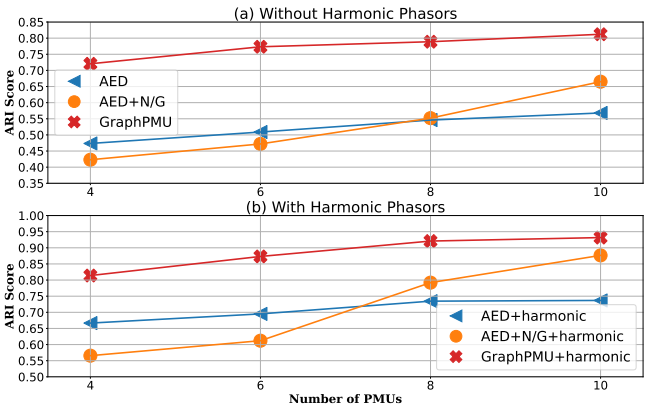


Fig. 7. ARI scores for AED, AED+N/G and GraphPMU methods vs. the number of D-PMUs, *with* and *without* using harmonic synchrophasors.

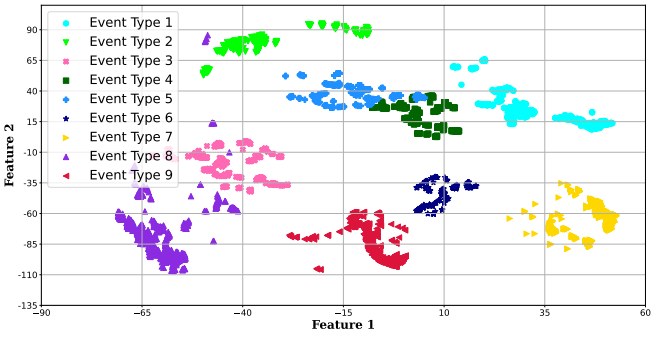


Fig. 8. The t-SNE for GraphPMU for all test events when using 10 D-PMUs.

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. 6(c), the points for event types 3, 4, 5, and 6 are separated from each other much better compared to the other figures.

E. Impact of Adding Harmonic Phasor Measurements

Table II 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 II with Table I, 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 V.

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.

It is worth reiterating the fact that, when we use H-PMUs, we take advantage of *both* the fundamental phasor measurements and the harmonic phasor measurements; because H-PMUs can provide both; e.g., see [3]–[6].

F. Impact of the Number of D-PMUs

Fig. 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* 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. 7(b), GraphPMU achieves a very high ARI score of 0.92 with only 8 H-PMUs in a network with 34 buses.

Fig. 8 shows the performance of GraphPMU in clustering different types of events, when there are 10 sensors available. If we compare Fig. 8 with Fig. 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.

Before we end this section, it is worth noting that, the proposed GraphPMU is applicable even if *fewer* D-PMUs are available. Note that, in principle, one would need at least *two* sensors to be available to take advantage of time-synchronization among the phasor measurements. Furthermore, one cannot take advantage of the relative locations of the sensors on the network's graph if less than two sensors are available. Therefore, the minimum number of the sensors in the context of this paper is two. Accordingly, we applied GraphPMU to the case where there are only two D-PMUs available. In this regard, we removed the D-PMUs at buses 824 and 846; to have $\mathcal{M} = \{806, 836\}$, instead of the default four D-PMUs in set $\mathcal{M}$ in (12). The ARI is obtained as 0.498 for GraphPMU, 0.355 for AED+N/G, and 0.442 for AED. As we can see, GraphPMU outperforms the other methods even if only two D-PMUs are available. Of course, the performance is not as good as the case where four D-PMUs are available, where GraphPMU would achieve an ARI of 0.720 as we saw in Table I. Similar results are obtained if we replace the two D-PMUs with two H-PMUs. In this case, the ARI is obtained as 0.654 for GraphPMU, 0.439 for AED+N/G, and 0.591 for AED. Again, GraphPMU outperforms the other methods even if only two H-PMUs are available. Of course, the performance is not as good as the case where four H-PMUs are available, where GraphPMU would achieve an ARI of 0.814.

TABLE II
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**

TABLE III
ARI SCORE FOR THE GRAPHPMU WHEN WE CHANGE THE LOCATION OF D-PMU 2.

Location of D-PMU 2	ARI
Bus 824	0.720
Bus 812	0.650
Bus 850	0.721
Bus 854	0.704
Bus 858	0.689

G. Impact of the Location of D-PMUs

In this section, we conduct a sensitivity analysis with respect to the location of the D-PMUs. We assume that four D-PMUs are available. Recall that the default locations of the D-PMUs are at the four buses that are listed in (12). We assume that the locations of three D-PMUs are fixed at buses 806 (D-PMU 1), 846 (D-PMU 3), and 836 (D-PMU 4). However, we change the location of D-PMU 2 to place it at five different locations. In each case, we report the ARI of the GraphPMU methodology. The results are shown in Table III. As we can see in this table, the performance of GraphPMU can change depending on the exact location of the D-PMUs. However, the changes are not very significant. In fact, compared to the default locations, changing the locations can decrease the ARI by at most by 9.7% when D-PMU 2 is placed at bus 812, and it can increase the ARI by at most 0.1% when D-PMU 2 is placed at bus 850.

H. Impact of Incorporating Traditional Measurements

In all the previous case studies, we assumed that we do not have access to any measurements other than the high-resolution and time-synchronized measurements that are obtained from the very few D-PMUs (or H-PMUs) on the network. However, in practice, one may have access to additional measurements from the traditional measurement devices; such as the measured power consumption at each bus. While such traditional measurements cannot directly provide much information about the events due to their very low granularity and lack of time-synchronization, they may still help improve the performance of GraphPMU to do a better event clustering based on the locationally scarce phasor measurements that are available.

The traditional measurements can be incorporated in GraphPMU as part of the NL step. In the original NL step, we use the nominal loading at each bus (from the utility's network model) as a very rough approximation of the true loading at that bus so as to approximate the power flow across the power distribution network at the time when the event occurs. However, if we have access to the power consumption measurements at each bus that are obtained by the traditional measurement devices, then we can use them to enhance our power flow approximations in the NL step. This can in turn potentially improve the overall performance of GraphPMU.

An example is shown in Table IV. The case study in this example is based on the default scenario, where there are four D-PMUs on the network as listed in (12). Note that, the two cases in Table IV are based on the exact same phasor measurements from the same D-PMUs. As we can see, the incorporation of the measurements from the traditional measurement devices can improve the performance of GraphPMU, increasing ARI by 5.97% from 0.720 to 0.763.

TABLE IV
ARI SCORE FOR THE GRAPHPMU WHEN WE INCORPORATE TRADITIONAL MEASUREMENTS.

Method	ARI
GraphPMU (Nominal Loading)	0.720
GraphPMU (Measured Loading)†	0.763

† By using traditional power meters.

VII. CONCLUSIONS AND FUTURE WORK

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.

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.

APPENDIX

In this appendix, we explain the definition of ARI and the way that it is calculated. The Rand Index is a similarity measure between two clustering results by considering all pairs of samples and counting pairs that are placed in the same or different clusters in the conducted and true clustering. Let M denotes the total number of events. Suppose $A = \{A_1, \ldots, A_P\}$ is the true clustering of the events, while $B = \{B_1, \ldots, B_Q\}$ is a clustering that is constructed by a clustering method that we seek to evaluate. Parameters P and Q denote the number of clusters in A and B, respectively. Let w be the number of pairs of events that are placed in the same cluster in A and in the same cluster in B, x be the number of pairs of events that are placed in the same cluster in A but *not* in the same cluster in B, y be the number of pairs of events that are placed in the same cluster in B but *not* in the same cluster in A, and z be the number of pairs of events in different clusters in both partitions. Quantities w and z can

be interpreted as agreements, x and y as disagreements. The Rand Index (RI) is defined as [48]:

$$\text{RI} = \frac{w+z}{w+x+y+z},$$

which is a number between 0 and 1. When the A and B perfectly match, then RI is 1. However, one issue with the RI is that its expected value when A and B are random partitions does not take a constant value , for instance zero. This issue is addressed in the Adjusted Rand Index (ARI), which is proposed in [45]:

$$\text{ARI} = \frac{\sum_{i,j}\binom{M_{ij}}{2} - \left[\sum_i \binom{M_i}{2}\sum_j \binom{M_j}{2}\right]/\binom{M}{2}}{\frac{1}{2}\left[\sum_i \binom{M_i}{2} + \sum_j \binom{M_j}{2}\right] - \left[\sum_i \binom{M_i}{2}\sum_j \binom{M_j}{2}\right]/\binom{M}{2}}$$

where parameter M_{ij} is the number of events that are in cluster $A_j \in A$ and also in cluster $B_i \in B$, parameter M_i is the number of events in cluster $B_i \in B$, and parameter M_j is the number of events in cluster $A_j \in A$. The notations with parentheses denote the mathematical combination. In general, an ARI value lies between 0 and 1. A smaller ARI means a lower agreement between the true clustering A and the obtained clustering B, while a larger ARI means a higher agreement between the true clustering A and the obtained clustering B.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[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] M. Pavlovski, M. Alqudah, T. Dokic, A. A. Hai, M. Kezunovic, and Z. c. Obradovi, "Hierarchical convolutional neural networks for event classification on pmu measurements ," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2021.

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

[11] Y. Tang and J. Yang, "Dynamic event monitoring using unsupervised feature learning towards smart grid big data," in *Proc. of the IEEE IJCNN*, Anchorage, AK, May 2017.

[12] I. Niazazari and H. Livani, "A PMU-data-driven disruptive event classification in distribution systems ," *Electric Power Systems Research*, vol. 157, pp. 251–260, 2018.

[13] H. Li, Y. Weng, E. Farantatos, and M. Patel, "A hybrid machine learning framework for enhancing pmu-based event identification with limited labels ," in *Proc. of International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, College Station, TX, Apr. 2019.

[14] Y. Liu, L. Yang, A. Ghasemkhani, H. Livani, V. A. Centeno, P.-Y. Chen, and J. Zhang, "Robust event classification using imperfect real-world pmu data ," *IEEE Internet of Things Journal* , 2022.

[15] J. Shi, K. Yamashita, and N. Yu, "Power System Event Identification with Transfer Learning Using Large-scale Real-world Synchrophasor Data in the United States," in *Proc. of IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, April 2022.

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

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

[18] E. Klinginsmith, R. Barella, X. Zhao, and S. Wallace, "Unsupervised clustering on PMU data for event characterization on smart grid," in *Proc. of the IEEE SMARTGREENS*, , Rome, Italy, April 2016.

[19] H. Li, Y. Weng, E. Farantatos, and M. Patel, "An unsupervised learning framework for event detection, type identification and localization using pmus without any historical labels ," in *Proc. of the IEEE Power & Energy Society General Meeting (PESGM)*, Atlanta, GA, July 2019.

[20] 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 Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2173–2186, 2019.

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

[22] 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.

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

[24] Y. Yuan, Z. Wang, and Y. Wang, "Learning latent interactions for event identification via graph neural networks and PMU data," *IEEE Trans. on Power Systems*, (Early Access), March 2022.

[25] M. Mansour-Lakouraj, M. Gautam, H. Livani, and M. Benidris, "A multi-rate sampling PMU-based event classification in active distribution grids with spectral graph neural network," *Electric Power Systems Research*, vol. 211, 2022.

[26] Y. Luo, C. Lu, L. Zhu, and J. Song, "Data-driven short-term voltage stability assessment based on spatial-temporal graph convolutional network," *Int. J. of Electrical Power & Energy Systems*, vol. 130, 2021.

[27] 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 Trans. on Power Delivery*, vol. 36, no. 3, pp. 1380–1389, 2020.

[28] 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 Trans. on Smart Grid*, vol. 11, no. 5, pp. 4380–4389, 2020.

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

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

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

[32] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on AI and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

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

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

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

[36] 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.

[37] 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.

[38] T. M. Cover and J. A. Thomas, "Entropy, relative entropy and mutual information," *Elements of information theory*, vol. 2, pp. 12–13, 1991.
[39] M. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *Proc. of the International Conference on Machine Learning*, July 2018.
[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[41] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.
[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.
[43] https://www.pscad.com/.
[44] 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.
[45] A. J. Gates and Y.-Y. Ahn, "The impact of random models on clustering similarity," *arXiv preprint arXiv:1701.06508*, 2017.
[46] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. of Int.l Conf. on Machine Learning*, 2016.
[47] 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.
[48] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

Armin Aligholian received the B.Sc. degree in Electrical Engineering from the Amirkabir University of Tehran, Tehran, Iran, in 2014, the M.Sc. degree in Energy Systems Engineering from the University of Tehran, Tehran, in 2017, and the Ph.D. degree in Electrical and Computer Engineering from the University of California at Riverside, CA, USA, in 2022. He has recently joined Recurve Inc. as Data Scientist. His research interests include application of deep learning, data-driven and optimization techniques in power system monitoring and control. He works on time-series data analysis and anomaly detection in power systems.

Hamed Mohsenian-Rad (M'09-SM'14-F'20) received the Ph.D. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2008. He is currently a Professor of electrical engineering and a Bourns Family Faculty Fellow at the University of California, Riverside, CA, USA. His research is on monitoring, data analysis, and optimization of power systems and smart grids. He is the author of the textbook *Smart Grid Sensors: Principles and Applications* by Cambridge University Press - 2022. He was the recipient of the National Science Foundation (NSF) CAREER Award, the Best Paper Award from the IEEE Power & Energy Society General Meeting, the Best Paper Award from the IEEE Conference on Smart Grid Communications, and a Technical Achievement Award from the IEEE Communications Society. He has been the PI on ten million dollars research grants in the area of smart grid. He has served as Editor for the IEEE TRANSACTIONS ON POWER SYSTEMS, IEEE TRANSACTIONS ON SMART GRID and the IEEE POWER ENGINEERING LETTERS.