Structural Tensor Learning for Event Identification with Limited Labels

Haoran Li, *Student Member, IEEE,* Zhihao Ma, *Student Member, IEEE,* Yang Weng, *Senior Member, IEEE,* Erik Blasch, *Fellow, IEEE,* Surya Santoso, *Fellow, IEEE*

***Abstract*—The increasing uncertainty of distributed energy resources promotes the risks of transient events for power systems. To capture event dynamics, Phasor Measurement Unit (PMU) data is widely utilized due to its high resolutions. Notably, Machine Learning (ML) methods can process PMU data with feature learning techniques to identify events. However, existing ML-based methods face the following challenges due to salient characteristics from both the measurement and the label sides: (1) PMU streams have a large size with redundancy and correlations across temporal, spatial, and measurement type dimensions. Nevertheless, existing work cannot effectively uncover the structural correlations to remove redundancy and learn useful features. (2) The number of event labels is limited, but most models focus on learning with labeled data, suffering risks of non-robustness to different system conditions. To overcome the above issues, we propose an approach called Kernelized Tensor Decomposition and Classification with Semi-supervision (KTDC-Se). Firstly, we show that the key is to tensorize data storage, information filtering via decomposition, and discriminative feature learning via classification. This leads to an efficient exploration of structural correlations via high-dimensional tensors. Secondly, the proposed KTDC-Se can incorporate rich unlabeled data to seek decomposed tensors invariant to varying operational conditions. Thirdly, we make KTDC-Se a joint model of decomposition and classification so that there are no biased selections of the two steps. Finally, to boost the model accuracy, we add kernels for non-linear feature learning. We demonstrate the KTDC-Se superiority over the state-of-the-art methods for event identification using PMU data.**

***Index Terms*—Event identification, large PMU streams, limited labels, tensor learning, semi-supervised learning, kernel method.**

I. Introduction

Modern power systems significantly incorporate highly uncertain generations and loads to facilitate clean and low-cost productions and consumptions. To better accommodate the growing uncertainty and maintain the system stability, the power system requires advanced tools for system event identification. To capture the event dynamics, Phasor Measurement Units (PMUs) provide synchronized phasor measurements with high-granularity (e.g., 30 or 60 samples per second) [1], [2]. Therefore, the PMU-based power system event identification is one of the central topics to improve the system reliability. With synchrophasors to record system dynamics, many efforts analyze measurement patterns and identify when, where, and what type of events are. To find the event initialization time, methods like change point detection [3] can detect abnormal intervals that imply events. However, to know more information about event types and locations, how to analyze PMU streams in the best way becomes challenging.

One idea to find event types and locations is to use expert information. For example, one can use signal transformation or filtering to map the time series data into some physically meaningful domain for comparing with some predefined thresholds. These methods use wavelet transformation [4], Kalman filtering [5], and Swing Door Trending (SDT) [6], etc. For example, [6] utilizes a swing door to compress data with a pre-defined door width, and the detectable events must have a certain level of slope rate. However, as these methods need to pre-define some measures or thresholds, the usage may be biased because of the specific design and test cases. Therefore, can we have a general model?

For obtaining a general form, previous work proposes to use existing events and their labels to train in a Supervised Learning (SL) manner. Such Machine Learning (ML) models typically extract features for minimizing the loss function. For instance, Decision Tree (DT) [7] treats each measurement as a factor to determine the final decision. Although transparent, such a method is inefficient to make use of complex measurement correlations. Therefore, [8] proposes Support Vector Machine (SVM) to assign each input measurement a weight to form the final feature. There are also more complex and powerful models such as Convolutional Neural Network (CNN) [9] and Graph Neural Network (GNN) [1]. They consider the spatial correlations with square and graph convolutions, respectively. One can also couple the temporal information in Long Short-Term Memory (LSTM) units. For example, [10] uses LSTM to extract periodic patterns and data inertia in time. However, for PMU data, it's desirable to simultaneously consider correlations among spatial, temporal, and measurement type dimensions. So, one can keep on increasing the model complexity. But, PMU streams accumulate quickly into terabyte (TB) level for training due to high volumes, large dimensionality, and complex correlations among the space, time, and measurement type (e.g., voltage magnitude, angle, frequency, etc.) dimensions.

To make the computation feasible, e.g., for real-time analysis, past work pre-processes PMU data with various dimension reduction techniques before the learning phase [11], [12], e,g., Principal Component Analysis (PCA) [13] and Independent Component Analysis (ICA) [14]. The pre-processing can also

Haoran Li, Zhiaho Ma, and Yang Weng are with the Department of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, 85281, USA. E-mail: {lhaoran,zhihaoma,yang.weng}@asu.edu. Erik Blasch is with Air Force Research Laboratory, Arlington, VA, 22203, USA. E-mail: erik.blasch@gmail.com. Surya Santoso is with The University of Texas at Austin, Austin, TX, 78712, USA. E-mail: ssantoso@mail.utexas.edu.

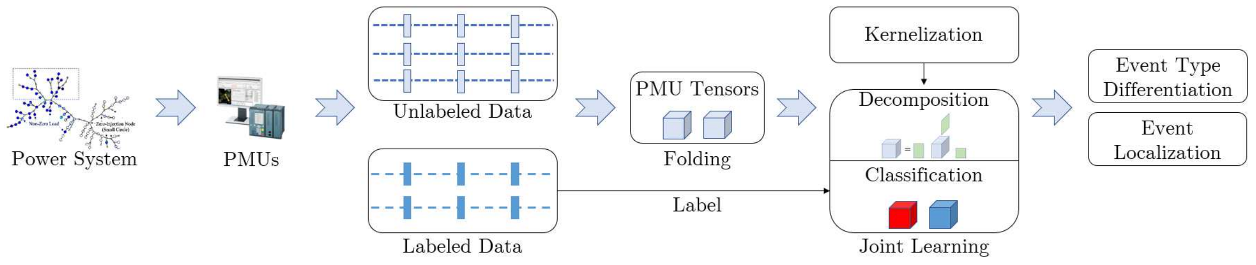


Fig. 1: KTDC-Se flowchart to tackle data (1) correlation, (2) volume, and (3) unlabeled processing.

select core statistics. For instances, [15] selects the mean values of segments via Symbolic Aggregation approXimation (SAX). [16] converts PMU signals to wavelet basis coefficients via Discrete Wavelet Transformation (DWT). These two methods filter information along with the time domain. On the other hand, [9] finds the compact vectors by considering both the time and the space domains using Markov Transition Field (MTF). However, they not only ignore the dimension of measurement type but also have their biases on how to compress the data.

In this paper, we propose to merge the two steps of dimension reduction and supervised learning into one step to avoid bias and improve efficiency. The key idea is to define a structure to hold key information in different dimensions, e.g., the measurement types ignored by [9], [15], [16]. Based on such an idea, we design a tensor learning framework to extract physically meaningful features with simple computations. Fig. 1 shows the design that easily includes the temporal, spatial, and measurement type dimensions. The classification process in the tensor learning can directly provide the classification results while the dimension reduction process for efficiency is done inexplicitly. This is because the tensor unfolding can covert the decomposed core tensor to vectors for classification, enabling an end-to-end model in Fig. 1. While such a process can extract key information quickly and systematically, we also want to preserve the nice property of nonlinear feature extraction, like the property in CNN and LSTM. For such a purpose, we mathematically derive kernalization of the classifier to process non-linear physical relationships in power systems.

While the proposed model is highly efficient, many realistic cases do not have enough labels for training. For example, [17] notes that out of $1,013$ PMU events recorded by a utility, only 84 events are labeled. But, limited labeled data will decrease the learning accuracy. One idea is to employ Semi-supervised learning for taking advantages of widely available unlabeled data in power systems, shown in the middle part of Fig. 1. But, there are challenges of integrating Semi-supervised learning and kernel-based tensor learning. For example, we need to restrict the same decomposition model for labeled and unlabeled data. We achieve the integration by aggregating all 3-D PMU tensors into a 4-D tensor. Then, a direct tensor decomposition of the 4-D tensor can maintain the same decomposition parameters for all 3-D tensors, no matter if the data is with a label or not. To train the proposed Kernelized Tensor Decomposition and Classification with Semi-supervision (KTDC-Se) from the above, we develop an efficient coordinate descent method. Finally, when the tensor number is significantly large, we modify our training method based on a mini-batch-based training scheme to save the computational storage.

For the numerical verification, the KTDC-Se method is tested extensively at various conditions on the Illinois 200-bus system, South Carolina 500-bus system [18], [19], and realistic data sets from our utility partners. These conditions include different loading conditions and PMU penetrations, etc. The benchmark methods include various supervised and semi-supervised learning approaches and cross-validation is used for evaluating model accuracy. The results show that our proposed method can efficiently obtain highly accurate event identification and localization in large systems with many data streams coming from PMUs. In general, we have the following contributions:

- We design KTDC-Se model to incorporate massive labeled and unlabeled PMU streams, where we employ tensors to uncover the complex multi-dimensional correlations and create compact and informative features to identify events.
- We derive a fast coordinate descent algorithm and its variational mini-batch version to train our KTDC-Se.
- We implement extensive experiments to demonstrate the high performance of KTDC-Se over other models with synthetic and real-world datasets.

To emphasize our contributions, we summarize the basic principles of our designs as follows. (1) The proposed model employs tensors to explore high-dimensional correlations and create more compact and informative features for accurate and fast inference. (2) The proposed model is a unified framework for dimension reduction and supervised learning, which prevents the bias induced by separate steps and metrics. (3) The proposed model can take in rich unlabeled data for better performance.

The remainder of the paper is organized as follows: Section II introduces the notations and tensor preliminaries for the model. Section III defines the problem. Section IV proposes our KTDC-Se. SectionV illustrates the learning algorithm. Section VI conducts experiments for baselines and KTDC-Se, and Section VII concludes the paper.

II. Notations and Preliminaries

To integrate PMU data reduction and machine learning in one tensorized framework, we first introduce basics of tensor algebra [20] and the corresponding notations. To summarize,

TABLE I: Overview of tensor notations and operators.

Notation	Interpretation
α	scalar
$\mathbf{a}$	vector
$\mathbf{A}$	matrix
$\mathcal{X}$	tensor, set, or space
$\mathbf{X}_{(n)}$	unfolding of tensor $\mathcal{X}$ along mode n
$\circ$	outer product
$\times_n$	mode-n product
$\otimes$	Kronecker product
$\lVert\cdot\rVert_2$	l_2 norm of a vector
$\lVert\cdot\rVert_F$	l_2 Frobenius norm of a matrix or a high-order tensor

Table I presents the basic notations for different types of variables and operations.

A. Tensor Notations

Multi-mode data can be stored in the so-called tensor [21], the multi-dimensional arrays. The number of dimensions for a tensor is referred to as order. For example, scalar (0-order tensor), vector (1-order tensor) and matrix (2-order tensor). Then, for a D-order tensor $\mathcal{X}$, $I_1 \times I_2 \times \cdots \times I_D$ are denoted as the dimensions, i.e., $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ or $\mathcal{X}^{I_1 \times I_2 \times \cdots \times I_D}$, where I_i ($\forall 1 \leq i \leq D$) is the dimensionality of the i^{th} dimension of $\mathcal{X}$.

B. Tensor Operations

There are many types of operations for a tensor like folding, unfolding, product, etc. This subsection provides some operations used in the KTDC-Se method.

mode-n unfolding of a tensor. A tensor can be unfolded to a matrix, a process that is also known as matrization. Specifically, for $\mathcal{X}^{I_1 \times I_2 \times \cdots \times I_D}$, one can unfold it along the n-dimension (mode) to obtain $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n \times \prod_{i=1, i\neq n} I_i}$. Mathematically, the result is:

$$\mathcal{X}(i_1, i_2, \cdots, i_D) = \boldsymbol{X}_{(n)}(i_n, j),$$
$$j = 1 + \sum_{k=1, k\neq n}^{D} (i_k - 1)J_k, \ J_k = \prod_{m-1, m\neq n}^{k-1} I_m,$$

where $\mathcal{X}(i_1, \cdots, i_n)$ is denoted as the $(i_1, \cdots, i_n)^{th}$ entry of tensor $\mathcal{X}$.

n-mode product. For a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ and a matrix $\boldsymbol{U} \in \mathbb{R}^{K \times I_n}$, the n-mode product is denoted as:

$$(\mathcal{X} \times_n \mathbf{U})(i_1, \cdots, i_{n-1}, k, i_{n+1}, \cdots, i_D)$$
$$= \sum_{i_n=1}^{I_n} \mathcal{X}(i_1, i_2, \cdots, i_D)\mathbf{U}(k, i_n),$$

where $\mathcal{X} \times_n \mathbf{U} \in \mathbb{R}^{I_1 \times \cdots I_{n-1} \times K \times I_{n+1} \cdots \times I_N}$ is a tensor.

Tensor Tucker Decomposition. For a D-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$, one key research topic is to find the approximation using a set of small tensors. For example, PMU data is of high volume and low rank [22]. Thus, the low-rank approximation is preferred to efficiently represent the PMU data and remove the redundant information. The target can be achieved via tensor decomposition. Specifically, the so-called Tucker decomposition is [20]:

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \cdots \times_D \boldsymbol{U}_D$$
$$\approx \sum_{r_1=1}^{R_1} \cdots \sum_{r_J=1}^{R_D} \mathcal{G}(r_1, \cdots, r_D)\mathbf{u}_1^{r_1} \circ \cdots \circ \mathbf{u}_D^{r_D},$$

where $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_D}$ is a core tensor in the factorization, and $\boldsymbol{U}_i \in \mathbb{R}^{I_i \times R_i}$ is a base matrix along mode i. $\mathbf{u}_i^{r_i}$ is the r_i^{th} column of $\boldsymbol{U}_i$ and $\circ$ is the outer product.

Finally, the Tucker decomposition can be rewritten in a matrix format:

$$\boldsymbol{X}_{(n)} = \boldsymbol{U}_n \boldsymbol{G}_{(n)}(\boldsymbol{U}_D \otimes \cdots \otimes \boldsymbol{U}_{n+1} \otimes \boldsymbol{U}_{n-1} \otimes \cdots \otimes \boldsymbol{U}_1)^\top,$$

where $\otimes$ is the so-called Kronecker product and $\top$ represents the matrix transpose. In summary, the introduced tensor operations lay foundations for our integrated model with certain physical interpretations. Specifically, tensor decomposition provides efficient feature extraction while maintaining certain physical structures in the core tensor $\mathcal{G}$. Further, tensor unfolding converts $\mathcal{G}$ to vectors that can be input to a classifier, which enables an end-to-end model of decomposition and classification.

III. Problem formulation

In this section, we define the target problem using the above notation. Before introducing the formulation, we first identify the study scope with the following points.

Data Preparation of PMU Tensors. For PMU streams, we follow the idea of [1], [15], [23] to extract a window of PMU signals with sufficient information to indicate the event dynamics for training a classifier. As shown in Fig. 2, each window of data can be formalized into a PMU tensor $\mathcal{X}' \in \mathbb{R}^{T \times L \times M}$ where T denotes the number of time slots for each window, L denotes the number of PMUs, and M denotes the number of measurement types (e.g., Voltage Magnitude (VM), Voltage Angle (VA), Frequency (F), etc.). Further, with a slight abuse of notation, the total N PMU tensors are denoted as the total event tensor $\mathcal{X} \in \mathbb{R}^{N \times T \times L \times M}$ and assume the first H ($H < N$) sub-tensors along the first dimension have labels. Correspondingly, the label vector is denoted as $\boldsymbol{y} \in \mathbb{Z}^{H \times 1}$. In addition, to make the tensor data comparable between different measurement types, we implement normalization to restrict each entry of the tensor within the range $[0, 1]$. More specifically, the normalization happens for each entry of the tensor in the total N tensors in Fig. 2. Then, among the N values, the maximum value of the specific entry is assigned to be 1, the minimum value is assigned to be 0, and the intermediate value is corresponding transformed to be an entry within $(0, 1)$.

Data Labels: Event Types and Locations. We focus on the tasks of distinguishing event types and locating events. Since we are proposing data-driven approaches without the requirement of domain knowledge, a diversified set of system event types and locations can be considered. For example, in the experiment, we consider five event types, including line trip, generator trip, single-phase-to-ground fault, phase-to-phase fault, and three-phase fault. Further, for each event type, we randomly create 2 different locations in the system. Similar treatments are implemented in many related studies [15], [16], [18], [23].

Treatment for Multiple Events: Our proposed method can handle multiple events. First, we admit that in the following modeling process, each of our trained classifier is binary to make better use of kernel methods with hinge loss for high accuracy, which is similar to Support Vector Machine (SVM) [24]. Nevertheless, power system event identification is essentially a multi-class classification problem, i.e., each unique combination of an event type and an event location can indicate a unique label. Thus, the one-against-one method is utilized to train multiple classifiers. Specifically, we train multiple binary models at the same time, and their majority vote leads to the final event label that can be an arbitrary integer. One can refer to [25] for this method which has better performances than other multi-class SVMs.

Treatment for New Data. If other types of events come without label information and if they never appear in the historical dataset, our model can not directly output the event type and location since this scenario is beyond our study scope. However, we can assign these new events a new label that means "to be determined". By doing this, we can view these new events as labeled data and input them into our model. This procedure is helpful in providing guidance.

Treatment for Imbalanced Dataset. Our model can successfully handle the imbalanced dataset due to the following reasons. First, as described above, each of our proposed models is binary with the focus on two classes of events. Thus, in the training process, we only select the data of two different labels for training. This mitigates the data-imbalance issue across different classes. Namely, the binary classifier will not be affected too much as long as the selected two classes of events have similar data numbers. Second, if the selected two classes of events are imbalanced to train the binary model, our proposed hinge loss can still guide the learning of an accurate decision boundary. This is because by minimizing hinge loss, the formulation of the decision boundary will only be determined by the so-called support vectors. These support vectors are data points that lie close to the decision boundary and can determine the equations of the boundary. However, for many interior points that are far from the decision boundary, they don't affect the final decision. To summarize, for imbalanced datasets, the hinge loss enables a small group of data to determine the parameters for the decision boundary, which resolves the issue of the imbalanced dataset.

Above notations and treatments summarize the common scenarios for power system event identification and how our proposed model can handle them. In general, we define our problem as follows.

- Problem type: semi-supervised event identification using PMU data.
- Given: a total event tensor $\mathcal{X}$ and a label vector $\boldsymbol{y}$.
- Find: an abstract mapping $f(\mathcal{X}) = \boldsymbol{y}$ to compress the information in $\mathcal{X}$ and use the compressed information to identify event labels in $\boldsymbol{y}$.

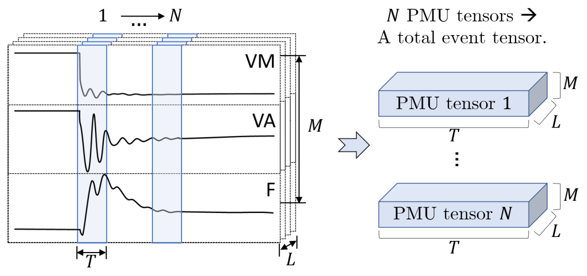


Fig. 2: Illustration of the moving window-based division to generate PMU tensors.

IV. Proposed model

The design of the above mapping f can be diversified. However, existing work suffers a key challenge of biased selections of data compression and event identification without proper integration. In this section, we design an end-to-end model that makes full use of tensor structure to achieve fast computations, physical interpretations, high capacity with non-linear feature extractions, and high accuracy under semi-supervision.

A. An Integrated Model with Efficiency and Physical Meanings

For an efficient model, we need to remove the redundancy in PMU measurements. Section I shows the drawback of traditional methods: they can hardly explore the high-dimensional correlations. Further, Section II illustrates that tensor is a natural container of high-dimensional data and tensor Tucker decomposition is an excellent approach to uncover the cross-dimension correlations. However, it is still unclear how we can design an efficient model to remove redundancy and capture event information for different PMU event tensors, and how we can guarantee the model robustness by tackling some labeled and rich unlabeled tensors.

For a detailed design, we show the motivation in Fig. 3. We utilize different colors to represent different components of data. More specifically, we utilize light blue to represent tensor data $\mathcal{X}^{'}$ and $\mathcal{G}$. Then, we utilize orange, blue, red, and green colors to represent the decomposed parameter matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$, respectively. These colors can help to distinguish the types of decomposed tensors. Second, to emphasize the dimension of the decomposed tensors, we bold the corresponding lines in $\mathcal{G}^{'}$ and utilize different colors in $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ to distinguish the bold lines. Then, the reader can easily understand how the dimensions match. (3) We change the figures for output y_1, y_2, and y_3 to circles, where the solid line represents $y_1 = 1$, the two types of dotted lines represent $y_2 = -1$ and $y_3 =?$ (i.e., unknown).

For each PMU tensor $\mathcal{X}^{'}$, the left part of Fig. 3 visualizes the process of a Tucker decomposition into base matrices $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ and a core tensor $\mathcal{G}$. $\mathcal{G}$ can maintain the structure as the PMU tensor $\mathcal{X}^{'}$, leading to specific physical interpretations. Specifically, the base matrices can be viewed as the bases along different dimensions, and the core tensor $\mathcal{G}$ represents the interactions among these bases [12]. Thus, we

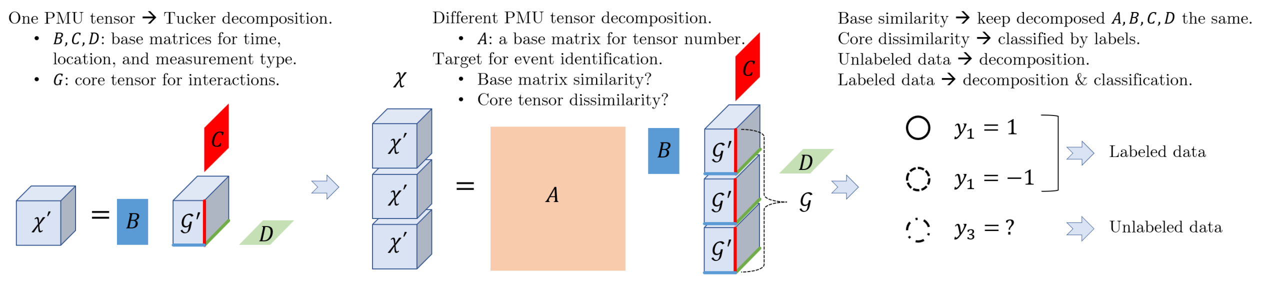

One PMU tensor → Tucker decomposition.
• B, C, D: base matrices for time, location, and measurement type.
• G: core tensor for interactions.
Different PMU tensor decomposition.
• A: a base matrix for tensor number.
Target for event identification.
• Base matrix similarity?
• Core tensor dissimilarity?
Base similarity → keep decomposed A, B, C, D the same.
Core dissimilarity → classified by labels.
Unlabeled data → decomposition.
Labeled data → decomposition & classification.
χ′ = B G′ C D
χ
χ′ χ′ χ′ = A B G′ G′ G′ C D G
y1 = 1
y1 = −1
y3 = ?
Labeled data
Unlabeled data

Fig. 3: The motivation of the KTDC-Se model.

can assume bases for different PMU tensors are similar as long as the number of bases is sufficiently enough. In contrast, the interaction tensor $\mathcal{G}$ contains discriminative event information.

Then, to maximally remove the redundancy, we directly keep the same bases $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ for different PMU tensors during the decomposition, shown in the middle part of Fig. 3. Namely, we utilize a direct Tucker decomposition for a 4-D total event tensor $\mathcal{X}$. Then, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are naturally kept to be the same. Further, we want the core tensor $\mathcal{G}$ to contain distinguished event information. Therefore, we employ the event labels to conduct a supervised learning-based classification for dissimilarity maximization. The above procedure is for labeled tensors. For unlabeled tensors, only the decomposition procedure is implemented to increase the model robustness to different loading conditions. The concrete mathematical model of joint optimization is formulated in the following subsection.

B. Semi-supervised Optimization for the Joint Model

In the semi-supervised learning setting, the 4-D total event tensor $\mathcal{X}$ in Section IV-A contains all labeled and unlabeled data. Mathematically, we implement the Tucker decomposition for $\mathcal{X}$ as follows:

$$\begin{aligned}\mathcal{X} &\approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \times_4 \boldsymbol{D}\\ &\approx \sum_{r_1=1}^{N}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\sum_{r_4=1}^{R_4} \mathcal{G}(r_1,r_2,r_3,r_4)\mathbf{a}^{r_1}\circ \mathbf{b}^{r_2}\circ \mathbf{c}^{r_3}\circ \mathbf{d}^{r_4},\end{aligned} \tag{1}$$

where $\mathcal{G} \in \mathbb{R}^{N\times R_2\times R_3\times R_4}$ is the core tensor, i.e., the compressed tensor with small information redundancy. The matrices to scale the core tensors are $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{T\times R_2}$, $\boldsymbol{C} \in \mathbb{R}^{L\times R_3}$, and $\boldsymbol{D} \in \mathbb{R}^{M\times R_4}$. $\boldsymbol{a}^{r_1}$, $\boldsymbol{b}^{r_2}$, $\boldsymbol{c}^{r_3}$, and $\boldsymbol{d}^{r_4}$ are the r_1^{th}, r_2^{th}, r_3^{th}, and r_4^{th} columns of $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$, respectively. $R_2 < T$, $R_3 < L$, and $R_4 < M$ are the pre-defined dimensions of the reduced tensors to achieve the information compression.

In the decomposition of Equation (1), the first dimension is fixed of the core tensor $\mathcal{G}$ to be N so that the decomposition can still bring N features to represent different PMU tensors. Furthermore, the decomposition can be rewritten as:

$$\boldsymbol{X}_{(1)} \approx \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{D}\otimes\boldsymbol{C}\otimes\boldsymbol{B})^\top, \tag{2}$$

where $\boldsymbol{X}_{(1)} \in \mathbb{R}^{N\times(T\cdot L\cdot M)}$ and $\boldsymbol{G}_{(1)} \in \mathbb{R}^{N\times(R_2\cdot R_3\cdot R_4)}$ represent the mode-1 unfolding matrix of tensors $\mathcal{X}$ and $\boldsymbol{G}$, respectively. Clearly, columns in $\boldsymbol{G}_{(1)}$ represent the compressed features that can be utilized for the classifier training. Under semi-supervision, we utilize the labeled tensors with labels for the classification. Then, we propose a joint decomposition-classification model.

$$\begin{aligned}\min_{\boldsymbol{G}_{(1)},\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D},\boldsymbol{w},b} J = &\underbrace{||\mathcal{X}-\mathcal{G}\times_1\boldsymbol{A}\times_2\boldsymbol{B}\times_3\boldsymbol{C}\times_4\boldsymbol{D}||_F^2}_{J_1,\ \text{Reconstruction Loss}}\\ &+\gamma_1\underbrace{l(\boldsymbol{E}\boldsymbol{G}_{(1)}\cdot\boldsymbol{w}+b,\boldsymbol{y})}_{J_2,\ \text{Classification Loss}}+\gamma_2\underbrace{||\boldsymbol{w}||_2^2}_{J_3,\ l_2\ \text{Norm}},\end{aligned} \tag{3}$$

where J, J_1, J_2, and J_3 denote the total loss, reconstruction loss, classification loss and regularization terms, respectively. $\boldsymbol{E} = [\boldsymbol{I}^{H\times H}, \boldsymbol{0}^{H\times(N-H)}] \in \mathbb{R}^{H\times N}$ denotes a selection matrix to select the first H feature instances (i.e., the instances with labels) in $\boldsymbol{G}_{(1)}$ for the classification. $||\cdot||_F$ and $||\cdot||_2$ denote the Frobenius norm and the l_2 norm, respectively. $l(\cdot,\cdot)$ represents the classification loss function. The hinge loss of Support Vector Machine (SVM) is considered in this paper, i.e., $l(\boldsymbol{E}\boldsymbol{G}_{(1)}\cdot\boldsymbol{w},\boldsymbol{y}) = \sum_{i=1}^{H}[1-y_i(\boldsymbol{f}_i^\top\boldsymbol{w}+b)]_+$, where $\boldsymbol{f}_i \in \mathbb{R}^{(R_2\cdot R_3\cdot R_4)\times 1}$ is the i^{th} instance of the transformed features and y_i is the i^{th} label in $\boldsymbol{y}$. Namely, $\boldsymbol{F} = [\boldsymbol{f}_1,\boldsymbol{f}_2,\cdots,\boldsymbol{f}_H]^\top = \boldsymbol{E}\boldsymbol{G}_{(1)}$. Further, the hinge loss is defined as $[1-t]_+ = \max(0,1-t)^p$. Usually, one can treat $p = 1$ or $p = 2$ for l_1- or l_2-SVM [11], respectively. γ_1 and γ_2 are positive hyper-parameters to reweight the three terms in Equation (3).

C. Efficient Kernelization for Powerful Non-linear Feature Extractions

In the last two subsections, we successfully merge the data reduction and machine learning model into one optimization under semi-supervision. However, many PMU measurements have non-linear correlations. It is challenging to add non-linearity due to the computational cost. For example, adding sigmoid or polynomial functions to the loss function l in Equation (3) significantly increases the computations. This motivates the usage of kernel trick for nonlinear feature calculations [26]. Specifically, (i) the kernel trick shows that we can find a kernel function $k(\boldsymbol{f}_1,\boldsymbol{f}_2)$ such that $k(\boldsymbol{f}_1,\boldsymbol{f}_2) = \phi(\boldsymbol{f}_1)^\top\phi(\boldsymbol{f}_2) = g(\boldsymbol{f}_1^\top\boldsymbol{f}_2)$. The inner product computations, i.e., the main calculation procedure for the loss in Equation (3), can be conducted in the original feature space rather than the features transformed from polynomial or sigmoid functions. (ii) Further, the input features of the classifier in Equation (3) is a feature vector $\boldsymbol{f}$ with the dimension $r = R_2\cdot R_3\cdot R_4$. Then, if we consider to utilize d-degree polynomial features to represent the original features, we can obtain a r^d-dimensional feature vector $\phi(\boldsymbol{f})$. Then, the inner product over the r^d-dimensional features can incorporate r^{2d} multiplications and $r^{2d}-1$ summations. In general, we need $2r^{2d}-1$ operations, which is expensive. (iii) However, the kernel trick enables another procedure of computations with a much smaller computational cost. Basically, let $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ denote two r-dimensional features. Then, one only need to consider the inner product within the original r-dimensional space with r^2 multiplications and r^2-1 summations. Then, based on the kernel trick, only extra d multiplications are needed to bring the same result as in (ii). In general, the total operation number is $2r^2-1+d$, which saves a lot of computational resources compared to the computations in (ii). Thus, we propose to utilize kernel function to lift the data to high-dimensional or even infinite-dimensional feature space, and the kernel trick can enable the calculation to happen in the original data space, which easily maintains efficient calculations [27].

Specifically, due to the Representer theorem [28], the inner product of the classification model can be rewritten as $\boldsymbol{f}^\top \boldsymbol{w} = \sum_{i=1}^{H} \alpha_i k(\boldsymbol{f}, \boldsymbol{f}_i)$, where $k(\cdot,\cdot)$ is the kernel function and $\boldsymbol{f} \in \mathbb{R}^{(R_2 \cdot R_3 \cdot R_4)\times 1}$ is a variable in the feature space. Based on this equation, the kernelized learning process is:

$$\begin{aligned}\min_{\boldsymbol{G}_{(1)},\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D},\boldsymbol{\alpha},b} J = & \underbrace{||\mathcal{X} - \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \times_4 \boldsymbol{D}||_F^2}_{J_1,\ \text{Reconstruction Loss}} \\ & + \underbrace{\gamma_1 \sum_{i=1}^{H} \big[1 - y_i(\sum_{j=1}^{H} \alpha_j k(\boldsymbol{f}_i, \boldsymbol{f}_j) + b)\big]_+}_{J_2,\ \text{Classification Loss}} \\ & + \underbrace{\gamma_2 \sum_{i=1}^{H}\sum_{j=1}^{H} \alpha_i \alpha_j k(\boldsymbol{f}_i, \boldsymbol{f}_j)}_{J_3,\ \text{Regularization}},\end{aligned} \tag{4}$$

where $\boldsymbol{\alpha}$ is the vector of all α_is. Eventually, we re-emphasize the nice properties of KTDC-Se by (1) using tensors to capture multi-dimensional correlations, (2) proposing a joint model for decomposition and classification, (3) introducing kernels for non-linear features, and (4) conveniently tackling both labeled and unlabeled data.

V. Learning algorithm

The optimization in Equation (4) is non-convex. Thus, an alternative optimization algorithm is proposed to update the individual variable in the optimization while fixing other variables, i.e., the so-called coordinate descent method. This method is prevailing in the domain of tensor learning due to its efficiency and good convergence property [11], [20], [29]. Further, to calculate the gradient and avoid the non-differentiable scenario, the l_2 SVM [11], [30] is utilized, i.e., $p=2$ in the loss function $l(\cdot,\cdot)$. Then, to update each variable, KTDC-Se only needs to calculate the gradients with respect to every single variable and utilize the gradient descent for updating. Thus, the calculation of the gradients is shown as follows.

A. Gradients of Matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$

Gradient of $\boldsymbol{A}$. Based on matrix format of Tucker decomposition and for the convenience of later derivations, we define $\boldsymbol{H}_1 = \boldsymbol{G}_{(1)}(\boldsymbol{D} \otimes \boldsymbol{C} \otimes \boldsymbol{B})^\top$. Then, the gradient of the loss function in Equation (4) is calculated with respect to $\boldsymbol{A}$. Mathematically, the gradient is:

$$\begin{aligned}\nabla_{\boldsymbol{A}} J = \nabla_{\boldsymbol{A}} J_1 &= \nabla_{\boldsymbol{A}} \text{tr}\big((\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{H}_1)^\top \cdot (\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{H}_1)\big) \\ &= 2(\boldsymbol{A}\boldsymbol{H}_1\boldsymbol{H}_1^\top - \boldsymbol{X}_{(1)}\boldsymbol{H}_1^\top),\end{aligned} \tag{5}$$

where tr$(\cdot)$ represents the operation to obtain the matrix trace.
Gradients of $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$. By symmetry, $\boldsymbol{H}_2 = \boldsymbol{G}_{(2)}(\boldsymbol{D} \otimes \boldsymbol{C} \otimes \boldsymbol{A})^\top$, $\boldsymbol{H}_3 = \boldsymbol{G}_{(3)}(\boldsymbol{D} \otimes \boldsymbol{B} \otimes \boldsymbol{A})^\top$, and $\boldsymbol{H}_4 = \boldsymbol{G}_{(4)}(\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})^\top$ can be defined. Then, $\nabla_{\boldsymbol{B}} J$, $\nabla_{\boldsymbol{C}} J$, and $\nabla_{\boldsymbol{D}} J$ are calculated as follows:

$$\begin{aligned}\nabla_{\boldsymbol{B}} J &= 2(\boldsymbol{B}\boldsymbol{H}_2\boldsymbol{H}_2^\top - \boldsymbol{X}_{(2)}\boldsymbol{H}_2^\top), \\ \nabla_{\boldsymbol{C}} J &= 2(\boldsymbol{C}\boldsymbol{H}_3\boldsymbol{H}_3^\top - \boldsymbol{X}_{(3)}\boldsymbol{H}_3^\top), \\ \nabla_{\boldsymbol{D}} J &= 2(\boldsymbol{D}\boldsymbol{H}_4\boldsymbol{H}_4^\top - \boldsymbol{X}_{(4)}\boldsymbol{H}_4^\top).\end{aligned} \tag{6}$$

B. Gradient of Mode-1 Unfolding Matrix $\boldsymbol{G}_{(1)}$ of the Core Tensor

To update $\boldsymbol{G}_{(1)}$, the following gradients are separately derived. For the reconstruction loss, $\tilde{\boldsymbol{H}} = (\boldsymbol{D} \otimes \boldsymbol{C} \otimes \boldsymbol{B})^\top$ is denoted. Then, the gradient is:

$$\begin{aligned}\nabla_{\boldsymbol{G}_{(1)}} J_1 &= \nabla_{\boldsymbol{G}_{(1)}} \text{tr}\big((\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{G}_{(1)}\tilde{\boldsymbol{H}})^\top \cdot (\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{G}_{(1)}\tilde{\boldsymbol{H}})\big) \\ &= 2(\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{G}_{(1)}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{H}}^\top - \boldsymbol{A}^\top\boldsymbol{X}_{(1)}\tilde{\boldsymbol{H}}^\top).\end{aligned} \tag{7}$$

For the classification loss, $\hat{y}_i = \sum_{j=1}^{H} \alpha_j k(\boldsymbol{f}_i, \boldsymbol{f}_j) + b$ is denoted for simplification. Based on the chain rule, the gradient is:

$$\nabla_{\boldsymbol{f}_i} J_2 = \begin{cases} 2(\hat{y}_i - y_i)\sum_{j=1}^{H} \alpha_j \frac{\partial k(\boldsymbol{f}_i,\boldsymbol{f}_j)}{\partial \boldsymbol{f}_i} & \\ +2\alpha_i \sum_{j\neq i}^{H} (\hat{y}_j - y_j)\frac{\partial k(\boldsymbol{f}_i,\boldsymbol{f}_j)}{\partial \boldsymbol{f}_i}, & \text{if } y_i\hat{y}_i < 1\ , \\ \boldsymbol{0}, & \text{if } y_i\hat{y}_i \geq 1\ . \end{cases} \tag{8}$$

To elaborate on the above equation, polynomial and Radial Basis Function (RBF) kernels are utilized as examples. For the polynomial kernel $k(\boldsymbol{f}_i, \boldsymbol{f}_j) = (\boldsymbol{f}_i^\top \boldsymbol{f}_j + c)^d$, where c is a constant and d is the degree of the polynomial function, then

$$\frac{\partial k(\boldsymbol{f}_i,\boldsymbol{f}_j)}{\partial \boldsymbol{f}_i} = \begin{cases} d(\boldsymbol{f}_i^\top \boldsymbol{f}_j + c)^{d-1}\boldsymbol{f}_j, & \text{if } i \neq j\ , \\ 2d(\boldsymbol{f}_i^\top \boldsymbol{f}_j + c)^{d-1}\boldsymbol{f}_i, & \text{if } i = j\ . \end{cases} \tag{9}$$

For the RBF kernel $k(\boldsymbol{f}_i, \boldsymbol{f}_j) = \exp(-\lambda||\boldsymbol{f}_i - \boldsymbol{f}_j||_2^2)$, where λ is a positive constant:

$$\frac{\partial k(\boldsymbol{f}_i,\boldsymbol{f}_j)}{\partial \boldsymbol{f}_i} = 2\lambda k(\boldsymbol{f}_i,\boldsymbol{f}_j)(\boldsymbol{f}_j - \boldsymbol{f}_i). \tag{10}$$

Recall that $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_H]^\top = \boldsymbol{E}\boldsymbol{G}_{(1)}$, $\nabla_{\boldsymbol{F}} J_2 = [\nabla_{\boldsymbol{f}_1} J_2, \nabla_{\boldsymbol{f}_2} J_2, \cdots, \nabla_{\boldsymbol{f}_H} J_2]^\top$ can be obtained. Thus, $\nabla_{\boldsymbol{G}_{(1)}} J_2 = [\nabla_{\boldsymbol{f}_1} J_2, \nabla_{\boldsymbol{f}_2} J_2, \cdots, \nabla_{\boldsymbol{f}_H} J_2, \boldsymbol{0}, \cdots, \boldsymbol{0}]^\top$.

For the Regularization term, the result is:

$$\nabla_{\boldsymbol{f}_i} J_3 = \alpha_i^2 \frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_i)}{\partial \boldsymbol{f}_i} + 2\alpha_i \sum_{j \neq i} \alpha_j \frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\partial \boldsymbol{f}_i}. \tag{11}$$

Similarly, $\nabla_{\boldsymbol{G}_{(1)}} J_3 = [\nabla_{\boldsymbol{f}_1} J_3, \nabla_{\boldsymbol{f}_2} J_3, \cdots, \nabla_{\boldsymbol{f}_H} J_3, \boldsymbol{0}, \cdots, \boldsymbol{0}]^\top$ can be obtained. Summing the gradients of the three loss functions can bring the total gradient, i.e., $\nabla_{\boldsymbol{G}_{(1)}} J = \nabla_{\boldsymbol{G}_{(1)}} J_1 + \gamma_1 \nabla_{\boldsymbol{G}_{(1)}} J_2 + \gamma_2 \nabla_{\boldsymbol{G}_{(1)}} J_3$.

C. Gradients of Classifier Parameters $\boldsymbol{\alpha}$ and b

Gradient of $\boldsymbol{\alpha}$. The learning weight $\boldsymbol{\alpha}$ is coupled with the classification loss and the regularization. For the classification loss, then

$$\nabla_{\alpha_i} J_2 = \begin{cases} \sum_{j=1}^{H} 2(\hat{y}_j - y_j) k(\boldsymbol{f}_i, \boldsymbol{f}_j), & \text{if } y_i \hat{y}_i < 1 \ , \\ \\ 0, & \text{if } y_i \hat{y}_i \geq 1 \ . \end{cases} \tag{12}$$

Note that $\hat{y}_j$ can be explicitly expressed by $\boldsymbol{\alpha}$, i.e., $\hat{y}_j = \boldsymbol{k}_j^\top \boldsymbol{\alpha} + b$, where $\boldsymbol{k}_i$ is the i^{th} column vector of the kernel matrix $\boldsymbol{K}$, and the kernel matrix is defined as $\boldsymbol{K}(i,j) = k(\boldsymbol{f}_i, \boldsymbol{f}_j)$. Thus, the above equation can be written to a matrix format. Specifically, if $y_i \hat{y}_i < 1$, $\nabla_{\alpha_i} J_2 = 2\boldsymbol{k}_i^\top \cdot \boldsymbol{K}\boldsymbol{\alpha} + 2\boldsymbol{k}_i^\top \cdot (b - y_i)\boldsymbol{1}$ can be written, where $\boldsymbol{1}$ is an all-one column vector. Further, one can obtain a general format:

$$\nabla_{\boldsymbol{\alpha}} J_2 = 2\boldsymbol{K}\boldsymbol{I}^0(\boldsymbol{K}\boldsymbol{\alpha} + b\boldsymbol{1} - \boldsymbol{y}), \tag{13}$$

where $\boldsymbol{I}^0$ satisfies

$$\boldsymbol{I}^0(i,j) = \begin{cases} 1, & \text{if } i = j \text{ and } y_i \hat{y}_i < 1 \ , \\ \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Further, for the regularization, it's easy to find that

$$\nabla_{\boldsymbol{\alpha}} J_3 = 2\boldsymbol{K}\boldsymbol{\alpha}. \tag{15}$$

Finally, the total gradient is $\nabla_{\boldsymbol{\alpha}} J = \gamma_1 \nabla_{\boldsymbol{\alpha}} J_2 + \gamma_2 \nabla_{\boldsymbol{\alpha}} J_3$.

Gradient of b. Similarly, we calculate the gradient with respect to b:

$$\nabla_b J = \boldsymbol{1}^\top \boldsymbol{I}^0 (\hat{\boldsymbol{y}} - \boldsymbol{y}), \tag{16}$$

where $\hat{\boldsymbol{y}}$ is the vector of all $\hat{y}_i$s. With the above derivations, the final learning algorithm and flowchart are presented in Algorithm 1 and Fig. 4.

D. Training on Mini-batches

The above learning process may suffer storage issues when the number of training data N is large. Specifically, when updating $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ in Equations (6), the Kronecker product to calculate $\boldsymbol{H}_2$, $\boldsymbol{H}_3$, and $\boldsymbol{H}_4$ requires a large cost of storage as $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ for a large N. To mitigate this issue, we modify Algorithm 1 to train on mini-batches to save the memory [31].

Mathematically, we divide the training tensor $\mathcal{X}$ and labels $\boldsymbol{y}$ into K mini-batches $\{\mathcal{X}^i\}_{i=1}^K$ and $\{\boldsymbol{y}^i\}_{i=1}^K$, respectively. Each $\mathcal{X}^i$ contains some labeled data with labels to be $\boldsymbol{y}^i$ and many unlabeled data. Then, in each iteration, we update the mini-batch-independent weights $\tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ and b. $\tilde{\boldsymbol{A}} \in \mathbb{R}^{\tilde{N} \times \tilde{N}}$ is the matrix along the first dimension of each

Algorithm 1 Train-KTDC-Se($\mathcal{X}, \boldsymbol{y}$).

```
Input: Training tensor $\mathcal{X}$ and labels $\boldsymbol{y}$.
Hyper-parameters: number of labeled data $H$, core tensor dimensions $R_2$, $R_3$, and $R_4$, regularization parameters $\gamma_1$ and $\gamma_2$, polynomial kernel parameters $d$ and $c$, RBF kernel parameters $\lambda$, and learning rate $lr$.
Output: Parameters $\boldsymbol{G}_{(1)}^k, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}$, and $b$.
1:  Initialize $\boldsymbol{G}_{(1)}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}$, and $b$.
2:  repeat
3:    Calculate $\nabla_{\boldsymbol{A}} J$, $\nabla_{\boldsymbol{B}} J$, $\nabla_{\boldsymbol{C}} J$, and $\nabla_{\boldsymbol{D}} J$ by Equations (5) and (6).
4:    $\boldsymbol{A} = \boldsymbol{A} - lr \cdot \nabla_{\boldsymbol{A}} J$.
5:    $\boldsymbol{B} = \boldsymbol{B} - lr \cdot \nabla_{\boldsymbol{B}} J$.
6:    $\boldsymbol{C} = \boldsymbol{C} - lr \cdot \nabla_{\boldsymbol{C}} J$.
7:    $\boldsymbol{D} = \boldsymbol{D} - lr \cdot \nabla_{\boldsymbol{D}} J$.
8:    Calculate $\nabla_{\boldsymbol{G}_{(1)}} J_1$ by Equation (7).
9:    for $i = 1$ to $H$ do
10:     Calculate $\nabla_{\boldsymbol{f}_i} J_2$ and $\nabla_{\boldsymbol{f}_i} J_3$ by Equations (8) and (11) while fixing other parameters.
11:   end for
12:   Formalize $\nabla_{\boldsymbol{G}_{(1)}} J_2$ and $\nabla_{\boldsymbol{G}_{(1)}} J_3$. Then, obtain $\nabla_{\boldsymbol{G}_{(1)}} J$.
13:   $\boldsymbol{G}_{(1)} = \boldsymbol{G}_{(1)} - lr \cdot \nabla_{\boldsymbol{G}_{(1)}} J$.
14:   Build matrix $\boldsymbol{I}^0$ by Equation (14).
15:   Calculate $\nabla_{\boldsymbol{\alpha}} J_2$ and $\nabla_{\boldsymbol{\alpha}} J_3$ by Equations (13) and (15). Then, obtain $\nabla_{\boldsymbol{\alpha}} J$.
16:   $\boldsymbol{\alpha} = \boldsymbol{\alpha} - lr \cdot \nabla_{\boldsymbol{\alpha}} J$.
17:   Calculate $\nabla_b J$ by Equation (16).
18:   $b = b - lr \cdot \nabla_b J$.
19: until convergence
```

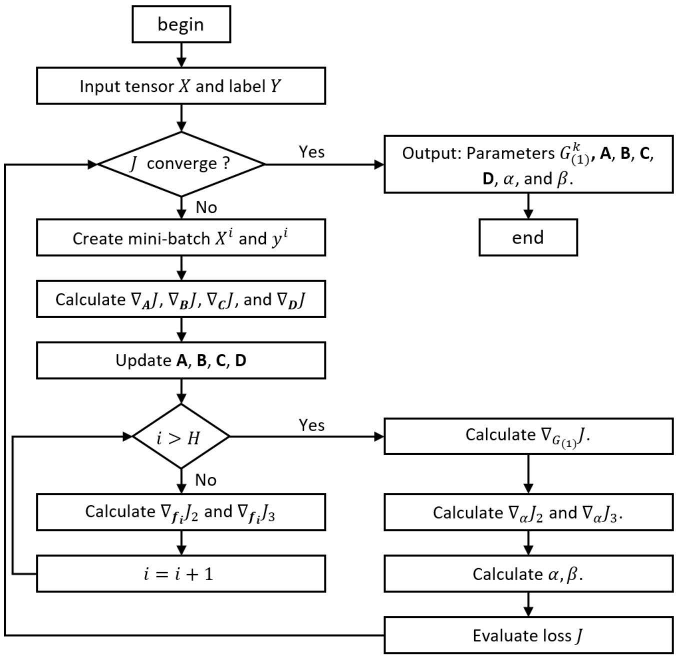

begin
Input tensor X and label Y
J converge ?
Yes
No
Output: Parameters $G_{(1)}^k$, **A**, **B**, **C**, **D**, α, and β.
end
Create mini-batch X^i and y^i
Calculate $\nabla_A J$, $\nabla_B J$, $\nabla_C J$, and $\nabla_D J$
Update **A**, **B**, **C**, **D**
$i > H$
Yes
No
Calculate $\nabla_{G_{(1)}} J$.
Calculate $\nabla_{f_i} J_2$ and $\nabla_{f_i} J_3$
$i = i + 1$
Calculate $\nabla_\alpha J_2$ and $\nabla_\alpha J_3$.
Calculate α, β.
Evaluate loss J

Fig. 4: The Flowchart of the KTDC-SE algorithm.

mini-batch tensor, where $\tilde{N} = \lfloor N/K \rfloor$ and $\lfloor \cdot \rfloor$ represents the floor function. Notably, keeping $\tilde{\boldsymbol{A}}$ the same for different mini-batches is an additional restriction to maintain similarity of $\boldsymbol{A}$, which doesn't appear in the direct training of Algorithm 1. However, this restriction further guarantees that the discriminative information is contained in core tensors.

Further, KTDC-Se uses mini-batch-dependent parameters $\boldsymbol{G}^i_{(1)} \in \mathbb{R}^{\tilde{N} \times (R_2 \cdot R_3 \cdot R_4)}$ and $\boldsymbol{\alpha}^i$ for the i^{th} mini-batch, where $\boldsymbol{\alpha}^i \in \mathbb{R}^{\tilde{H} \times 1}$ and $\tilde{H} = \lfloor H/K \rfloor$. Especially, $\boldsymbol{G}^i_{(1)}$ is a sub-block of $\boldsymbol{G}_{(1)}$, i.e., $\boldsymbol{G}_{(1)} = [(\boldsymbol{G}^1_{(1)})^\top, \cdots, (\boldsymbol{G}^K_{(1)})^\top]^\top$. Correspondingly, we have $\boldsymbol{F} = [(\boldsymbol{F}^1)^\top, \cdots, (\boldsymbol{F}^K)^\top]^\top$, where $\boldsymbol{F}^i = \boldsymbol{E}^i \boldsymbol{G}^i_{(1)}$ and $\boldsymbol{E}^i = [\boldsymbol{I}^{\tilde{H} \times \tilde{H}}, \boldsymbol{0}^{\tilde{H} \times (\tilde{N} - \tilde{H})}]$. Essentially, $\boldsymbol{\alpha}^i$ is a sub vector of the final weight vector $\boldsymbol{\alpha} = [(\boldsymbol{\alpha}^1)^\top, \cdots, (\boldsymbol{\alpha}^K)^\top]^\top$.

Then, for the i^{th} mini-batch, our training algorithm should obtain features $\boldsymbol{G}^i_{(1)}$, check the support vectors in these features, and update their corresponding weights in $\boldsymbol{\alpha}^i$. To maintain the coupling between $\boldsymbol{G}^i_{(1)}(\boldsymbol{\alpha}^i)$ and the rest $\boldsymbol{G}^j_{(1)}$s ($\boldsymbol{\alpha}^j$s), where $j \neq i$, all the information in $\boldsymbol{F}$, $\boldsymbol{y}$ and $\boldsymbol{\alpha}$ should be utilized to obtain $\nabla_{\boldsymbol{G}^i_{(1)}} J_2$, $\nabla_{\boldsymbol{G}^i_{(1)}} J_3$, $\nabla_{\boldsymbol{\alpha}^i} J_2$, and $\nabla_{\boldsymbol{\alpha}^i} J_3$ by Equations (8), (11), (13), and (15), respectively. Then, we can fix the correspondingly gradients with respect to support vectors in other mini-batches to $\boldsymbol{0}$s. Consequently, the complete algorithm can be seen in Algorithm 2.

Algorithm 2 Train-mini-batch-KTDC-Se($\mathcal{X}, \boldsymbol{y}$).

```
    Input: Training tensor $\mathcal{X} = \{\mathcal{X}^i\}_{i=1}^K$ and labels $\boldsymbol{y} = \{\boldsymbol{y}^i\}_{i=1}^K$.
    Output: Parameters $\{\boldsymbol{G}^i_{(1)}\}_{i=1}^K, \tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}$, and $b$.
1: repeat
2: for $i = 1$ to $K$ do
3:     Utilize the complete information in $\boldsymbol{G}$, $\boldsymbol{y}$, and $\boldsymbol{\alpha}$ and the mini-batch data to obtain: $\boldsymbol{G}^i_{(1)}, \tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}^i, b =$ Train-KTDC-Se($\mathcal{X}^i, \boldsymbol{y}^i | \boldsymbol{F}, \boldsymbol{y}, \boldsymbol{\alpha}$).
4:     $\boldsymbol{F} = [(\boldsymbol{F}^1)^\top, \cdots, (\boldsymbol{F}^K)^\top]^\top$.
5:     $\boldsymbol{\alpha} = [(\boldsymbol{\alpha}^1)^\top, \cdots, (\boldsymbol{\alpha}^K)^\top]^\top$.
6: end for
7: until convergence
```

E. Testing on Mini-batches

For the cross-validation process or online testing, we have another total test tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{\tilde{N} \times T \times L \times M}$ that needs to experience the decomposition and classification to obtain the label $\tilde{\boldsymbol{y}} \in \mathbb{Z}^{\tilde{N} \times 1}$, where we fix $\tilde{N}$ to be the number of PMU tensors in one mini-batch. The reason of fixing $\tilde{N}$ is that our mini-batch training yields a parameter matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{\tilde{N} \times \tilde{N}}$ that must be utilized for the test tensor decomposition. Therefore, there should be $\tilde{N}$ PMU tensors in the total test tensor. For real-time testing, if the testing PMU tensor number is not sufficient, we can repeat the testing tensor or utilize some data from the historical dataset to complete the testing mini-batch. Thus the testing procedures are as follows.

Obtain Test Feature Matrix $\tilde{\boldsymbol{G}}_{(1)}$. To obtain $\tilde{\boldsymbol{G}}_{(1)}$, we utilize the learned parameters $\tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$. By setting the gradient in Equation (7) to $\boldsymbol{0}$, we can obtain

$$\tilde{\boldsymbol{G}}_{(1)} = (\tilde{\boldsymbol{A}}^\top \tilde{\boldsymbol{A}})^{-1} \cdot (\tilde{\boldsymbol{A}}^\top \tilde{\boldsymbol{X}}_{(1)} \tilde{\boldsymbol{H}}^\top) \cdot (\tilde{\boldsymbol{H}} \tilde{\boldsymbol{H}}^\top)^{-1}, \tag{17}$$

where $\tilde{\boldsymbol{X}}_{(1)}$ represents mode-1 unfolding of tensor $\tilde{\mathcal{X}}$.

Predict label vector $\tilde{\boldsymbol{y}}$. Based on the Representer theorem, we need the learned weights $\boldsymbol{\alpha}$ and b, historical features in $\boldsymbol{F} = \boldsymbol{G}_{(1)}$, and test features in $\tilde{\boldsymbol{G}}_{(1)} = [\tilde{\boldsymbol{f}}_1^\top, \cdots, \tilde{\boldsymbol{f}}_{\tilde{N}}^\top]^\top$ to predict labels. Specifically, we can first calculate a test kernel matrix $\tilde{\boldsymbol{K}}(i,j) = k(\tilde{\boldsymbol{f}}_i, \boldsymbol{f}_j)$, where $\tilde{\boldsymbol{f}}_i \in \tilde{\boldsymbol{G}}_{(1)}$ and $\boldsymbol{f}_j \in \boldsymbol{F}$. Then, the predicted label can be obtained by:

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{K}} \boldsymbol{\alpha} + b. \tag{18}$$

VI. Experiments

For validation, we test over synthetic data sets such as the Illinois 200-bus system South Carolina 500-bus system [18]. We also test our result with realistic data from our utility partners.

A. Dataset Description

We utilize Illinois 200-bus system and South Carolina 500-bus system [18] to generate event data. Five event types are considered, including line trip, generator trip, single-phase-to-ground fault, phase-to-phase fault, and three-phase fault. For each event type, we consider 2 different event locations. Thus, there are 10 unique combinations of event types and locations, i.e., 10 event labels.

Then, we vary the loading conditions for the simulation to generate diversified event files. Totally, we have 80 event files each of which has 10s event data. Further, we consider the data resolution to be 60 samples per second, yielding 600 samples for each event file. To extract tensors from these streams, we utilize the moving window with the length to be 0.5s (i.e., 30 samples) and the moving gap to be 0.083s (i.e., 5 samples) to cut the PMU streams. We utilize a small moving window to obtain data points. In our experiment, each window covers 0.5s (i.e., 30 samples of PMU measurements). Fig. 5 illustrates why we select 0.5s as an appropriate window length. Specifically, the plot is a visualization of the PMU streams over time, where the x-axis represents the time and the y-axis represents the measurements (VM denotes voltage magnitude, VA denotes voltage angle, and F denotes frequency). We find that using 0.5s as the window length can appropriately include partial event information to identify events. Next, the length is not too long to prevent fast and real-time detection. Finally, the determination of the window length can be further studied by treating the window length as a hyper-parameter and conducting the cross-validation. However, this won't affect our main result and we treat this procedure as future work.

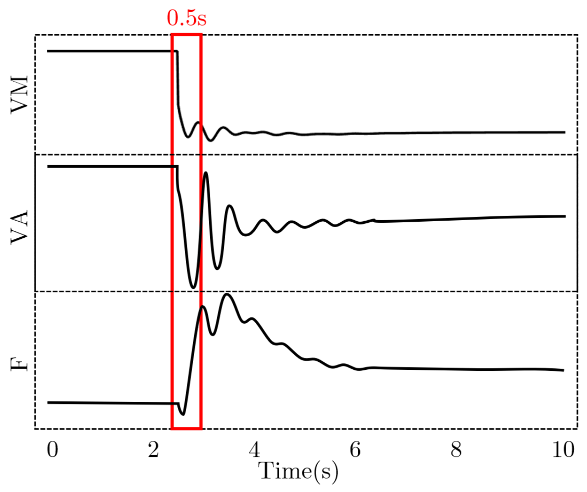


Fig. 5: The demonstration for window length selection.

Therefore, we have 5840 PMU tensors in total. For each PMU tensor, we have $T = 30$ for the time dimension, $L = 200\eta_1$ or $L = 500\eta_1$ for the 200-bus and the 500-bus system, respectively, where $\eta_1 \in \{0.05, 0.1, 0.15, 0.2\}$ represents the PMU penetrations for the grid. For each fixed η_1, PMU locations are randomly chosen for 5 times. Then, we set $M = 3$ for the measurement types, i.e., voltage magnitude, voltage angle, and frequency. To summarize, we have $\mathcal{X} \in \mathbb{R}^{5840\times 30\times 200\eta_1\times 3}$ or $\mathcal{X} \in \mathbb{R}^{5840\times 30\times 500\eta_1\times 3}$ for training and testing. To mimic a semi-supervised setting, we consider labeled data with the ratio of $\eta_2 = \{0.1, 0.2, 0.3, 0.4, 0.5\}$, leading to a label vector $\boldsymbol{y} \in \mathbb{Z}^{5840\eta_2\times 1}$.

Finally, we also test our proposed method using real-world PMU data from our partner in Arizona, USA. These files totally have 5 labels covering 3 types of line faults at 2 locations. After tensorization of data in 35 PMUs, we can obtain $\mathcal{X} \in \mathbb{R}^{511\times 30\times 35\times 3}$ and $\boldsymbol{y} \in \mathbb{Z}^{511\times 1}$.

Software For the simulation, we employ a commercial-grade simulator, Positive Sequence Load Flow (PSLF) [32] from General Electric (GE) company. For the model development and validation, we use Python with Pycharm IDE.

Summary of data and feature dimensions

- For Illinois 200-bus system, the total event tensor is $\mathcal{X} \in \mathbb{R}^{5840\times 30\times 200\eta_1\times 3}$ and the input data dimensions are $\{900, 1800, 2700, 3600\}$. Subsequently, we set $R_2 = 6$, $R_3 \in \{5, 5, 6, 8\}$, and $R_4 = 2$, and the feature dimensions are $\{60, 60, 72, 96\}$. For other benchmark methods, we employ PCA to reduce the dimensionality of the raw data and obtain features with the dimension of $\{120, 140, 150, 180\}$.
- For South Carolina 500-bus system, the total event tensor is $\mathcal{X} \in \mathbb{R}^{5840\times 30\times 500\eta_1\times 3}$ and the input data dimensions are $\{2250, 4500, 6750, 9000\}$. Subsequently, we set $R_2 = 6$, $R_3 \in \{5, 6, 8, 10\}$, and $R_4 = 2$, and the feature dimensions are $\{60, 72, 96, 120\}$. For other benchmark methods, we employ PCA to reduce the dimensionality of the raw data and obtain features with the dimension of $\{140, 180, 200, 240\}$.
- For datasets of the utility in Arizona, USA, the total event tensor is $\mathcal{X} \in \mathbb{R}^{511\times 30\times 35\times 3}$ and the input data dimension is 3150. Subsequently, we set $R_2 = 6$, $R_3 = 6$, and $R_4 = 2$, and the feature dimension is 72. For other benchmark methods, we employ PCA to reduce the dimensionality of the raw data and obtain features with the dimension of 160.

B. Benchmark Methods

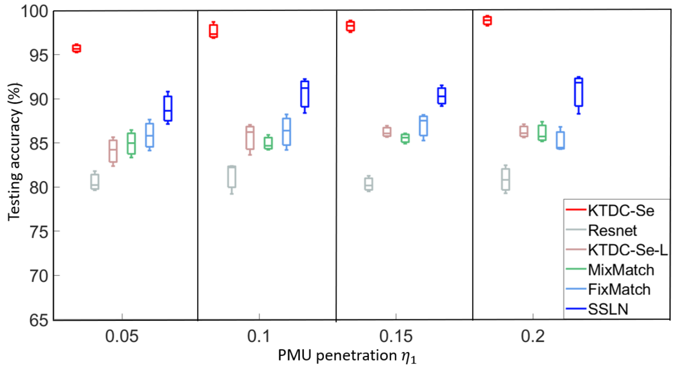
First, we train our KTDC-Se within the labeled data as a benchmark to demonstrate the impacts of the unlabeled data. Further, we employ state-of-the-art Semi-Supervised Learning (SSL) methods as benchmarks. The details of these methods are shown as follows.

- Deep Residual Network (Resnet) [33]: Resnet is an efficient deep learning model for classification. For this supervised learning model, we utilize only labeled data as comparison. As PMU data have high dimensionality (e.g., 9000 for the 500-bus system with $\eta_1 = 0.2$), Principal Component Analysis (PCA) is utilized to pre-process data before training the Resnet.
- KTDC-Se-L: KTDC-Se-L is to train a KTDC-Se model with only labeled data by setting $N = H$ in the model, which demonstrates the effectiveness of employing unlabeled data for training a classifier.
- MixMatch [34]: MixMatch can guess low-entropy labels for unlabeled instances with data augmentation. Then, MixMatch develops a probabilistic procedure to mix the labeled and unlabeled data to train a deep learning classifier. Similarly, we employ PCA to reduce the dimensionality of the mixed dataset from MixMatch and input them into a Resnet [33] as the final classifier. For a fair comparison, the Resnet has the same architecture as the first benchmark.
- FixMatch [35]: FixMatch first generates pseudo labels for data with weak data augmentation. Then, FixMatch develops a criterion to decide the pseudo label is retained or not. Finally, data with retained pseudo labels experience a strong data augmentation for the classifier training. Similar to MixMatch, we utilize PCA + Resnet as the final classifier. For fair comparison, the Resnet has the same architecture as the first benchmark.
- Semi-supervised Ladder Network (SSLN) [36]: SSLN combines supervised and unsupervised learning in deep neural networks with a joint loss function in a ladder network with an auto-encoder model structure. Similarly, we pre-process the data with the PCA method.

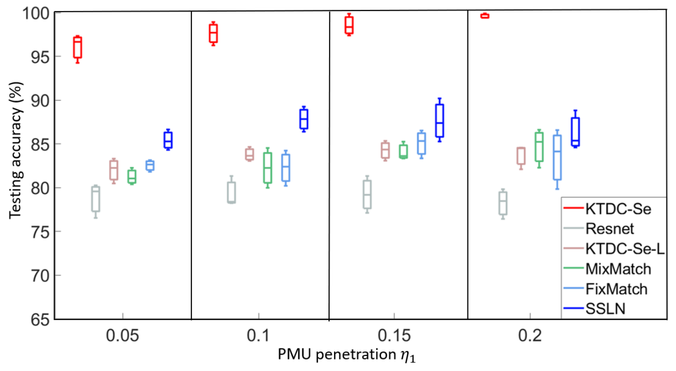
During the testing, the hyper-parameters for all models are fine-tuned in the 3-fold cross-validation to achieve the best accuracy. In general, by comparing the testing accuracy of KTDC-Se with Resnet, and KTDC-Se-L, we can illustrate the effectiveness of using unlabeled data. By comparing the testing accuracy of KTDC-Se and other methods, we can evaluate the performance of using an integrated model and two-stage models. Especially, Resnet, MixMatch, FixMatch, and SSLN have two separate steps of data pre-processing and learning, which have their biased selections. By comparing the label predicting time of KTDC-Se and other methods, we can evaluate the efficiency of the methods for real-time inference.

TABLE II: Testing accuracy (%) (mean ± standard deviation) for real-world PMU data.

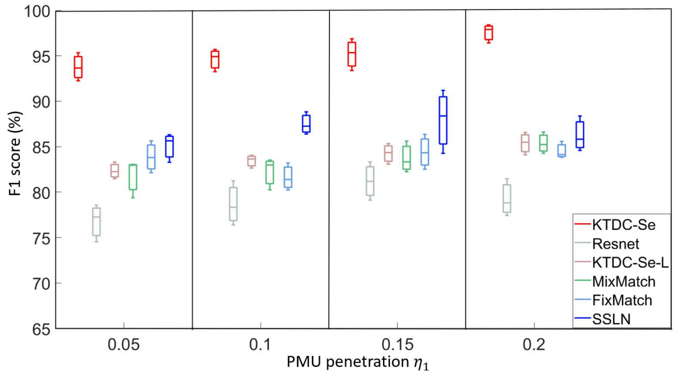
	KTDC-SE	RESNET	KTDC-SE-L	MIXMATCH	FIXMATCH	SSLN
ACCURACY	**92.3 ± 0.8**	77.1 ± 0.6	80.5 ± 1.1	82.6 ± 0.8	83.3 ± 1.6	83.5 ± 2.1



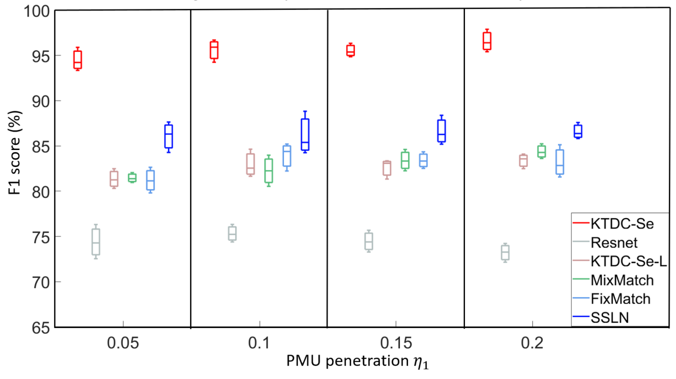
(a) Testing accuracy (%) for the 200-bus system.



(b) Testing accuracy (%) for the 500-bus system.



(c) F1 score (%) for the 200-bus system.



(d) F1 score (%) for the 500-bus system.

Fig. 6: Performances of event identification using different methods.

Finally, by comparing the choice of kernel selection, we can understand how the non-linear kernels boost the performance.

C. Joint Optimization of KTDC-Se is Better Than Two-stage Models

In this subsection, we evaluate the integration design by comparing our KTDC-Se and other two-stage models. Thanks to the integration without biased selection for the two-stage compression and classification, our model performs much better than benchmarks. Specifically, we report the results of simulated and real-world data as follows.

For the simulated data, we first fix $\eta_2 = 0.3$ to divide the labeled and unlabeled datasets for training and testing. Fig. 6a and 6b demonstrate the results for the two systems. Since for each η_1 of one system, we conduct 5 times randomization and 3-fold cross validation of the PMU location selection, there can be multiple different values of the testing accuracy for each testing scenario. Thus, we present the box plot in Fig. 6a and 6b to show the average and the variance.

We find that our KTDC-Se has an average accuracy promotion of 13.3%, 13.6%, and 9.3%, compared to MixMatch, FixMatch, and SSLN, respectively. Notably, the latter 3 benchmark models utilize PCA to pre-process data so that they can be trained with a reasonable cost. Even though these 4 methods utilize the same labeled and unlabeled data for training, the better performance of KTDC-Se shows that an integrated model can be better than the two-stage models. This is because that the integrated model avoids the biased selection of the two separate models. Further, the joint model enables the identification of discriminative core tensors that are sensitive to the event labels.

In our experiment, we implement 5 times random sampling for PMU locations for a given PMU penetration η_1 for two systems. Intuitively, if the PMUs are closer to the event, the accuracy should be higher. Then, for a fixed η_1 and fixed event locations, the relative PMU locations with respect to the event location cause the accuracy variance. Therefore, 500-bus system usually has higher accuracy variance than that of 200-bus system since 500-bus system has a larger range.

This information is also validated from Fig. 4 in the manuscript when $\eta_1 \in \{0.05, 0.1, 0.15\}$. However, when $\eta_2 = 0.2$, we find that the 500-bus system has a higher accuracy mean and lower accuracy variance. After careful checking, we find the reason is that 5 is a small number for random sampling of PMU locations. Specifically, when $\eta_2 = 0.2$, there are 4 out of 5 tests in the 500-bus system to have PMU locations close to the event location. However, for the 200-bus system, there are only 2 tests when PMU locations are close to the event locations. This shows that we may meet the situation when the 500-bus system data brings a better performance.

We also utilize F1 score to evaluate the performance for 200- and 500-bus systems. F1 score is the harmonic mean of precision and recall metrics, which gives a much better evaluation for datasets of imbalanced classes than accuracy [37]. Under this setting, we re-evaluate Section VI-D using F1 score. Fig. 6c and 6c illustrate that our proposed KTDC-Se model still has the best performance in different scenarios under F1 score, which demonstrates the superiority of our model. Compared to test accuracy, the result of F1 score for all methods is relatively lower. This is because F1 score has an overall consideration of the precision and the recall rate. However, under the metric of F1 score, KDTC-Se is still the best method.

For the real-world data, we report the testing accuracy in Table II. We observe that the average accuracy promotions of KTDC-Se are 9.7%, 9%, and 8.8%, compared to MixMatch, FixMatch, and SSLN, respectively. This still supports the advantage of using an integrated model.

We further evaluate the accuracy for each event type for the 200-bus system with $\eta_1 = 0.1$. The results can be seen in Table III. Other scenarios can bring similar results. We denote LT to represent line trip, GT to represent generator trip, SP to represent single-phase fault, PP to represent phase-to-phase fault, and TP to represent three-phase fault. Next, the numbers 1 and 2 represent the two different locations. The result illustrates that three-phase faults are easier to identify since they are more severe than others. On the other hand, the single-phase and phase-to-phase faults have lower accuracy since they have similar fault behaviors.

D. Semi-supervised Learning Boosts KTDC-Se Model Performance

In this subsection, we compare KTDC-Se, Resnet, and KTDC-Se-L to illustrate the effectiveness of semi-supervised learning. Specifically, for synthetic data, the average accuracy promotions are 18% and 13.3%, compared to Resnet and KTDC-Se-L, respectively. For real-world data, the corresponding accuracy promotions are 15.2% and 11.8%. These results show that there is a significant improvement when using unlabeled data.

The performance can be explained as follows. First, we can compare KTDC-Se and KTDC-Se-L. When doing the tensor decomposition of the PMU tensors, the unlabeled tensors in KDTC-Se enable the core tensors to understand different loading conditions and maintain similarity for different conditions as long as the event label is the same. Then, the trained classifier can successfully tackle different operational scenarios. Second, we can compare KTDC-Se-L and Resnet. We find that Resnet has an even worse performance as Resnet also employs PCA to pre-process data. As illustrated in Section VI-C, the two-stage model performs worse.

To further understand how many labeled data are needed for a good performance of KTDC-Se, we utilize the simulated data to test and fix $\eta_1 = 0.1$ and vary $\eta_2 \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ for the number of labeled data. As a comparison, we also implement KTDC-Se-L for the labeled data. Further, we also present the optimal testing accuracy via employing labels for all the PMU tensors to train the model, i.e., $\eta_2 = 1$. The results of average testing accuracy are shown in Fig. 7a and 7b.

Based on the plots, we have the following observations. (1) Training with extra unlabeled data significantly improves the accuracy, which has been explained before. (2) KTDC-Se can efficiently utilize the limited labels. For example, when $\eta_2 = 0.1$, the testing accuracy is still higher than 94%. This implies that KTDC-Se can filter information and group data by event labels in the feature space. Such a good result comes from the joint learning process to generate compact and discriminative features. (3) As η_2 increases, the testing accuracy of KTDC-Se gradually increases up to the optimal accuracy. Further, with only 30% of the labeled data, KTDC-Se can bring an accuracy almost close to the optimal one. This again indicates the high efficiency of KTDC-Se to utilize limited labels.

E. Tensor-based Framework Enables Fast Inference

The model efficiency can be evaluated via the training and testing time. We utilize the simulated data in Section VI-C and report the computational time of all methods on the testing dataset in Table IV. Based on the results, we have the following conclusions. (1) SSLN has the largest computational time due to its largest size with ladder networks, decoders, and encoders. (2) KTDC-Se has the second largest computational time since each KTDC-Se is a binary classifier and we utilize multiple binary models to create the final decision. Thus, the total training time for multiple models is higher. (3) The total training time of KTDC-Se is still comparable to other models. This is because training each binary model requires much smaller computations than other benchmark methods. More specifically, the input dataset for each KTDC-Se model only covers two event classes and has a much smaller size. (4) Resnet, MixMatch, and FixMatch have relatively close training times since they have the same input data processed via PCA and similar DNN models for classification. (5) KTDC-Se-L has the smallest training time since it only utilizes labeled data.

We further find that KTDC-Se has the second lower testing time, which demonstrates its efficiency. Further, KTDC-Se-L has the lowest time because it uses less data (i.e., only labeled data) for training. Thus, the test kernel matrix has a much smaller size compared to that in KTDC-Se. According to the prediction function in Equation (18), KTDC-Se-L has a lower testing time compared to KTDC-Se.

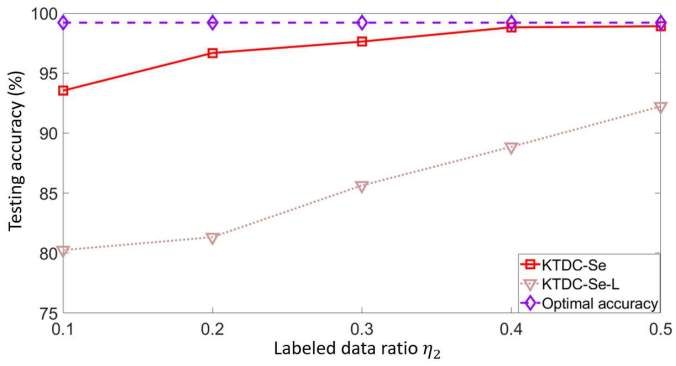
However, if we utilize both labeled and unlabeled data, KTDC-Se is much more efficient to do the inference than other methods. The reason is that KTDC-Se employs tensor decomposition to explore cross-dimension correlations and obtain a very compact core tensor for tests, but other methods utilize PCA to compress data, which needs more features to achieve the best accuracy. Secondly, other methods utilize a deep model to extract non-linear features, requiring more computational time. Thirdly, Resnet, FixMatch, and MixMatch

TABLE III: The testing accuracy (%) for each event type.

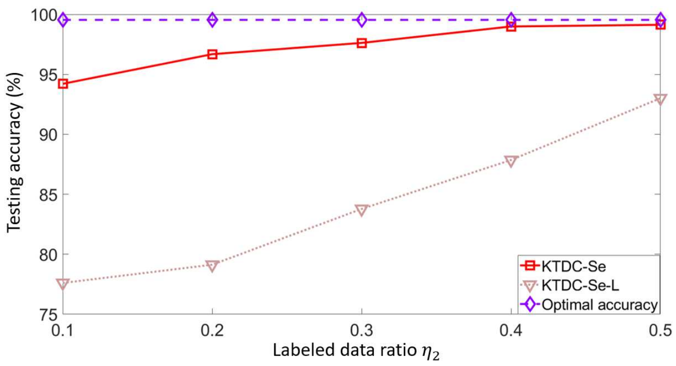
LABEL	KTDC-SE	RESNET	KTDC-SE-L	MIXMATCH	FIXMATCH	SSLN
LT1	96.8	81.5	86.3	84.5	85.2	92.6
LT2	97.2	81.7	87.2	84.2	84.9	93.1
GT1	98.3	82.3	87.4	85.1	85.4	94.2
GT2	97.8	83.2	86.8	84.0	84.9	92.5
SP1	95.7	80.6	84.6	82.5	83.3	89.8
SP2	96.1	81.0	84.8	82.7	83.6	91.3
PP1	96.5	81.5	85.2	83.1	84.5	90.6
PP2	97.2	81.2	85.4	82.6	85.1	91.5
TP1	98.5	85.1	88.7	86.2	87.2	92.6
TP2	99.3	85.7	87.9	87.4	87.7	93.0

TABLE IV: The average training/testing time (s) of the training/testing dataset for different methods.

	KTDC-SE	RESNET	KTDC-SE-L	MIXMATCH	FIXMATCH	SSLN
TRAINING TIME	359.6	325.7	45.8	333.6	345.4	431.3
TESTING TIME	1.3	3.8	0.4	3.8	3.7	6.4



(a) Sensitivity analysis for the 200-bus system.



(b) Sensitivity analysis for the 500-bus system.

Fig. 7: Results of the sensitivity analysis with respect to labeled data ratios.

share the same DNN architecture and consume close time for testing. On the other hand, the ladder network in SSLN has a larger size and needs more time to predict the label.

F. Non-linear Kernelization Largely Increases the Accuracy

In this subsection, we study the effectiveness of kernelization to the capacity of the classifier. Specifically, we report the testing accuracy for the 200-bus system when $\eta_1 = 0.1$ and $\eta_2 = 0.3$. Other results are similar and ignored due to the space limit. Then, we study the case with a constant kernel (e.g., $d = 1$ for the polynomial kernel) and cases with different non-linear kernels. When kernel is a polynomial function, we vary $d \in \{1, 2, 3\}$. When kernel is a RBF function, we vary $\lambda \in \{0.05, 0.1, 0.15\}$.

The results are shown in Table V. First, if we compare the constant kernels with other kernels, we find that adding non-linear kernels can significantly increase the accuracy. This is because the PMU measurements have high non-linear correlations. Secondly, the polynomial kernel can lead to an accuracy 95.5% when $d \geq 2$. For the RBF kernel, the accuracy is around 94.7% when $\lambda \geq 0.1$. This shows that the polynomial kernel, especially when $d = 2$, is better than others. The partial reason is that the quadratic correlations largely lie in the power flow equations.

TABLE V: The testing accuracy (%) (mean $\pm$ standard deviation) for different kernels.

KERNEL NAME	d OR λ	ACCURACY
POLYNOMIAL	1	89.22 ± 0.6
POLYNOMIAL	2	$\mathbf{95.7 \pm 0.3}$
POLYNOMIAL	3	95.4 ± 0.2
RBF	0.05	93.3 ± 0.6
RBF	0.1	94.7 ± 0.3
RBF	0.15	94.6 ± 0.5

VII. CONCLUSION

The increasing placement of PMUs leads to better power system situational awareness and event identification. Specifically, the ML-based methods can fast identify the event types and locations. However, the high volume, complex correlations of PMU measurements cause inefficiency to existing ML methods. Secondly, recent approaches focus on supervised learning while many event data are unlabeled. The inability to utilize rich unlabeled data hurt the model robustness to diversified loading conditions. To tackle these challenges, we propose our Kernelized Tensor Decomposition and Classification with Semi-supervision (KTDC-Se). Specifically, we treat PMU measurements as tensors and employ an advanced tensor decomposition to remove redundant information and save useful event features, significantly boosting model efficiency. Simultaneously, labeled data in the core tensors are input to a classification model with kernels for accurate classification, and unlabeled data contribute to the decomposition process. This guarantees the model robustness and accuracy. Numerically, we show that KTDC-Se achieves the best accuracy compared to other semi-supervised learning methods in both synthetic systems and real-world systems.

References

[1] Y. Yuan, Z. Wang, and Y. Wang, "Learning Latent Interactions for Event Identification via Graph Neural Networks and PMU Data," *arXiv preprint arXiv:2010.01616*, 2020.

[2] H. Li, Y. Weng, Y. Liao, B. Keel, and K. E. Brown, "Distribution grid impedance & topology estimation with limited or no micro-pmus," *International Journal of Electrical Power & Energy Systems*, 2021.

[3] 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 *IEEE Power & Energy Society General Meeting*, 2019.

[4] D.-I. Kim, T. Y. Chun, S.-H. Yoon, G. Lee, and Y.-J. Shin, "Wavelet-based event detection method using PMU data," *IEEE Transactions on Smart Grid*, 2015.

[5] E. Pérez and J. Barros, "A proposal for on-line detection and classification of voltage events in power systems," *IEEE Transactions on Power Delivery*, 2008.

[6] M. Cui, J. Wang, J. Tan, A. Florita, and Y. Zhang, "A novel event detection method using PMU data with high precision," *IEEE Transactions on Power Systems*, 2018.

[7] H. Li, Y. Weng, E. Farantatos, and M. Patel, "A hybrid machine learning framework for enhancing PMU-based event identification with limited labels," in *IEEE International Conference on Smart Grid Synchronized Measurements and Analytics*, 2019.

[8] D. De Yong, S. Bhowmik, and F. Magnago, "An effective power quality classifier using wavelet transform and support vector machines," *Expert Systems with Applications*, 2015.

[9] 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*, 2021.

[10] S. Zhang, Y. Wang, M. Liu, and Z. Bao, "Data-Based line trip fault prediction in power systems using LSTM networks and SVM," *IEEE Access*, 2018.

[11] L. He, C.-T. Lu, G. Ma, S. Wang, L. Shen, S. Y. Philip, and A. B. Ragin, "Kernelized support tensor machines," in *International Conference on Machine Learning*, 2017.

[12] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, 2017.

[13] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[14] K. Nordhausen and H. Oja, "Independent component analysis: A statistical perspective," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 10, no. 5, p. e1440, 2018.

[15] R. Ma, S. Basumallik, and S. Eftekharnejad, "A PMU-based data-driven approach for classifying power system events considering cyberattacks," *IEEE Systems Journal*, 2020.

[16] D.-I. Kim, "Complementary feature extractions for event identification in power systems using multi-channel convolutional neural network," *Energies*, vol. 14, no. 15, 2021.

[17] S. Brahma, R. Kavasseri, H. Cao, N. R. Chaudhuri, T. Alexopoulos, and Y. Cui, "Real-time identification of dynamic events in power systems using PMU data, and potential applications, models, promises, and challenges," *IEEE Transactions on Power Delivery*, 2017.

[18] H. Li, Z. Ma, and Y. Weng, "A transfer learning framework for power system event identification," *IEEE Transactions on Power Systems*, pp. 1–1, 2022.

[19] H. Li, Y. Weng, and H. Tong, "Heterogeneous Transfer Learning on Power Systems: A Merged Multi-modal Gaussian Graphical Model," in *IEEE International Conference on Data Mining*, 2020.

[20] T. Kolda and B. Bader, "Tensor decompositions and applications," *Journal of Applied Mathematics*, 2009.

[21] E. F. Lock, "Tensor-on-tensor regression," *Journal of Computational and Graphical Statistics*, 2018.

[22] M. Liao, D. Shi, Z. Yu, Z. Yi, Z. Wang, and Y. Xiang, "An alternating direction method of multipliers based approach for PMU data recovery," *IEEE Transactions on Smart Grid*, 2018.

[23] J. Shi, B. Foggo, and N. Yu, "Power system event identification based on deep neural network with information loading," *IEEE Transactions on Power Systems*, 2021.

[24] V. D. Sánchez A, "Advanced support vector machines and kernel methods," *Neurocomputing*, vol. 55, no. 1-2, pp. 5–20, 2003.

[25] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, 2002.

[26] H. Li, H. Tong, and Y. Weng, "Domain adaptation in physical systems via graph kernel," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 868–876.

[27] S. Y. Kung, *Kernel methods and machine learning*. Cambridge University Press, 2014.

[28] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International conference on computational learning theory*. Springer, 2001.

[29] B. Cao, C.-T. Lu, X. Wei, S. Y. Philip, and A. D. Leow, "Semi-supervised tensor factorization for brain network analysis," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016.

[30] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2015.

[31] T. Zhao, M. Yu, Y. Wang, R. Arora, and H. Liu, "Accelerated mini-batch randomized block coordinate descent method," *Advances in Neural Information Processing Systems*, 2014.

[32] General Electric Energy Consulting, "General Electric Concorda PSLF," 2018. [Online]. Available: https://www.geenergyconsulting.com/practice-area/software-products/pslf

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[34] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in Neural Information Processing Systems*, 2019.

[35] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in Neural Information Processing Systems*, 2020.

[36] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," *arXiv*, 2015.

[37] Joos Korstanje, "The F1 score," 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6