Intrinsic and Extrinsic Knowledge Transfer for Robust Data-Driven Event Identification

by

Zhihao Ma

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2023 by the
Graduate Supervisory Committee:

Yang Weng, Chair
Hongbin Yu
Meng Wu
Amarsagar Reddy Ramapuram Matavalam

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

The integration of Distributed Energy Resources (DER), including wind energy and photovoltaic (PV) panels, into power systems, increases the potential for events that could lead to outages and cascading failures. This risk is heightened by the limited dynamic information in energy grid datasets, primarily due to sparse Phasor Measurement Units (PMUs) placement. This data quality issue underscores the need for effective methodologies to manage these challenges.

One significant challenge is the data gaps in low-resolution (LR) data from RTU and smart meters, hindering robust machine learning (ML) applications. To address this, a systematic approach involves preparing data effectively and designing efficient event detection methods, utilizing both intrinsic physics and extrinsic correlations from power systems.

The process begins by interpolating LR data using high-resolution (HR) data, aiming to create virtual PMUs for improved grid management. Current interpolation methods often overlook extrinsic spatial-temporal correlations and intrinsic governing equations like Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs). Physics-Informed Neural Networks (PINNs) are used for this purpose, though they face challenges with limited LR samples. The solution involves exploring the embedding space governed by ODEs/DAEs, generating extrinsic correlations for initial LR data imputation, and enforcing intrinsic physical constraints for refinement.

After data preparation, event data dimensions such as spatial, temporal, and measurement categories are recovered in a tensor. To prevent overfitting, common in traditional ML methods, tensor decomposition is used. This technique merges intrinsic and physical information across dimensions, yielding informative and compact feature vectors for efficient feature extraction and learning in event detection.

Lastly, in grids with insufficient data, knowledge transfer from grids with similar event patterns is a viable solution. This involves optimizing projected and transferred vectors from tensor decomposition to maximize common knowledge utilization across grids. This strategy identifies common features, enhancing the robustness and efficiency of ML event detection models, even in scenarios with limited event data.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my adviser, Dr. Yang Weng, for his invaluable guidance and support during my academic journey. I had the opportunity to explore cutting-edge Machine Learning Techniques in physical systems under his guidance. His invaluable guidance, consistent encouragement, and deep insights have been instrumental in the fruition of this work.

Heartfelt thanks to my committee members: Dr. Amarsagar Reddy Ramapuram Matavalam, Dr. Hongbin Yu, and Dr. Meng Wu. Their insightful feedback and knack for directing me to the right resources were pivotal to the successful completion of this project. I'm equally indebted to the esteemed faculty of Arizona State University. Their teachings have been foundational, shaping my research capabilities.

Finally, I would like to express my profound appreciation to my spouse, Mrs. Mengyu Liu, as well as my parents, Mr. Jianhong Ma and Mrs. Songqing Mo, for their invaluable support and contributions. The steadfast belief and support they have provided have served as a constant source of stability and motivation along the course of our endeavor.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

GENERAL INTRODUCTION

Modern physical systems are continuously expanding to serve society more effectively. For example, the Internet of Everything (IoE) is integrating devices into an intelligent web at an unparalleled rate. Also, with more Distributed Energy Resources (DER) components added to the system. This increase poses a heightened risk of event surges, outages, and cascading failures unless there are robust methodologies and accurate system knowledge in place. Practical challenges, such as untimely system updates, limited sensor coverage, sensor resolution differences, and bad sensor data, often impede achieving comprehensive situational awareness. Additionally, as power systems pivot towards uncertain energy sources like Photovoltaic (PV) and wind power and grapple with unpredictable elements like Electric Vehicle (EV) charging, the demand for advanced tools, like Phasor Measurement Units (PMUs), grows. Yet, existing machine learning (ML) techniques often struggle with the intricacies of these data streams. Given the ever-changing nature of power systems, it's imperative to develop frameworks that ensure high situation awareness. This framework should incorporate both intrinsic and extrinsic data interpolation and feature extraction, enabling efficient tensor learning event detection and leveraging the full potential of both intrinsic and extrinsic power system attributes.

The primary objective of this thesis is to devise a robust event detection method capable of functioning effectively amidst compromised data quality. A crucial aspect of navigating these challenges lies in the harmonious integration of both intrinsic and extrinsic learning.

Building upon this foundational understanding, addressing extrinsic data quality becomes pivotal. The study leverages data from PMU and SCADA sensors to reconstruct missing dynamic states within the power generation system using a semi-supervised learning framework. This endeavor faces two predominant challenges based on the distinctive data properties. Firstly, the diverse locations of PMU and SCADA sensors necessitate approximating data correlations to estimate missing data precisely. Secondly, the paucity of dynamic transitions in SCADA samples underscores the imperative need for efficient SCADA data utilization. To tackle the spatial challenge, Deep Neural Networks (DNNs) with enhanced capacities are deployed to encapsulate spatial-temporal information, predicting

dynamic states. For the temporal hurdle, a novel mechanism is designed for efficient SCADA data use. Specifically, by iteratively repurposing the predicted dynamic states within SCADA data to refine the DNN model, performance gradually escalates. The potency of this proposed training technique garners validation within the framework of Expectation-Maximization (EM).

Having established the extrinsic framework, the next logical progression is to explore the intrinsic realm. After external validation by discerning the spatial-temporal behaviors across diverse sensors, there emerges a palpable potential in harnessing intrinsic data relations. This would both trim computational expenses and amplify interpolation precision. Hence, an innovative interpolation method is introduced, aiming to transmute Low-Resolution (LR) meters into "virtual" PMUs. By tapping into the intrinsic physics-informed connections defined by Differential Algebraic Equations (DAEs) in the power system and leveraging high-resolution data from accessible PMUs, this technique algorithmically fills in gaps from LR meters. This strategy not only optimizes data utilization but also ushers in a cost-effective yet potent monitoring and control approach for power systems. Such a system promises informed decisions, ensuring robust system stability and resilience without the financial strains of conventional installations.

Transitioning from data optimization, the focus shifts to the potency of robust event detection systems, which often lie in the data's underlying patterns and knowledge. The burgeoning advancements in ML and Data Mining (DM), armed with proficient learning and rapid inference capabilities, provide an ideal platform to distill this knowledge. Consequently, this thesis introduces advanced learning-based models tailored for this very purpose.

Yet, a new dimension of complexity emerges when dealing with newer grids where data scarcity hinders ML model-based event detection. The solution rests in knowledge transfer from a data-abundant grid (source) to a data-deficient one (target). This study utilizes Transfer Learning (TL) as the conduit for this knowledge migration. In this context, a comprehensive TL framework is proposed for power system transfer learning, grounded in robust theoretical underpinnings and expansive generalizability.

To round off the discussion, impeccable data quality is paramount, especially when considering the intricacies associated with spatial and temporal aspects and sensors with varied resolutions. Addressing these concerns, this research introduces an innovative approach to enhance data quality both extrinsically and intrinsically. Armed with this refined data, the subsequent focus shifts to how

the proposed intrinsic and extrinsic techniques function in event detection, ensuring they operate at their zenith.

1.1 Research Objectives

1. **Data Interpolation: Semi-Supervised Learning for Extrinsic Data Correlation:** Propose techniques to improve data quality extrinsically by merging PMU and SCADA sensor data. Address spatial-temporal challenges by leveraging DNNs and iteratively refining the model using SCADA data. This objective seeks to predict and recover missing dynamic states in the power generation system, especially in the context of differing sensor resolutions and locations.
2. **Data Interpolation: Intrinsic Physics-Informed Approach to Enhance Data Quality:** Design a novel method to leverage intrinsic correlation among different nodes, transforming Low-Resolution meters into "virtual PMUs". By utilizing inherent physics-informed relationships and high-resolution data, this objective aims to algorithmically interpolate missing values algorithmically, ensuring a cost-effective yet high-quality power system monitoring solution.
3. **Event Detection: Exploring Intrinsic Data Structures with Tensor Learning:** Develop cutting-edge ML models to intrinsically identify and classify events within power systems by analyzing underlying patterns and system data. Ensure that these models are equipped with the capabilities of fast inference and efficient learning, with a focus on differentiation and localization of event types.
4. **Event Detection: Extrinsic Knowledge Enhancement through Transfer Learning:** Establish a framework for robust event detection in grids with limited data. The primary approach involves utilizing TL to migrate knowledge from a data-rich grid to a data-scarce one. The framework should address challenges like data space homogeneity/heterogeneity and ensure label space consistency even in scenarios of data space disjointing.

1.2 Organization

The organization of this report is as follows:

1. Chapter 1 provides an overview of the research, including the motivation, objectives, and organization of the study. It highlights the need for robust event detection methods with bad data quality in power systems and introduces the concepts of intrinsic and extrinsic learning.
2. Chapter 2 conducts a survey on improving data quality extrinsically by merging PMU and SCADA sensor data. The chapter addresses spatial-temporal challenges by leveraging DNNs and iteratively refining the model using SCADA data under a semi-supervised learning framework. The goal is to predict and recover missing dynamic states in the power generation system, especially in the context of differing sensor resolutions and locations. The chapter proposes data interpolation and fusion techniques, utilizing DNNs to capture complex relationships between the sensor data. The effectiveness of the proposed methods is demonstrated through extensive numerical validations.
3. Chapter 3 surveys on leveraging intrinsic properties to interpolate and enhance data in power systems. The chapter focuses on transforming low-resolution meters into "virtual" PMUs by utilizing inherent physics-informed relationships and high-resolution data. The goal is to algorithmically interpolate missing values algorithmically, ensuring a cost-effective yet high-quality power system monitoring solution. The chapter presents the design of the interpolation method and discusses its advantages in terms of accuracy and efficiency. The proposed method is validated through numerical experiments, demonstrating its effectiveness in enhancing data quality and providing reliable power system monitoring.
4. Chapter 4 focuses on developing advanced ML models to intrinsically identify and classify events within power systems. The chapter begins with a literature review of equation learning and explores the maintenance of physical consistency in learning models. It discusses the use of convex optimization techniques to ensure physical consistency in fully-observable and partially-observable systems. The chapter also presents comprehensive experiments on diversified systems to illustrate the outstanding performance of the proposed models. The results demonstrate the effectiveness of the developed models in analyzing system data, recognizing underlying patterns, and achieving high performance in event detection.

5. Chapter 5 proposes an extrinsic framework for robust event detection using TL with limited data. The chapter explores the use of TL to migrate knowledge from a data-rich grid to a data-limited grid. It discusses two frameworks: one for supervised learning and another for unsupervised learning. The supervised learning framework utilizes an ensemble model to aggregate different supervised learning models with different biases. The unsupervised learning framework leverages physical properties to provide labels in scenarios without label information. Additionally, the chapter introduces a tensor-learning model that converts large-scale tensors to compact and informative feature vectors for fast event identification. The proposed methods are extensively validated through numerical experiments, demonstrating their high performance in event detection.
6. Chapter 6 presents the conclusions and contributions of the research. Then, Chapter 6 discusses the future directions of the research work.

Chapter 2

DATA INTERPOLATION: LEVERAGING SEMI-SUPERVISED LEARNING FOR EXTRINSIC DATA CORRELATION

2.1 Introduction

A power-generation system integrates stochastic components and loads to facilitate clean and low-cost production and consumption of power, such as Photovoltaic (PV) power and wind power that highly depend on uncertain weather conditions. These components make the power system vulnerable to dynamic environmental events. To better capture system dynamics, Phasor Measurement Units (PMUs) are continually integrated into the power system to provide synchronized phasor measurements with 30-120 samples per second (General Electric Energy Consulting 2018; "IEEE Standard for Synchrophasor Measurements for Power Systems" 2011). This high resolution enables accurate monitoring of the dynamic states.

However, installing a single PMU is relatively expensive and costs around several thousands of dollars. According to the US Department of Energy (Young and Silverstein 2014), the cost of a single PMU in the United States ranges from $40,000 to $180,000. Meanwhile, the conventional measuring system of power systems, i.e., Remote Terminal Unit (RTU) based SCADA system, has been largely deployed over the last decade (Simões Costa, Albuquerque, and Bez 2013). As a comparison, the installation cost for an RTU sensor ranges from $200 to $5,000(Sayed and Gabbar 2017). In addition, the expense of installing PMUs is not restricted to the equipment alone. Other costs include communication infrastructure, data management systems, and cybersecurity precautions to assure the security and dependability of the obtained data. These expenses may rapidly accumulate, making it difficult for utilities to adopt PMUs on a large scale (Madani et al. 2011; Basetti and Chandel 2016). Compared to PMUs, SCADA measurements have relatively low resolution and can't fully record system dynamics. For example, the SCADA system provides only 0.5-2 measurement samples per second (Ghosal and Rao 2015).

Therefore, numerous studies focus on optimal PMU placement to balance the PMU costs and the benefits from PMUs (e.g., system observability). For example, (Baldwin et al. 1993) uses a

dual search algorithm to achieve observability. (Aminifar, Fotuhi-Firuzabad, and Safdarian 2013) presents a way to quantify the benefits associated with the development of a Wide-Area Measurement System (WAMS) to reach an optimal PMU placement solution. However, in our chapter, I propose a brand-new perspective to interpolate missing dynamic information of SCADA measurements based on a few PMUs. Hence, with limited computational costs, SCADA sensors can be converted to "virtual PMUs" with the capacity to infer the intermediate missing values. Moreover, our DNN in Hd-Deep-EM can achieve real-time inference for the intermediate data since the forward computations of DNN, i.e., the DNN prediction, are very fast. The goal is achieved by combining the information of SCADA data and PMU measurements to accurately estimate the missing dynamic states between SCADA data, under the domain of Dynamic State Estimation (DSE).

DSE is an important topic in power systems with the growing uncertainties in the system from both the generation and the demand sides (Shih and Huang 2002). DSE is challenging due to the complex dynamic transitions of system states. Existing methods can be categorized into the following groups. The first group utilizes PMU measurements and full system physical equations to estimate dynamic states. The process includes prediction to forecast the future system states and filtering to remove bad data and decrease errors based on current measurements, i.e., the so-called Kalman Filter (KF) (Chang, Taranto, and Chow 1997; Scholtz 2004; Dorfler, Pasqualetti, and Bullo 2013). KF-based methods require strict assumptions, e.g., lossless network with voltage magnitude to be fixed at 1 p.u. To make the methods more appropriate for real-world systems, Extended Kalman Filter (EKF) (Ghahremani and Kamwa 2011a; Netto, Zhao, and Mili 2016; Zhao, Netto, and Mili 2016) and Unscented Kalman Filter (UKF) (Ghahremani and Kamwa 2011b; Valverde and Terzija 2011) are employed. However, these methods still require the system dynamic model.

Therefore, the second group of methods utilizes PMU data without system parameters. Specifically, they introduce Koopman operator-based KF (KKF), a data-driven approach with Maximum Likelihood Estimation (MLE) to estimate the system transition and observation matrices (Surana 2016, 2020; Netto and Mili 2018). In particular, Koopman operator can convert nonlinear dynamics to a proper linear approximation, making the computation easy and enabling the embedded MLE process. However, these methods still require certain statistical assumptions on the state distributions (e.g., Student's t-distribution). This may not be generalizable to diversified system conditions with

the increasing penetration of renewable energy. Moreover, they still demand a certain number of PMUs to achieve system-level observability.

Then, the third group of work assumes limited PMUs with a large number of SCADA sensors across the system. Then, they try to estimate the missing values of SCADA data via methods like information fusion (Qu, Wang, and Shen 2021), Bayesian methods (Camões et al. 2022), and regression methods (Song and Shepperd 2007; Batista and Monard 2003; Le and Benjapolakul 2018). These methods, however, are hard to capture complex spatial-temporal correlations of power system datasets. Additionally, they typically suffer overfitting issues since the SCADA data contain limited dynamic information.

In this chapter, I focus on the Deep Learning (DL) techniques that can effectively capture the complex spatial-temporal correlation of PMU and SCADA data. These methods utilize hierarchical feature learning to extract features spatially or temporally. To capture spatial correlations, methods like Convolutional Neural Networks (CNNs)(Plathottam, Salehfar, and Ranganathan 2017; Gupta, Gurrala, and Sastry 2018; C. Kim et al. 2019) and Graph Neural Network (GNN) can employ square or graph-based kernels for convolution and extraction of local features. To obtain temporal correlations, Recursive Neural Networks (RNNs)(Vermaak and Botha 1998; Ayad et al. 2018) and Long Short-Term Memory (LSTM) (Graves et al. 2008) employ gates to filter sequential information and learn useful patterns. In this chapter, I make use of the capacity of DNN to analyze spatial-temporal data for dynamic state estimation. However, DNN-based methods require relatively large datasets with diverse information. DNNs may overfit when the information is limited to uncover the dynamic states. Notably, such a scenario often happens for SCADA data due to the low resolution. Thus, it's quite challenging to build a good DNN model to predict the dynamic states of SCADA systems.

For the overfitting issues, I claim that they always exist no matter how I improve the DNN model. Specifically, the overfitting is due to the limited samples in the low-resolution RTU sensors due to the different sampling rates. If I only use the limited RTU data as output data in the training dataset, most of the dynamic information is missing. *Therefore, DNN tends to overly fit the RTU samples without the motivation to explore how to estimate intermediate states.*

This chapter addresses the system problem by explicitly introducing "more" training data with dynamic information. Specifically, the new training data is derived from the predicted hidden states

and the corresponding input PMU data, containing rich dynamic information. Subsequently, the DNN can be trained with all the data to further improve the prediction. These two steps form an iterative way of improving the neural network model's performance. To elaborate on the effectiveness of the procedure, I show that the iterative process is essentially an Expectation-Maximization (EM) method with good convergence performance. Thus, I named our method Heterogeneous data Deep EM (Hd-Deep-EM).

EM algorithms impact numerous topics in power systems. They mostly target probabilistic load or power source monitoring(Pankratov et al. 2019; Xiang, Wang, and Wang 2019; Ganjavi et al. 2017), where EM helps to estimate the underlying distributions with the maximal likelihood. For example, (Pankratov et al. 2019; Xiang, Wang, and Wang 2019; Ganjavi et al. 2017) develop a Gaussian Mixture Model (GMM) to model the density function of the load/source, and utilize EM algorithm to find a suboptimal solution. Some other applications appear in PMU-based cyber attack detection (Lee and Kundur 2014) and joint impedance and topology estimation (Yu, Weng, and Rajagopal 2017). For example, in (Lee and Kundur 2014), the objective becomes the log-likelihood of data with underlying cyber attacks.

There are also other studies to combine deep learning and EM algorithms for different domains. Specifically, in signal processing, (Y. Zhang et al. 2022) develops a generalized expectation maximization to estimate channel states and detect signals. (Shao et al. 2022) detects signals by leveraging the deep unfolding to represent the iterative EM algorithm and improve the detection accuracy. In computer vision, (Jiang et al. 2014) utilizes online EM algorithm to improve the inference in the deep Bayesian network, achieving good image classification performances. (Jiang et al. 2014) reconstructs images via EM network and employs the power of EM algorithm to tackle noises. In general, those methods can hardly be applied to power system PMU and SCADA data due to different resolutions. They also don't provide solid theoretical guarantees. In this chapter, I theoretically and experimentally clarify the effectiveness and efficiency of Hd-Deep-EM for dynamic hidden state estimation.

For the numerical verification, the Hd-Deep-EM method is tested extensively at various conditions on the synthetic 200- and 500-bus systems datasets, where the loading conditions and PMU/SCADA sensor penetrations are diversified for extensive testing. To show the power of Hd-Deep-EM in facing the spatial and temporal difference, I compare the proposed Hd-Deep-EM method with shallow EM

algorithm and EM algorithm with a Vanilla neural network. To show the power of Hd-Deep-EM in tackling the temporal trend from the SCADA measurements, I compare Hd-Deep-EM with the traditional neural network. The benchmark methods include EM and deep learning approaches, and Mean Square Error (MSE) is used for evaluating interpolation model accuracy. The results show that our proposed Hd-Deep-EM method can efficiently interpolate missing data to the SCADA measurement.

To summarize, I have the following contributions. (1) I formulate a data-driven problem to utilize synchrophasors from limited PMUs and widespread RTU sensor measurements to estimate the RTU data's dynamic states and interpolate the missing values. The goal is promising since I can utilize limited PMUs to boost the data quality of all RTU sensors. Additionally, the idea is practical due to the high data correlations within a system. (2) I propose our Hd-Deep-EM to solve the problem. Hd-Deep-EM has 3 strong properties to achieve high performance. First, Hd-Deep-EM employs Deep Neural Networks (DNNs) to approximate the correlation from PMUs to RTU sensors. While PMUs and RTU sensors locate at different places, the universal approximation power of DNNs enables Hd-Deep-EM to capture such correlations. Second, I utilize a window of PMU data to predict one sample of RTU sensors at a time slot. Therefore, the DNN in the Hd-Deep-EM can capture the temporal trends for the prediction. This also enables the prediction to be robust to noise, bad data, and the loss of synchronization. Third, the training of the DNN easily faces overfitting issues due to the limited data in RTU sensors with little dynamic information. Therefore, I propose an iterative process to utilize the predicted data in the RTU sensors to re-train the DNN, following an Expectation-Maximization (EM) manner. Thus, the strong theoretical guarantees from the EM algorithm support the convergence of Hd-Deep-EM. (3) I conduct comprehensive numerical validations on transmission grids and show Hd-Deep-EM has the best performance compared to other methods.

The rest of the chapter is organized as follows: Section 4.3 defines the problem. Section 4.4 proposes our Hd-Deep-EM. Section 4.6 conducts experiments for baselines and Hd-Deep-EM.

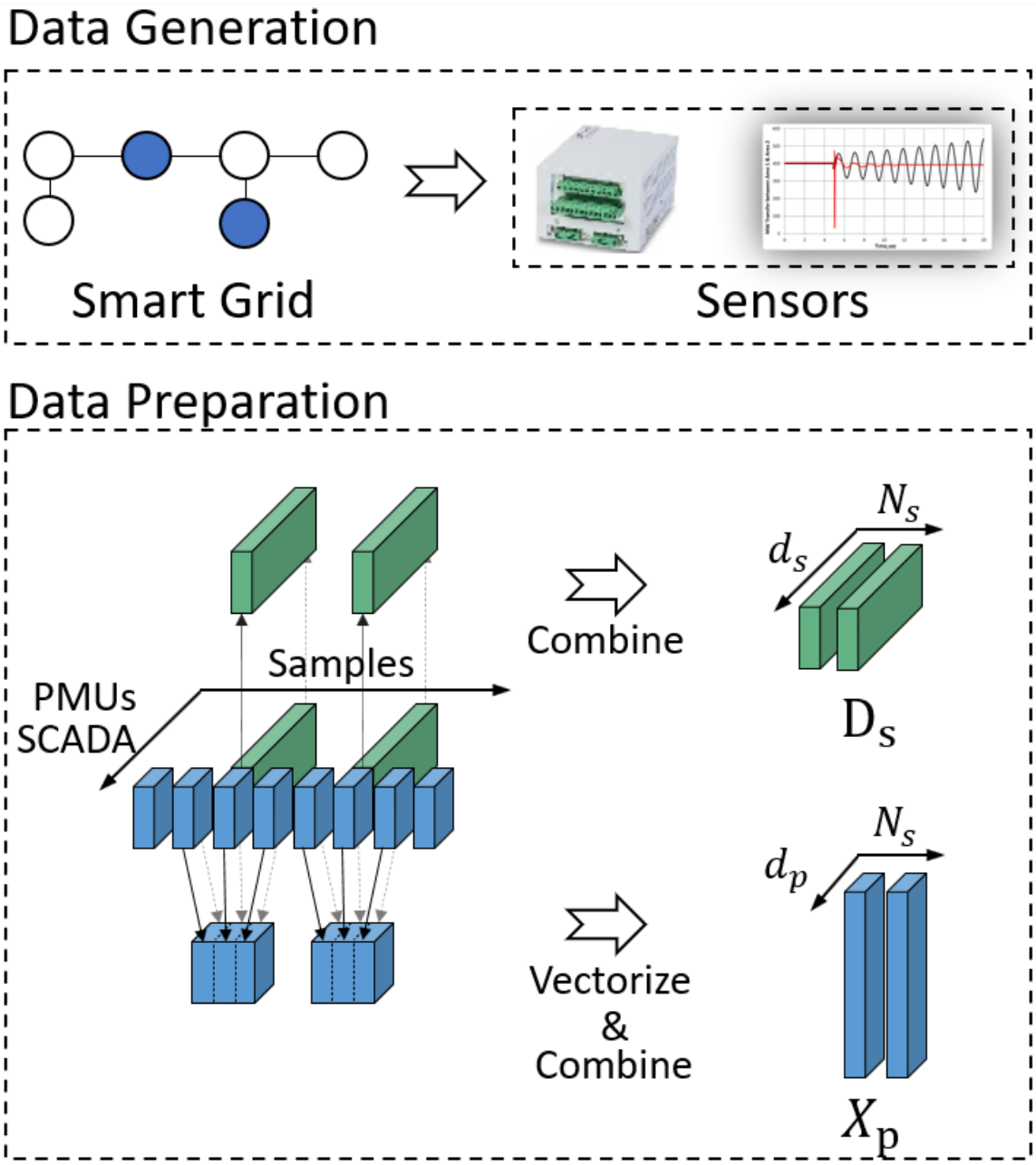


Figure 1. A Moving Window-based Segmentation Procedure to Process the PMU Streams.

2.2 Problem Formulation

I denote that N_p samples are for d_p numbers of PMUs and N_s samples are for d_s numbers of SCADA data. Then, let $\boldsymbol{D}_p \in \mathbb{R}^{N_p \times d_p}$ represent the PMU data matrix and $\boldsymbol{D}_s \in \mathbb{R}^{N_s \times d_s}$ for the

SCADA data. Moreover, due to the difference in resolution among PMUs and SCADA data, I have $N_p \gg N_s$. Then, I denote the missing data matrix as $\boldsymbol{Z} \in \mathbb{R}^{(N_p - N_s) \times d_s}$. In the transmission system, it's reasonable to assume the SCADA sensors can achieve the full system observability (Manousakis and Korres 2018), i.e., d_s is comparable to the system node number. Our contribution is to use few PMUs to interpolate the data for a large number of SCADA sensors, achieving the dynamic state estimation via the dynamic information from PMUs.

I also describe the measurement types of PMUs and RTUs here. PMUs are highly accurate devices that can measure the phasor quantities of voltage and current, such as magnitude, phase angle, and frequency, in real-time. PMUs can also measure the rate of change of these parameters, which is known as the frequency deviation or ROCOF (Rate of Change of Frequency) (Gharavi and Hu 2017). These measurements are used to monitor the stability of the power system and detect any disturbances or anomalies that may occur. RTUs, on the other hand, are used for remote monitoring and control of power system equipment such as transformers, switches, and breakers. They can measure a range of parameters, including voltage, current, power, and energy consumption (Thomas and McDonald 2017). RTUs can also provide information on the status of equipment, such as whether it is on or off and any faults or errors that may occur. In general, the above raw data can be input to the DNN model. Then, with nonlinear activations like Rectified Linear Unit (ReLU) (Glorot, Bordes, and Bengio 2011), the intermediate neurons after activations can represent the nonlinear features of the model. In Experiment, I mainly focus on the prediction of voltage magnitude as system states. However, other data have similar performance.

With the defined PMU and SCADA data flows, I need to process the raw data and formalize the training data to estimate the dynamic states. While one can build a one-to-one mapping from the PMU measurements to the SCADA data simultaneously, the mapping may not be robust if data on either side have outliers. Thus, to improve the robustness, I propose utilizing the PMU data of neighboring time slots as the input. Fig. 19 illustrates the process where I utilize a group of PMU data (i.e., the blue box) as input to estimate the SCADA sample (i.e., the green box). Naturally, if I input the intermediate PMU data that don't have the corresponding SCADA data, the prediction will be an estimate of the missing dynamic states of the SCADA data.

To predict the measurement states, mathematically, I employ a moving window to segment PMU streams based on the resolution of the SCADA data.

As shown in Fig. 19, for each timestamp of SCADA data, the PMUs data will extract a $k \times d_p$ moving window that has a center of the target timestamp. Subsequently, each extracted moving window will be vectorized into an input vector. This procedure makes the processed PMUs and SCADA data have the same number of samples N_s. Therefore, I have $\boldsymbol{X}_p \in \mathbb{R}^{N_s \times (k \times d_p)}$ as the input PMU data matrix and $\boldsymbol{D}_s \in \mathbb{R}^{N_s \times d_s}$ as the output SCADA data. In order to recover the $\boldsymbol{Z}$ with function f, I also apply the moving window to the intermediate PMU streams to predict the missing states $\boldsymbol{Z}$. Specifically, I denote the PMU data matrix for recovering $\boldsymbol{Z}$ as $\tilde{\boldsymbol{X}}_p \in \mathbb{R}^{(N_p - N_s) \times (k \times d_p)}$.

Finally, the strategy of this chapter is to learn the mapping rule

$$f : \boldsymbol{X}_p \cup \tilde{\boldsymbol{X}}_p \rightarrow \boldsymbol{D}_s \cup \boldsymbol{Z}. \tag{2.1}$$

The complete problem definition can be summarized as follows:

- Problem: Deep expectation maximization for data interpolation.
- Given: a set of PMU-based samples with $\boldsymbol{D}_p \in \mathbb{R}^{N_p \times d_p}$ as the high resolution data and $\boldsymbol{D}_s \in \mathbb{R}^{N_s \times d_s}$ as the low resolution data, respectively.
- Output: a regression model to learn data mapping f between PMU data and SCADA data and interpolate the missing value of the SCADA data.

2.3 Proposed Model

The design of the above mapping f can be diversified. However, existing works can not sufficiently capture the spatial-temporal correlations of PMU and SCADA data. In this section, I first illustrate that a well-designed DNN for the function f can efficiently handle the spatial-temporal correlations and estimate the dynamic states. However, the training of the DNN may suffer overfitting due to the scarce dynamic information in SCADA data. Then, I show the key is to resolve the issue is to completely make use of the hidden dynamic states to update the mapping. Such a hidden value estimation and parameter updating process lie in the domain of the Expectation-Maximization (EM) algorithm.

2.3.1 Expectation-Maximization Algorithm for Hidden State Estimation

EM algorithms are developed to tackle problems with latent variables, e.g., the hidden states in SCADA data streams. Specifically, the EM algorithm is an iterative process to separately estimate the statistics of latent variables and the corresponding parameters that determine the likelihood of the latent data. Thus, to apply EM algorithms to our problem, the first step is to identify what parameters should be used to determine the latent variables (i.e., the hidden states).

In general, one can categorize the parametric models into the generative and the discriminative models. The generative model estimates the joint distributions between the known and the unknown variables to generate new data. However, it's hard to choose an appropriate model to estimate hidden states. For example, GMM is usually used in most EM algorithms. However, the GMM model is not capable to represent the distribution of hidden states in power systems with complex spatial-temporal correlations.

Thus, in this chapter, I propose to utilize a discriminative mapping f to directly map from PMU data to SCADA data. The advantage of a discriminative design is to maintain the spatial-temporal correlations in the PMU data, thus boosting the approximation accuracy of the hidden states. Mathematically, I can write the EM steps as follows.

- E step: Estimate the expected values of the hidden states in SCADA data streams. Namely, input the corresponding PMU data to f and output hidden states.
- M step: Update parameters of f for better estimation.

Obviously, the design of f should be carefully considered. In the following section, I display our design using deep learning techniques to achieve accurate dynamic state estimation.

Fig. 2 visualizes the process of the EM algorithms. To investigate the convergence of the EM algorithm, many studies are done and one can refer to (Wu 1983) for more details.

2.3.2 Deep Neural Networks to Capture Spatial-Temporal Correlations

The core of our Hd-Deep-EM algorithm is the deep neural network (DNN) model, which is applied to approximate the data interpolation mapping between PMU to SCADA measurement. Intuitively,

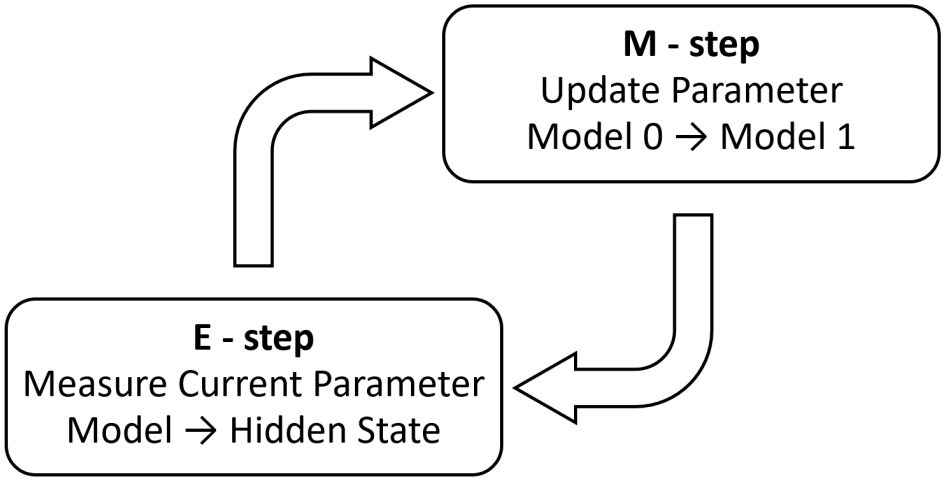


Figure 2. An Illustration of the EM Algorithm.

a DNN model can help capture the temporal-spatial correlations and learn useful features for the task. Specifically, the DNN model has multi-layer feed-forward neural network structure, which consists of typical three-level network architecture: one input layer, several hidden layers, and one output layer. Since the input data comes from PMU data $\boldsymbol{X}_p$ and the output data comes from the SCADA data $\boldsymbol{D}_s$, I denote $\boldsymbol{x}$ as the variable for $\boldsymbol{X}_p$, $\boldsymbol{y}$ as the truth variable for $\boldsymbol{D}_s$, and $\hat{\boldsymbol{y}}$ as the predicted variable using the DNN. Then, the DNN function can be written as follows.

$$\begin{aligned} \boldsymbol{h}_0 &= \boldsymbol{x}, \\ \boldsymbol{h}_i &= \boldsymbol{g}_i\left(\boldsymbol{W}_i \boldsymbol{h}_{i-1} + b_{i-1}\right), \forall\ i = 1, \cdots, N_l, \\ \hat{\boldsymbol{y}} &= \boldsymbol{g}_{N_l+1}\left(\boldsymbol{W}_{N_l+1} \boldsymbol{h}_{N_l} + b_{N_l}\right), \end{aligned} \tag{2.2}$$

where N_l denotes the number of layers for the neural network, $\boldsymbol{h}_0$ denotes the input matrix of the network, $\boldsymbol{h}_i$ is the output matrix of the i-th hidden layer, b_i is the bias term, and $\boldsymbol{g}$ is the activation function for each hidden layer. I elaborate more on the model of the DNN based on the following perspectives.

2.3.2.1 Structure and Activations

DNN has a hierarchical structure to gradually extract non-linear features for the task. Generally, the i-th hidden layer models the interactions between the $(i-1)$-th features by introducing a

connection weight matrix $\boldsymbol{W}_i$ and a bias vector b_i. Then, the result is activated via a non-linear activation function $g(\cdot)$. The choice of g can vary, and some common options include Rectified Linear Unit (ReLU), sigmoid function, and tangent function, etc. Notably, ReLU is the most commonly used because ReLU-based DNN can efficiently tackle the gradient vanishing problem and has a high efficiency to compute(Agarap 2019).

2.3.2.2 Loss function

After defining the structure and the inner activation functions, I need to train the DNN with a loss function. For this regression problem, the Mean Square Error (MSE) can smoothly measure the difference between the predicted output $\hat{\boldsymbol{y}}$ and the true output $\boldsymbol{y}$:

$$L(W, B, \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{|N_s|} \sum_{i \in N_s} (\hat{\boldsymbol{y}} - \boldsymbol{y})^2, \tag{2.3}$$

where $W = \{\boldsymbol{W}_i\}_i$ is the set of layer-wise weights of the DNN, and $B = \{b_i\}_i$ is the set of bias terms.

2.3.2.3 Training Process

With the defined loss function, training is conducted via minimizing the loss with given input/output data. The minimization can be written as:

$$\min_{W,B} L(W, B, \boldsymbol{x}, \boldsymbol{y}). \tag{2.4}$$

To solve the optimization, many algorithms can be utilized. For example, one can utilize either Stochastic Gradient Descent (SGD)(Bottou 2012) or the Adam(Kingma and Ba 2017) to train the model.

The defined DNN has the ability to learn the local features and determine the spatial-temporal correlations. To well-train the DNN model to approximate the dynamic states, I require the data to contain enough dynamic information. Nevertheless, the SCADA data has little dynamic information, decreasing the DNN model performance.

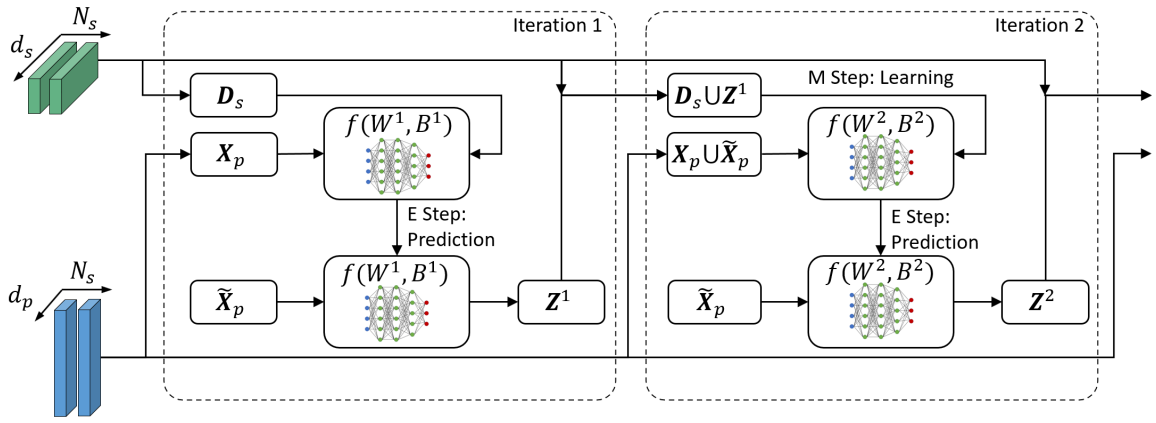


Figure 3. Illustration of the Computations for Proposed Hd-deep-em Method.

2.3.3 Deep Expectation Maximization to Boost the Training of the DNN

To resolve SCADA data scarcity, the key is to introduce more SCADA data with system dynamic information. As the dynamic systems states of the SCADA system are hidden, the problem can be decomposed into (1) estimating hidden values and (2) updating the model parameters as shown in Fig. 25. Mathematically, this iterative procedure is an Expectation-Maximization (EM) algorithm. This subsection derives the Hd-Deep-EM algorithm from training the DNN model, leading to the so-called Hd-Deep-EM model.

In general, the EM algorithm tries to solve an optimization with missing quantities. For the optimization to train the DNN in Equation (2.4), the hidden dynamic states of the SCADA data are introduced. Thus, the optimization can be rewritten as:

$$\min_{W,B} L(W, B, \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{y}), \tag{2.5}$$

where $\boldsymbol{z}$ represents the variable of missing values in the SCADA data. Namely, $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1}^{N_p - N_s}$. Clearly, $\boldsymbol{z}$ is also the output of the DNN f given the input PMU data $\boldsymbol{x}$ at the corresponding time. However, the unknown knowledge of $\boldsymbol{z}$ makes the direct optimization of Equation (2.5) impossible.

Therefore, the EM algorithm solves Equation (2.5) in an iterative manner. In the Expectation (E) step, the algorithm tries to approximate the expected values of $\boldsymbol{z}$ using the current model and data. Then, in the Maximization (M) step, the model tries to maximize the profit, i.e., minimize the loss, to obtain a better model. More specifically, for the k-th iteration, EM algorithm has the following formulations.

- E step: Estimate the missing values of SCADA data $\boldsymbol{z}^k$. Based on the definition of $\boldsymbol{z}$, the expected estimation of $\boldsymbol{z}^k$ can be written as:

$$\boldsymbol{z}^k = f(\boldsymbol{x}; W^k, B^k), \tag{2.6}$$

where W^k and B^k represent the parameters of the DNN at the k-th iteration. The input data $\boldsymbol{x}$ comes from the intermediate PMU matrix $\tilde{\boldsymbol{X}}_p$ that corresponds to the hidden timestamps of the SCADA system. Then, the estimated values of $\boldsymbol{z}^k$ can be added to train the $(k+1)$-th DNN model, which improves the training with more accurate dynamic information in $\boldsymbol{z}^k$. Thus, the training process is the M step, shown as follows.

- M step: Retrain the DNN model to minimize the loss:

$$(W^{k+1}, B^{k+1}) = \arg \min_{W^{k+1}, B^{k+1}} L(W^k, B^k, \boldsymbol{z}^k, \boldsymbol{x}, \boldsymbol{y}). \tag{2.7}$$

The obtained parameters W^{k+1} and B^{k+1} can formalize the $(k+1)$-th DNN model. Since $\boldsymbol{z}^k$ is known values, Equation (2.7) can be conveniently trained using SGD or Adam algorithms. All quantities of variable $\boldsymbol{z}^k$ can formulate the missing state matrix $\boldsymbol{Z}$. For different iterations, the values of $\boldsymbol{Z}$ can change. Thus, I also add the superscript and utilize $\boldsymbol{Z}^k$ to represent the predicted missing values in the k^{th} iteration, as shown in Fig. 25.

Finally, I summarize the complete algorithm for the Hd-Deep-EM model in Algorithm 1.

Algorithm 1 Hd-Deep-EM Algorithm.

Input: PMU data matrix $\boldsymbol{X}_p$, SCADA data matrix $\boldsymbol{D}_s$, PMU data matrix for recovering data $\tilde{\boldsymbol{X}}_p$.
Hyper-parameters: number of iterations K and DNN-related parameters like batch size and learning rate.
Output: Missing data matrix $\boldsymbol{Z}$ for the dynamic states.

Initial parameters of the DNN f. $k = 1$ to K E step: Use Equation (2.6) to estimate the k-th dynamic states. M step: Optimize Equation (2.7) to update parameters of the k-th DNN. =0

2.3.4 Double Expectation Maximization Algorithms for Noisy Data

For realistic datasets, noise may lower the performance of the state estimation. To tackle noise data, I modify our Hd-Deep-EM algorithm for better estimation. Specifically, I can assume the

noises are independent of measurements and can be easily modeled via a distribution model like GMM. Then, a natural idea is to decompose the model for estimating hidden states and noises. Since both data can be estimated via EM algorithms, I modify our Hd-Deep-EM and propose a Double Hd-Deep-EM algorithms for noisy data.

Specifically, in each iteration, I utilize the DNN model f to represent the mapping from input PMU data to the output. In the meantime, I propose to leverage another Gaussian model to estimate the noises, formulating the double EM algorithms (Fig 25). Mathematically, the double EM steps can be written as follows:

- E1 step: Estimate the missing values of SCADA data $\boldsymbol{z}^k$ using Equation (2.6).
- E2 step: Compute the noise data

$$\boldsymbol{n}^k = \boldsymbol{y} - f(\boldsymbol{x}; W^k, B^k). \tag{2.8}$$

- M1 step: Update the parameters of Gaussian model using $\boldsymbol{n}^k$. Then, obtain $\boldsymbol{n}^{k+1}$ using the Gaussian model.
- M2 step: Update parameters of DNN model f via:

$$\min_{W^{k+1}, B^{k+1}} L(W^k, B^k, \boldsymbol{z}^k, \boldsymbol{x}, \boldsymbol{y} - \boldsymbol{n}^{k+1}). \tag{2.9}$$

  Note that for known SCADA data, I utilize $\boldsymbol{y} - \boldsymbol{n}^{k+1}$ to take place of $\boldsymbol{y}$ to achieve the denoising.

In general, the algorithm can be summarized as Algorithm 2. I denote our algorithm as the Double Hd-Deep-EM algorithm.

Algorithm 2 Double Hd-deep-em Algorithm for Noisy Data.

Input: PMU data matrix $\boldsymbol{X}_p$, SCADA data matrix $\boldsymbol{D}_s$, PMU data matrix for recovering data $\tilde{\boldsymbol{X}}_p$.
Hyper-parameters: number of iterations K and DNN-related parameters like batch size and learning rate.
Output: Missing data matrix $\boldsymbol{Z}$ without noise for the dynamic states.

$k = 1$ to K Initial parameters of the DNN f and the Gaussian model. E step: Use Equations (2.6) and (2.8) to estimate the k-th dynamic states and the noise. M step: Utilize the noise data to update the Gaussian model. Then, optimize Equation (2.9) to update parameters of the k-th DNN. =0

2.3.5 Theoretical Support of Double Hd-Deep-EM Algorithm for Denoising

In the Double Hd-Deep-EM algorithm, I estimate parameters for both the DNN model and the underlying distribution of noises, which requires certain guarantees for convergence to the true noise distribution. The EM algorithm can provide the convergence theorem which states the decreasing distance between the true likelihood of the noise distribution and the estimated likelihood using the data in Equation (2.8). Specifically, I modify the convergence theorem from (Wu 1983) and propose the following theorem:

Theorem 1. Let θ denote the parameters of the noise distribution. Then, let $\{\theta^t\}$ denote the parameter sequence in the EM iteration and t represents the iteration index. If θ^* is a limit point of $\{\theta^t\}$, then (1) θ^* is a stationary point of the likelihood function of the noise data in Equation (2.8), and (2) the sequence $\{l(\theta^t)\}$ is non-decreasing and converges to $l(\theta^*)$, where $l(\theta)$ represents the likelihood of the noise distribution.

The proofs of Theorem 1 can be found in (Wu 1983). Basically, Theorem 1 justifies the convergent dynamics of the EM algorithm. More specifically, Theorem 1 proves that the convergent point θ exists for the parameters of the true distribution of the noises. Then, Theorem 1 illustrates the convergence to θ using the EM algorithm.

The convergence is further supported by numerical validations where I investigate the proposed Hd-Deep-EM algorithm and the Double Hd-Deep-EM for noiseless and noisy data, respectively. The results demonstrate that our method has excellent denoising capacities under different noise levels.

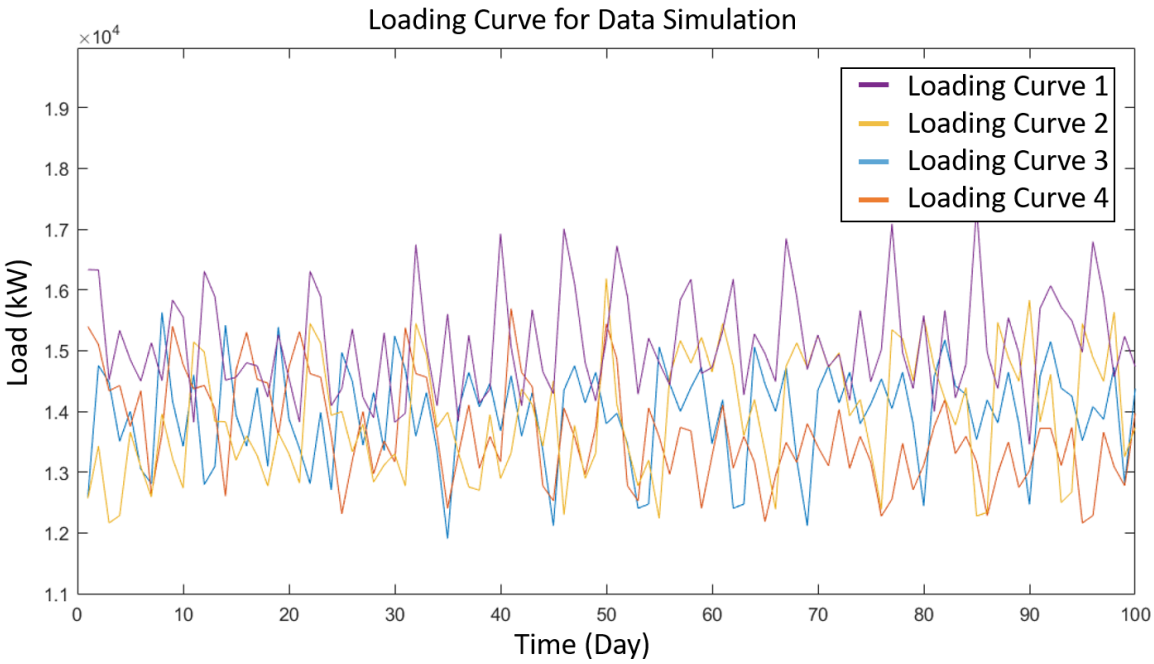


Figure 4. Selected Loading Scenarios for PSLF Data Generation.

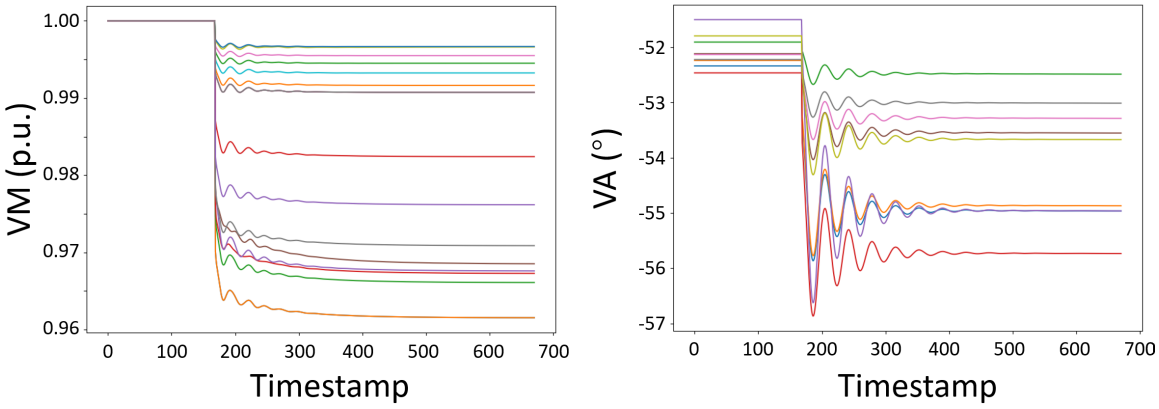


Figure 5. The Visualization of 10 Pmus' Voltage Magnitude (Vm) Measurements for Three Different Events.

2.4 Experiments

2.4.1 Dataset Description

In the experiment, I employ synthetic datasets from 200- and 500- bus systems (Engineering Texas A&M University 2016a, 2016b). For data preparation, I employ a commercial-grade simulator, Positive Sequence Load Flow (PSLF) to simulate PMU data with 60 samples per second. To diversify datasets, I vary the loading conditions during the simulation, which leads to different system dynamics. Specifically, I can change the initial condition of the load flow in PSLF by different loading curves. I employ the load profiles in real-world PJM Interconnection (*Data Miner 2 — dataminer2.pjm.com*) as different initialization points for the simulated systems. For example, Fig. 4 shows 4 loading curves from the PJM data. I randomly select their points at different time slots for the initialization, which boost the diversity of our experiments. Moreover, I introduce three different fault events to our data set to validate the robustness of our algorithm. As shown in Fig. 39, I visualize the event using Voltage Magnitude (VM) data obtained from PSLF. Further, I assume PMUs only locate at partial buses, with a penetration $\eta = 0.05$. For the rest data, I assume they are SCADA data with a sampling process. I obtain the SCADA data with 0.5 samples per second.

Totally, I have over 40 files of 10-second simulated data for each test system. To obtain $\boldsymbol{X}_p$ from these time series, I utilize the moving window with the length to be 0.33 seconds (i.e., $k = 20$ samples) to reformalize the training data. Thus, I have $d_p = 200 \times \eta = 10$; $d_s = 200 \times (1 - \eta) = 190$ for 200−bus test system and $d_p = 500 \times \eta = 25$; $d_s = 500 \times (1 - \eta) = 475$ for 500−bus test system. In general, I have $\boldsymbol{D}_s \in \mathbb{R}^{20\times 190}$ and $\boldsymbol{X}_p \in \mathbb{R}^{20\times(20\times 10)}$ for the 200-bus system and $\boldsymbol{D}_s \in \mathbb{R}^{20\times 475}$ and $\boldsymbol{X}_p \in \mathbb{R}^{20\times(20\times 25)}$ for the 500-bus system. They are matrices of PMUs and SCADA measurements data for one window. Further, I have $\boldsymbol{Z} \in \mathbb{R}^{580\times 190}$ $\tilde{\boldsymbol{X}} \in \mathbb{R}^{580\times 200}$ for the 200-bus system and $\boldsymbol{Z} \in \mathbb{R}^{580\times 475}$ $\tilde{\boldsymbol{X}} \in \mathbb{R}^{580\times 500}$ for the 500-bus system. They are total extracted matrices for missing SCADA data and PMU data to recover $\boldsymbol{Z}$.

2.4.2 Benchmark Method

As benchmarks for the effectiveness of the proposed model, I employ three distinct methods. The details of these methods are as follows.

- EM algorithm + Linear Regression (LR): LR fits a linear model with coefficients to minimize the error (Pedregosa et al. 2011). However, the LR model does not consider the spatial differences between PMU and SCADA data.
- EM algorithm + Vanilla Deep Neural Network: This method considers a Hd-Deep-EM framework with a vanilla DNN. Vanilla DNN does not take temporal correlations into account. In particular, window-based segmentation is configured with $k = 1$ and does not consider neighboring data for training. To differentiate from the suggested model, this method is referred to as Hd-Deep-EM0 while the proposed model is referred to as Hd-Deep-EM1. In addition, the proposed approach with noise consideration is referred to as the Double Hd-Deep-EM model.
- Deep Neural Network (DNN)(C.-Y. Cheng et al. 2020): DNN is extracted from the Hd-Deep-EM to directly train the mapping without EM algorithm. DNN can utilize spatial and temporal correlations effectively. However, the lack of dynamic information in SCADA data may hinder the DNN's effectiveness.

In general, by comparing the testing accuracy of the proposed model and the benchmark models, I can evaluate the effectiveness of Hd-Deep-EM. As claimed in the proposed model, our Hd-Deep-EM can (1) capture the complex spatial correlations between PMU and SCADA data, (2) incorporate temporal correlations for a robust estimator, and (3) use the EM framework to iteratively incorporate the temporal dynamics of the SCADA data for better DNN training. Especially compare our Double Hd-Deep-EM with the LR+ EM to illustrate the high representational power of the DNN to capture spatial correlations. Then, I compare Double Hd-Deep-EM with Hd-Deep-EM0 to understand to impacts of considering temporal correlations to estimate the hidden states. Finally, I compare Double Hd-Deep-EM with DNN to illustrate the design of the EM framework.

During the testing, the hyper-parameters for all models are fine-tuned to achieve the best accuracy. Especially, I report the detailed results with respect to Mean Square Error (MSE) between the truth and predict $\boldsymbol{Z}$ matrix (i.e., missing dynamic states of the SCADA system) for all methods.

The true $\boldsymbol{Z}$ matrix is the missing value not reported in the RTU sensors. They are in fact unknown to us. Therefore, I propose the following procedures to prepare data for validation.

- **Simulation**. I run a business-level simulator, Positive Sequence Load Flow (PSLF), to simulate PMU data with 60 samples per second for 200-bus and 500-bus systems (Engineering Texas A&M University 2016a, 2016b). For each node, I can obtain the PMU streams.
- **Down-Sampling**. I selected a limited number of buses and assumed they are placed with PMUs. Then, the rest buses should be equipped with SCADA sensors. I achieve this target by doing down-sampling for the streams of 60 samples per second. I extract 1 out of 30 samples in the streams as the blue points. These samples are treated as the measurements of the SCADA sensors. Correspondingly, the unselected data can formulate the true $\boldsymbol{Z}$ matrix. Notably, $\boldsymbol{Z}$ is only used in testing to evaluate the prediction error.

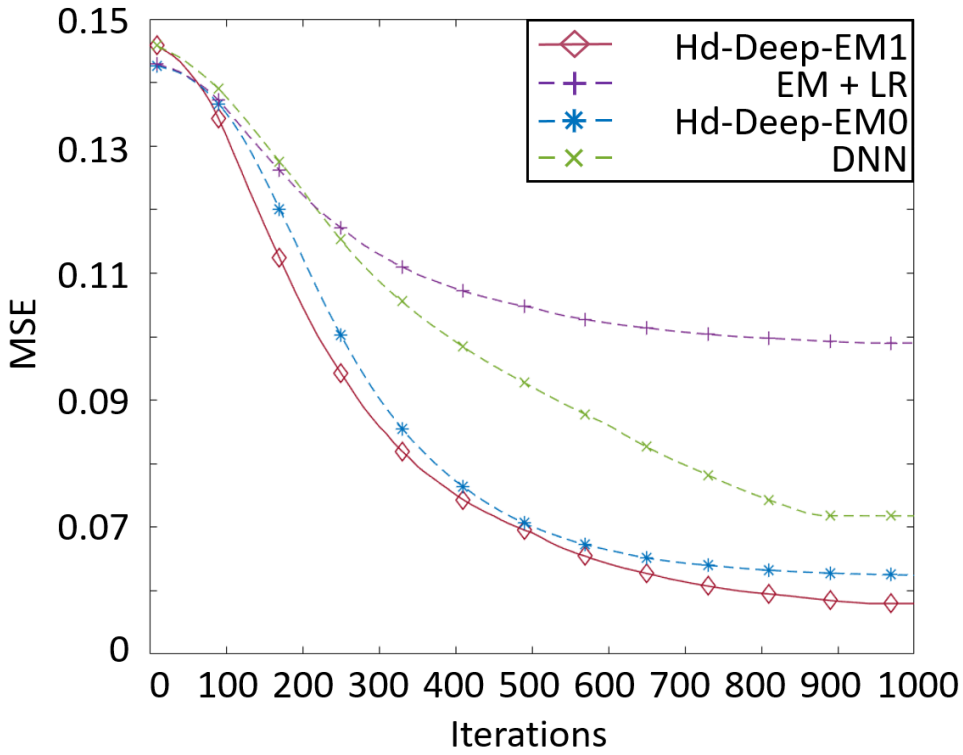


Figure 6. Testing Error for the 200-bus System.

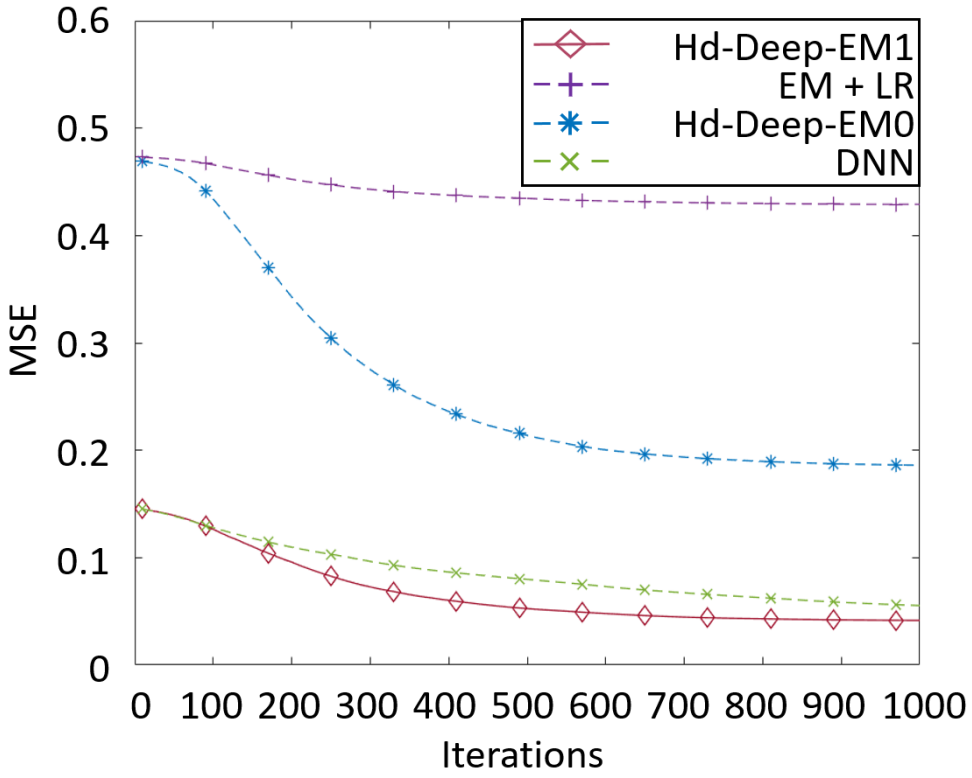


Figure 7. Testing Error for the 500-bus System.

2.4.3 Deep Model Is Better Than Linear Model

In this subsection, I evaluate the effectiveness of our Hd-Deep-EM algorithm in tackling the spatial difference by comparing our Hd-Deep-EM algorithm to LR + EM. The results show that our model performs better than benchmarks. Specifically, I report the prediction performance of simulated data as follows.

Fig. 6 and Fig. 7 demonstrates the performances for the two methods in two different test system. The y-axis is the MSE error, and the x-axis represents the number of iterations during training. From the above figures, I find that our Hd-Deep-EM algorithm is lower than the MSE error of 0.05 and 0.4, compared to shallow EM+ LR in 200- and 500- bus test system after convergence. The better performance of the Hd-Deep-EM algorithm shows that the deep learning model can be better than the LR model in our complex scenario.

2.4.4 Consider Temporal Correlations Increase DNN Performance

In this subsection, I evaluate the effectiveness of our Hd-Deep-EM algorithm in integrating the temporal correlations. Specifically, I compare Hd-Deep-EM0 and Double Hd-Deep-EM. Due to the window-based segmentation, our Double Hd-Deep-EM can successfully integrate neighboring measurements in the temporal domain to estimate the dynamic states at one time slot. However, Hd-Deep-EM0 doesn't have this treatment. Then, I report the results of simulated data as follows.

Fig. 6 and Fig. 7 demonstrate the performances for the two methods. From the above figures, I find that our Double Hd-Deep-EM has an average of 0.05 and 0.06 testing MSE error for both test systems after convergence. In comparison, Hd-Deep-EM0 has an error of 0.07 and 0.2. The performance of these two methods can be explained as follow. Window segmentation can capture the data before and after the SCADA time stamp, which can benefit from finding the temporal trending of the PMU data and avoiding the negative impacts of the outliers. The better performance of the Hd-Deep-EM algorithm with window segmented data shows that our proposed deep learning model can easily capture the temporal trend for PMU data in our complex scenario. Finally, I observe that in Fig. 7, DNN model can perform better than Hd-Deep-EM0 that doesn't consider the temporal correlations. This verifies our claims that considering proper temporal correlations in the DNN and Hd-Deep-EM1 (noiseless) can improve the performance of estimating dynamic hidden states.

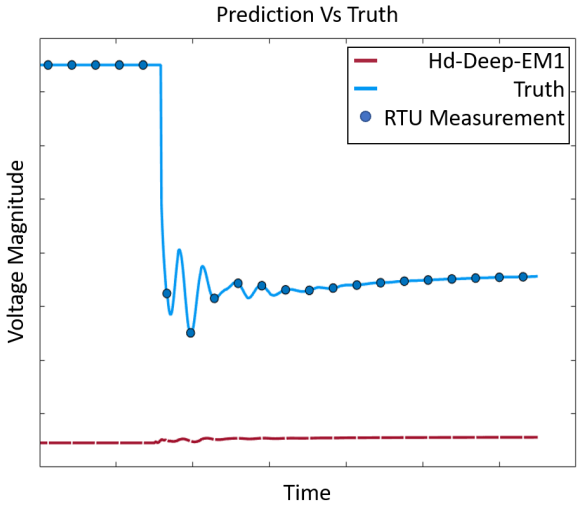


Figure 8. Hd-Deep-EM Prediction on Iteration 1.

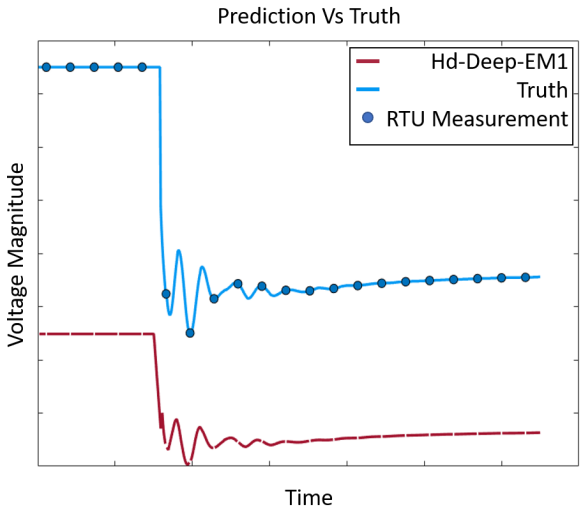


Figure 9. Hd-Deep-EM Prediction on Iteration $3,300$.

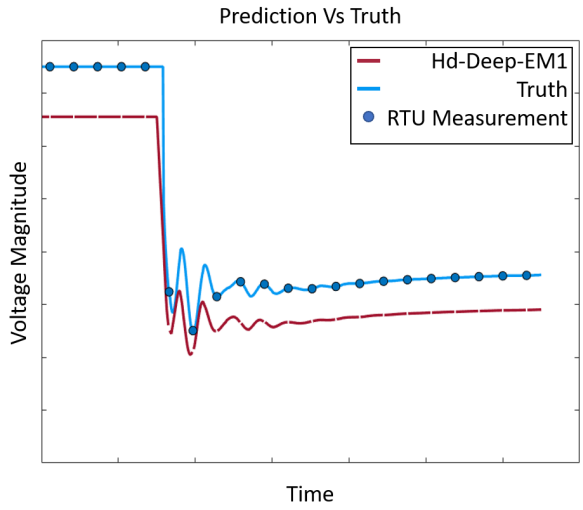


Figure 10. Hd-Deep-EM Prediction on Iteration $6,600$.

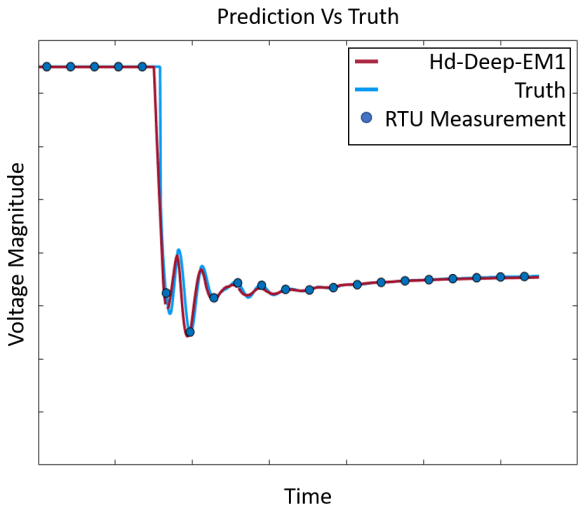


Figure 11. Hd-Deep-EM Prediction on Iteration $10,000$.

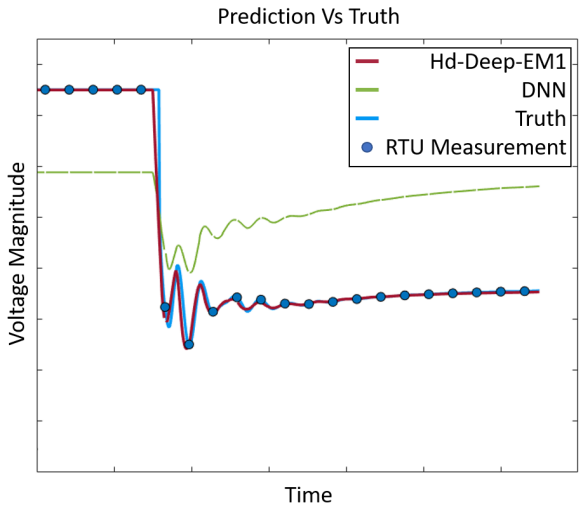


Figure 12. Data Interpolation Result after 10,000 Epoch.

2.4.5 EM Procedure Provides Better Training of the DNN

In this subsection, I evaluate the effectiveness of introducing the EM procedure by comparing Hd-Deep-EM1 and a DNN model. First, I show a case study of how the EM algorithm can improve the training of DNN. I consider a line trip in Fig. **??** and Fig. 12. Specifically, Fig. **??** shows the iterative process and our EM framework helps the DNN to make more use of the dynamic data and enable the DNN prediction to gradually converge to the true blue curve. Fig. 12 demonstrates the

final comparison of our Hd-Deep-EM and vanilla DNN. I can find that only when I combine EM algorithm with DNN model can I achieve a generalizable model to accurately predict the dynamic data for the RTU sensors.

Then, I report the statistical results of simulated data as follows. Fig. 6 and Fig. 7 demonstrate the performances for the two methods. I find that our Double Hd-Deep-EM has an average testing error reduction of 0.01 and 0.02, compared to the DNN method. The reason, when training the DNN model, the output SCADA data has limited dynamic information. Thus, direct training can't guarantee the DNN model performance to predict dynamic states. On the other hand, our Hd-Deep-EM can iteratively predict the dynamic states and reuse the predicted results for training, leading to a better training procedure to incorporate dynamic information.

2.4.6 Window-based Prediction Enable Certain Robustness to Non-synchronization

I claim that our method has certain robustness to the non-synchronization issues. The main reason is that I utilize a window of PMU data, rather than single time-slot data, as the input to the DNN model in Hd-Deep-EM. Therefore, our DNN model utilizes an interval of PMU data to predict one point in the SCADA data streams. Due to the high temporal correlations, our DNN contains certain tolerance for the mismatch. If the non-synchronization doesn't exceed the range of the window, DNN can deliver a good estimation. To support our claim, I conduct numerical validations for the 200-bus case with 10 PMUs, shown in Table 1. I find that as long as the delay time between PMU and SCADA sensors is less than 1s, our Hd-Deep-EM can always attain good performance.

Table 1. Testing Error for Different Delay Time.

Delay Time(s)	Hd-Deep-EM1	EM + LR	Hd-Deep-Em0	DNN	Double Hd-Deep-EM
0	0.039	0.080	0.055	0.052	0.039
0.5	0.041	0.085	0.058	0.057	0.042
1	0.042	0.089	0.057	0.059	0.044
1.5	0.057	0.120	0.085	0.078	0.056
2	0.121	0.145	0.215	0.232	0.129

2.4.7 Double Hd-Deep-EM Provides Better Performance in Tackling Noise

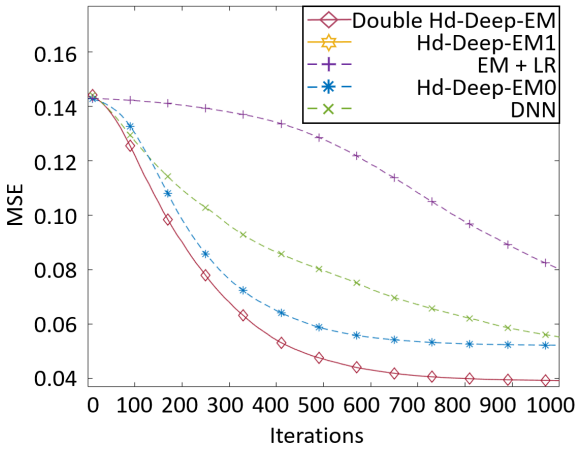


Figure 13. Testing Error for the 200-bus System with Noisy Data.

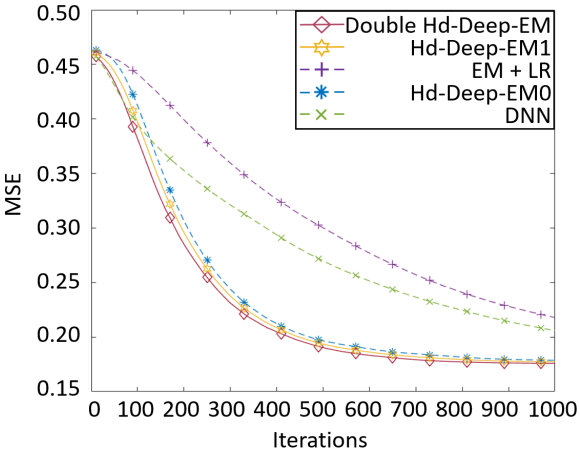


Figure 14. Testing Error for the 500-bus System with Noisy Data.

In this subsection, I evaluate the effectiveness of interpolating noisy data by comparing Double Hd-Deep-EM among all three benchmark models and our Hd-Deep-EM1 method. In the experiment, I introduce noise into our dataset to validate the effectiveness in tackling noisy data. Specifically, I report the results of simulated data as follows. As shown in Fig. 13 and Fig. 14, I find that our Double Hd-Deep-EM has an average testing error reduction of 0.04, 0.02, 0.02 and 0.01, compared to EM+LR, Hd-Deep-EM0, DNN method and our proposed Hd-Deep-EM1 for the 200 bus system. Furthermore, Double Hd-Deep-EM has an average testing error reduction of 0.01, 0.02, 0.03 and

0.005, compared to bench mark methods and Hd-Deep-EM1 for the 500 bus system. The results show that our Double Hd-Deep-EM has the best performance of robustness to noisy data.

2.4.8 Hd-Deep-EM Has Comparable Testing Time

In this subsection, I compare the training and testing time for different methods, shown in Table 2. The results show that the EM framework will generally increase the training time even when I utilize the same iteration numbers. This is reasonable because vanilla DNN will only utilize RTU samples, i.e., the blue dots in Fig. 12. However, the EM framework can bring more training samples, making the training process more time-consuming. However, the training time is still affordable for the offline process. Moreover, if I compare the testing time, I find that all Machine Learning (ML) methods have a very quick inference time, which is due to the fast forward computation of the ML model.

Table 2. The Training and Testing Time (S) for Different Data Interpolation Methods for the Scenario on the 200-bus System with 1000 Iterations.

method	training time	testing time
Hd-Deep-EM1	508.32	0.03
EM + LR	481.52	0.01
Hd-Deep-Em0	492.80	0.03
DNN	19.21	0.01
Double Hd-Deep-EM	537.31	0.04

2.4.9 Sensitivity Analysis with Respect to Different PMU Locations

An essential point is about the placement and number of Phasor Measurement Units (PMUs) compared to Supervisory Control and Data Acquisition (SCADA) sensors. Specifically, the locations of PMUs and SCADA sensors can largely influence on the performance of Deep Learning (DL) algorithms in power systems.

Then, I add the following experiments on the Illinois 200-bus system to check how the locations

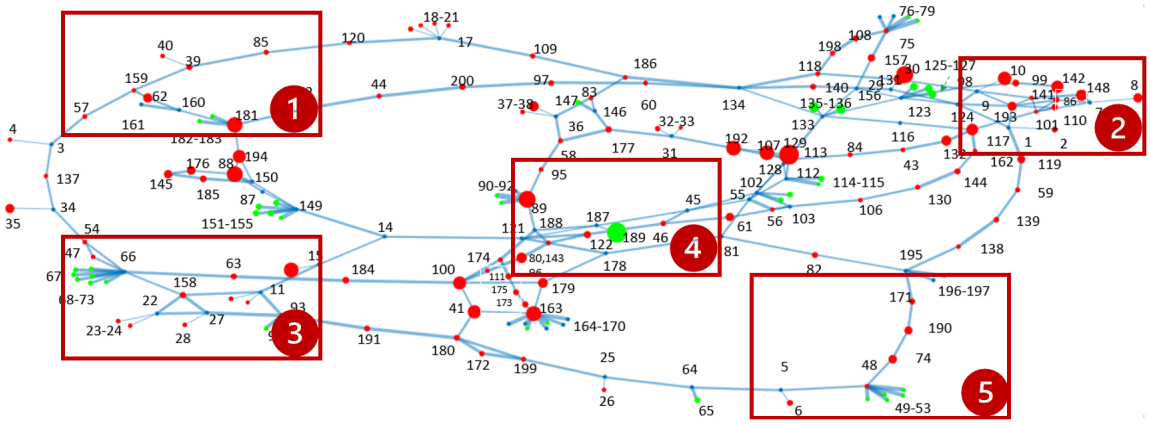


Figure 15. Different PMU Placement Test Case on Illinois 200-bus System.

of PMUs can affect the final results. Specifically, in our experiments in the manuscript, I conduct uniform selections for 10 PMUs over the complete grids for 10 times, making PMUs widely located in the grid. In this revised version, I propose a biased selection for 10 PMUs, shown in Fig. 15. I define 5 different regions in the grid with 10 PMU placement and make the rest nodes placed with SCADA sensors to monitor the whole grid. With obtained data, I conduct different methods and report the results of MSE error in Table 3.

Table 3. Testing Error for Different PMU Placement Test Cases.

Cases	Hd-Deep-EM1	EM + LR	Hd-Deep-Em0	DNN	Double Hd-Deep-EM
Region 1	0.041	0.072	0.050	0.049	0.042
Region 2	0.039	0.080	0.055	0.052	0.039
Region 3	0.040	0.091	0.049	0.045	0.040
Region 4	0.032	0.064	0.049	0.042	0.036
Region 5	0.048	0.101	0.067	0.051	0.048
Uniform Selection	0.031	0.062	0.043	0.039	0.031

First, I observe that for all biased selections of PMUs, all methods perform worse than the uniform selection. This is because PMU measurements have higher correlations to the neighboring SCADA data. Thus, if PMUs can be placed in a wide range, their information can be better utilized to interpolate the SCADA data of the complete grid. Second, I find that Region 4 has relatively better performance since it locates in the central area of the grid, which makes the PMUs better to monitor the whole grid. Thirdly, Region 2 has the third good performance. This is because I conduct

4 out of 10 faults around Region 2. In general, if PMUs are widely placed or close to dynamic events, the performance will be better.

2.4.10 Sensitivity Analysis with Respect to Noise Levels

In addition, I further evaluate Double Hd-Deep-EM performance with other benchmarks models in 3 different level of noise injection to the simulated data. To quantify the level of the noise, I utilize the Signal-to-Noise Ratio (SNR) which compares the level of a desired signal to the level of noise. Signal-to-Noise Ratio can be defined as the ratio of signal power to noise power. The result of MSE error is shown in the tables below.

Table 4. The MSE for Different Data Interpolation Methods in Different Noise Level for Illinois 200-bus Systems.

SNR Ratio (dB)	Double Hd-Deep-EM	Hd-Deep-EM1	EM+ LR	Deep EM0	DNN
25	**0.03914**	0.04874	0.08002	0.05214	0.05513
15	**0.03923**	0.05045	0.08204	0.05426	0.05525
10	**0.04035**	0.05765	0.08234	0.05987	0.05947

Table 5. The MSE for Different Data Interpolation Methods in Different Noise Level for South Carolina 500-bus Systems.

SNR Ratio (dB)	Double Hd-Deep-EM	Deep EM1	EM+ LR	Deep EM0	DNN
25	**0.17596**	0.17723	0.21794	0.17873	0.20601
15	**0.17601**	0.17983	0.21819	0.18073	0.20730
10	**0.17616**	0.18177	0.22056	0.18395	0.21049

According to the performance in Table 4 and Table 5, I find that our Double Hd-Deep-EM has an average testing error reduction over different noise level of 0.013, 0.017, 0.014 and 0.042, compared to Hd-Deep-EM1, EM+LR, Hd-Deep-EM0 and DNN method for 200 bus system. And Double

Hd-Deep-EM has an average testing error reduction over different noise level of 0.004, 0.031, 0.005 and 0.043, compared to bench mark methods for 500 bus system.

2.4.11 Event Identification Accuracy Comparison with Different Dataset

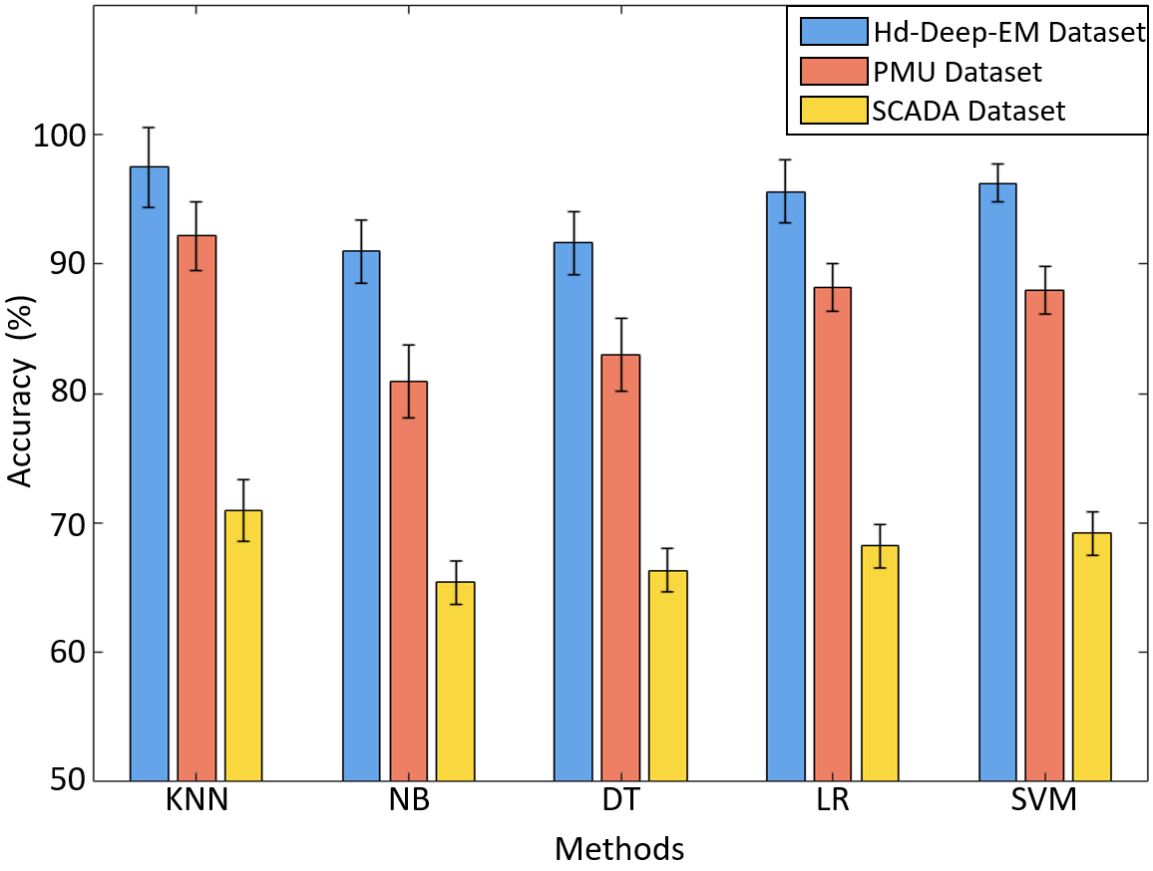


Figure 16. Event Identification for Different Datasets.

To evaluate the effectiveness of interpolation, I propose to compare the interpolated SCADA data, original SCADA data and PMU data for the data-driven task: identifying system events. Specifically, the same event identification algorithm is performed among the three datasets. I utilize the 5 different popular Supervised Learning methods to evaluate the results. The selected methods are K Nearest Neighbor (KNN), Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR), and Support Vector Machine (SVM). As shown in Fig. 16, I find that our interpolated data can best

improve the different learning methods, with KNN being the best. Averagely, our merged dataset has an average increase of 7.92%, and 36.35%, compared to the original PMU and SCADA datasets.

2.4.12 Investigations of Different Activation Functions and Initialization Methods

For the DNN models, the choices of activation functions and initialization methods for the weights are biases are important. Therefore, in this subsection, I investigate the performance of different designs. For the effectiveness of activation functions, I conduct the following experiments, as shown in table 6. I find that the changes in the final results are insignificant for different hybrid activation functions (Radford, Metz, and Chintala 2015) compared to ReLU functions. Thus, it's reasonable to utilize ReLU as the activation function given its lowest complexity.

Table 6. Testing Error for Different Activation Functions.

Activation Function	Hd-Deep-EM1
ReLU	0.039
Softplus + ReLU	0.042
Softplus + ELU	0.041
Softplus + Leaky ReLU	0.039
Sigmoid + ReLu + Softplus	0.040
Hyperbolic Tangent + Sigmoid + ReLu	0.045

For the initialization methods, our experiments before utilize the He initialization method (K. He et al. 2015). However, I agree that different initialization may cause different results. Some other initialization methods in Pytorch include Uniform Initialization, Normal Initialization, Xavier Initialization, etc (Paszke et al. 2019). To further quantify the impacts of these initialization methods, I conduct the tests for the 200-bus case with 10 PMUs. The result is shown in Table 7. I find that although I change different initialization for the weights and biases, the results are similar.

Table 7. Testing Error for Different Initialization Function.

Initialization	Hd-Deep-EM1	Hd-Deep-Em0	DNN	Double Hd-Deep-EM
He	0.039	0.055	0.052	0.39
Uniform	0.041	0.053	0.051	0.40
Normal	0.038	0.057	0.053	0.42
Xavier	0.040	0.058	0.053	0.41

Chapter 3

DATA INTERPOLATION: INTRINSIC PHYSICS-INFORMED APPROACH TO ENHANCE DATA QUALITY

3.1 Introduction

The increasing integration of clean energy, such as Photovoltaic (PV) and wind generations, introduces large uncertainty and intermittence to power systems. Hence, the system suffers more and more disturbances, oscillations, and outages. It's critical to understand and control system dynamics to maintain system stability and resilience. To achieve the goal, tremendous methods arise for transient analysis (Q. Zhu et al. 2018), outage detection (Haoran Li et al. 2019b, 2019a; Li, Ma, and Weng 2022; Li, Ma, Weng, Blasch, et al. 2022), dynamic state estimation (Netto, Zhao, and Mili 2016; Le and Benjapolakul 2018; Qu, Wang, and Shen 2021) , parameter estimation (Li and Weng 2021; Haoran Li et al. 2023), control (Machowski et al. 2020), etc. Most of these studies rely on the dynamic measurements from Phasor Measurement Units (PMUs) that produce $30 \sim 120$ samples per second ("IEEE Standard for Synchrophasor Measurements for Power Systems" 2011).

However, PMU installation is costly. For example, the cost of a single PMU in the United States ranges from $40,000 to $180,000 (Young and Silverstein 2014). Other costs are related to communication infrastructure, data management systems, and cybersecurity precautions. These expenses grow significantly as the number of PMUs increases, preventing extensive PMU installations (Madani et al. 2011). In contrast, traditional Low-Resolution (LR) meters are relatively cheap. For example, the installation cost for a Remote Terminal Unit (RTU) sensor ranges from $200 to $5,000 (Sayed and Gabbar 2017). RTUs provide $0.5 \sim 2$ samples per second for the Supervisory Control and Data Acquisition (SCADA) system. While largely distributed, such LR meters can hardly provide enough dynamic information. To boost dynamic observability without additional PMU placement, one can algorithmically interpolate the missing values of LR meters, thus converting LR meters to "virtual" PMUs. In this chapter, I define such virtual PMUs as interpolated LR meters that have PMU-level data resolutions as well as a certain tolerance of interpolation errors. They are

not required to provide measurements as accurate as PMUs but can provide certain patterns of the system dynamics.

In addition to SCADA systems in the transmission grid, I also provide two possible use cases in distribution grids to build virtual PMUs. In case 1, I consider the next-generation smart meter with a sampling rate much larger than 1 sample per 15min (Jones; Hawkins 2022). Thus, one can promote the future smart meters into virtual PMUs as long as the future smart meters have around $1 \sim 2$ samples per second. In case 2, I can further reduce the communication costs via down-sampling the micro-PMUs, e.g., producing 10 samplers per second. However, our model can convert the down-sampled streams and the next-generation smart meter streams into virtual PMU streams. In general, this chapter can serve as a reference for the development and trade-off of the future distribution grid measuring infrastructures, communications, and AI computations.

Existing methods use the data correlations to estimate the underlying quantities, categorized into model-based, optimization-based, signal analysis-based, and Machine Learning (ML)-based methods. The first group introduces pre-defined models to connect samples. Typical models include identity function, linear or polynomial functions (De Boor 1994), spline function (Habermann and Kindermann 2007), etc. Obviously, the presumed formulations are biased to depict the power system dynamic transitions. The second group leverages the data matrix's properties to formulate constrained optimization to infer missing entries. For example, (Gao et al. 2015) employ low-rank property for data recovery. Specifically, they minimize the low-rankness of the measurement matrix. However, the insufficient LR measurements make data less redundant and violate the low-rank postulation.

In addition to data similarity, signal trends, structures, frequency, and modes can be mined for interpolation, bringing signal analysis-based methods. For example, (Dong et al. 2013) introduces an autoregressive model to extract trends for data imputation. Reference (Radchenko and Viazovska 2019) utilizes Fourier transformation to compute the basis for interpolation. (Mewett, Reynolds, and Nazeran 2004) proposes a spectrum interpolation in the frequency domain. Instead of direct computations for patterns or signatures, ML models learn useful features to estimate the missing data. For instance, (Fukami, Fukagata, and Taira 2021) employs a spatial-temporal Convolutional Neural Network (CNN) to conduct super-resolution for turbulent flows. (Jia, Yu, and Ma 2018) mines such correlations using kernel methods so that seismic data are interpolated. (Z. Ma et al. 2023)

combines Deep Learning (DL) and the Expectation-Maximization (EM) algorithm to capture the dynamic patterns for power measurements iteratively. However, these methods can't guarantee physics consistency.

To tackle this fundamental problem, I note that power system dynamic measurements are governed by Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs). Hence, the solution of the DAEs can provide accurate data imputation. Consequently, the problem can be viewed as solving the DAEs, given multi-resolution samples from several LR meters and few PMUs. Generally, the given DAEs can be solved via classic differential equation solvers (Tucker 2002), time-domain simulations (Wang and Crow 2011), advanced transformation and decomposition techniques (Liu and Sun 2020), ML models (Misyris, Venzke, and Chatzivasileiadis 2020), etc. In particular, Physics-Informed Neural Network (PINN) (Raissi, Perdikaris, and Karniadakis 2019) has attracted great attention to solve DAEs, due to the high flexibility to incorporate data, differentiability, fast inference, efficient storage, etc. Therefore, employing a PINN for the virtual PMU conversion is practical.

Specifically, PINN aims to approximate the solution of DAEs via a Deep Neural Network (DNN), where the input is the time and the output is the solution at the given time. Furthermore, PINN introduces a function-form loss that forces the DNN to maintain the DAE functions. The derivatives of the DAE can be calculated by the so-called Automatic Differentiation (AD) (Chiu et al. 2022). The above designs enable both feasibility and efficiency for PINN as the DAE solver. While gaining a much better performance than other methods, PINN still has a convergence issue if the sampling rate is low (Shin, Darbon, and Karniadakis 2020).

Thus, can I improve the PINN performance to achieve virtual PMU conversion? The key is to utilize the fact that for power systems, HR/LR quantities lie in a low-dimensional embedding space (i.e., a manifold) constrained by ODEs/DAEs (Hairer 2011). For example, Fig. 39 illustrates the strong correlations of voltage magnitudes and angles. Therefore, I propose a systematic approach. In step 1, I find a good initial guess of the missing LR data in the embedding space. In step 2, I refine the interpolated data with the governing ODEs/DAEs, shown in the left and the middle part of Fig. 18. More specifically, I build a mapping between HR and LR variables to represent the spatial-temporal correlations, parameterized by another NN in the left part of Fig. 18. Nevertheless, training the NN is also challenged by the limited LR samples. Luckily, the rich PMU-data patterns facilitate a

Semi-Supervised Learning (SSL) paradigm, ensuring the trained NN gains high generalizability with few LR quantities. Hence, SSL enables us to provide a good initial guess for the missing data. Then, I can refine initial values with a PINN, making the interpolation satisfy DAEs, shown in the middle part of Fig. 18.

PINN requires exact DAE functions. When the DAE parameters are unknown, I can estimate them by leveraging interpolated and true measurements. The idea is possible due to the following reasons: (1) **Availability**. The AD layer in CoPIE can provide derivatives for DAE parameter estimation. (2) **Fidelity**. By our above designs, the fidelity of interpolated data is guaranteed to some extent. (3) **Robustness**. Traditionally, the Least Absolute Shrinkage and Selection Operator (LASSO) is widely utilized to identify nonlinear system dynamics (Brunton, Proctor, and Kutz 2016; Stanković et al. 2020). However, our interpolated data may contain certain errors that may degrade the accuracy. To promote the robustness, I employ Error-Corrected LASSO (EC-LASSO) (Loh and Wainwright 2011) to conduct parameter estimation, thus tolerating moderate errors from interpolation, shown in the middle part of Fig. 18. In general, I term our framework as CoPIE: A Coordinate framework for Physics-informed Interpolation and Estimation.

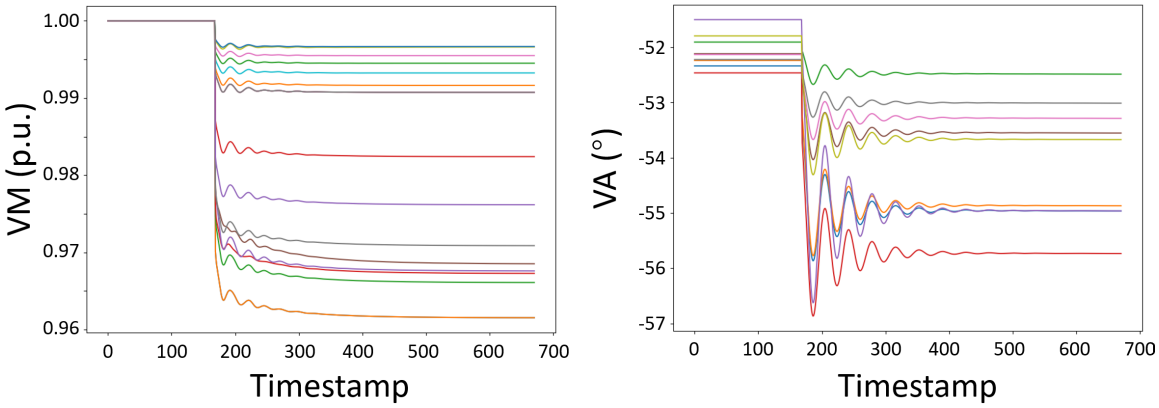


Figure 17. The Visualization of Voltage Magnitude (Vm) and Voltage Angle (Va) Measurements for a Line Trip Event.

To summarize, I have the following contributions. (1) I formulate the problem of converting LR meters into virtual PMUs. The problem is physics-informed data-driven with HR synchrophasors from limited PMUs and widespread LR measurements. (2) To solve the problem, I propose our CoPIE to maximally leverage the data patterns, spatial-temporal correlations, and the physics (i.e., DAEs). (3) I propose strict theoretical validations on our CoPIE and claim that the error

is limited. (4) I conduct comprehensive numerical validations on diversified grids and show that CoPIE performs better than other methods.

The rest of the chapter is organized as follows: Section 4.3 defines the problem. Section 4.4 proposes our CoPIE. Section 3.4 provides the theoretical analysis. Section 4.6 conducts experiments. Section **??** concludes the chapter.

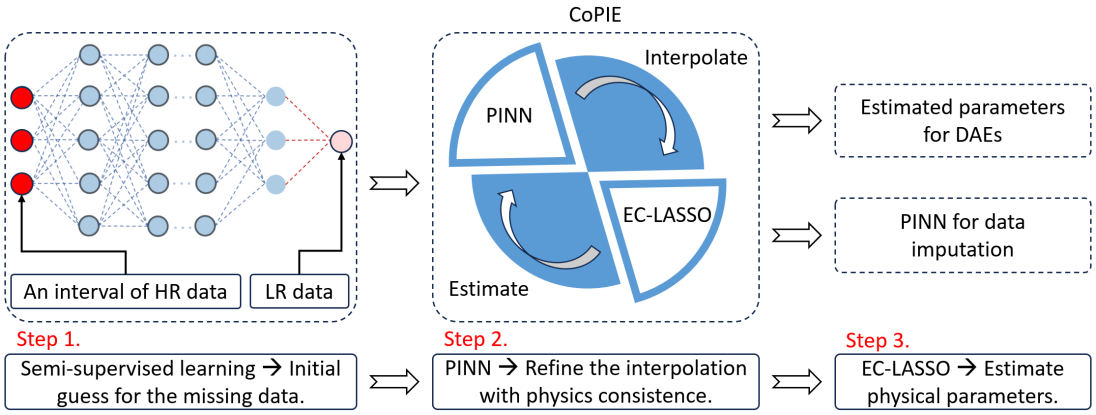


Figure 18. An Illustration of the CoPIE Framework.

3.2 Problem Formulation

In this subsection, I formalize the problem of virtual PMU conversion through data interpolation. To begin with, I illustrate what information is given. Generally, the information is included in a DAE that governs the system dynamics:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{y}(t)\big), \\ \mathbf{0} &= \boldsymbol{g}\big(\boldsymbol{x}(t), \boldsymbol{y}(t)\big), \end{aligned} \tag{3.1}$$

where the differential variable $\boldsymbol{x}(t) \in \mathbb{R}^{d_x}$, the algebraic variable $\boldsymbol{y} \in \mathbb{R}^{d_y}$, $\boldsymbol{f}(\cdot)$ are nonlinear functions to compute the derivative $\dot{\boldsymbol{x}}(t) = \frac{d\boldsymbol{x}(t)}{dt}$, and $\boldsymbol{g}(\cdot)$ are algebraic functions of system constraints. In our setting, parameters of equations $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are unknown, and I only have samples of $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$, measured by LR meters or PMUs.

To further demonstrate the data availability, I assume there are few PMUs and a large number of LR meters. These two types of sensors can together provide all synchronized measurements of $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ at some specific time slot t. Therefore, I denote $\boldsymbol{u}(t) \in \mathbb{R}^{d_u}$ and $\boldsymbol{w}(t) \in \mathbb{R}^{d_w}$ as the

measurements for PMUs and LR meters, respectively, where $d_u \ll d_w$ and $d_u + d_w = d_x + d_y$. Let $\phi(\cdot)$ be a partition function such that $[\boldsymbol{x}, \boldsymbol{y}] = \phi(\boldsymbol{u}, \boldsymbol{w})$, i.e., $\phi(\cdot)$ split the vector $[\boldsymbol{u}, \boldsymbol{w}]$ into $\boldsymbol{x}$ and $\boldsymbol{y}$. Subsequently, by the difference of sampling rates, I assume that there are N_u samples for $\boldsymbol{u}$ and N_w samples for $\boldsymbol{w}$, where $N_u \gg N_w$. In general, Fig. 19 visualizes the available datasets. I denote a set of PMU samples as $\{\boldsymbol{u}(t_i)\}_{i=1}^{N_u}$ and LR meter samples as $\{\boldsymbol{w}(\tilde{t}_i)\}_{i=1}^{N_w}$, where I utilize $\tilde{\cdot}$ imply that $t_i \neq \tilde{t}_i$. Eventually, I can define our problem as follows.

- Problem: Convert LR meters to virtual PMUs via physics-informed data interpolation.
- Given: A set of PMU samples $\{\boldsymbol{u}(t_i)\}_{i=1}^{N_u}$ and LR meter samples $\{\boldsymbol{w}(\tilde{t}_i)\}_{i=1}^{N_w}$.
- Find: A parametric model to interpolate the missing dynamic data in LR meters.

With only measurements, CoPIE uses erroneous data to robustly estimate the parameters of the DAE, boosting the data interpolation. Moreover, CoPIE maximizes the usage of data patterns and spatial-temporal correlations in an SSL manner, obtaining a tight bound and state-of-the-art numerical performance. I demonstrate CoPIE as follows.

3.3 Proposed Model

In this section, I first introduce the preliminary of a PINN. Then, I introduce our CoPIE framework with both interpolation and estimation. Finally, I demonstrate the CoPIE's training algorithm.

3.3.1 Preliminary of a PINN

PINN is composed of a DNN and an AD layer. The DNN aims to approximate the solution of the DAE in Equation (3.1) with the universal approximation power. Mathematically, I denote $[\boldsymbol{u}(t;\Theta), \boldsymbol{w}(t;\Theta)]$ as the DNN, parameterized by Θ, which maps from the time index t to the DAE solutions $[\boldsymbol{u}, \boldsymbol{w}]$. Usually, the DNN uses Sigmoid activation to guarantee both nonlinearity and certain smoothness for approximating the physical solutions. For the loss function, one can enforce the boundary-condition loss, i.e., the Mean Square Error (MSE), to fit the measurements:

$$\mathcal{L}_1(\Theta) = \sum_{i=1}^{N_u} ||\boldsymbol{u}(t_i) - \boldsymbol{u}(t_i;\Theta)||_2^2 + \sum_{i=1}^{N_w} ||\boldsymbol{w}(\tilde{t}_i) - \boldsymbol{w}(\tilde{t}_i;\Theta)||_2^2, \tag{3.2}$$

where $||\cdot||_p$ represents the l_p norm.

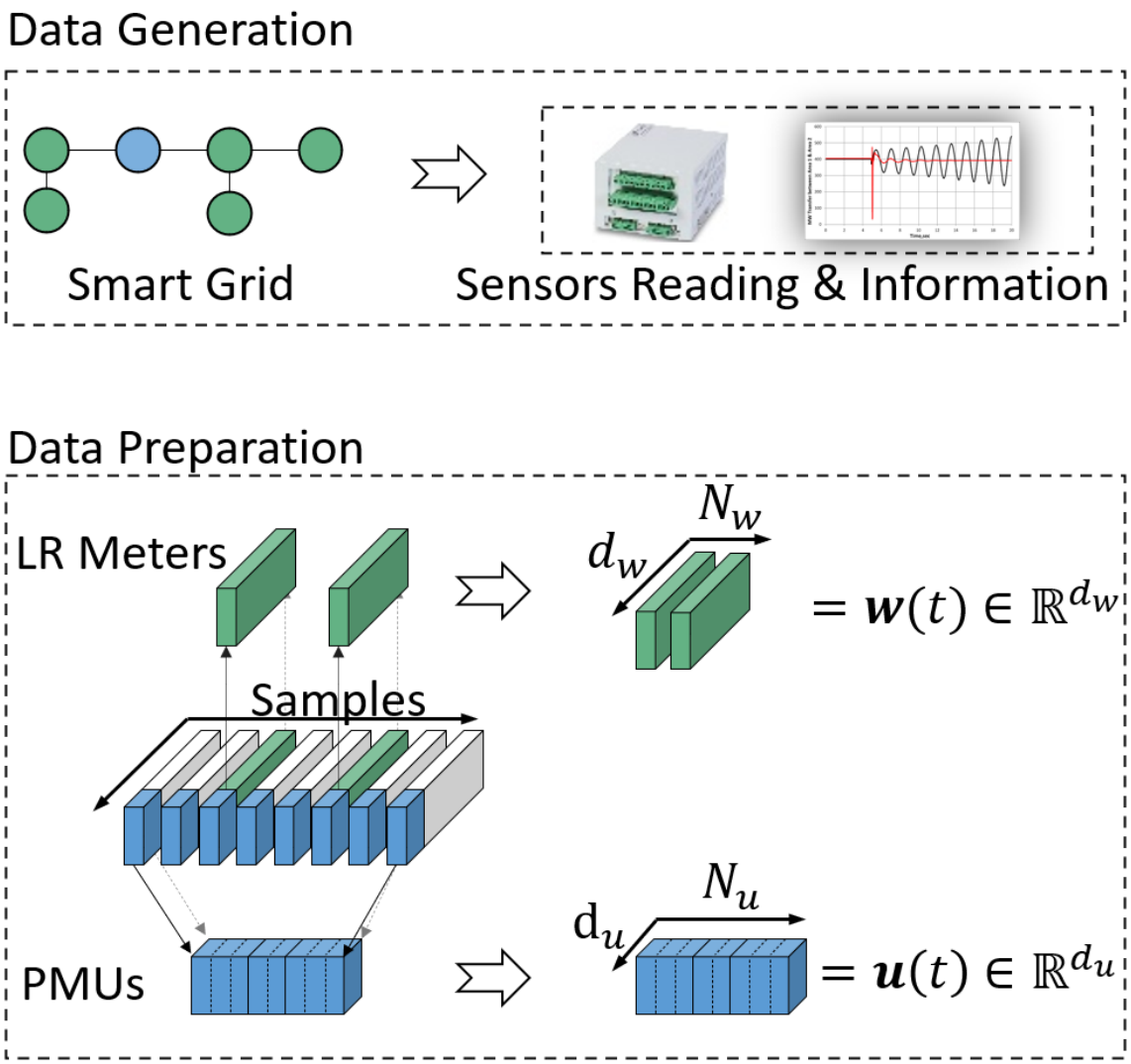


Figure 19. A Moving Window-based Segmentation Procedure to Process the PMU Streams.

Equation (3.2) slightly modifies the original boundary-condition loss and utilizes both the LR and HR samples. I prove in Section 3.4 that additional HR samples reduce the approximation error. Nevertheless, training with only $\mathcal{L}_1(\Theta)$ can easily meet overfitting. To make the approximation generalize well, PINN includes an additional function-form loss.

$$
\begin{aligned}
[\boldsymbol{x}(t;\Theta), \boldsymbol{y}(t;\Theta)] =& \phi\big(\boldsymbol{u}(t;\Theta), \boldsymbol{w}(t;\Theta)\big), \\
\mathcal{L}_2(\Theta) = & \int_0^T \big|\big|\boldsymbol{g}\big(\boldsymbol{x}(t;\Theta), \boldsymbol{y}(t;\Theta)\big)\big|\big|_2^2 \\
& + \big|\big|\dot{\boldsymbol{x}}(t;\Theta) - \boldsymbol{f}\big(\boldsymbol{x}(t;\Theta), \boldsymbol{y}(t;\Theta)\big)\big|\big|_2^2 dt,
\end{aligned}
\tag{3.3}
$$

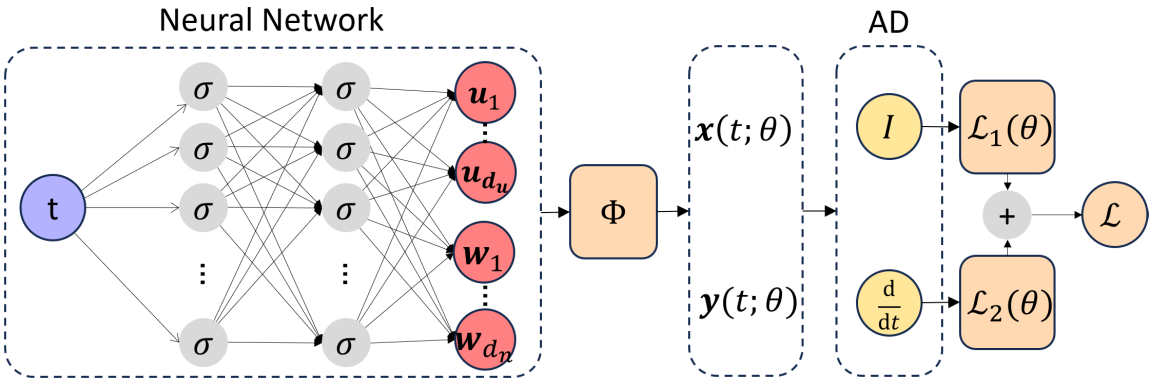


Figure 20. An Illustration of the PINN Model. I Utilize I to Represent the Identity Function in the AD Layer.

where $\phi(\cdot)$ as the split function is defined in Section 4.3, T is the end time of the system dynamics, and $\dot{\boldsymbol{x}}(t;\Theta)$ is the numerical differentiation of $\boldsymbol{x}(t;\Theta)$, generated by the AD layer using the chain rule (Raissi, Perdikaris, and Karniadakis 2019).

The integration in Equation (3.3) can be numerically solved via a small step. Basically, $\mathcal{L}_2(\Theta)$ maintains the DAE function form (i.e., Equation (3.1)) at each time slot in $[0, T]$. Finally, one can utilize a penalty term $\lambda_1 > 0$ to balance the two losses, leading to the final PINN loss:

$$\mathcal{L}(\Theta) = \mathcal{L}_1(\Theta) + \lambda_1 \mathcal{L}_2(\Theta). \tag{3.4}$$

The structure of a PINN is shown in Fig. 20. If well trained, PINN can accurately predict data for arbitrary $t \in [0, T]$, i.e., conducting data imputation. Moreover, as long as I have enough data, PINN can quickly converge to the true solution (Shin, Darbon, and Karniadakis 2020). However, limited LR samples obstruct the convergence.

3.3.2 CoPIE to Improve Convergence

First, I clarify our design philosophy in a conceptual visualization in Fig. 21. The red points represent measurements. The grey regions of the top part of Fig. 21 represent the estimate errors, and the LR variables have a significant error region. However, if I can utilize the correlations and the HR samples to project the interpolated data of LR variables, I can obtain a more restrictive error region, as shown in the bottom figure. The solid and dotted arrows represent the projection, and

the pink nodes are the projected results. This essentially makes use of the fact LR/HR data lie in a low-dimensional embedding space, and a good projection helps to find the missing values in the embedding. Subsequently, I can input the red and the pink nodes to the PINN and obtain further refinement. Finally, for the PINN, if the parameters of DAE are unknown, I introduce EC-LASSO to estimate parameters. The general framework is shown in Fig. 18.

Mathematically, the projection can be represented as a mapping from HR samples to LR samples, denoted as $\boldsymbol{h}(\cdot) : \mathbb{R}^{K\times d_u} \to \mathbb{R}^{d_w}$, where K is the number of time slots (an even number) employed to capture the temporal trend. Similarly, I utilize a neural network with a parameter set Ω to represent the mapping, namely, $\boldsymbol{h}(\cdot;\Omega)$. Notably, there can be a lot of designs for this NN to conduct spatial-temporal inference, e.g., the Transformer (Vaswani et al. 2017). However, I find that a fully connected NN is enough in our Experiments, thanks to the strong correlations of voltage magnitudes and angles. Then, to predict each LR sample $\boldsymbol{w}(t)$, I input the correspondingly synchronized HR sample $\boldsymbol{u}(t)$ and the neighboring K samples $[\boldsymbol{u}(t-\Delta\cdot K/2), \boldsymbol{u}(t-\Delta\cdot(K/2-1)), \cdots, \boldsymbol{u}(t+\Delta\cdot K/2)]$, where Δ is the sampling interval for PMUs. Basically, the spatial-temporal NN is defined as $\boldsymbol{h}\big(\boldsymbol{u}(t-\Delta\cdot K/2), \boldsymbol{u}(t-\Delta\cdot(K/2+1)), \cdots, \boldsymbol{u}(t+\Delta\cdot K/2);\Omega\big)$. The training of $\boldsymbol{h}(\cdot;\Omega)$, however, may still suffer low generalizability since the output samples (LR samples) are limited. To address the issue, I introduce SSL incorporating rich PMU data to promote generalizability.

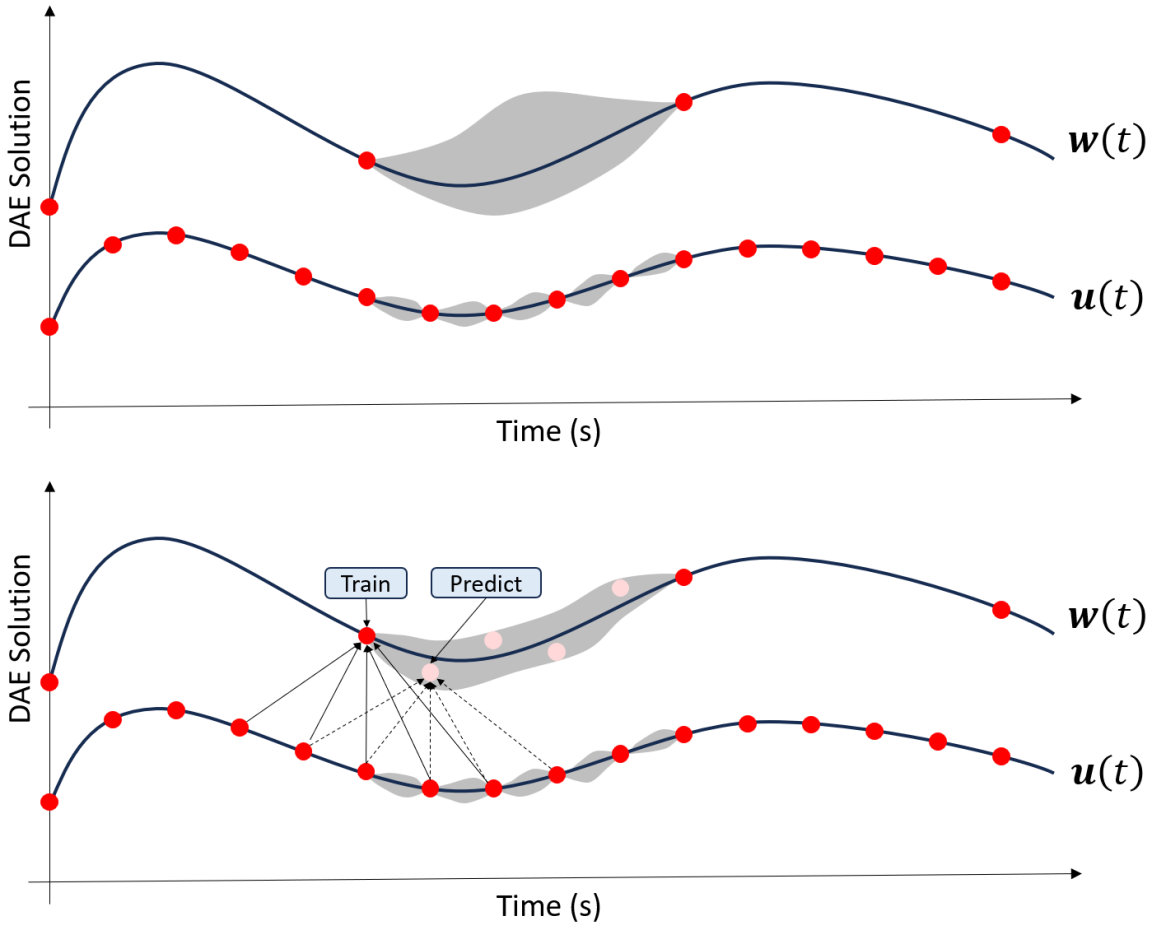


Figure 21. Conceptual Illustration of the Effectiveness of Spatial-temporal Restrictions.

Specifically, I divide the input data $\left\{\left[\boldsymbol{u}(t_i - \Delta \cdot K/2), \cdots, \boldsymbol{u}(t_i + \Delta \cdot K/2)\right]\right\}_{i=K/2}^{N_u - K/2}$ into the labeled dataset $\mathcal{U}_1$ and the unlabeled dataset $\mathcal{U}_2$. Similarly, I denote the label (i.e., output value) set $\{\boldsymbol{w}(\tilde{t}_i)\}_{i=1}^{N_w}$ as $\mathcal{W}_1$. With $\mathcal{U}_1$, $\mathcal{U}_2$, and $\mathcal{W}_1$, there are many SSL algorithms to produce a highly generalizable regression model. In particular, pseudo label-based method is a simple yet effective procedure. One can refer to (Min and Tai 2022) for the theoretical proofs of the effectiveness of pseudo label-based SSL. Authors in (Min and Tai 2022) provide a good convergence property as long as unlabeled data set is large enough, satisfied in our setting. Hence, I propose to iteratively generate pseudo label set $\mathcal{W}_2^k$ for the k^{th} iteration and retrain $\boldsymbol{h}(\cdot)$ using $\mathcal{U}_1$, $\mathcal{U}_2$, $\mathcal{W}_1$, and $\mathcal{W}_2^k$ for the $(k+1)^{th}$ iteration, shown in Algorithm 3. The left part of Fig. 18 illustrates the SSL training for the spatial-temporal NN.

If well-trained, $\boldsymbol{h}(\cdot, \Omega)$ delivers a good estimation of interpolated data, namely, the pink nodes in

Algorithm 3 Pseudo Label-based Training.

Input: Labeled input data $\mathcal{U}_1$, unlabeled input data $\mathcal{U}_2$, and output label (value) set $\mathcal{W}_1$.

$k = 0$ Update Ω_k using input $\mathcal{U}_1$ and output $\mathcal{W}_1$. Not converge Predict $\mathcal{W}_2^k$ using $\boldsymbol{h}(\cdot; \Omega_k)$ and $\mathcal{U}_2$. $k \leftarrow k + 1$. Update Ω_k using input $\mathcal{U}_1/\mathcal{U}_2$ and output $\mathcal{W}_1/\mathcal{W}_2^k$. =0

Output: the trained spatial-temporal NN $\boldsymbol{h}(\cdot; \Omega^*)$.

Fig. 21. Then, the LR data with initial interpolation and HR data are synchronized. With slight abuse of notation, I denote $\{\bar{\boldsymbol{w}}(t_i)\}_{i=1}^{N_u}$ as the set of pink and red nodes. This dataset can be further used for training the PINN, which serves as adding physical constraints to the data imputation. Specifically, I have:

$$\begin{aligned}
\mathcal{L}_1(\Theta) &= \sum_i^{N_u} ||\boldsymbol{u}(t_i) - \boldsymbol{u}(t_i; \Theta)||_2^2 \\
&\quad + \sum_i^{N_u} ||\bar{\boldsymbol{w}}(t_i) - \boldsymbol{w}(t_i; \Theta)||_2^2. \\
\big[\boldsymbol{x}(t; \Theta), \boldsymbol{y}(t; \Theta)\big] &= \phi\big(\boldsymbol{u}(t; \Theta), \boldsymbol{w}(t; \Theta)\big), \\
\mathcal{L}_2(\Theta; \boldsymbol{f}, \boldsymbol{g}) &= \int_0^T \big|\big|\boldsymbol{g}\big(\boldsymbol{x}(t; \Theta), \boldsymbol{y}(t; \Theta)\big)\big|\big|_2^2 \\
&\quad + \big|\big|\dot{\boldsymbol{x}}(t; \Theta) - \boldsymbol{f}\big(\boldsymbol{x}(t; \Theta), \boldsymbol{y}(t; \Theta)\big)\big|\big|_2^2 dt, \\
\mathcal{L}(\Theta; \boldsymbol{f}, \boldsymbol{g}) &= \mathcal{L}_1(\Theta) + \lambda_1 \mathcal{L}_2(\Theta; \boldsymbol{f}, \boldsymbol{g}).
\end{aligned} \tag{3.5}$$

When the $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are unknown in Equation (3.5), the related parameters can be estimated via LASSO (Brunton, Proctor, and Kutz 2016). (Stanković et al. 2020) provides a good example of applying LASSO in power systems for dynamic identification. I hence introduce EC-LASSO for the estimation (Loh and Wainwright 2011) , which provides a certain tolerance for the errors of the interpolated data. Specifically, I utilize the first equation of $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{y}(t))$ as an example and all other equations have similar procedures. For the first equation with N_u samples or imputed data, I have the matrix format:

$$\dot{\boldsymbol{x}}_1 = (Z + E)\boldsymbol{\beta} + \boldsymbol{n}, \tag{3.6}$$

where $\dot{\boldsymbol{x}}_1 \in \mathbb{R}^{N_u}$ is the derivative of the first state variable in N_u time slots, $Z \in \mathbb{R}^{N_u \times M}$ is an augmented library of all nonlinear bases for state variables $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ (Brunton, Proctor, and Kutz 2016), where M is the number of nonlinear bases. For example, in power systems, I have second-order polynomials for the voltage magnitude and sinusoidal functions for the voltage angle. $E \in \mathbb{R}^{N_u \times M}$ is a error matrix because of data interpolation. $\boldsymbol{\beta} \in \mathbb{R}^M$ and $\boldsymbol{n} \in \mathbb{R}^{N_u}$ represent the

system dynamical parameters and the noise vector, respectively. Hence, EC-LASSO provides the following LASSO estimation with an error correction mechanism:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \frac{1}{N_u} ||\dot{\boldsymbol{x}}_1 - Z\boldsymbol{\beta}||_2^2 - \boldsymbol{\beta}^\top \Sigma_E \boldsymbol{\beta} + \lambda_2 ||\boldsymbol{\beta}||_1, \tag{3.7}$$

where $\top$ is the transpose operator, $\Sigma_E \in \mathbb{R}^{M \times M}$ is the covariance matrix of E and λ_2 is a positive penalty term. Usually, I don't have the exact covariance Σ_E. I thus introduce a proper estimation based on the statistics of Z for K times, i.e., $Z^1 \sim Z^K$. These repeated matrices can be obtained during our interpolation-estimation iteration, illustrated in Section 3.3.3. Mathematically, I can compute:

$$\hat{\Sigma}_E = \frac{1}{N_u} \sum_{n=1}^{N_u} \frac{\sum_{k=1}^{K} (Z_n^k - \bar{Z}_{n\cdot})(Z_n^k - \bar{Z}_{n\cdot})^\top}{(K-1)}, \tag{3.8}$$

where Z_n^k represents the n^{th} row of Z^k, and $\bar{Z}_{n\cdot} = \frac{1}{K} \sum_{k=1}^{K} Z_n^k$. Basically, I can utilize the sample covariance of $Z^1 \sim Z^K$ to approximate the error covariance (Loh and Wainwright 2011). Equations (3.7) and (3.8) provide a convex optimization problem for EC-LASSO as long as $N_u \geq M$. Finally, I combine PINN and EC-LASSO to coordinately interpolate data and estimate parameters, i.e., our CoPIE framework.

3.3.3 Optimization Algorithm for CoPIE

I integrate the above techniques to optimize CoPIE. First, the SSL-based learning delivers a warm start for interpolation. Then, I iteratively implement data imputation and physical parameter estimation using LASSO. Notably, for every K times, I record the nonlinear matrices $Z^1 \sim Z^K$. Then, I can conduct EC-LASSO to improve the parameter estimation for one time. Such a procedure is repeated until convergence. Eventually, the complete optimization algorithm for CoPIE is demonstrated in Algorithm 4.

3.4 Theoretical validations on pinn and CoPIE

In this section, I provide theoretical validations to quantify errors for PINN and CoPIE. In particular, I demonstrate what causes large errors for PINN and how CoPIE can resolve the issue with good initialization of missing values.

Algorithm 4 Optimization for the CoPIE Framework.

Input: A set of PMU samples $\{\boldsymbol{u}(t_i)\}_{i=1}^{N_u}$ and LR meter samples $\{\boldsymbol{w}(\tilde{t}_i)\}_{i=1}^{N_w}$.
Hyper-parameters: Penalty term λ_1 for Equation (3.5), λ_1 for Equation (3.7), number of iterations K, and DNN-related parameters like number of epochs, number of iterations, batch size, and learning rate.

Use Algorithm 3 to gain the trained spatial-temporal NN $\boldsymbol{h}(\cdot;\Omega^*)$ and the interpolated dataset $\{\bar{\boldsymbol{w}}(t_i)\}_{i=1}^{N_u}$. $k=0$. Use LASSO to estimate parameters of the DAE. Namely, use interpolated and true data to formulate Equation (3.7). Then, set $\Sigma_E=0$ and solve the optimization to obtain $\boldsymbol{f}^k(\cdot)$ and $\boldsymbol{g}^k(\cdot)$. Train CoPIE with $\boldsymbol{f}^k(\cdot)$ and $\boldsymbol{g}^k(\cdot)$ by minimizing $\mathcal{L}(\Theta;\boldsymbol{f}^k,\boldsymbol{g}^k)$ in Equation (3.5). Not converge $k=1$ to K Use current PINN to update the dataset $\boldsymbol{w}(t)$. Prepare the augmented matrix Z^k. Use LASSO to estimate the parameters of the DAE. Namely, use interpolated and true data to formulate Equation (3.7). Then, set $\Sigma_E=0$ and solve the optimization to obtain $\boldsymbol{f}^k(\cdot)$ and $\boldsymbol{g}^k(\cdot)$. Train CoPIE with $\boldsymbol{f}^k(\cdot)$ and $\boldsymbol{g}^k(\cdot)$ by minimizing $\mathcal{L}(\Theta;\boldsymbol{f}^k,\boldsymbol{g}^k)$ in Equation (3.5). Use prepared matrices $Z^1\sim Z^k$ to calculate estimate error covariance with Equation (3.8) and obtain $\hat{\Sigma}_E$. Use EC-LASSO to estimate the parameters of the DAE. Namely, replace Σ_E with $\hat{\Sigma}_E$ in Equation (3.7). Then, solve the optimization to obtain $\boldsymbol{f}^k(\cdot)$ and $\boldsymbol{g}^k(\cdot)$. Train CoPIE with $\boldsymbol{f}^k(\cdot)$ and $\boldsymbol{g}^k(\cdot)$ by minimizing $\mathcal{L}(\Theta;\boldsymbol{f}^k,\boldsymbol{g}^k)$ in Equation (3.5). $=0$

Output: the estimated DAEs and the optimal parameter set Θ^* in CoPIE to interpolate the missing values.

While Equation (3.1) makes it hard for further derivations, I assume a linearization for $\boldsymbol{g}(\cdot)$, e.g., linearized power flow equations. Then, I can substitute $\boldsymbol{y}(t)$ with respect to a linear expression of $\boldsymbol{x}(t)$ so that I convert the DAE in Equation (3.1) to Ordinary Differential Equations (ODEs):

$$\dot{\boldsymbol{x}}(t)=\tilde{\boldsymbol{f}}(\boldsymbol{x}(t)). \tag{3.9}$$

I further make the following assumption on $\tilde{\boldsymbol{f}}(\cdot)$ to guarantee the unique solution:

Assumption 1. *The function $\tilde{\boldsymbol{f}}(\cdot)$ is Lipschitz continuous with respect to $\boldsymbol{x}(t)$.*

I denote $L>0$ as the Lipschitz constant. With slight abuse of notation, I denote the PINN for the ODE as $\boldsymbol{x}(t;\Theta)$, parameterized by Θ. Activated by sigmoid functions, $\boldsymbol{x}(t;\Theta)$ is sufficiently smooth. Then, the residual of the approximated solution minus the true solution exists. Furthermore, I denote minimal the residual term and error term as:

$$\begin{aligned}\boldsymbol{r}(t)&=\dot{\boldsymbol{x}}(t;\Theta^*)-\tilde{\boldsymbol{f}}(\boldsymbol{x}(t;\Theta^*)),\\ \boldsymbol{e}(t)&=\boldsymbol{x}(t)-\hat{\boldsymbol{x}}(t;\Theta^*),\end{aligned} \tag{3.10}$$

where I denote Θ^* as the (sub)-optimal parameter set after full training. Subsequently, I propose the following theorem to demonstrate the error bound for PINN.

Theorem 2. *With the given N_w samples of $\boldsymbol{x}(t)$, constructed by the synchronized measurements of PMUs and LR meters, the approximation error term of the PINN has the following upper bound:*

$$\begin{aligned} ||\boldsymbol{e}(t)||_2 \leq e^{\frac{LT}{N_w}} \max(\{||\boldsymbol{e}(t_i)||_2\}_{i=1}^{N_w}) \\ + \frac{e^{\frac{LT}{N_w}}}{L} \max(\{||\boldsymbol{r}(t)||_2 : 0 \leq t \leq T\}), \end{aligned} \tag{3.11}$$

where T introduced in Section 3.3.1 is the end point of the system dynamical interval.

Proof. The proof is a direct extension of Theorem III.1 in (Hillebrecht and Unger 2022). □

Despite that the PINN tries to minimize $||\boldsymbol{e}(t_i)||_2$ and $||\boldsymbol{r}(t)||_2$ via boundary-condition loss and function-form loss, a relatively small N_w makes the exponential term large. Generally, the PINN with only N_w samples suffer large errors. If I consider the additional $N_u - N_w$ measurements of PMUs, i.e., a small subset of entries for $\boldsymbol{x}(t)$, I can reduce the error bound, indicated by the following theorem:

Theorem 3. *With the given N_u samples of PMUs and N_w samples of LR meters, the approximation error term of the PINN has the following upper bound:*

$$\begin{aligned} ||\boldsymbol{e}(t)||_2 &= ||\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}(t;\Theta^*) + \tilde{\boldsymbol{x}}(t;\Theta^*) - \boldsymbol{x}(t;\Theta^*)||_2 \\ &\leq e^{\frac{LT}{N_u}} \max(\{||\boldsymbol{e}_2(t_i)||_2 + ||\boldsymbol{e}_1(t_i)||_2\}_{i=1}^{N_u}) \\ &+ \frac{2e^{\frac{LT}{N_u}}}{L} \max(\{||\boldsymbol{r}(t)||_2 : 0 \leq t \leq T\}), \end{aligned} \tag{3.12}$$

where I utilize the subscript 1 and 2 to represent the PMU and LR entries, respectively. Furthermore, I define $\tilde{\boldsymbol{x}}(t;\Theta)$ as follows:

$$\tilde{\boldsymbol{x}}(t;\Theta) = [\boldsymbol{x}_1(t), \boldsymbol{x}_2(t;\Theta^*)]. \tag{3.13}$$

Proof. To prove Theorem 3, I use a triangle inequality to upper bound the error. Then, I only need to employ the conclusion of Theorem (2) twice to conclude the proof. □

Since $N_u \gg N_w$, the error in Equation (3.12) is much smaller than that in Equation (3.11). However, it should be noted that the term $||\boldsymbol{e}_2(t_i)||_2$ is not minimized during the training, given that there are no measurements of LR variables for the $N_w - N_u$ time slots. Therefore, the approximation error is still non-negligible.

To further reduce the error, I introduce our CoPIE. Obviously, the significant improvement of CoPIE lies in the spatial-temporal restriction and a powerful SSL framework that enforce $\boldsymbol{x}_2(t;\Theta^*) \approx$

$\boldsymbol{x}_2(t)$. Hence, I can assume there is a small constant $\delta > 0$ such that $||\boldsymbol{e}_2(t_i)||_2 \leq \delta, \forall i \in \{1, 2, \cdots, N_u\}$. Subsequently, I have the following error bound for CoPIE.

Theorem 4. *Assume there exists a small constant $\delta > 0$ such that $||\boldsymbol{e}_2(t_i)||_2 \leq \delta, \forall i \in \{1, 2, \cdots, N_u\}$, where $||\boldsymbol{e}_2(t_i)||_2$ is defined in Equation* (3.12)*. With the given N_u samples of PMUs and N_w samples of LR meters, the approximation error of the CoPIE has the following upper bound:*

$$\begin{aligned} ||\boldsymbol{e}(t)||_2 &= ||\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}(t; \Theta^*) + \tilde{\boldsymbol{x}}(t; \Theta^*) - \boldsymbol{x}(t; \Theta^*)||_2 \\ &\leq e^{\frac{LT}{N_u}} \max(\{\delta + ||\boldsymbol{e}_1(t_i)||_2\}_{i=1}^{N_u}) \\ &+ \frac{2e^{\frac{LT}{N_u}}}{L} \max(\{||\boldsymbol{r}(t)||_2 : 0 \leq t \leq T\}), \end{aligned} \tag{3.14}$$

where I utilize the subscript 1 *and* 2 *to represent the PMU and LR entries, respectively. Furthermore, $\tilde{\boldsymbol{x}}(t; \Theta)$ is defined in Equation* (3.13)*.*

Proof. The proof is a direct extension of Theorem 3. □

Since δ is a very small constant, the approximation error for CoPIE is very small. This validates our CoPIE design.

3.5 Experiments

3.5.1 System Description

In the experiments, I include 5 different systems: an RLC circuit, IEEE 4-bus, 9-bus, 34-bus, 118-bus, and 123-bus system. These models cover circuit, transmission and distribution ODEs to depict the system dynamics.

RLC Circuit: The RLC circuit is a second-order ODE where the current is the state variable. The resistance, inductance, and capacitance are the physical parameters. The ground-truth result of the voltage data with respect to time can be written in a closed-form solution. I assume one node has 60 samples per second sampling rate while the other has 2 samples per second.

IEEE 4-bus system: The 4-bus system is introduced in (Misyris, Venzke, and Chatzivasileiadis 2020), where there are 2 generators with the second-order swing equation and the frequency dependent loads with the first-order ODE. Voltage angle is treated as the state variable. The parameters and

the initial values of the system can be found in (Misyris, Venzke, and Chatzivasileiadis 2020). I can use the ode45 solver in MATLAB to obtain the ground-truth solution. The solution is accurate as long as I set a very small step size.

IEEE 9-bus & 118-bus transmission systems: The parameters of IEEE 9-bus can be seen in (Wang, Liu, and Sun 2015). The system considers the swing equation as well as the power balance constraints in the DAE equations. Moreover, line parameters after a short-circuit fault are also provided. Hence, I can conduct interpolation for the fault dynamics. The IEEE 118-bus system can be referred to (Schneider et al. 2017). For both systems, I utilize the DAE solver in MATLAB to conduct the simulation. Similarly, I can achieve accurate results as long as I set the step size to be very small. For the transmission systems, I put PMUs at the generators with 60 samples per second. Then, I put RTUs at load buses with 2 samples per second.

IEEE 34-bus & 123-bus distribution system: I also simulate distribution feeders of the IEEE 34-bus and 123-bus systems (Schneider et al. 2017). Similarly, I put micro-PMUs and LR meters like smart meters at the generation and load buses, respectively. However, I assume micro-PMUs have 60 samples per second and increase the LR meters resolution to a relatively reasonable value, e.g., 2 samples per second.

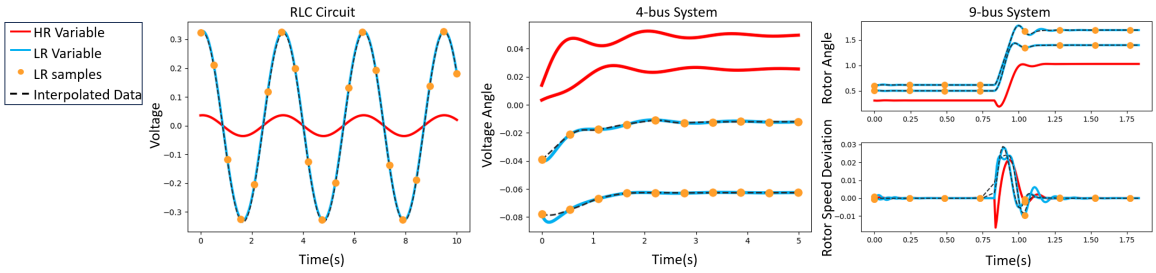


Figure 22. Visualizations of the Interpolated Results of Our CoPIE. Samples of Hr Variables Are Hidden Since They Are Too Dense.

3.5.2 Benchmark Methods

In this subsection, I introduce alternative benchmark methods as follows:

- **Deep Neural Network (DNN)**(C.-Y. Cheng et al. 2020): DNN can be directly utilized to train the mapping from HR to LR data. DNN can capture spatial and temporal correlations effectively. However, the lack of dynamic information in LR data hinders the DNN's effectiveness.

- **Physics Informed Neural Networks (PINN)** (Raissi, Perdikaris, and Karniadakis 2019): PINN, as a DAE solver, can efficiently incorporate measurements and enforce DAE constraints to the solution. However, PINN has convergence issues in the face of LR data.
- **Hd-Deep-EM** (Z. Ma et al. 2023): Hd-Deep-Em utilizes the EM algorithm to gradually interpolate data and utilize data to train a mapping from HR to LR variables. Hence, this model is similar to our SSL-based training for the spatial-temporal NN $\boldsymbol{h}(\cdot;\Omega)$. However, this method can't provide physics consistency, and there is no guarantee for the performance.
- **Spline Interpolation**: Spline interpolation (Wahba 1981) is an effective interpolation method that employs piecewise-defined polynomials. Spline interpolation is revered for its flexibility and smoothness. It represents an optimal choice when the data demonstrate a smooth nature but may under-perform when there are abrupt changes or discontinuities. Moreover, the result may not satisfy the underlying DAEs.
- **Cubic Spline Interpolation**: It's a specific type of spline interpolation that applies cubic polynomials to approximate the function between data points (McKinley and Levine 1998). Cubic spline interpolation guarantees the continuity of the first and second derivatives across adjacent intervals, yielding a smooth curve that well represents the data. However, it requires substantial computation when dealing with a large dataset. Similarly, there is no physics consistency.

Finally, I evaluate different methods by calculating the Mean Absolute Percentage Error (MAPE (%)) between the interpolated data and the true data. I also use MAPE to measure the parameter estimation performance of EC-LASSO.

3.5.3 CoPIE Visualization for Data Imputation: Excellent Match for Different Systems

In this subsection, I first visualize the interpolated data using CoPIE. Fig. 22 illustrates the interpolated data for the RLC circuit, 4-bus, and 9-bus systems. The left plot is the result of the RLC circuit. Clearly, the voltage signal is a sinusoidal waveform. I assume one node has 60 samples per second in the red curve while the other has 2 samples per second (orange nodes) in the blue curve. Clearly, I find that the interpolated data (dotted line) almost perfectly fits the blue curve. This shows that our PINN can approximate the sinusoidal function well.

For the IEEE 4-bus system, the red curves with dense HR samples are the generator voltage angles while the blue curves are the load bus voltage angles, shown in the middle plot of Fig. 22. The swing equation can gradually make the voltage angle at different buses stable. After I input these data to CoPIE, I observe the interpolated line (i.e., dotted line). I observe that there is a small mismatch at the beginning of the curves. However, other periods have a good match. The mismatch is mainly due to our training setting: I fix the training iterations for all scenarios. If I keep training, it's clear that the dotted line will drop and fit the blue curves.

For the IEEE 9-bus system, the dynamics are more complicated since I add a three-phase fault near bus 7 at the end of line $5-7$. In the right plot of Fig. 22, I can observe the fault dynamics at around $0.75 \sim 1.25$s. The comparison between the dotted lines and the blue lines reveals that the only mismatch happens around $0.75 \sim 0.8$s for the rotor speed deviations. Again, if I enable more training, such mismatch will be reduced.

3.5.4 The General Performance for Different Data Imputation Methods

In this subsection, I still fix the sampling rates for the HR and LR samples to be 60Hz and 2Hz, respectively. Then, I evaluate the performance of different methods. It's noteworthy that PINN is the only method that requires the DAE, and our CoPIE has the option to use the DAE or not since I have a built-in estimation block. Table 8 presents the result.

Table 8. Interpolation Mape(%) of All Methods on Different Test Systems.

Method	RLC	4-bus	9-bus	34-bus	118-bus	123-bus
CoPIE w/ DAE	0.85	0.05	0.30	1.53	1.35	1.49
CoPIE w/o DAE	0.86	0.06	0.31	1.55	1.72	1.88
PINN w/ DAE	1.21	1.81	3.06	5.39	11.20	12.27
Hd-Deep-EM	1.30	2.04	3.67	8.20	13.35	11.80
DNN	2.92	2.94	5.74	20.86	23.81	22.67
Cubic Spline	0.93	0.04	0.31	24.65	37.47	23.86
Spline	0.95	0.06	0.36	27.57	32.53	24.00

I have the following observations. First, if I compare CoPIE with or without DAE parameters, I find that the difference is insignificant. This is because our embedded EC-LASSO block can efficiently recover the DAE parameters (see more details in Section 3.5.5). As a consequence, the performance of the two methods will converge with enough iterations and an optimized learning rate. Second,

while PINN uses the DAE functions, its errors are much larger than those of CoPIE, especially for large systems. This is because the LR samples can hardly help to reduce the accumulative errors for PINN as a DAE solver, which becomes a bottleneck for all solvers.

Third, the correlation-based methods yield relatively large errors because they don't have physics consistency. Since Hd-Deep-EM has an SSL framework, it's better than other correlation-based methods. The worst case is the Spline method without biased assumptions and no physical guarantees. Therefore, I can draw the conclusion that only CoPIE brings the possibility to turn LR meters into virtual PMUs.

3.5.5 Effectiveness of EC-LASSO to Estimate Parameters

In this subsection, I investigate the effectiveness of EC-LASSO for the DAE parameter estimation. Specifically, I still fix the HR/LR sampling rates to be 60/2Hz, respectively. Then, I set $K = 5$ in Algorithm 4 and record the result of each iteration that runs K times the inner loop. Namely, I present the MAPE(%) between the estimated and the true parameters during each iteration. Fig. 23 illustrates the result. For most cases, it takes only less than 5 iterations to make the MAPE less than 5%. This is because EC-LASSO can effectively utilize the interpolated data with an error correction and generate accurate estimation.

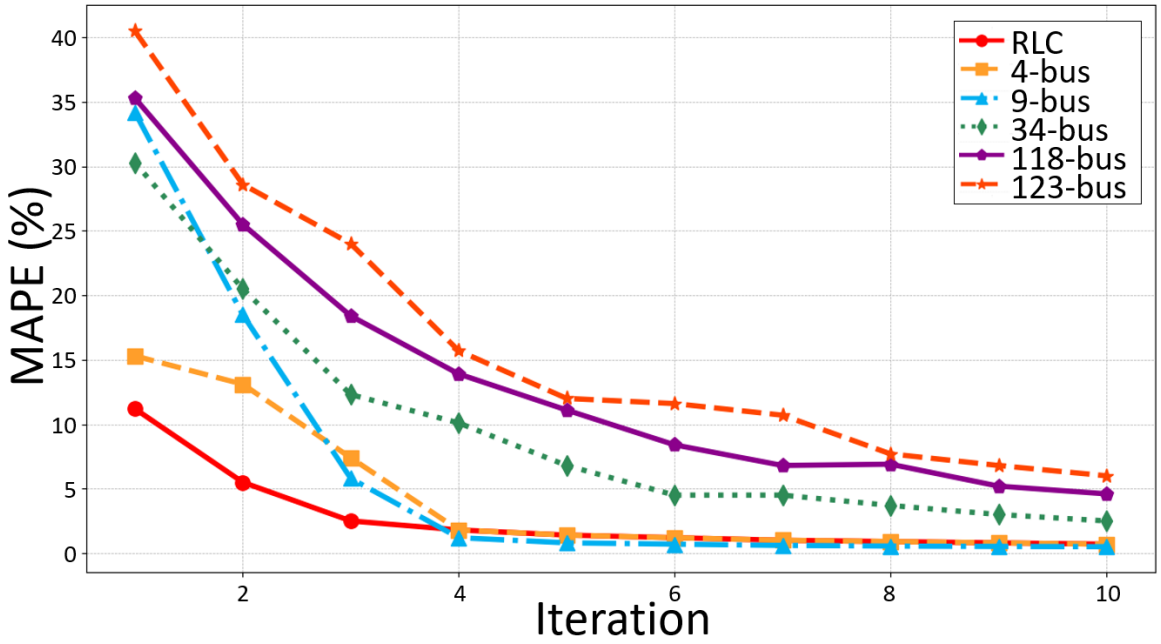


Figure 23. Mape(%) of the Parameter Estimation in Each Iteration.

3.5.6 Sensitivity Analysis with Respect to Different Sampling Resolution Ratios

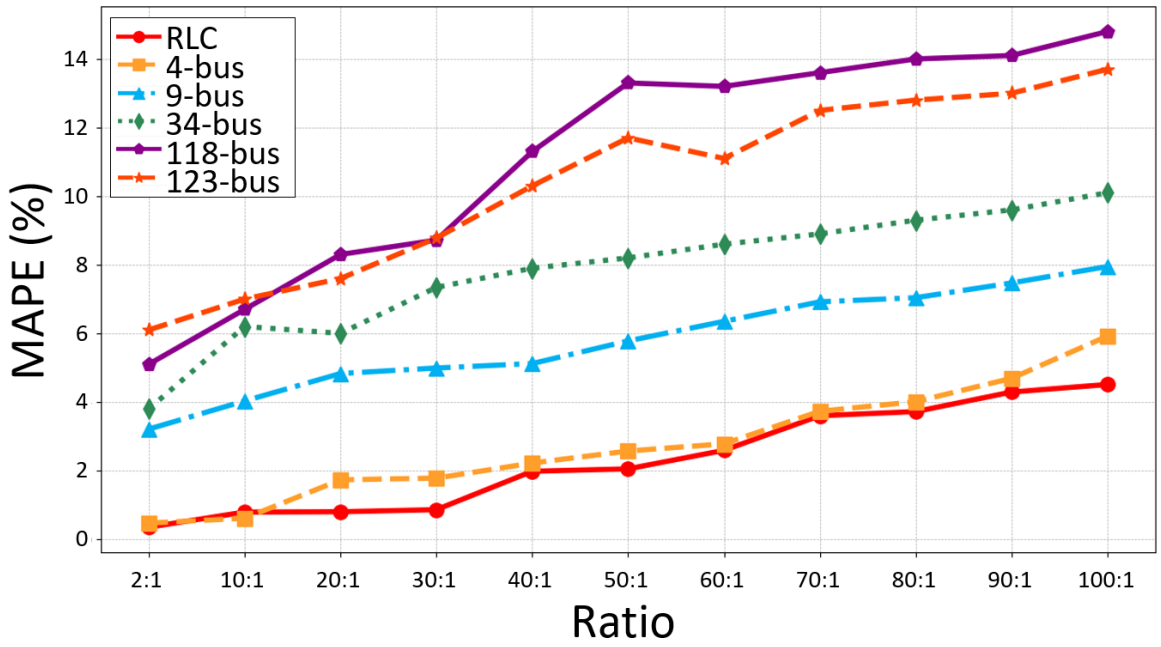


Figure 24. Mape(%) with Respect to the Sampling Rate Ratio.

In this subsection, I test the performance of CoPIE with different sampling rate. Specifically, I fix the HR sampling rate to be 60Hz and vary the sampling rate ratio of HR to LR as follows: $\{2:1, 10:2, 20:1, \cdots, 90:1, 100:1\}$. Clearly, when the ratio is larger, there are more LR samples and the interpolation error should be lower. Experimentally, I plot the MAPE (%) with respect to the ratio in Fig. 24. The results imply that the MAPE can be maintained at less than 15% even when the ratio is $100:1$. Thus, I only require a very limited LR sampling rate so that I can turn LR meters into virtual PMUs with an assured error bound. Second, I discover that the error increases near linearly or sub-linearly. According to our Theorem 2, traditional PINN has exponential growth. Only when I utilize CoPIE can I achieve linear or sublinear error bounds. More specifically, Theorem 4 shows that this error is only related to the HR samples and the approximation error. Hence, if I increase the LR sample rate, I essentially reduce the approximation error of the spatial-temporal NN, i.e., δ in Theorem 4, which is almost a linear reduction.

Chapter 4

EVENT DETECTION: EXPLORING INTRINSIC DATA STRUCTURES WITH TENSOR LEARNING

4.1 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, a 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) (Yuan, Wang, and Wang 2020). 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 (Haoran Li et al. 2019b) can detect abnormal intervals that imply events (e.g., line trip, generator trip, single-phase-to-ground fault, phase-to-phase fault, and three-phase fault). However, to know more information about event types and locations, it's challenging to analyze high-dimensional PMU streams in the best way. One idea to find event types and locations is to

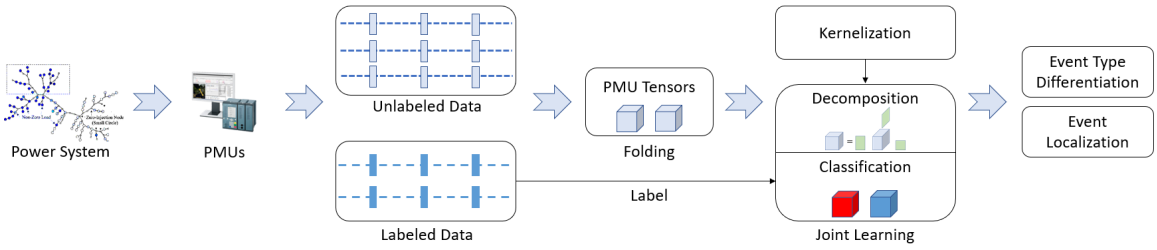


Figure 25. KTDC-Se Flowchart to Tackle Data (1) Correlation, (2) Volume, and (3) Unlabeled Processing.

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 predefined thresholds. These methods use wavelet transformation (D.-I. Kim et al. 2015), Kalman filtering (Pérez and Barros

2008), and Swing Door Trending (SDT) (Cui et al. 2018), etc. For example, (Cui et al. 2018) 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 I 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) (Haoran Li et al. 2019a) 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, (De Yong, Bhowmik, and Magnago 2015) 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) (Yuxuan Yuan et al. 2021a), and Graph Neural Network (GNN) (Yuan, Wang, and Wang 2020). 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, (S. Zhang et al. 2018) 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) quantities 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 (L. He et al. 2017; Sidiropoulos et al. 2017), e,g., Principal Component Analysis (PCA) (Sidiropoulos et al. 2017; Abdi and Williams 2010) and Independent Component Analysis (ICA) (Nordhausen and Oja 2018). The pre-processing can also select core statistics, including mean values (Ma, Basumallik, and Eftekharnejad 2020) and wavelet basis coefficients (Kim 2021), etc. I further categorize these data processing techniques by their treatments along the temporal and spatial dimensions. In the time domain, (Kapoor et al. 2018) utilizes Kullback–Leibler (KL) divergence to measure the distribution change for time-series measurements to detect events. (Hafiz et al. 2017) studies wavelet transformation families to

transform signals into features. Then, these features are input to K-Nearest Neighbors (KNN) and Naive Bayes (NB) to classify events. Similarly, (Shaw and Jena 2020) employs wavelet transformation and Kalman filtering to process frequency measurements and produce features. (Yadav, Pradhan, and Kamwa 2018; Z. Li et al. 2021) utilize Teager-Kaiser energy operator on PMU streams to identify event time and locate events. Subsequently, PMUs that are close to events are selected as input to KNN and LSTM.

In the spatial domain, (Weikang Wang et al. 2020) converts the frequency and angle measurements into an image and utilizes CNN to capture the local features. (D. Ma et al. 2019) develops a graph theory-based network partitioning algorithm to group different buses and locate events. More studies investigate both dimensions together. (Shi, Foggo, and Yu 2021; Y. Cheng et al. 2022) develop a Graph Signal Processing (GSP) method to capture temporal correlations of PMU measurements and reorganize PMUs based on the magnitude of the temporal correlations. This re-organization enables the following supervised learning model to better extract spatial information and identify events. Some researchers store data in a 2-dimensional matrix and study the corresponding matrix-based algorithms. (S. Liu et al. 2021) develops the two-Dimensional Orthogonal Locality Preserving Projection (2-D-OLPP)-based method to extract spatial-temporal correlations. (Li, Wang, and Chow 2018) calculates the subspace of the PMU data matrix and produces measures over the subspace for event identification. (Ge et al. 2015; Rafferty et al. 2016; S. Liu et al. 2019) employ PCA for the data matrix to yield compact feature vectors. In general, these methods are diversified and have different feature extraction capacities based on their own biases. There is a need to come up with a unified and efficient design to capture the complex correlations of synchrophasor data among the space, time, and measurement type domains.

In this chapter, I propose to merge the two steps of 3-D 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 (Ma, Basumallik, and Eftekharnejad 2020; Kim 2021; Yuxuan Yuan et al. 2021a). Based on such an idea, I design a tensor learning framework to extract physically meaningful features with simple computations. Fig. 25 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 via tensor decomposition. Hence, tensor unfolding

can covert the decomposed core tensor to vectors for classification, enabling an end-to-end model in Fig. 25. While tensor learning can extract key information quickly and systematically, I also want to preserve the nice property of nonlinear feature extraction, like the property in CNN. For such a purpose, I mathematically derive kernelization 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, (Brahma et al. 2017) 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 to take advantage of widely available unlabeled data in power systems, shown in the middle part of Fig. 25. But, there are challenges in integrating Semi-supervised learning and kernel-based tensor learning. For example, I need to restrict the same decomposition model for labeled and unlabeled data. I achieve the integration by aggregating all 3-D PMU tensors into a 4-D tensor, where the additional dimension is the number of PMU tensors. 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, I develop an efficient coordinate descent method. Finally, when the tensor number is significantly large, I modify our training method based on a mini-batch-based training scheme to save 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 (Li, Ma, and Weng 2022), 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, I have the following contributions:

- Design the KTDC-Se model to incorporate massive labeled and unlabeled PMU streams. Employ tensors to uncover the complex multi-dimensional correlations and create compact and informative features to identify events;
- Derive a fast coordinate descent algorithm and its variational mini-batch version to train our KTDC-Se; and

- Conduct extensive experiments to demonstrate the high performance of KTDC-Se over other models with synthetic and real-world datasets.

To emphasize the contributions, I 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 chapter is organized as follows: Section 4.2 introduces the notations and tensor preliminaries for the model. Section 4.3 defines the problem. Section 4.4 proposes our KTDC-Se. Section4.5 illustrates the learning algorithm. Section 4.6 conducts experiments for baselines and KTDC-Se.

4.2 Notations and Preliminaries

To integrate PMU data reduction and machine learning in one tensorized framework, I first introduce the basics of tensor algebra and the corresponding notations (Kolda and Bader 2009; Blasch et al. 2021). To summarize, Table 9 presents the basic notations for different types of variables and operations.

4.2.1 Tensor Notations

Multi-mode data can be stored in the so-called tensor (Lock 2018), 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 $\mathcal{X}$.

h!

Table 9. Overview of Tensor Notations and Operators.

Notation	Interpretation
a	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
$\|\cdot\|_2$	l_2 norm of a vector
$\|\cdot\|_F$	l_2 Frobenius norm of a matrix or a high-order tensor

4.2.2 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 (Liao et al. 2018). 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 (Kolda and Bader 2009):

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

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 the KTDC-Se 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.

4.3 Problem Formulation

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

Data Preparation of PMU Tensors. For PMU streams, I follow the idea of (Yuan, Wang, and Wang 2020; Shi, Foggo, and Yu 2021; Ma, Basumallik, and Eftekharnejad 2020) to extract a window of PMU signals with sufficient information to indicate the event dynamics for training a classifier. As shown in Fig. 36, 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, I 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. 36. 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. I focus on the tasks of distinguishing event types and locating events. Since I 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, I 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, I randomly create 2 different locations in the system. Similar treatments are implemented in many related studies (**li2022domain**; Ma, Basumallik, and Eftekharnejad 2020; Kim 2021; Shi, Foggo, and Yu 2021; Li, Ma, and Weng 2022).

Model Capacity of Tackling Multi-class Events. Our proposed method can handle events of multi-classes. W e emphasize that in the following modeling process, each of our trained classifiers is binary to make better use of kernel methods with hinge loss for high accuracy, which is similar to Support Vector Machine (SVM) (Sánchez A 2003). 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 for training multiple classifiers. Specifically, I 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 (Hsu and Lin 2002) for this method which has better performances than other multi-class SVMs.

Model Capacity of Tackling Cascading Events. Identifying cascading events is important for power systems. For cascading events, the time interval between two events usually lasts for minutes or hours (Bhatt et al. 2009). However, our model considers a time interval within a few seconds (e.g., 0.5s in our experiments) because our goal is to conduct fast event identification. Thus, one approach is to treat cascading events as independent events. Then, our training process can find the correct decision boundary to classify events, including both single and cascading events. The above process focuses on quickly *identifying* events but ignores sequentially *predicting* cascading events. While the latter task is out of our study scope, our model still has a certain capacity to predict cascading failures with enough event files and labels.

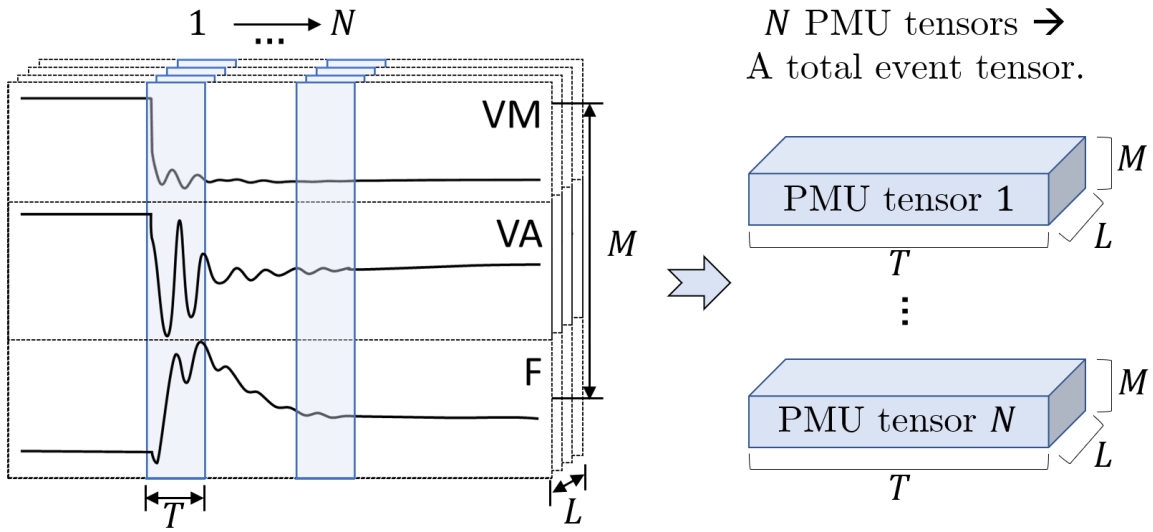


Figure 26. Illustration of the Moving Window-based Division to Generate PMU Tensors.

Specifically, one can utilize a two-digit label y_1y_2 where y_1 represents the current event status and y_2 represents the following event in an event sequence. If the current event status is a single event, I can treat $y_2 = 0$, which indicates the following system condition is normal. Subsequently, the two-digit label needs to be converted to an integer by considering all the labels in the training dataset for multi-class classification. Finally, as claimed above, I train multiple binary models to vote for the integer label. In general, the above process utilizes a data-driven way to estimate the correlations of cascading events during training for cascading event predictions. As long as I have enough data, our tensor-based information compression and feature extraction lead to efficient event identification and prediction.

Above notations and treatments summarize the common scenarios for power system event identification and how KTDC-Se proposed model can handle them. In general, I 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 $\boldsymbol{y}$.

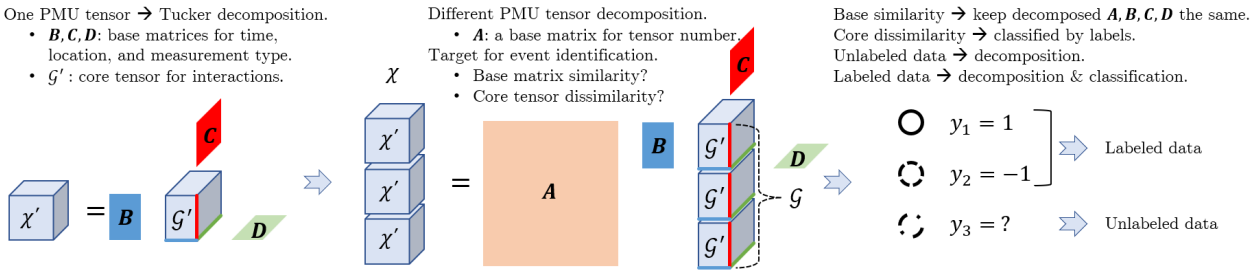


Figure 27. The Motivation of the KTDC-Se Model.

4.4 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, I 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.

4.4.1 An Integrated Model with Efficiency and Physical Meanings

For an efficient model, I need to remove the redundancy in PMU measurements. Section 4.1 shows the drawback of traditional methods: they can hardly explore the high-dimensional correlations. Further, Section 4.2 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 I can design an efficient model to remove redundancy and capture event information for different PMU event tensors, and how I can guarantee the model robustness by tackling some labeled and rich unlabeled tensors.

For a detailed design, I show the motivation in Fig. 27. I utilize different colors to represent different components of data. More specifically, I utilize light blue to represent tensor data $\mathcal{X}^{'}$ and $\mathcal{G}$. Then, I 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, I 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) I 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. 27 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 (Sidiropoulos et al. 2017). Thus, I can assume bases for different PMU tensors are similar as long as the number of bases is sufficient enough. In contrast, the interaction tensor $\mathcal{G}$ contains discriminative event information.

Then, to maximally remove the redundancy, I 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. 27. Namely, I 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, I want the core tensor $\mathcal{G}$ to contain distinguished event information. Therefore, I 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.

4.4.2 Semi-supervised Optimization for the Joint Model

In the semi-supervised learning setting, the 4-D total event tensor $\mathcal{X}$ in Section 4.4.1 contains all labeled and unlabeled data. Mathematically, I implement the Tucker decomposition for $\mathcal{X}$ as follows:

$$\begin{aligned} \mathcal{X} &\approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \times_4 \boldsymbol{D} \\ &\approx \sum_{r_1=1}^{N} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} \mathcal{G}(r_1, r_2, r_3, r_4) \mathbf{a}^{r_1} \circ \mathbf{b}^{r_2} \circ \mathbf{c}^{r_3} \circ \mathbf{d}^{r_4}, \end{aligned} \tag{4.1}$$

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 (4.1), 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:

$$\boldsymbol{X}_{(1)} \approx \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{D} \otimes \boldsymbol{C} \otimes \boldsymbol{B})^{\top}, \tag{4.2}$$

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, I utilize the labeled tensors with labels for the classification. Then, I propose a joint decomposition-classification model.

$$\begin{aligned} \min_{\boldsymbol{G}_{(1)}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{w}, b} J = & \underbrace{||\mathcal{X} - \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \times_4 \boldsymbol{D}||_F^2}_{J_1,\ \text{Reconstruction Loss}} \\ & + \gamma_1 \underbrace{l(\boldsymbol{E}\boldsymbol{G}_{(1)} \cdot \boldsymbol{w} + b, \boldsymbol{y})}_{J_2,\ \text{Classification Loss}} + \gamma_2 \underbrace{||\boldsymbol{w}||_2^2}_{J_3,\ l_2\ \text{Norm}}, \end{aligned} \tag{4.3}$$

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 (L. He et al. 2017), respectively. γ_1 and γ_2 are positive hyper-parameters to reweight the three terms in Equation (4.3).

4.4.3 Efficient Kernelization for Powerful Non-linear Feature Extractions

In the last two subsections, I 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 (4.3) significantly increases the computations. This motivates the usage of kernel trick for nonlinear feature calculations.

Specifically, (i) the kernel trick shows that I 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 (4.3), 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 (4.3) is a feature vector $\boldsymbol{f}$ with the dimension $r = R_2 \cdot R_3 \cdot R_4$. Then, if I consider utilizing d-degree polynomial features to represent the original features, I 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, I 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, I 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 (Kung 2014).

Specifically, due to the Representer theorem (Schölkopf, Herbrich, and Smola 2001), 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:

$$
\begin{aligned}
\min_{\boldsymbol{G}_{(1)}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}, b} J = & \underbrace{||\mathcal{X} - \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \times_4 \boldsymbol{D}||_F^2}_{J_1,\ \text{Reconstruction Loss}} \\
& + \gamma_1 \underbrace{\sum_{i=1}^{H} \left[1 - y_i (\sum_{j=1}^{H} \alpha_j k(\boldsymbol{f}_i, \boldsymbol{f}_j) + b)\right]_+}_{J_2,\ \text{Classification Loss}} \\
& + \gamma_2 \underbrace{\sum_{i=1}^{H} \sum_{j=1}^{H} \alpha_i \alpha_j k(\boldsymbol{f}_i, \boldsymbol{f}_j)}_{J_3,\ \text{Regularization}},
\end{aligned}
\tag{4.4}
$$

where $\boldsymbol{\alpha}$ is the vector of all α_is. Eventually, I 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.

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, I can assign these new events a new label that means "to be determined"■ By doing this, I 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. For power systems, the frequency of different events can vary significantly, leading to an imbalanced dataset. Then, the classifier may be biased towards event types with more data. To mitigate the imbalanced issue, there are approaches including reweighting (Kotsiantis 2011; Wentao Wang et al. 2021; Ren et al. 2018), cost-sensitive training (Shen et al. 2015; Tang et al. 2009; Zadrozny, Langford, and Abe 2003), and learning rate adaptation (He and Garcia 2009). The reweighting technique aims to balance the different classes by modifying the weights of the samples. The cost-sensitive training tries to assign higher costs for misclassifying the minority class. The learning rate adaptation adjusts the learning rate for gradient-based methods and encourages more weight updating for the minority class. Luckily, these three types of methods are applicable to our framework.

4.5 Learning Algorithm

The optimization in Equation (4.4) 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(Kolda and Bader 2009; L. He et al. 2017; Cao et al. 2016). Further, to calculate the gradient and avoid the non-differentiable scenario, the l_2 SVM (Tang 2015; L. He et al. 2017) 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.

4.5.1 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, I 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 (4.4) is calculated with respect to $\boldsymbol{A}$. Mathematically, the gradient is:

$$\begin{aligned} \nabla_{\boldsymbol{A}} J = \nabla_{\boldsymbol{A}} J_1 &= \nabla_{\boldsymbol{A}} \mathrm{tr}\big((\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{H}_1)^\top \cdot (\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{H}_1)\big) \\ &= 2(\boldsymbol{A}\boldsymbol{H}_1\boldsymbol{H}_1^\top - \boldsymbol{X}_{(1)}\boldsymbol{H}_1^\top), \end{aligned} \tag{4.5}$$

where $\mathrm{tr}(\cdot)$ 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:

$$\begin{aligned} \nabla_{\boldsymbol{B}} J &= 2(\boldsymbol{B}\boldsymbol{H}_2\boldsymbol{H}_2^\top - \boldsymbol{X}_{(2)}\boldsymbol{H}_2^\top), \\ \nabla_{\boldsymbol{C}} J &= 2(\boldsymbol{C}\boldsymbol{H}_3\boldsymbol{H}_3^\top - \boldsymbol{X}_{(3)}\boldsymbol{H}_3^\top), \\ \nabla_{\boldsymbol{D}} J &= 2(\boldsymbol{D}\boldsymbol{H}_4\boldsymbol{H}_4^\top - \boldsymbol{X}_{(4)}\boldsymbol{H}_4^\top). \end{aligned} \tag{4.6}$$

4.5.2 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:

$$\begin{aligned} \nabla_{\boldsymbol{G}_{(1)}} J_1 &= \nabla_{\boldsymbol{G}_{(1)}} \mathrm{tr}\big((\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{G}_{(1)}\tilde{\boldsymbol{H}})^\top \cdot (\boldsymbol{X}_{(1)} - \boldsymbol{A}\boldsymbol{G}_{(1)}\tilde{\boldsymbol{H}})\big) \\ &= 2(\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{G}_{(1)}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{H}}^\top - \boldsymbol{A}^\top\boldsymbol{X}_{(1)}\tilde{\boldsymbol{H}}^\top). \end{aligned} \tag{4.7}$$

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:

$$\nabla_{\boldsymbol{f}_i} J_2 = \begin{cases} 2(\hat{y}_i - y_i) \sum_{j=1}^{H} \alpha_j \frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\partial \boldsymbol{f}_i} & \\ +2\alpha_i \sum_{j \neq i}^{H} (\hat{y}_j - y_j) \frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\partial \boldsymbol{f}_i}, & \text{if } y_i\hat{y}_i < 1\ , \\ \boldsymbol{0}, & \text{if } y_i\hat{y}_i \geq 1\ . \end{cases} \tag{4.8}$$

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

$$\frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\partial \boldsymbol{f}_i} = \begin{cases} d(\boldsymbol{f}_i^\top \boldsymbol{f}_j + c)^{d-1} \boldsymbol{f}_j, & \text{if } i \neq j \ , \\ \\ 2d(\boldsymbol{f}_i^\top \boldsymbol{f}_j + c)^{d-1} \boldsymbol{f}_i, & \text{if } i = j \ . \end{cases} \tag{4.9}$$

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:

$$\frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\partial \boldsymbol{f}_i} = 2\lambda k(\boldsymbol{f}_i, \boldsymbol{f}_j)(\boldsymbol{f}_j - \boldsymbol{f}_i). \tag{4.10}$$

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:

$$\nabla_{\boldsymbol{f}_i} J_3 = \alpha_i^2 \frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_i)}{\partial \boldsymbol{f}_i} + 2\alpha_i \sum_{j \neq i} \alpha_j \frac{\partial k(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\partial \boldsymbol{f}_i}. \tag{4.11}$$

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., $\nabla_{\boldsymbol{G}_{(1)}} J = \nabla_{\boldsymbol{G}_{(1)}} J_1 + \gamma_1 \nabla_{\boldsymbol{G}_{(1)}} J_2 + \gamma_2 \nabla_{\boldsymbol{G}_{(1)}} J_3$.

4.5.3 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

$$\nabla_{\alpha_i} J_2 = \begin{cases} \sum_{j=1}^{H} 2(\hat{y}_j - y_j) k(\boldsymbol{f}_i, \boldsymbol{f}_j), & \text{if } y_i \hat{y}_i < 1 \ , \\ \\ 0, & \text{if } y_i \hat{y}_i \geq 1 \ . \end{cases} \tag{4.12}$$

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 $\mathbf{1}$ is an all-one column vector. Further, one can obtain a general format:

$$\nabla_{\boldsymbol{\alpha}} J_2 = 2\boldsymbol{K}\boldsymbol{I}^0(\boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y}), \tag{4.13}$$

where $\boldsymbol{I}^0$ satisfies

$$\boldsymbol{I}^0(i,j) = \begin{cases} 1, & \text{if } i = j \text{ and } y_i\hat{y}_i < 1 \ , \\ \\ 0, & \text{otherwise.} \end{cases} \tag{4.14}$$

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

$$\nabla_{\boldsymbol{\alpha}} J_3 = 2\boldsymbol{K}\boldsymbol{\alpha}. \tag{4.15}$$

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, I calculate the gradient with respect to b:

$$\nabla_b J = \mathbf{1}^\top \boldsymbol{I}^0(\hat{\boldsymbol{y}} - \boldsymbol{y}), \tag{4.16}$$

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 5 and Fig. 28.

Algorithm 5 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 γ_1 and γ_2, polynomial kernel parameters d and c, RBF kernel parameters λ, and learning rate lr.

Output: Parameters $\boldsymbol{G}^k_{(1)}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}$, and b.

4.5.4 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 (4.6), the Kronecker product to calculate $\boldsymbol{H}_2$, $\boldsymbol{H}_3$, and $\boldsymbol{H}_4$ requires a large storage for $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. To mitigate this issue, I modify Algorithm 5 to train on mini-batches to save the memory(Zhao et al. 2014; Li, Weng, and Tong 2022).

Mathematically, I 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

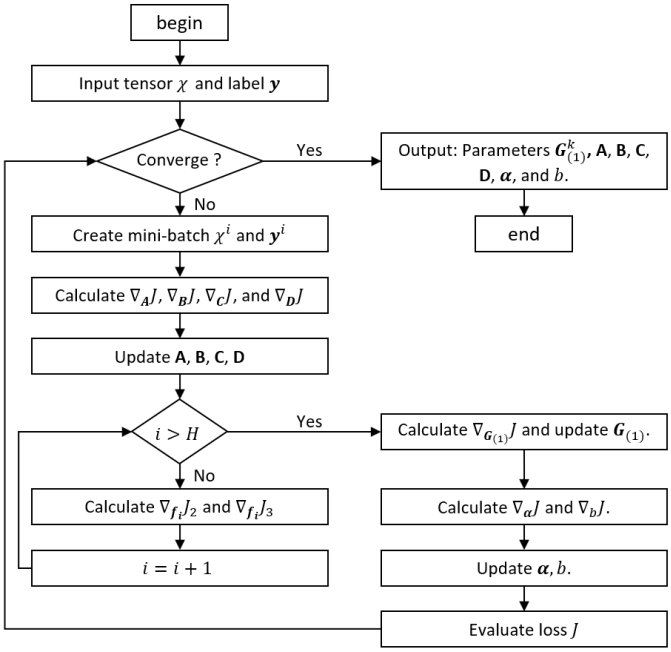


Figure 28. The Flowchart of the KTDC-SE Algorithm.

data. Then, in each iteration, I 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 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 5. However, this restriction further guarantees that the discriminative information is contained in core tensors.

Further, KTDC-Se uses mini-batch-dependent parameters $\boldsymbol{G}_{(1)}^i \in \mathbb{R}^{\tilde{N}\times(R_2\cdot R_3\cdot R_4)}$ 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, $\boldsymbol{G}_{(1)}^i$ is a sub-block of $\boldsymbol{G}_{(1)}$, i.e., $\boldsymbol{G}_{(1)} = [(\boldsymbol{G}_{(1)}^1)^\top, \cdots, (\boldsymbol{G}_{(1)}^K)^\top]^\top$. Correspondingly, I have $\boldsymbol{F} = [(\boldsymbol{F}^1)^\top, \cdots, (\boldsymbol{F}^K)^\top]^\top$, where $\boldsymbol{F}^i = \boldsymbol{E}^i\boldsymbol{G}_{(1)}^i$ 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 $\boldsymbol{G}_{(1)}^i$, check the support vectors in these features, and update their corresponding weights in $\boldsymbol{\alpha}^i$. To maintain the coupling between $\boldsymbol{G}_{(1)}^i(\boldsymbol{\alpha}^i)$ and the rest $\boldsymbol{G}_{(1)}^j$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 $\nabla_{\boldsymbol{G}_{(1)}^i}J_2$, $\nabla_{\boldsymbol{G}_{(1)}^i}J_3$, $\nabla_{\boldsymbol{\alpha}^i}J_2$, and $\nabla_{\boldsymbol{\alpha}^i}J_3$ by Equations (4.8), (4.11), (4.13), and

(4.15), respectively. Then, I can fix the corresponding gradients with respect to support vectors in other mini-batches to **0**s. Consequently, the complete algorithm can be seen in Algorithm 6.

Algorithm 6 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 $\{\boldsymbol{G}_{(1)}^i\}_{i=1}^K, \tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}$, and b.
repeat $i = 1$ to K Utilize the complete information in $\boldsymbol{G}$, $\boldsymbol{y}$, and $\boldsymbol{\alpha}$ and the mini-batch data to obtain: $\boldsymbol{G}_{(1)}^i, \tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{\alpha}^i, b$ = Train-KTDC-Se($\mathcal{X}^i, \boldsymbol{y}^i | \boldsymbol{F}, \boldsymbol{y}, \boldsymbol{\alpha}$). $\boldsymbol{F} = [(\boldsymbol{F}^1)^\top, \cdots, (\boldsymbol{F}^K)^\top]^\top$. $\boldsymbol{\alpha} = [(\boldsymbol{\alpha}^1)^\top, \cdots, (\boldsymbol{\alpha}^K)^\top]^\top$. **until** convergence =0

4.5.5 Testing on Mini-batches

For the cross-validation process or online testing, I 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 I 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, I 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 $\tilde{\boldsymbol{G}}_{(1)}$, I utilize the learned parameters $\tilde{\boldsymbol{A}}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$. By setting the gradient in Equation (4.7) to **0**, I can obtain

$$\tilde{\boldsymbol{G}}_{(1)} = (\tilde{\boldsymbol{A}}^\top \tilde{\boldsymbol{A}})^{-1} \cdot (\tilde{\boldsymbol{A}}^\top \tilde{\boldsymbol{X}}_{(1)} \tilde{\boldsymbol{H}}^\top) \cdot (\tilde{\boldsymbol{H}} \tilde{\boldsymbol{H}}^\top)^{-1}, \tag{4.17}$$

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, I 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, I 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:

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{K}} \boldsymbol{\alpha} + b. \tag{4.18}$$

Remark: Since I have to train multiple binary models, our method requires a certain level of computation for training. However, the model prediction is fast. For example, I show the average

training and testing time in Table 13 in the experiments. Our method (KTDC-Se) has the second-largest training time and the second-lowest testing time. The fast testing performance is because I significantly reduce the number of model parameters by tensor Tucker decomposition. This structural tensor decomposition guarantees that KTDC-Se can have compact and informative feature vectors, bringing fast and accurate event identification. Therefore, in realistic applications, I can conduct offline training using more time and computational resources. Then, our light model can be fast implemented for online inference.

I can further improve efficiency by changing the structure of the model design. For example, I can have one base model and restrict all sub-models to sharing the base model. These sub-models represent different branches for classification. The multi-branch design can enable one-time training for all classifiers. Moreover, this design can easily fit into our framework because of our gradient-based training algorithms (i.e., Algorithms 5 and 6). More specifically, the chain rule for gradient calculations enables the information of gradients to be passed from branches to the base, similar to multi-branch neural networks (Al Rahhal et al. 2018). However, due to the space limit, I will discuss this topic in future work.

4.6 Experiments

For validation, I test over synthetic data sets such as the Illinois 200-bus system and South Carolina 500-bus system (Li, Ma, and Weng 2022; Li, Ma, Weng, and Farantatos 2022). I also test our results with realistic data from our utility partners.

4.6.1 Dataset Description

I utilize Illinois 200-bus system and South Carolina 500-bus system (Li, Ma, and Weng 2022) 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, I consider 2 different event locations. Thus, there are 10 unique combinations of event types and locations, i.e., 10 event labels.

Then, I vary the loading conditions for the simulation to generate diversified event files. Totally, I

have 80 event files each of which has 10s event data. Further, I consider the data resolution to be 60 samples per second, yielding 600 samples for each event file. To extract tensors from these streams, I 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. I 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. 29 illustrates why I 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). I 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, in Section 4.6.7, I conduct the sensitivity analysis for the window length, which shows that the length between $[0.4, 0.7]$ can help to train an accurate model.

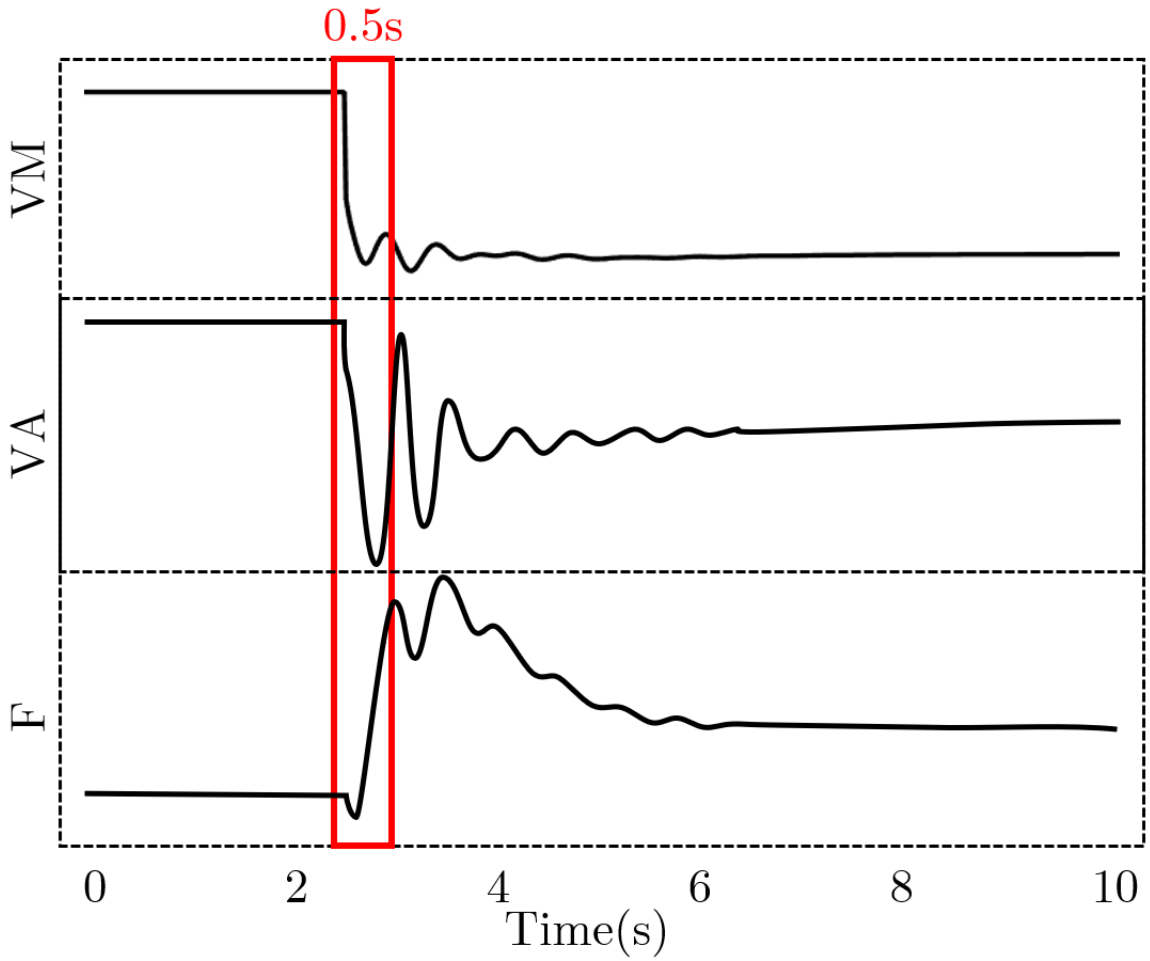


Figure 29. The Demonstration for Window Length Selection.

In general, I have 5840 PMU tensors in total. For each PMU tensor, I 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, I set $M = 3$ for the measurement types, i.e., voltage magnitude, voltage angle, and frequency. To summarize, I have $\mathcal{X} \in \mathbb{R}^{5840\times30\times200\eta_1\times3}$ or $\mathcal{X} \in \mathbb{R}^{5840\times30\times500\eta_1\times3}$ for training and testing. To mimic a semi-supervised setting, I consider labeled data with the ratio of $\eta_2 = \{0.1, 0.2, 0.3, 0.4, 0.5\}$, leading to a label vector $\boldsymbol{y} \in \mathbb{Z}^{5840\eta_2\times1}$.

Finally, I 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 (single-phase-to-ground fault,

phase-to-phase fault, and three-phase fault) at 2 locations. After tensorization of data in 35 PMUs, I can obtain $\mathcal{X} \in \mathbb{R}^{511\times30\times35\times3}$ and $\boldsymbol{y} \in \mathbb{Z}^{511\times1}$.

Software For the simulation, I employ a commercial-grade simulator, Positive Sequence Load Flow (PSLF) (General Electric Energy Consulting 2018) from General Electric (GE) company. For the model development and validation, I use Python with Pycharm IDE.

Summary of data and feature dimensions

- For Illinois 200-bus system, the total event tensor is $\mathcal{X} \in \mathbb{R}^{5840\times30\times200\eta_1\times3}$ and the input data dimensions are $\{900, 1800, 2700, 3600\}$. Subsequently, I 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, I 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 $\mathcal{X} \in \mathbb{R}^{5840\times30\times500\eta_1\times3}$ and the input data dimensions are $\{2250, 4500, 6750, 9000\}$. Subsequently, I 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, I 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 $\mathcal{X} \in \mathbb{R}^{511\times30\times35\times3}$ and the input data dimension is 3150. Subsequently, I set $R_2 = 6$, $R_3 = 6$, and $R_4 = 2$, and the feature dimension is 72. For other benchmark methods, I employ PCA to reduce the dimensionality of the raw data and obtain features with the dimension of 160.

Table 10. 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

4.6.2 Benchmark Methods

First, I train our KTDC-Se within the labeled data as a benchmark to demonstrate the impacts of the unlabeled data. Further, I 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) (K. He et al. 2016): Resnet is an efficient deep learning model for classification. For this supervised learning model, I 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 (Berthelot et al. 2019): 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, I employ PCA to reduce the dimensionality of the mixed dataset from MixMatch and input them into a Resnet (K. He et al. 2016) as the final classifier. For a fair comparison, the Resnet has the same architecture as the first benchmark.
- FixMatch (Sohn et al. 2020): 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, I 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) (Rasmus et al. 2015): 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, I 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, I can illustrate the effectiveness of using unlabeled data. By comparing the testing accuracy of KTDC-Se and other methods, I 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, I can evaluate the efficiency of the methods for real-time inference. Finally, by comparing the choice of kernel selection, I can understand how the non-linear kernels boost the performance.

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

In this subsection, I evaluate the integration design by comparing our KTDC-Se and other two-stage models. By default, I set the kernel of our KTDC-Se to be polynomial with the degree $d = 2$. Thanks to the integration without biased selection for the two-stage compression and classification, our model performs much better than benchmarks. Specifically, I report the results of simulated and real-world data as follows.

For the simulated data, I first fix $\eta_2 = 0.3$ to divide the labeled and unlabeled datasets for training and testing. Fig. 41 and 42 demonstrate the results for the two systems. Since for each η_1 of one system, I 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, I present the box plot in Fig. 41 and 42 to show the average and the variance.

I 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 at 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, I 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

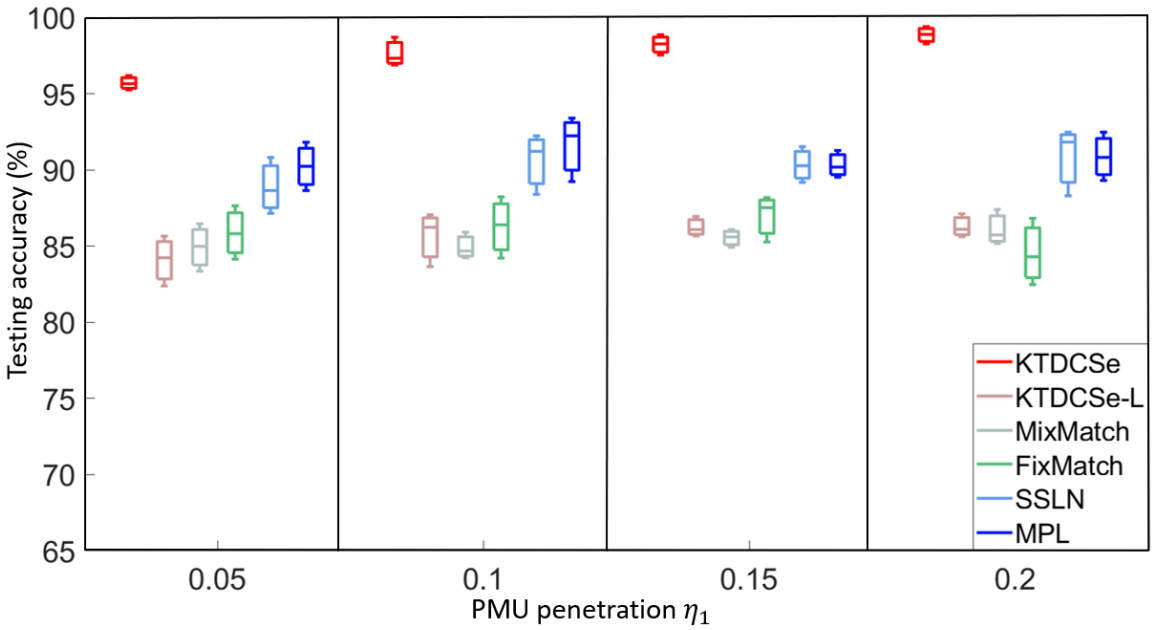


Figure 30. Testing Accuracy (%) for the 200-bus System.

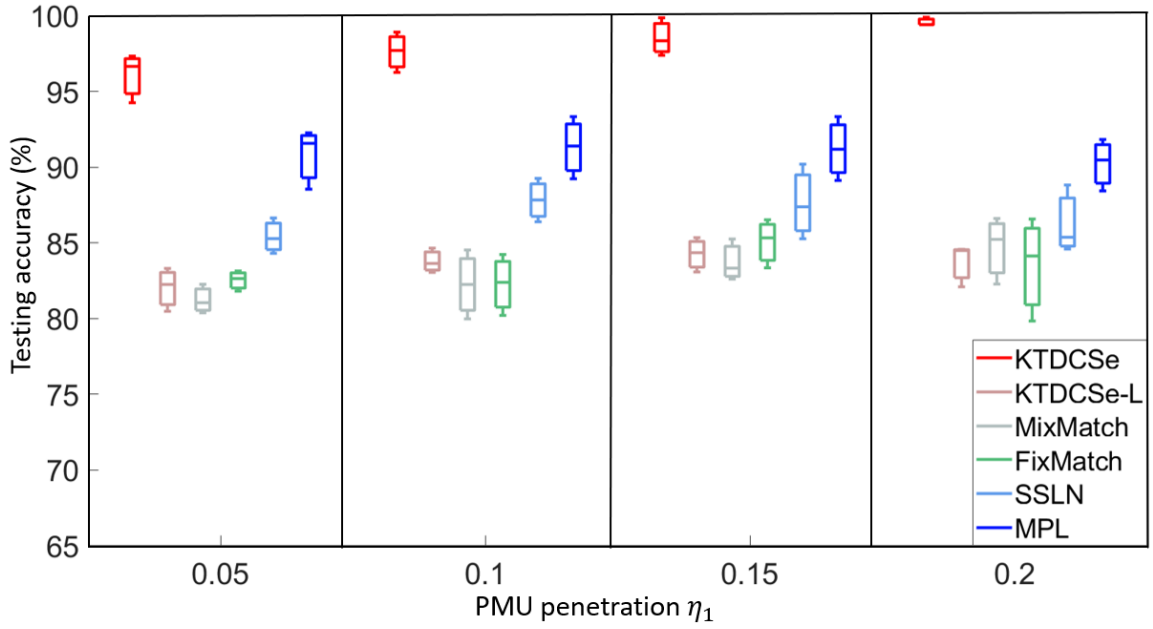


Figure 31. Testing Accuracy (%) for the 500-bus System.

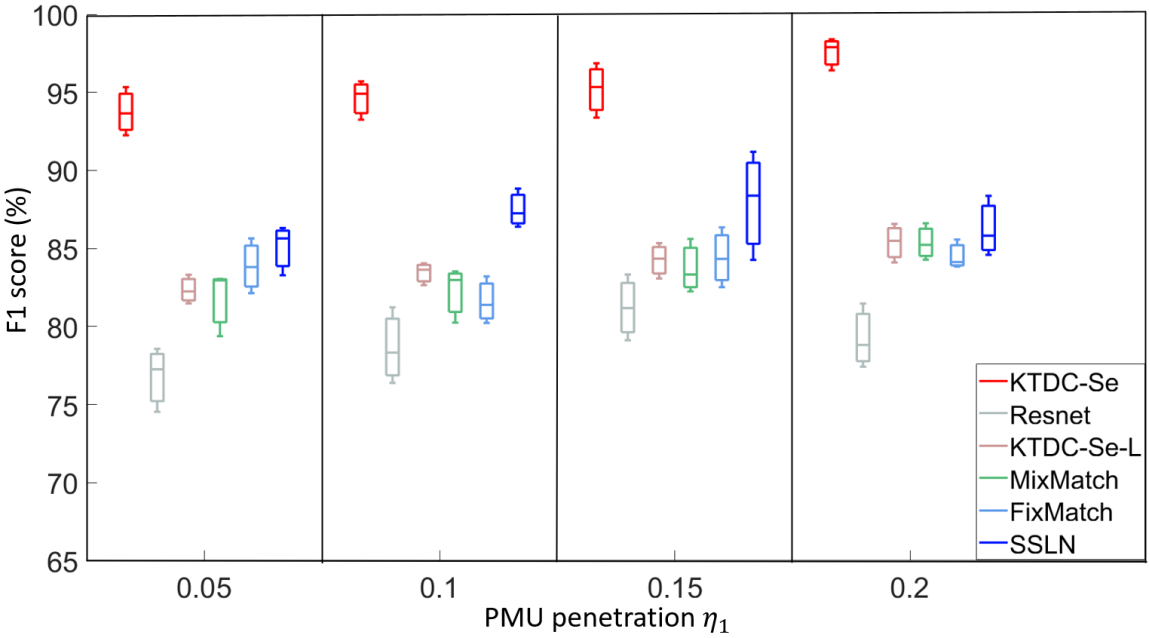


Figure 32. F1 Score (%) for the 200-bus System.

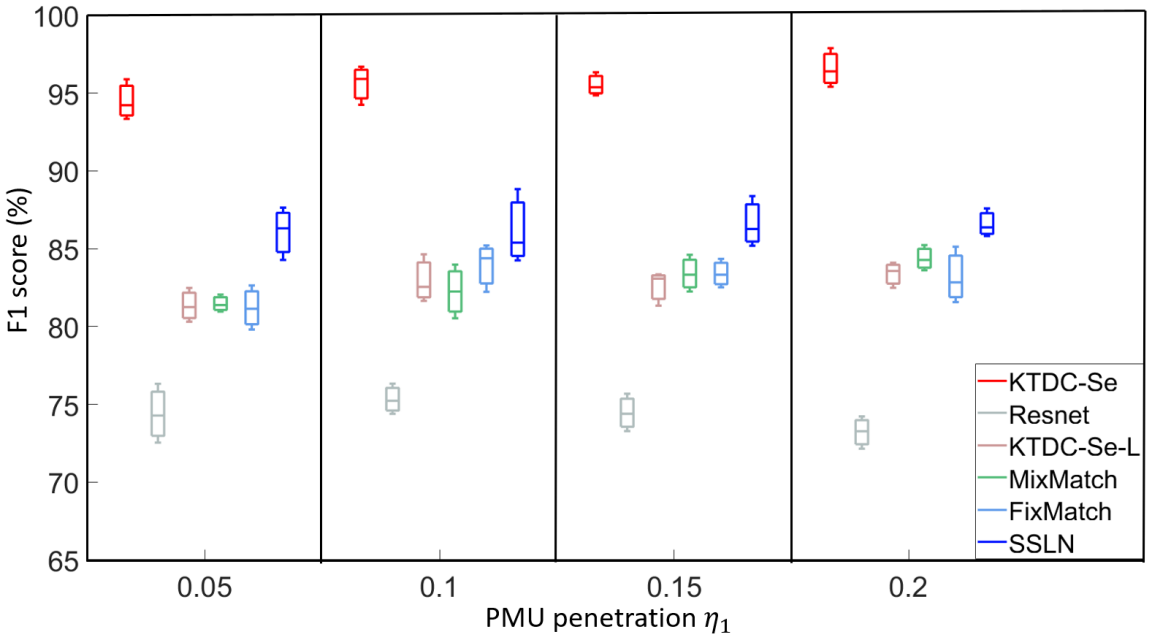


Figure 33. F1 Score (%) for the 500-bus System.

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 $\eta_2 = 0.2$, I find that the 500-bus system has a higher accuracy mean and lower accuracy variance. After careful checking, I 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 I may meet the situation when the 500-bus system data brings better performance.

I 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 (Joos Korstanje 2021). Under this setting, I re-evaluate Section VI-D using F1 score. Fig. 32 and 32 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, I report the testing accuracy in Table 10. I 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.

I 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 11. Other scenarios can bring similar results. I 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.

Table 11. Testing Accuracy (%) (Mean ± Standard Deviation) for Real-world PMU Data.

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 12. The Average Predicting Time (S) of the Testing Dataset for Different Methods.

	KTDC-Se	Resnet	KTDC-Se-l	MixMatch	FixMatch	SSLN
Time	1.3	3.8	0.4	3.8	3.7	6.4

Table 13. 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

4.6.4 Semi-supervised Learning Boosts KTDC-Se Model Performance

In this subsection, I 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, I 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, I can compare KTDC-Se-L and Resnet. I find that Resnet

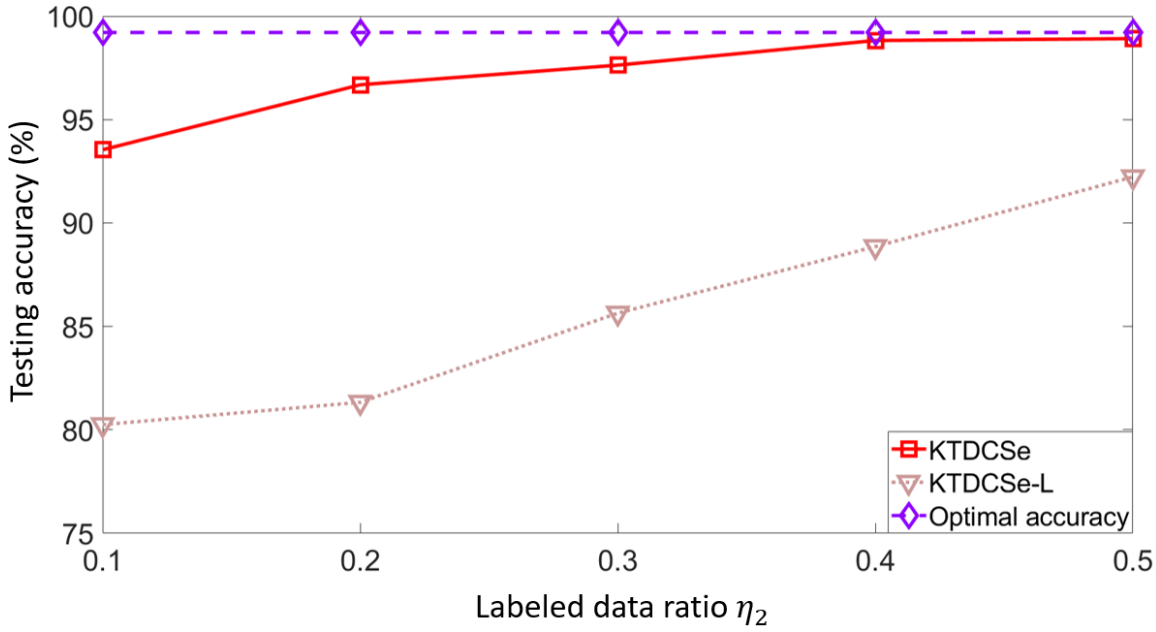
has an even worse performance as Resnet also employs PCA to pre-process data. As illustrated in Section 4.6.3, the two-stage model performs worse.

To further understand how much labeled data is needed for a good performance of KTDC-Se, I 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, I also implement KTDC-Se-L for the labeled data. Further, I 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. 34a and 34b.

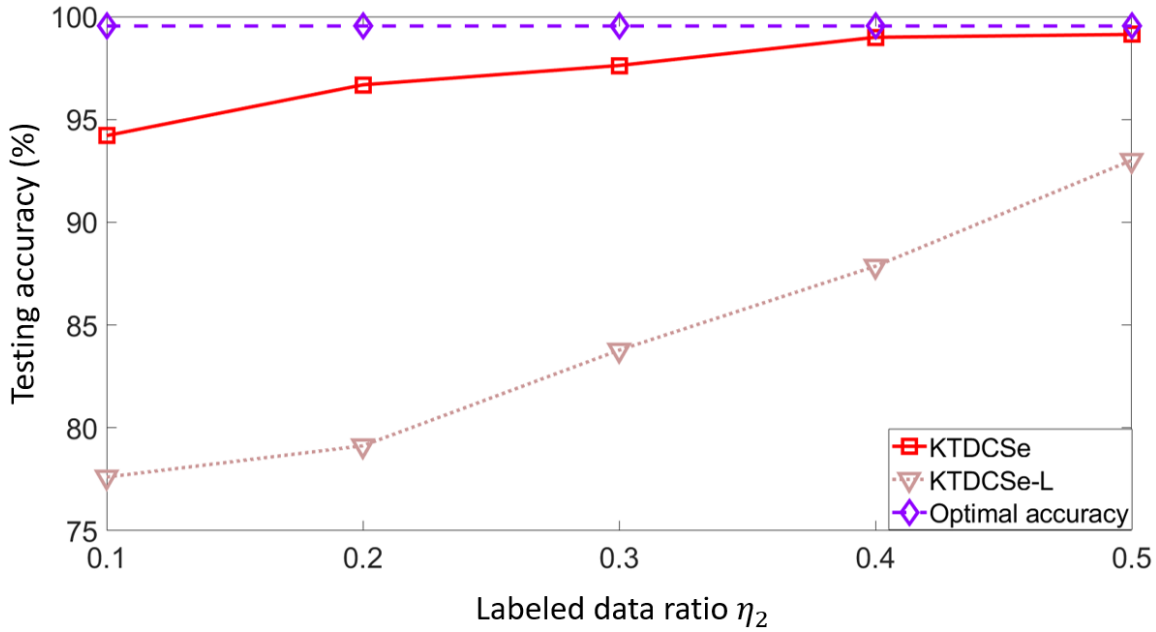
Based on the plots, I have the following observations. (1) Training with extra unlabeled data significantly improves the accuracy, which has been explained before. (2) KTDC-Se can efficiently utilize the limited labels. For example, when $\eta_2 = 0.1$, the testing accuracy is still higher than 94%. 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.

4.6.5 Tensor-based Framework Enables Fast Inference

The model efficiency can be evaluated via the training and testing time. I utilize the simulated data in Section 4.6.3 and report the computational time of all methods on the testing dataset in Table 13. Based on the results, I 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 I 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.



(a) Sensitivity Analysis for the 200-bus System.



(b) Sensitivity Analysis for the 500-bus System.

Figure 34. Results of the Sensitivity Analysis with Respect to Labeled Data Ratios.

I 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 (4.18), KTDC-Se-L has a lower testing time compared to KTDC-Se.

However, if I utilize both labeled and unlabeled data, KTDC-Se is much more effective in doing 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 share the same DNN architecture and consume similar times for testing. On the other hand, the ladder network in SSLN has a larger size and needs more time to predict the label.

4.6.6 Non-linear Kernelization Largely Increases the Accuracy

In this subsection, I study the effectiveness of kernelization on the capacity of the classifier. Specifically, I 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, I 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, I vary $d \in \{1, 2, 3\}$. When kernel is a RBF function, I vary $\lambda \in \{0.05, 0.1, 0.15\}$.

The results are shown in Table 14. First, if I compare the constant kernels with other kernels, I 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 of 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.

4.6.7 Sensitivity Analysis of the Window Length

Finally, I vary the length of the moving window to test the suitable region. Specifically, I conduct

Table 14. The Testing Accuracy (%) (Mean ± 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

the experiments for the 200-bus system with $\eta_1 = 0.1$ and $\eta_2 = 0.3$. Further, I set the kernel of our KTDC-Se to be polynomial with the degree $d = 2$. The result is shown in Table 15. I have the following observations on the results. First, when the window length is less than 0.3s, the accuracy will decrease. This shows that when the window width is too small, the event dynamics may not be completely covered and the classification will encounter errors. Second, the test accuracy will decrease slightly when the window width is larger than 0.8s. This is because a large window will contain both normal and event information, which reduces the classification accuracy. Third, the best value of window length is 0.6s, and there is a robust region $[0.4, 0.7]$ to produce an accurate model.

Table 15. The Testing Accuracy (%) (Mean ± Standard Deviation) for Different Window Lengths.

Window length	Accuracy
0.1	88.3 ± 0.3
0.2	89.2 ± 0.4
0.3	92.3 ± 0.3
0.4	95.3 ± 0.4
0.5	95.7 ± 0.3
0.6	$\mathbf{95.9 \pm 0.3}$
0.7	95.2 ± 0.2
0.8	93.5 ± 0.3
0.9	93.6 ± 0.2
1.0	94.1 ± 0.2

Chapter 5

EVENT DETECTION: EXTRINSIC KNOWLEDGE ENHANCEMENT THROUGH TRANSFER LEARNING

5.1 Introduction

Power systems are dramatically integrating highly uncertain components for cleaner, more efficient, and lower-cost energy generations and consumption. For example, renewable energies like Photovoltaic (PV) and wind power depend largely on the changing weather. The charging of Electric Vehicles (EVs) also brings randomness to the electric grid. This unprecedented system evolution, despite promising values, deteriorates traditional system monitoring and prevents the achievement of situational awareness.

To enhance the system monitoring under changing operating points, Machine Learning (ML) methods are more and more popular due to their high capacity of extracting informative features from dynamic data streams. This advantage is especially expanded with the increasing deployment of Phasor Measurement Units (PMUs) to provide high-resolution phasor data. PMU-based ML models achieve great success in state estimation (Weng, Negi, and Ilić 2019), event identification (Haoran Li et al. 2019a; H. Li et al. 2019; Yuxuan Yuan et al. 2021b), resilience improvement (Alimi, Ouahada, and Abu-Mahfouz 2020), and cyber-attack detection (Mohammadpourfard et al. 2020), etc. These applications, however, strictly assume that there are enough data for training.

Such an assumption is easily violated in the following scenarios. 1) For new grids, data scarcity prevents ML model training. 2) For old grids with new meters or components (e.g., lines, generators, and loads), the data dimensionality and distributions change after the new construction. Thus, old ML models will fail, while new models are hard to train with limited data. 3) Even for stable grids with long-term operations and few events, the number of event labels is limited. Thus, directly training of a Machine Learning (ML) model is hard due to the limited labels. For example, the fault data is intrinsically limited for a highly reliable grid. In general, I demand new principles to fundamentally change the dilemma.

Therefore, I propose to utilize Transfer Learning (TL) for knowledge migration from a data-rich

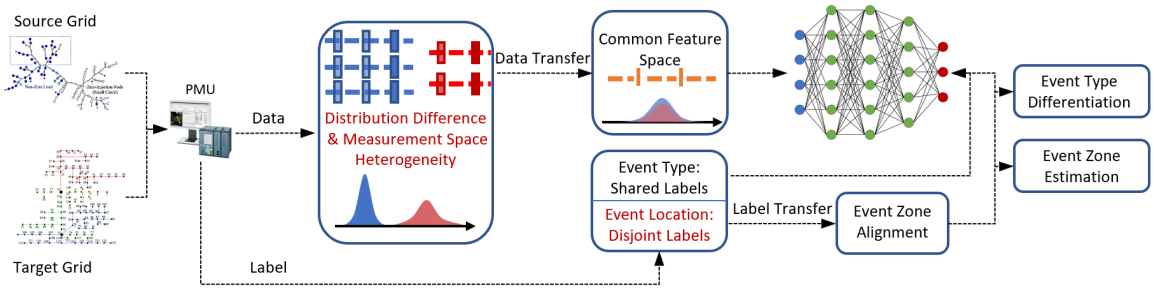


Figure 35. The Flowchart of the Proposed Transfer Learning Framework.

grid (source grid) to a data-limited grid (target grid). TL employs well-established feature extraction techniques to obtain common features for enhancing the decision-making in the target domain (Pan and Yang 2009). This approach promotes many TL applications to fields like object localization (Guillaumin and Ferrari 2012), image classification (Y. Zhu et al. 2011), video pattern recognition (Duan et al. 2009), sentiment analysis (Blitzer, McDonald, and Pereira 2006), and vehicle routing detection (Feng et al. 2012), etc. For power systems, TL also has few applications on dynamic security assessment (Ren and Xu 2019), load file generation (Li, Weng, and Tong 2020) and event forecasting (X. He et al. 2020). However, these methods are restricted to a small scope and have limited theoretical foundations. In this chapter, I aim to propose a general TL framework for power system transfer learning with both strong theoretical foundations and high generalizability. Notably, although I focus on event type differentiation and localization, these two tasks are representative for situations of data space homogeneity/heterogeneity and label space identicalness/disjointing.

More specifically, I assume two grids can have either the same or different number of PMUs for homogeneous and heterogeneous measurement spaces (Day and Khoshgoftaar 2017), respectively. As for the label space, two grids share the same event-type set but completely different event-location sets. For example, the location label of a line trip for one grid doesn't have any physical meanings for the other grid. Taking all the above characteristics into consideration, I try to answer the following questions. (i) Can I propose a unified framework to incorporate datasets with the same or different dimensionalities and minimize the knowledge transfer error? (ii) Can I bridge two disjoint label spaces for physically meaningful label transfer?

For question (i), I focus on heterogeneous TL and treat the homogeneous scenario as one special case. To align the dimensionality, existing work aims to find proper projections. (Zhou et al. 2014) find the mapping from the source to the target by minimizing the Euclidean distance. This projection

is inefficient and inaccurate for power system datasets since a) the Euclidean distance is an inaccurate measure to evaluate the data distribution difference (Day and Khoshgoftaar 2017) which implies that there is a distance between the distribution of the source and the target spaces. For example, for two power systems, the PMU measurements usually have certain distribution distances due to different operating conditions. b) the transformation requires a perfect reconstruction of target data that has ultra-high dimensionality in power systems. For adapting distribution difference, Domain Adaptation (DA) in TL minimizes distribution discrepancy for the source and the target domains to obtain the common knowledge, which is theoretically sound and computationally inexpensive (Pan and Yang 2009). For example, (Yan et al. 2018) improves the problem a) by introducing Entropic Gromov-Wasserstein Discrepancy for a better evaluation but still meets the problem of b). (Tsai, Yeh, and Wang 2016) utilizes the so-called Maximum Mean Discrepancy (MMD) (Long et al. 2013) as the measure for problem a). For problem b), Principal Component Analysis (PCA) is employed to pre-process the target domain's data, thus making the conversion from the source to the target to be an efficient dimensionality reduction process. However, PCA fixes the mapping for the target grid to a subspace, thus reducing the capacity of finding common knowledge.

To increase the capacity, I propose to find two projections for both the source and the target data to a common low-dimensional space. Within this space, I utilize MMD to measure the distribution divergence. Further, to support the proposed method, I exhibit strong mathematical foundations for the optimization formulation and the introduction of MMD measure. After projections, any ML models can be further trained using the transformed data to perform tasks in the target grid as long as the label spaces are the same. For disjoint label spaces in question (ii), a label-transfer process is required. I find that usually, the disjoint labels indicate a local change to the system, e.g., the label for a local event. These changes have higher impacts on PMUs that are closer to the change location. Thus, I can select a fixed number of representative PMUs for two grids to relabel the local change. Consequently, disjoint label spaces are mapped to one common label space for the label transfer, thus complementing the total TL framework.

The complete framework is illustrated in Fig. 35. I aim to transfer knowledge from a long-operated source grid to a new target grid, which boosts the performance of an ML model to identify events in the target grid. To capture event patterns, I utilize PMU data with high resolutions and labels to train a supervised ML model like a Deep Neural Network (DNN). However, as shown in the middle of

Fig. 35, the measurement spaces of the source and the target system (1) have different distributions due to different operating conditions, and (2) have different dimensions that cause heterogeneity. Further, though two systems can have shared event type labels, they own (3) disjoint event-location label spaces. Thus, traditional ML models fail to utilize all the information together, and I propose a heterogeneous Transfer Learning (TL) framework to tackle these problems. Specifically, as shown in the middle of Fig. 35, the data transfer process handles (1) and (2) via finding the common feature space with the same dimensions and similar feature distributions from the source and the target systems. Thus, these features can be input to the ML model (e.g., the DNN model at the right of Fig. 35) for training. Secondly, the label transfer process deals with (3) via categorizing event locations into zones that have the same number and relative locations (e.g., Northwest and Southeast, etc.) in the source and the target grid. Then, zone indices are treated as labels. Finally, the right part of Fig. 35 illustrates that the event type or event zone labels are the output of the ML models for training and predictions. In general, I have the following contributions:

- I formalize the problem of power system TL with a thorough consideration of the data-space and label-space differences. With domain knowledge, I show that though data/label differences exist, there is shared knowledge, and positive transfer can always happen.
- I propose a general framework to minimize the differences with strong mathematical guarantees for highly efficient knowledge transfer.
- I will conduct extensive experiments to demonstrate the high performance of our framework with datasets from different systems.

Remark: to the best of the author's knowledge, this chapter is the first to consider both the data space heterogeneity and label space non-overlaps. Primarily, I combine the domain knowledge of event responses and rigorous mathematical proofs to build the model. With extensive experiments, I show the effectiveness of our proposed models.

The rest of the chapter is organized as follows: Section 5.2 formalizes the problem. Section 5.3 demonstrates the data transfer process. Section 5.4 illustrates the label transfer approach. Section 5.5 evaluates the performance of the proposed framework.

5.2 Problem Definition

In this chapter, the goal is to train a classifier that can analyze a period of PMU streams and output the underlying event type and location. Note that some smart meter data with high resolutions and qualities (Stegner et al. 2016; G. Liu et al. 2020; Zhu and Reinhardt 2019) can also be utilized as long as the measurements can capture event dynamics. However, in this chapter, I focus on PMU data. To prepare the input data, I employ a moving window to segment PMU streams. As shown in Fig. 36, the window has fixed width and moving gap for the source and the target measurements. Then, each time a small period of electric measurements (e.g., voltage, current, and frequency) for all PMUs is gathered and vectorized into an input vector. Specifically, I denote $\boldsymbol{x} \in \mathbb{R}^{d_S}$ and $\tilde{\boldsymbol{x}} \in \mathbb{R}^{d_T}$ as the input vector variable for the source and the target grid, respectively, where d_S and d_T represent the corresponding dimensionalities. Generally, I consider the scenario when $d_S \neq d_T$ due to different PMU numbers in two grids. However, our models can also handle the case when $d_S = d_T$. Then, let y and $\tilde{y}$ be the label variable of the input vectors for the source and the target grid. The label can be the normal status, event types, and event locations. For event type labels, I have $y, \tilde{y} \in \{1, 2, \cdots, K\}$, where K is the total number of event types in the power systems. For the event location label, I have $y \in \{I_1, I_2, \cdots, I_h\}$ and $\tilde{y} \in \{J_1, J_2, \cdots, J_l\}$, where h and l are the number of event locations for the source and the target grids, respectively. Note that $\{I_1, I_2, \cdots, I_h\} \cap \{J_1, J_2, \cdots, J_l\} =$ for two disconnected power networks.

After the moving window segmentation, I denote $\{(\boldsymbol{x}_i, y_i)_{i=1}^{n_S}\}$ and $\{(\tilde{\boldsymbol{x}}_i, \tilde{y}_i)_{i=1}^{n_T}\}$ as a set of labeled training instances from the source grid and the target grid, respectively, where n_S and n_T are the number of samples for the source and the target grids, and I assume $n_S \gg n_T$. With above notations, I have the following problem formulations

- Problem: PMU-based Transfer learning for event type differentiation and event zone estimation.
- Given: a set of PMU-based samples with labels $\{(\boldsymbol{x}_i, y_i)_{i=1}^{n_S}\}$ and $\{(\tilde{\boldsymbol{x}}_i, \tilde{y}_i)_{i=1}^{n_T}\}$ from the source and the target grids, respectively.
- Output: a classifier f to distinguish event types or identify event locations in the target grid.

The defined problem with high d_S and d_T forces us to find a cost-efficient data transfer process.

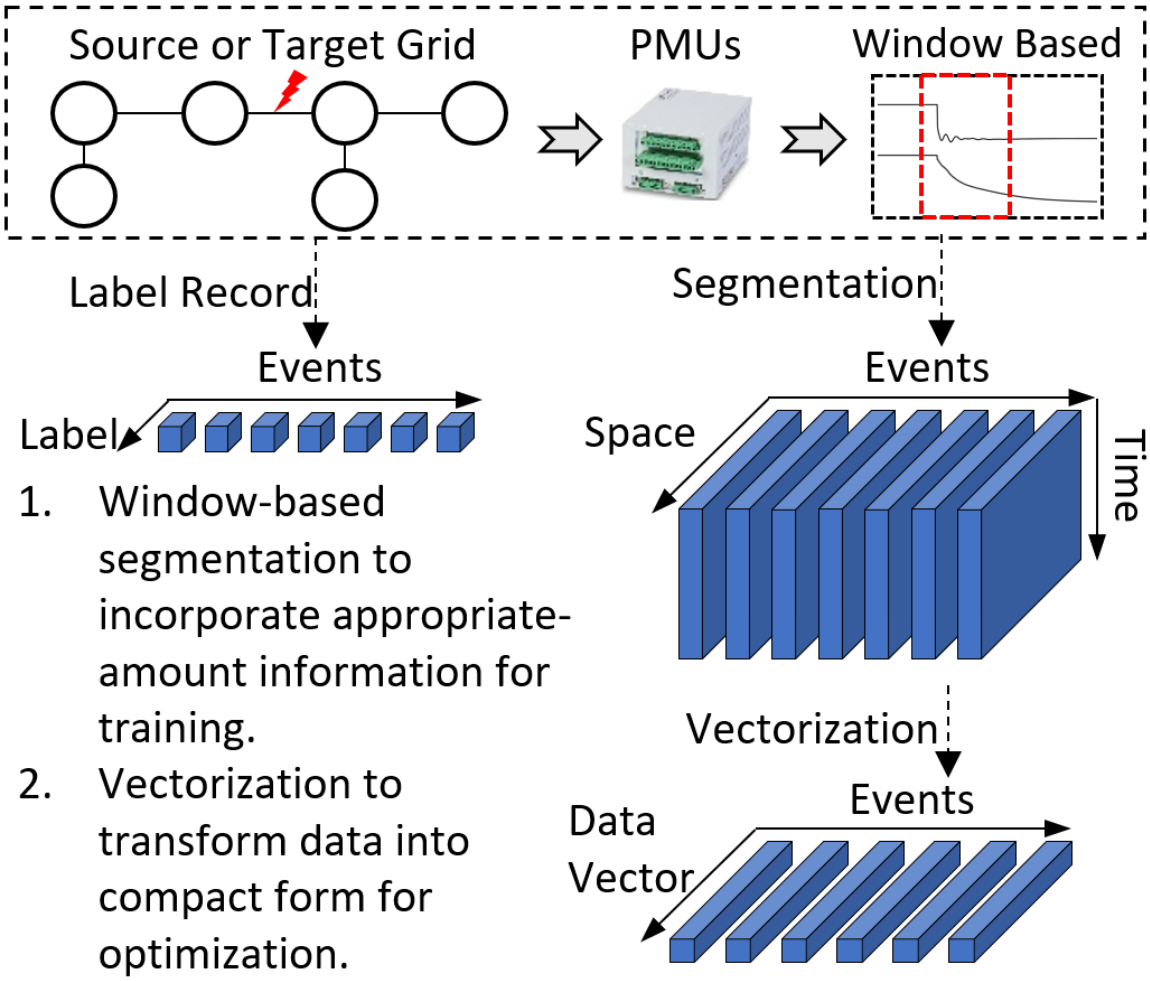


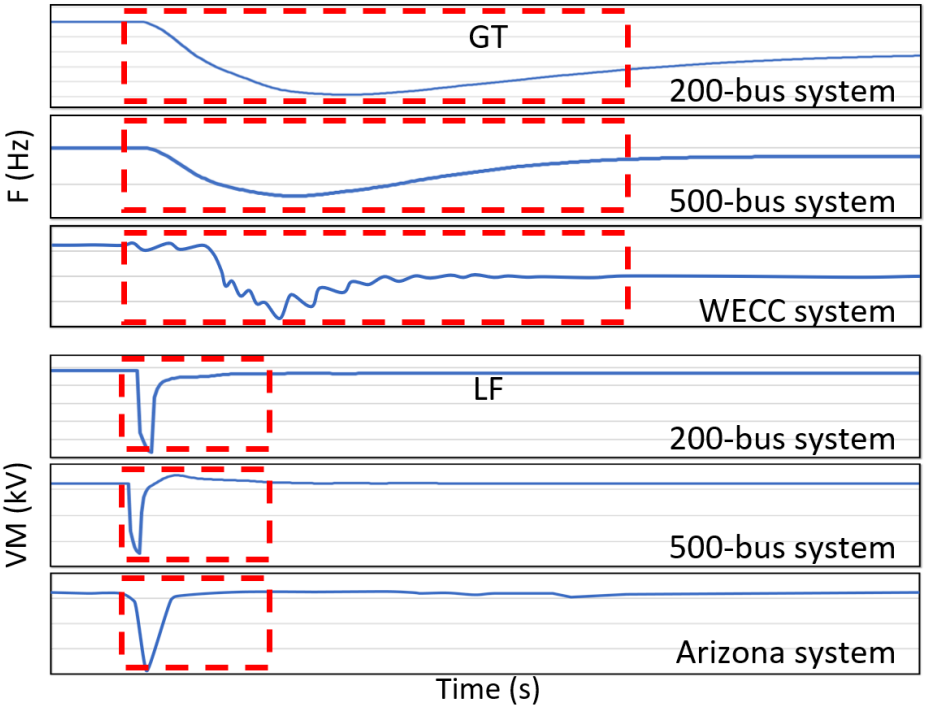
Figure 36. A Moving Window-based Segmentation and Vectorization Procedure to Process PMU Streams.

I show in the next section that this procedure is achieved via finding a common low-dimensional feature space.

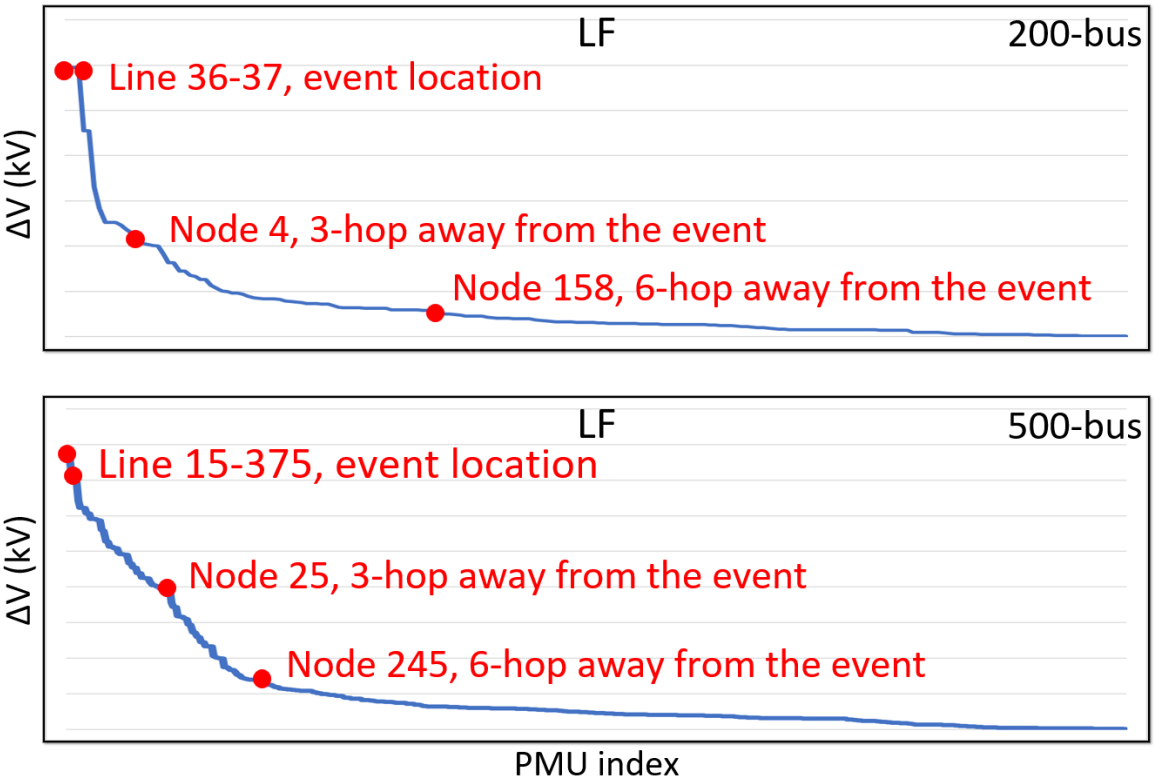
5.3 Data Transfer: Projection to Low-dimensional Feature Space

5.3.1 Validation of the Transfer Positivity

Before elaborating on data transfer details, a more fundamental issue should be first figured out: will the data transfer be positive to enhance rather than deteriorate the ML model training for the target grid? Namely, I want to avoid the so-called *negative transfer* (Rosenstein et al. 2005). To



(a) Event Type Similarity of PMU Streams Between 200 Bus System, 500 Bus System and Actual System Data.



(b) Event Location Similarity of Voltage Magnitude Drops Between 200 Bus System and 500 Bus System.

Figure 37. Illustrations of the Existence of Shared Knowledge for Event Type Differentiation and Localization.

illustrate the positivity of our transfer learning, I start from physics. Generally, as shown in Fig. 37a, I utilize simulated data from Illinois 200-bus system (Engineering Texas A&M University 2016a) and South Carolina 500-bus system (Engineering Texas A&M University 2016b), and realistic data from Western Electricity Coordinating Council (WECC) system (Usman and Faruque 2019) and our utility partner in Arizona to demonstrate the Generator Trip (GT) and Line Fault (LF) events. Note that I have another IEEE 14-bus simulation system (Illinois Center for a Smarter Electric Grid 2013) and its event performances are similar. Due to space limit, I don't show this system's event curves. Further, with geographical information of the 200-bus and 500-bus systems, I utilize Fig. 37b to demonstrate LF event impacts on different PMUs from event centers to edges. Thus, I show the similarity of the event geographical propagation.

Specifically, I have the following statements to support transfer positivity for most of the power systems. (1) for different systems, the dynamical behaviors of each electric device are similar (North American Electric Reliability Corporation 2017). Then, the aggregation of them in a system can bring similar dynamics in response to a certain event. (2) the protection and response mechanism to a specific event is similar (Qin, Li, and Zhu 2021). For example, for GT events, the inertial and governor responses help to stabilize the frequency, leading to similar behaviors of frequency for the first 3 curves in Fig. 37a. Based on (1) and (2), I observe similar behaviors of the GT and LF in simulated data. For realistic data, I find that though they have disturbances, they have similar patterns of the curve change tendency after a specific event.

For event zone estimation, I claim that the common knowledge lies in the event impact propagation from the event locations to other areas. Specifically, I present the voltage magnitude drop after an LF event for 200- and 500-bus systems, as shown in Fig. 37b. The impact of the event decreases as the location is further and further away from the event center. Such a propagation, based on the system damping characteristic, is general for different systems. Finally, the red dotted block in Fig. 37a and the sharp decreasing in the left part of Fig. 37b illustrate that the shared knowledge lies in the low-dimensional space compared to the ultra high-dimensional input data. Thus, I propose the following assumption that lays the foundations for further transfer learning.

Assumption 2. *There exists a low-dimensional feature space that is extracted from the source and the target data spaces and represents the common knowledge to the system event.*

Remark: there may be some special systems that lose the event similarity compared to most of

the power systems, leading to negative transfer (Rosenstein et al. 2005). However, in this chapter, I focus on the potentials of transfer learning for most of the standard systems and the method design. The investigation and quantification of negative transfer on power systems can be treated as a future topic.

5.3.2 Data Transfer Objective Formulation

Based on Assumption 2, I propose to project $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ to the d-dimensional latent feature space with linear transformations $\boldsymbol{T} \in \mathbb{R}^{d \times d_S}$ and $\tilde{\boldsymbol{T}} \in \mathbb{R}^{d \times d_T}$, where $d \ll d_S$ and $d \ll d_T$. Namely, I have

$$F(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{x}, \tilde{F}(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}. \tag{5.1}$$

I can add kernel mapping to $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ for non-linearity. However, I find the linear mapping is generally good enough in testing, and adding kernels don't bring significant improvements. In the following derivations, I focus on the linear mapping.

The optimal $\boldsymbol{T}$ and $\tilde{\boldsymbol{T}}$ ultimately help to train a good classifier for event identification in the target grid. Namely, I want the conditional probabilities to be approximately equal:

$$P(y|F(\boldsymbol{x})) \approx P(\tilde{y}|\tilde{F}(\tilde{\boldsymbol{x}})), \tag{5.2}$$

where $P(\cdot)$ represents the probability or probability density for the random variables. Equation (5.2) indicates that the decision boundaries of the source and the target for each label are close. Thus, I can utilize all the feature data to train the classifier.

Remark: Equation (5.2) as the goal can only be proposed and analyzed when the label spaces for the source and the target grid are the same, e.g., label space for event types. For disjoint label spaces for event locations, I propose to first align the label space in the next section. Then, all the derivations based on equation (5.2) can be continued. For the convenience of later derivations, I still assume the final common label space includes K unique labels.

To further understand the objective in equation (5.2), I rewrite the equation according to Bayes' rule:

$$\begin{aligned} &\frac{P(F(\boldsymbol{x})|y) * P(y)}{\sum_i P(F(\boldsymbol{x})|y=i) * P(y=i)} \\ \approx &\frac{P(\tilde{F}(\tilde{\boldsymbol{x}})|\tilde{y}) * P(\tilde{y})}{\sum_i P(\tilde{F}(\tilde{\boldsymbol{x}})|\tilde{y}=i) * P(\tilde{y}=i)}. \end{aligned} \tag{5.3}$$

Equation (5.3) shows that for the source and the target grids' data, if $P(F(\boldsymbol{x})|y) \approx P(\tilde{F}(\tilde{\boldsymbol{x}})|\tilde{y})$ and $P(y) \approx P(\tilde{y})$, equation (5.2) can be achieved. The former equation inspires a minimization of the conditional probability of features given labels, but the latter condition is generally not true. For instance, if the source grid is much older than the target grid, the probability of an event is generally higher. Luckily, based on the probability theory, I have the following equations.

$$P(F(\boldsymbol{x})) = \sum_i P(F(\boldsymbol{x})|y=i) * P(y=i),$$
$$P(\tilde{F}(\tilde{\boldsymbol{x}})) = \sum_i P(\tilde{F}(\tilde{\boldsymbol{x}})|\tilde{y}=i) * P(\tilde{y}=i).$$

(5.4)

Therefore, guaranteeing $P(F(\boldsymbol{x})) \approx P(\tilde{F}(\tilde{\boldsymbol{x}}))$ and $P(F(\boldsymbol{x})|y) \approx P(\tilde{F}(\tilde{\boldsymbol{x}})|\tilde{y})$ together can lead to the final goal of equation (5.2). Based on this observation, I can jointly minimize the distance between $P(F(\boldsymbol{x}))$ and $P(\tilde{F}(\tilde{\boldsymbol{x}}))$ and the distance between $P(F(\boldsymbol{x})|y)$ and $P(\tilde{F}(\tilde{\boldsymbol{x}})|\tilde{y})$. Namely, I have the following optimization problem:

$$\min_{\boldsymbol{T},\tilde{\boldsymbol{T}}} \sum_{y,\tilde{y}} D(\boldsymbol{T}\boldsymbol{x}, \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}|y=\tilde{y}) + \eta D(\boldsymbol{T}\boldsymbol{x}, \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}) + \lambda||\boldsymbol{T}||_F^2 + \lambda||\tilde{\boldsymbol{T}}||_F^2,$$

(5.5)

where D represents the distance of two distributions and η is a hyper-parameter to weight two objectives and avoid numerical issues. λ is the penalty term and $||\cdot||_F$ represents the Frobenius norm. The last two terms help to prevent overfitting. I term our optimization in equation 5.3.2 as Heterogeneous Joint conditional and marginal Distribution Adaptation (HJDA). **Remark:** HJDA is different from Joint Distribution Adaptation (JDA) in (Long et al. 2013), which utilizes the same projection for two domains. HJDA considers different projection matrices $\boldsymbol{T}$ and $\tilde{\boldsymbol{T}}$ since (1) nodes in two grids are misaligned and can't be treated as one input variable and (2) cross-grid transfer learning is often heterogeneous, i.e., $d_S \neq d_T$.

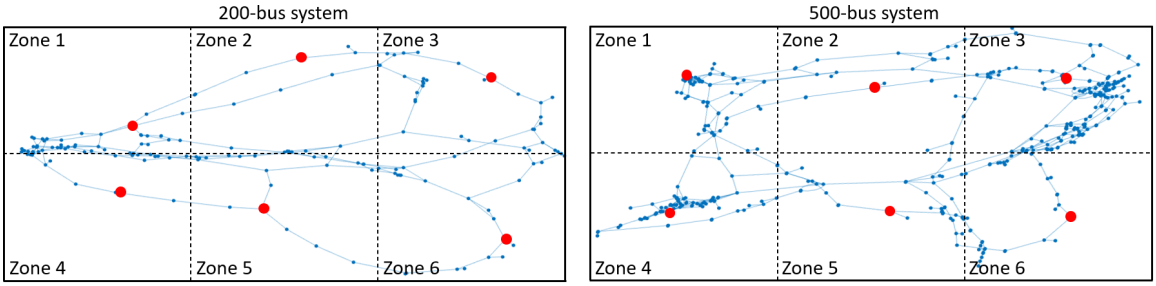


Figure 38. Event Zone Categorization to Align Event Location Labels.

There are many measures for probability distance D in (5.3.2). The most popular one is a nonparametric distance estimate: Maximum Mean Discrepancy (MMD) for its simplicity in calculations and solidness with mathematical foundations (Long et al. 2013; Gretton et al. 2012; Tsai, Yeh, and Wang 2016). In general, MMD evaluates the sample distribution difference over any unit-ball function class in the characteristic Reproducing Kernel Hilbert Spaces (RKHS). Thus, the MMD metric has two distinguished properties: (1) the unit-ball function class is rich enough to make MMD powerful to represent the distribution difference. (2) RKHS brings fast convergence of the estimate MMD to the true value as the sample size increases. One can find more strict mathematical proofs of MMD's properties in (Gretton et al. 2012). Based on the definition of MMD, I specify the optimization problems as follows.

$$\begin{aligned}
&\min_{\boldsymbol{T},\tilde{\boldsymbol{T}}} \sum_{y,\tilde{y}} D(\boldsymbol{T}\boldsymbol{x}, \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}|y=\tilde{y}) + \eta D(\boldsymbol{T}\boldsymbol{x}, \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}) + \lambda||\boldsymbol{T}||^2 + \lambda||\tilde{\boldsymbol{T}}||^2, \\
&= \sum_{y=\tilde{y}=k} ||\frac{1}{n_S^{(k)}} \sum_{i=1}^{n_S^{(k)}} \boldsymbol{T}\boldsymbol{x}_i - \frac{1}{n_T^{(k)}} \sum_{i=1}^{n_T^{(k)}} \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}_i||^2 \\
&+ \eta||\frac{1}{n_S} \sum_{i=1}^{n_S} \boldsymbol{T}\boldsymbol{x}_i - \frac{1}{n_T} \sum_{i=1}^{n_T} \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}_i||^2 + \lambda||\boldsymbol{T}||^2 + \lambda||\tilde{\boldsymbol{T}}||^2,
\end{aligned} \tag{5.6}$$

where $n_S^{(k)}$ and $n_T^{(k)}$ represent number of samples for label k, where $1 \leq k \leq K$.

5.3.3 Data Transfer Optimization Algorithm

I can utilize gradient descent method to solve the optimization. The gradient with respect to $\boldsymbol{T}$ is as follows.

$$
\begin{aligned}
\nabla_{\boldsymbol{T}} = \sum_k \Big(& \frac{4}{\left(n_S^{(k)}\right)^2} \sum_{i,j=1}^{n_S^{(k)}} \boldsymbol{T}\boldsymbol{x}_i(\boldsymbol{x}_j)^\top \\
& - \frac{2}{n_S^{(k)} n_T^{(k)}} \sum_{i=1}^{n_S^{(k)}} \sum_{j=1}^{n_T^{(k)}} \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}_j(\boldsymbol{x}_i)^\top \Big) \\
& + \frac{4\eta}{n_S^2} \sum_{i,j=1}^{n_S} \boldsymbol{T}\boldsymbol{x}_i(\boldsymbol{x}_j)^\top \\
& - \frac{2\eta}{n_S n_T} \sum_{i=1}^{n_S} \sum_{j=1}^{n_T} \tilde{\boldsymbol{T}}\tilde{\boldsymbol{x}}_j(\boldsymbol{x}_i)^\top + 2\lambda \boldsymbol{T}, \\
= & \sum_k \left(\boldsymbol{T}\boldsymbol{X}_k \boldsymbol{M}_k \boldsymbol{X}_k^\top - \tilde{\boldsymbol{T}}\tilde{\boldsymbol{X}}_k \boldsymbol{N}_k \tilde{\boldsymbol{X}}_k^\top \right) \\
& + \eta\left(\boldsymbol{T}\boldsymbol{X}\boldsymbol{M}_0\boldsymbol{X}^\top - \tilde{\boldsymbol{T}}\tilde{\boldsymbol{X}}\boldsymbol{N}_0\tilde{\boldsymbol{X}}^\top \right) + 2\lambda \boldsymbol{T},
\end{aligned}
\tag{5.7}
$$

where $\boldsymbol{X}$ is the stack of all source samples, i.e., $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n_S}]$, and $\boldsymbol{X}_k$ is the stack of source samples with label k. The same stacking happens on $\tilde{X}$ and $\tilde{X}_k$ using target samples. Each element for $\boldsymbol{M}_k$, $\boldsymbol{N}_k$, $\boldsymbol{M}_0$, and $\boldsymbol{N}_0$ is $4/\left(n_s^{(k)}\right)^2$, $2/\left(n_s^{(k)} n_t^{(k)}\right)$, $4/n_s^2$, and $2/(n_s n_t)$, respectively. Similarly, I can calculate $\nabla_{\tilde{\boldsymbol{T}}}$ by symmetry. Finally, I solve the optimization by iteratively updating $\boldsymbol{T}$ and $\tilde{\boldsymbol{T}}$ using the calculated gradients.

5.4 Label transfer: Label Alignment via PMU Indices

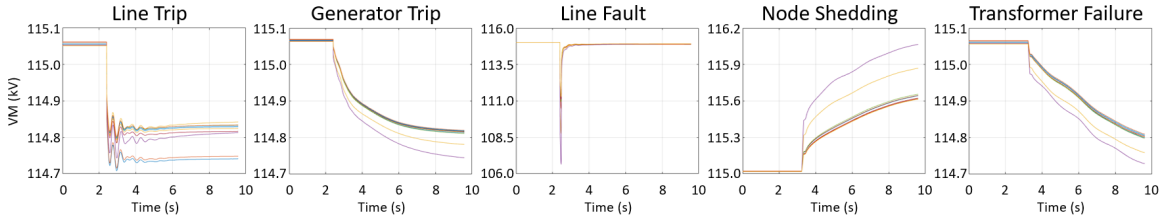


Figure 39. The Visualization of Five Types of Events in Experiments.

While HJDA in Section 5.3 guarantees a well-established data transfer process with solid mathematical explanations, this procedure can only handle the identical label spaces for the source and the target grid. For the event zone estimation problem, however, the label spaces are disjoint. Thus, I propose a label alignment approach to achieve the cross-system label transfer.

As shown in Fig. 37b, system event has a decreasing impact from the event center to further

areas. Thus, the event location can be reflected by the PMU locations. Therefore, I can select some responsible PMUs, each of which aims at monitoring potential events in one local area. For example, in Fig. 38, each system can be divided into 6 zones, i.e., the Northwest, North, Northeast, Southwest, South, and Southeast areas for zone $1 \sim 6$.

I divide the zone based on the similarity measure of event impacts to responsible PMUs. Namely, I select the responsible PMUs as the event zone centers. Then, the event impact similarity helps to identify event zone boundaries, leading to the event zone division. Specifically, I have the following steps. (i) **I choose the event zone center via selecting responsible PMUs.** In general, the selection is based on the prior knowledge of the system, including the system topology, the factors to determine the event frequency like loads, generation reserves, environments, and device ages, and the availability of PMUs, etc. In this chapter, I focus more on demonstrating the potentials of event zone divisions for Transfer Learning. Thus, I assume the responsible PMUs are evenly distributed among the system. Then, for both the source and the target systems, I pick up the same number of responsible PMUs, thus bringing the same number of event zones. Mathematically, the indices of the responsible PMUs (i.e., $1, 2, \cdots, M$ for M responsible PMUs) can form the common set of event zone labels for two grids. (ii) **I identify the event zone boundaries via measuring the similarity of event impacts on the responsible PMUs.** Mathematically, the problem can be solved via finding the PMU that suffers the largest distribution change before and after a certain event. Thus, I utilize Maximum Mean Discrepancy (MMD) for evaluations. Specifically, for the j^{th} PMU in the source or target grid, let $\boldsymbol{w}_j^i$ be the sample at the i^{th} time slot. Then, I can calculate the MMD distance of samples before and after an event:

$$D_j = ||\frac{1}{n_0}\sum_{i=1}^{n_0}\boldsymbol{w}_j^i - \frac{1}{n_1}\sum_{k=n_0}^{n_0+n_1}\boldsymbol{w}_j^k||^2, \tag{5.8}$$

where n_0 and n_1 are the total number of time slots before and after the event, respectively. Therefore, the MMD value D_j represents the corresponding event's impacts on PMU j. Using equation (5.8) to calculate one event's impacts on all the responsible PMUs, I can rank and find the mostly-affected PMU as the true responsible PMU for the specific event.

The complete process essentially relabels one event with its responsible PMU index (i.e., the zone index), and the new label with the categorized geographical information can be directly applied to another grid. Finally, though I only utilize a subset of PMUs for relabeling, all PMU measurements can be inputted into the data transfer framework. This is because in the data transfer process, I

have two projections to maximize the common features. This data-driven process will automatically decide which PMU is important by weighting the PMU values using the transformation matrices.

Remark: in our label transfer process, I have a fixed number of event zones (i.e., M event zones for M responsible PMUs) for the source and the target systems. However, I can do the fine-grained selection by the prior knowledge of the system, including the system topology, the factors to determine the event frequency like loads, generation reserves, environments, and device ages, and the availability of PMUs, etc. Then, the zone numbers can be different for the source and the target grids. In future work, I will extend the topic for Transfer Learning under overlapped but not the same zone label spaces.

5.5 Experiment

5.5.1 Dataset Description

To diversify the testing scenarios, I utilize IEEE 14-bus system (Illinois Center for a Smarter Electric Grid 2013), Illinois 200-bus system (Engineering Texas A&M University 2016a), and South Carolina 500-bus system (Engineering Texas A&M University 2016b) to simulate events. The simulation platform is based on a business-level software Positive Sequence Load Flow (PSLF) (General Electric Energy Consulting 2018) from General Electric (GE) company. For each system event simulation, I change the loading conditions and event locations to demonstrate the robustness of our models. Table 16 displays all event types in our simulations. Further, as shown in Fig. 39, I visualize the event using Voltage Magnitude (VM) data obtained from PSLF. The data are randomly selected from 10 PMUs in the 200-bus system. As for the plots, similar event patterns can be found in (Ma, Basumallik, and Eftekharnejad 2020), which demonstrates the correctness of the simulation. Subsequently, I generate electric measurement streams with 30 samples per second to mimic the realistic PMU streams, including voltage magnitude, voltage angle, current magnitude, current angle, and frequency. To simulate the data-rich source grid and the data-limit target grid, I consider $\frac{n_S}{n_T} = 8$. Then, I first set $n_S = 400$ and $n_T = 50$ to test different scenarios, where I add noise and vary PMU penetration levels. Secondly, I conduct sensitivity analysis with respect to the data points. I denote $N = n_S + n_T$ and vary $N \in \{400, 450, 500, 550, 600, 650, 700\}$ and keep the ratio $\frac{n_S}{n_T} = 8$.

I add white Gaussian noise with signal-to-noise-ratio (SNR) to be 125 to mimic realistic PMU noise (Ye Yuan et al. 2016). Then, I denote $\gamma \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$ as the PMU penetration level to evaluate the model performance with different PMU numbers. For example, for the 200-bus system, $\gamma = 0.05$ implies that I randomly select 10 PMUs to prepare the training data. For this case, Fig. 40 visualizes the PMU locations in the 200-bus system, where the red and green circles represent the nodes with generators and loads, respectively. The circle size represents the relative generation/load values. Finally, I add labels from PMU 1 to PMU 10 for the PMU locations. For the event zone estimation problem, the selected PMUs should cover the 6 monitoring regions in Fig. 38 so that I can form the common zone-index space for label transfer. Finally, I denote $A \to B$ as transfer learning from source grid A to target grid B.

Label	Event Type
1	Line trip
2	Generator trip
3	Line fault
4	Load shedding
5	Transformer failure

Table 16. Categorization of Event Types for Simulations.

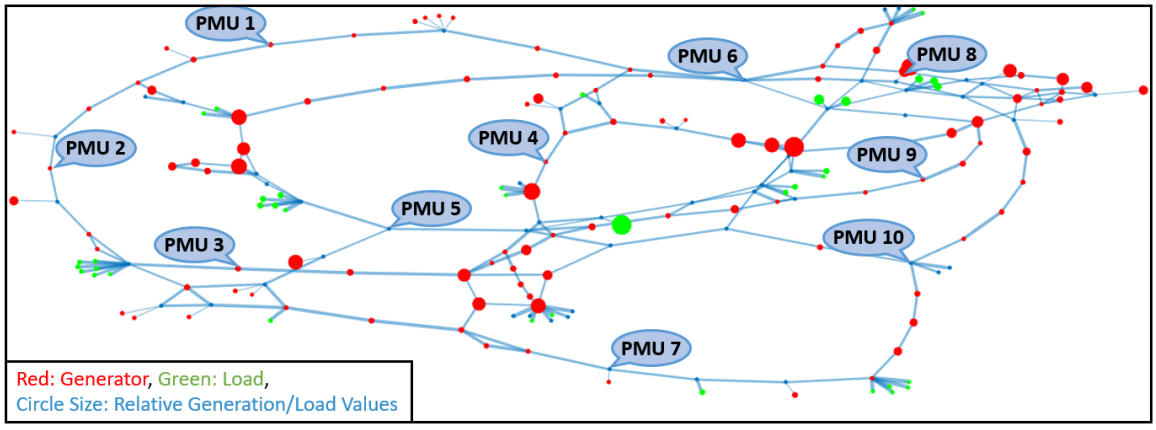


Figure 40. The Visualization of Pmu Locations for the 200-bus System.

5.5.2 Model Description and Implementation Details

To comprehensively demonstrate the high-performance of our models, I have implemented the following models for comparison.

- Principal Component Analysis (PCA) + Resnet (K. He et al. 2016). This is a benchmark method to train a classifier *without* transfer learning. PCA aims to reduce the dimensionality of the PMU data, and Deep Residual Network (Resnet) is the classifier for event type differentiation and event zone estimation. The reason for using Resnet is that Resnet has excellent performances in various ML tasks (Guo and Du 2019; L. Ma et al. 2020; Choi, Ryu, and Kim 2018).
- Heterogeneous Joint Distribution Adaptation (HJDA) + Resnet. I can use our proposed HJDA to convert source and target data to a common feature space. Then, Resnet can be employed to train classifiers based on the common features.
- Transfer Component Analysis (TCA) (Pan et al. 2010) + Resnet. TCA also employs MMD measure to find projections to a latent space from the source and the target measurement spaces. However, the objective only considers minimizing the marginal data distribution. Then, Resnet can be employed to train the features in the latent space.
- Cross-Domain Landmark Selection (CDLS) (Tsai, Yeh, and Wang 2016) + Resnet. The supervised version of CDLS finds the optimal matrix to project the source data to the PCA-based subspace of the target data, where MMD is also employed as the objective for both the marginal and the conditional distributions. With transferred datasets, Resnet can be employed to train the classifier.

All the above methods can be utilized to event zone estimation problems with the help of our proposed label transfer process.

I utilize 3-fold cross-validation to fine-tune the hyper-parameters and obtain the testing accuracy for the final classifiers. Notably, the hyper-parameters of Resnet should stay the same: i.e., the same hidden layers, number of neurons, activation function, learning rate, and so on. Thus, I create fair comparisons among different transfer learning distribution adaptation methods.

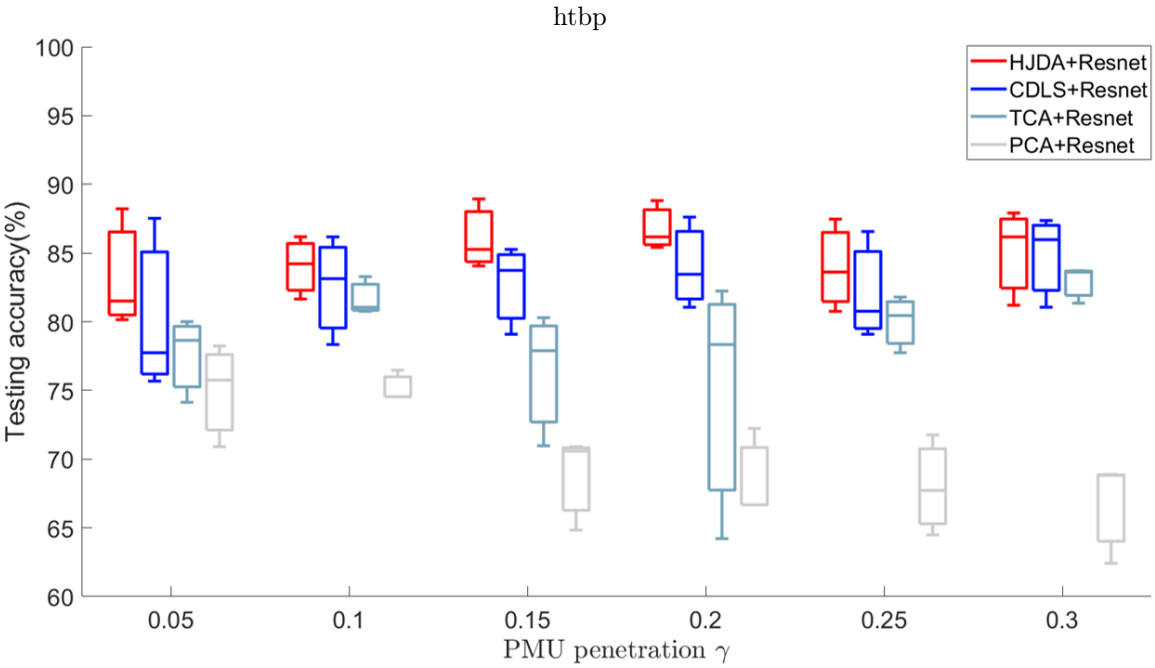


Figure 41. Testing Accuracy (%) for Scenario 14 → 200.

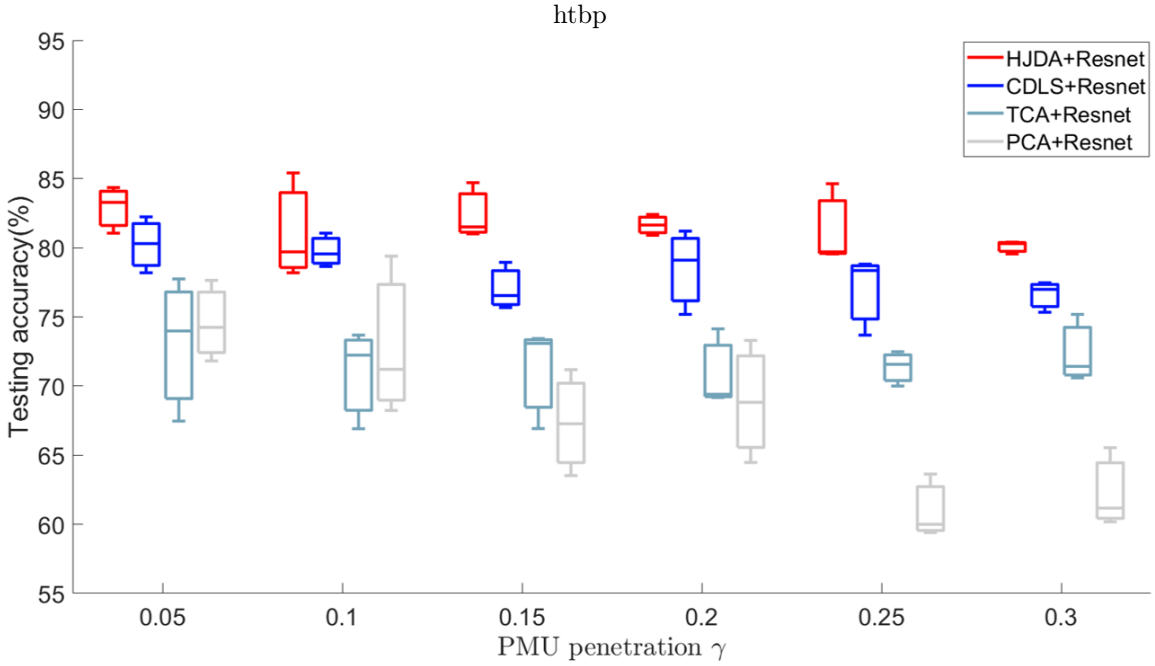


Figure 42. Testing Accuracy (%) for Scenario 200 → 14.

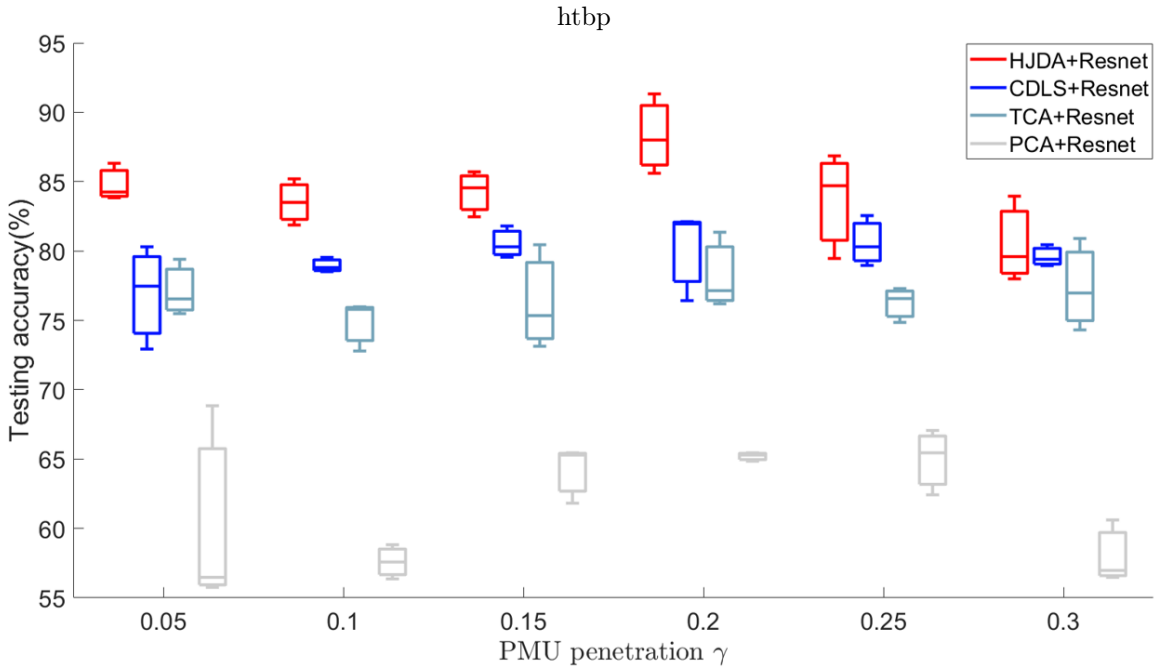


Figure 43. Testing Accuracy (%) for Scenario 14 → 500.

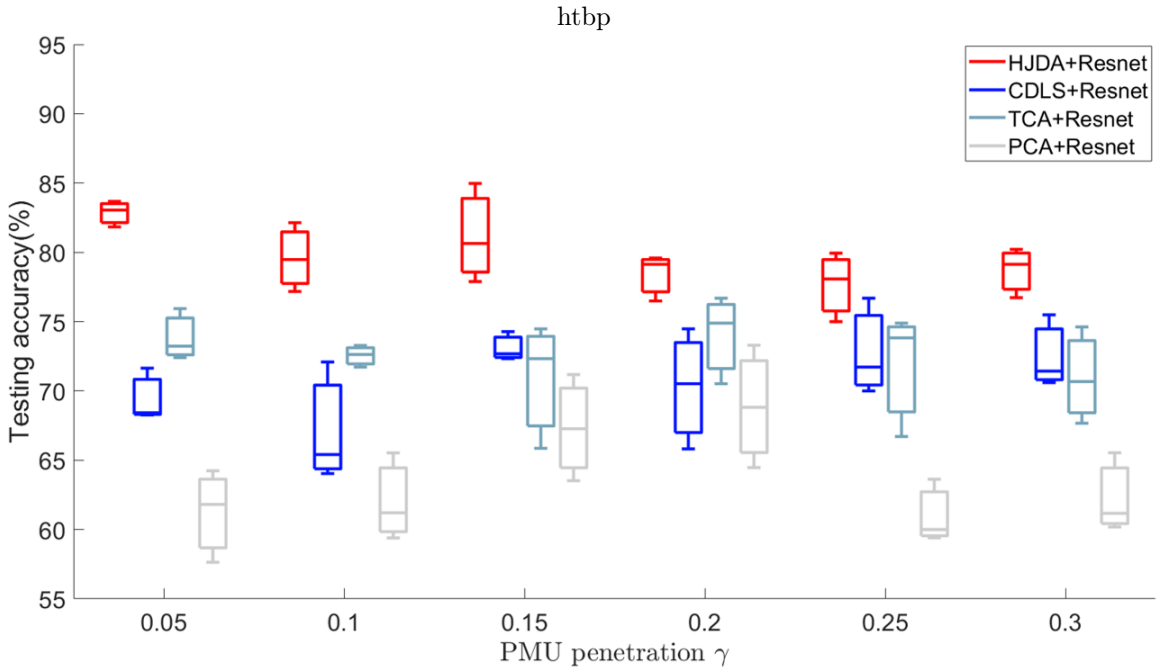


Figure 44. Testing Accuracy (%) for Scenario 500 → 14.

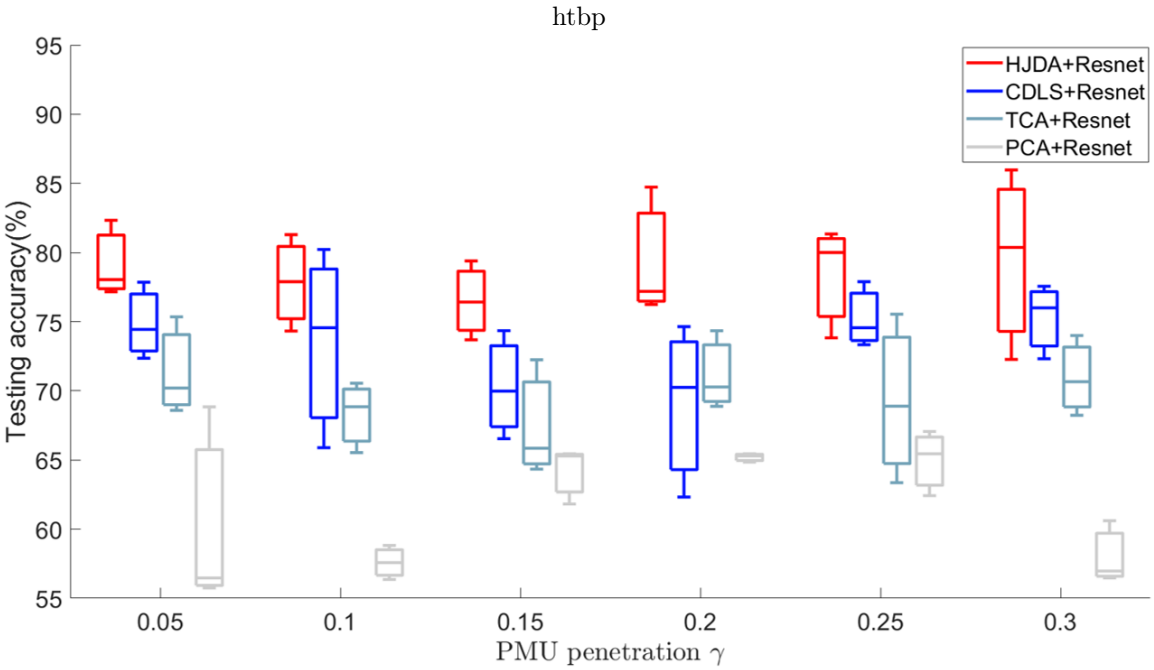


Figure 45. Testing Accuracy (%) for Scenario 200 → 500.

5.5.3 Transfer Learning-based Event Type Differentiation: Highly-Accurate Performance of the Proposed HJDA

Fig. 41 to Fig. 46 illustrate the results of power system event identification using different methods. I conduct comprehensive experiments for every two of the synthetic systems and with changing γ. For each scenario, I find that our proposed HJDA (red box) performs the best over other methods. Specifically, the average testing accuracy among all scenarios is 81.7%, 76.7%, 73.6%, and 64.6% for HJDA, CDLS, TCA, and PCA, respectively. This proves the general high performance of our method. Further, I have the following analyses to verify the principles of our transfer learning framework.

Positive transferability. Compared to the PCA method (grey box) without transfer learning, other 3 transfer learning methods have a significant improvement for the testing accuracy, which demonstrates the positive transferability of the power system PMU data for event type differentiation. As I discussed in Section 5.3.1, the positive effect of transfer learning exists for power systems, which is highly supported by our results.

HJDA vs. TCA. In each trial, HJDA has significant accuracy increasing compared to TCA

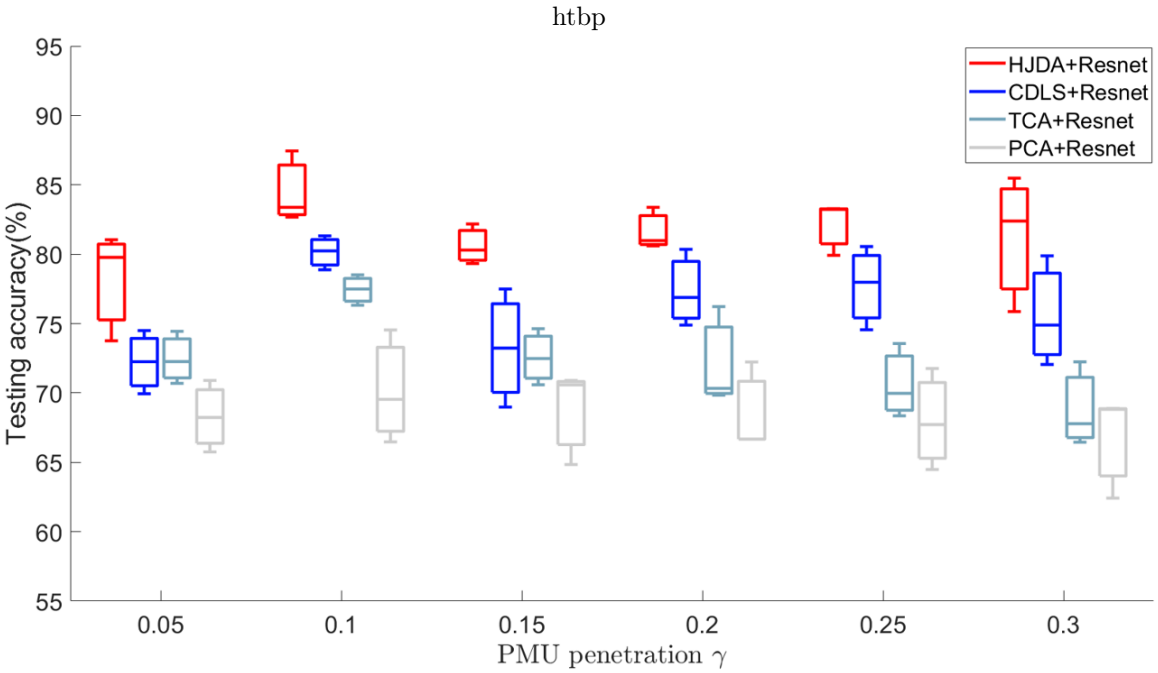


Figure 46. Testing Accuracy (%) for Scenario 500 → 200.

(shallow green box). The average line in the box plot for HJDA is always higher than TCA, and the box plots for these two methods do not have overlaps. The reason is that TCA only focuses on minimizing the distance of marginal data distributions in equation (5.3.2). However, this does not mean the general goal of conditional distribution in equation (5.2) can be minimized. Especially, if the label distributions are different for two systems, equations (5.3) and (5.3.2) indicate that conditional distributions in equation (5.2) can not match. In our simulation, I randomly implement the event, so the label distribution for the source and the target is not the same. Thus, TCA does not perform well. On the other hand, HJDA has a thorough consideration of the joint marginal and conditional distributions, thus obtaining a much better performance.

HJDA vs. CDLS. Compared to the CDLS method (blue box), HJDA has a big improvement when I test transfer learning between 14-bus and 500-bus systems, and a minor improvement when I test transfer learning between 14-bus and 200-bus system or 200-bus and 500-bus system. I have the following explanations for our observations. (i) CDLS method employs PCA to reduce the dimensionality of the target data and finds the optimal projection from the source space to the PCA-based target space with the same objectives of our methods. Thus, our method wins with the ability to search for a better target projection rather than a fixed PCA transformation. Especially,

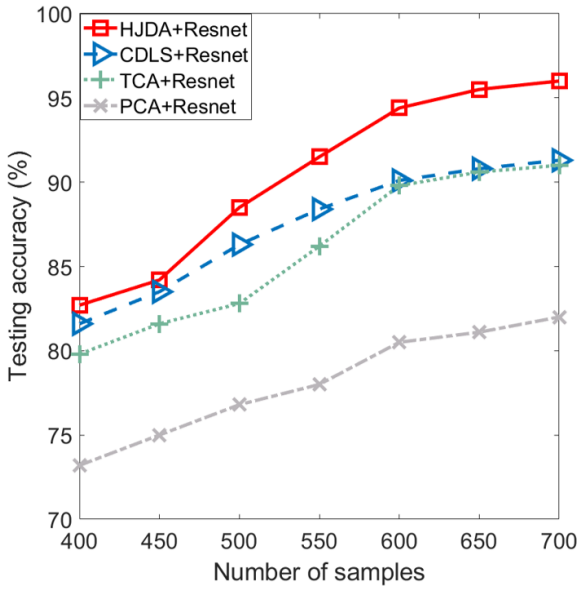
the possibility of finding a better projection is promoted under the event label supervision. (ii) the data distribution difference is largest between 14-bus and 500-bus system, and it is vital to utilize a better projection for the target grid. Thus, HJDA performs much better for transfer learning in 14-bus and 500-bus systems. (iii) since CDLS employ the same objectives for optimization with strong mathematical supports, CDLS can report good results for other transfer learning scenarios when the source and the target grids' data have relatively smaller distribution difference.

Performance with respect to system size. Intuitively, a larger system seems to have richer information and can bring a better result. However, I observe an interesting and counter-intuitive phenomenon: using a smaller-size system as the source grid can achieve a better performance than treating it as the target grid. This is especially observed for transfer learning between 14-bus and 200-bus/500-bus systems.

The reason is that when I minimize the conditional distribution given labels, i.e., the first sum term in equation (5.3.2), using an integer to represent a label with high complexity may cause troubles. More specifically, for one specific event type, different event locations and magnitudes essentially demand a more careful event categorization in minimizing the conditional distribution differences. For example, the active power of generators for the 14-bus system has less variability than that for the 500-bus system, making the data variance of the 500-bus system larger but with the same label "generator trip". Therefore, HJDA suffers an over-minimization that compresses too much information in the 500-bus system to match the distribution in the 14-bus system. If I treat the 500-bus system as the source grid, more data are used and over-compressed, thus deteriorating the performance. To mitigate this issue, methods like label complexity analysis, regularization, or non-linear domain adaptation can be utilized, and I will explore these methods in the future.

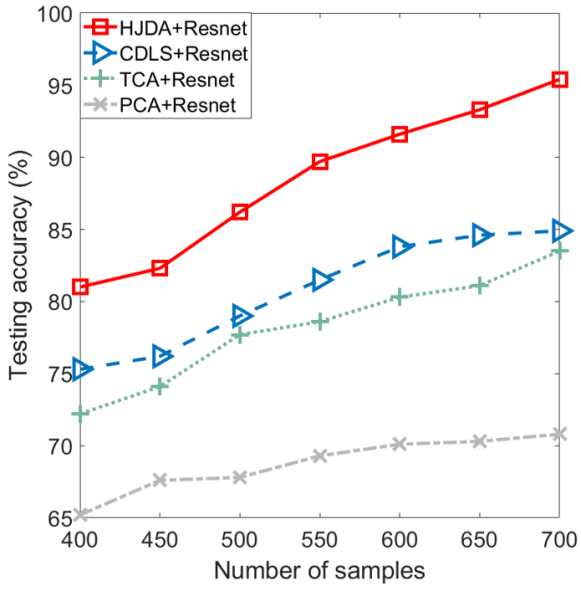
5.5.4 Transfer Learning-based Event Zone Estimation: Relabeling Technique Enable Positive Transfer and High Accuracy of the Proposed HJDA

For event zone estimation, I divide each system into 6 zones to align the label, as shown in Fig. 38. For each zone, I assume there is a PMU (i.e., the responsible PMU) for monitoring. However, as I discussed, the grid division can be flexible according to the availability of PMUs. Then, I conduct line fault for different lines that are uniformly distributed in the monitoring regions. Subsequently,



htbp

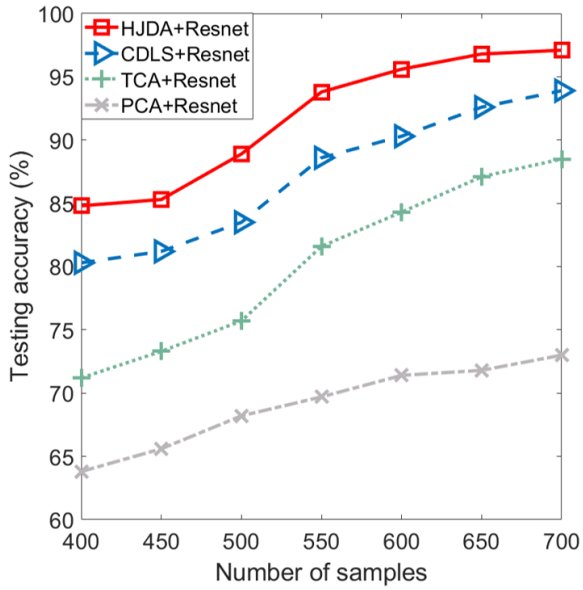
Figure 47. Testing Accuracy (%) for Scenario 14 → 200.



htbp

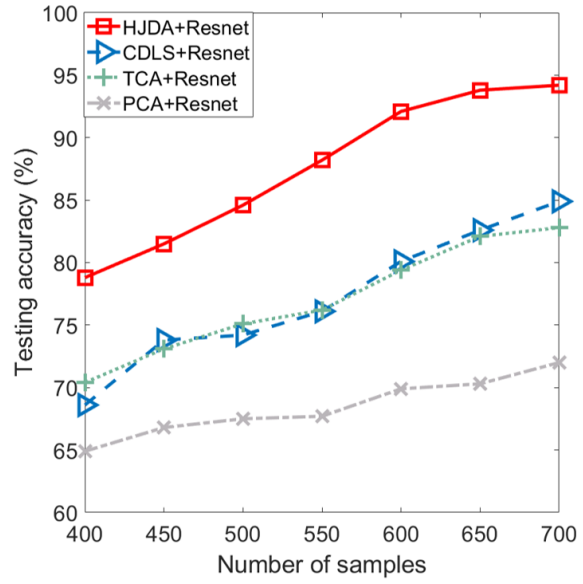
Figure 48. Testing Accuracy (%) for Scenario 200 → 14.

for each line-fault event, I calculate its responsible PMU based on the ranking of the calculated MMD value in equation (5.8). If the k^{th} $(1 \leq k \leq 6)$ responsible PMU has the largest MMD value, I relabel the certain line fault as k.



htbp

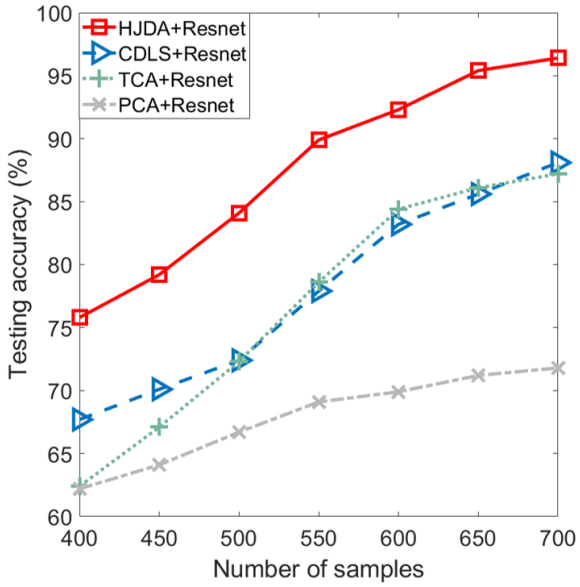
Figure 49. Testing Accuracy (%) for Scenario 14 → 500.



htbp

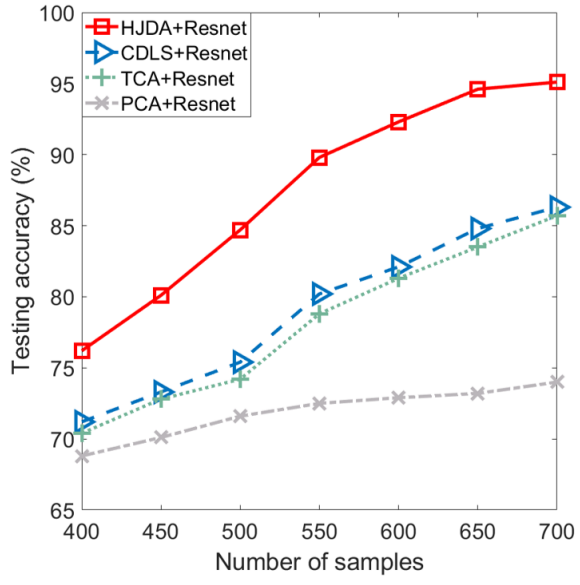
Figure 50. Testing Accuracy (%) for Scenario 500 → 14.

After our relabeling techniques, I conduct different methods for data transfer and implement one fixed Resnet to be the event zone estimation classifier. The final result is shown in Table 17. I observe similar behaviors discussed in Section 5.5.3. Specifically, I have (*i*) HJDA performs better than PCA, TCA, and CDLS. The averaged testing accuracy among all scenarios is 74.7%, 70.1%, 68.7%, and 60.6% for HJDA, CDLS, TCA, and PCA, respectively. (*ii*) information migration from a



htbp

Figure 51. Testing Accuracy (%) for Scenario 200 → 500.



htbp

Figure 52. Testing Accuracy (%) for Scenario 500 → 200.

Table 17. The Testing Accuracy (%) (Mean ± Standard Deviation) of Different Methods + Resnet for Event Zone Estimation.

	HJDA	CDLS	TCA	PCA
$14 \rightarrow 200$	$\mathbf{80.2 \pm 4.4}$	74.4 ± 6.2	68.7 ± 3.5	60.6 ± 4.3
$200 \rightarrow 14$	$\mathbf{73.6 \pm 3.6}$	66.5 ± 5.8	64.4 ± 4.7	56.3 ± 7.8
$14 \rightarrow 500$	$\mathbf{76.5 \pm 5.4}$	73.5 ± 3.3	69.7 ± 4.3	58.3 ± 8.4
$500 \rightarrow 14$	$\mathbf{70.9 \pm 2.4}$	68.4 ± 3.3	60.6 ± 6.9	56.3 ± 5.6
$200 \rightarrow 500$	$\mathbf{75.0 \pm 6.7}$	68.3 ± 4.9	70.0 ± 6.7	58.3 ± 8.4
$500 \rightarrow 200$	$\mathbf{72.6 \pm 4.1}$	69.7 ± 2.3	64.6 ± 5.0	60.6 ± 5.9

small system to a large system still brings better results. This indicates that the over-compression phenomenon still exists, explained in Section 5.5.3. Basically, for a larger system, the monitoring sub-region is wider with more variability for its inner events. For example, some line faults have a bigger impact on the systems while others don't. Thus, compressing all of these event data to match the conditional distribution hurt the final classifier performance. I demand further theoretical frameworks to analyze and prevent the over-compression problem.

5.5.5 Sensitivity Analysis with Respect to Data Numbers: Increasing Data Points Boost the Performance of HJDA

In Section 5.5.3, I utilize $N = 450$ to demonstrate the results and show large improvements of our HJDA over other methods. To further illustrate the improvements of all methods giving more data, I conduct a sensitivity analysis with respect to data numbers for event type differentiation. Specifically, I vary $N \in \{400, 450, 500, 550, 600, 650, 700\}$. Then, I report the mean of the testing accuracy, and the results are shown from Fig. 47 to Fig. 52. I find that if $N \geq 550$, our method can reach the average accuracy larger or equal to 90.5%. When $N = 700$, the average accuracy can be 95.7%. This shows that I can provide a reliable result for the utility partners. Further, I observe that for each scenario, our method enjoys the largest improvement when the number of data points increases. This illustrates the advantages of our method to efficiently absorb new information from new data samples. Such a significant result is due to our method's complete consideration of conditional distribution and marginal distribution differences (compared to TCA + Resnet method) and flexible projection process (compared to CDLS + Resnet method), as illustrated in Section V-C.

5.5.6 Computational Time for Event Simulation and Model Training and Testing

The computational time is one of the most important evaluations for real-time applications. Thus, I provide the time for developing both the power network for the dynamic simulation and the deep residual network (Resnet) for classification. The former represents the simulation time in PSLF and the latter includes the training/testing time of the classification model.

For the PSLF simulation of the power networks, I observe that the simulation time depends on the system size. Specifically, for 14-bus, 200-bus, and 500-bus systems, the average simulation times are 2.5s, 3.3s and 4.2s, respectively. This implies that PSLF can quickly generate high-quality event data for the research usage.

For Resnet, our experiments include different transfer learning methods, including HJDA + Resnet, CDLS + Resnet, TCA + Resnet, and PCA + Resnet. Specifically, HJDA, CDLS, TCA, and PCA all include the dimensionality reduction process and I restrict the projected data to have the same dimensionality and the Resnet to have the same architecture. Thus, I expect close training and testing time for HJDA + Resnet, CDLS + Resnet, TCA + Resnet. As for the benchmark PCA + Resnet without transfer learning, I only utilize the target data to train and test, leading to much smaller training and testing time. Table 18 illustrates the result when I utilize the example of $200 \rightarrow 500$ with $N = 450$ for transfer learning-based methods and $N = n_T = 50$ for the benchmark method. Finally, I consider 3-fold cross validation. Thus, for transfer learning-based methods, I obtain 300 data points for training and 150 points for testing. For the benchmark method, I obtain 33 data points for training and 17 data points for testing.

Table 18. The Training and Testing Time (S) for Resnet with Different Transfer Learning Methods for the Scenario $200 \rightarrow 500$.

	HJDA	CDLS	TCA	PCA
Training time	12.51	13.65	13.01	1.64
Testing time	0.14	0.15	0.15	0.03

I find that (1) it takes around 13s for training with 300 data points. This training time is affordable as the training process can be treated off-line to prepare the classifier, which doesn't affect

the real-time classification. (2) it takes around 0.23s for testing with 150 data points, which indicates that I can achieve the real-time event identification.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, I have addressed the challenges of event detection and data quality in power systems by integrating intrinsic and extrinsic knowledge transfer.

In Chapter 2, we introduced intrinsic data interpolation via sensor fusion to improve data quality in power systems. By merging PMU and SCADA sensor data, we addressed spatial-temporal challenges and leveraged deep neural networks to predict and recover missing dynamic states. This approach provided a novel solution to the problem of differing sensor resolutions and locations, ensuring accurate and reliable data for event detection and monitoring.

Chapter 3 presented extrinsic data interpolation and enhancement, transforming low-resolution meters into "virtual" PMUs. By utilizing inherent physics-informed relationships and high-resolution data, missing values were algorithmically interpolated, resulting in a cost-effective yet high-quality power system monitoring solution. The proposed method demonstrated its effectiveness in enhancing data quality and providing reliable power system monitoring.

In Chapter 4, we developed advanced machine learning models for intrinsic event detection, leveraging extensive datasets and tensor learning techniques to improve precision and enable real-time reactions. By effectively analyzing large datasets, these models reduced redundant information and irrelevant factors, improving the accuracy of event detection. The proposed models demonstrated outstanding performance in recognizing underlying patterns and achieving high precision in event classification.

Chapter 5 focused on extrinsic event identification using transfer learning. We proposed a framework that allowed knowledge to be transferred from data-rich grids to data-limited grids, enabling the latter to gain valuable insights from established grids. This approach addressed the challenges of limited event data in new grids and provided a robust event detection method. The framework incorporated transfer learning techniques to migrate knowledge and adapt it to the target grid, ensuring accurate event detection and classification.

Through extensive numerical validations, we demonstrated the effectiveness and high performance of the proposed models and techniques in event detection and data quality improvement. The results validated the robustness and accuracy of the developed models, showcasing their potential for real-world applications in power systems.

6.2 Future Work

In the future, the following aspects should be further investigated to make more contributions to the engineering and academic domains.

1. To learn symbolic equations of physical systems, the coherency of input data may cause the issue of not able to identify the system theoretically. Thus, further investigations must be made to study the identifiability of the physical system.
2. With the learned physics-consistent ML model, the next step is to achieve better planning, economic dispatch, and system control.
3. More types of tensor decomposition techniques should be investigated to improve the tensor-based event identification.

REFERENCES

Abdi, Hervé, and Lynne J Williams. 2010. 'Principal component analysis.' *Wiley interdisciplinary reviews: computational statistics* 2 (4): 433–459.

Agarap, Abien Fred. 2019. 'Deep Learning Using Rectified Linear Units (ReLU).' *arXiv preprint arXiv:1803.08375.*

Al Rahhal, Mohamad M, Yakoub Bazi, Taghreed Abdullah, Mohamed L Mekhalfi, Haikel AlHichri, and Mansour Zuair. 2018. 'Learning a multi-branch neural network from multiple sources for knowledge adaptation in remote sensing imagery.' *Remote Sensing* 10 (12): 1890.

Alimi, Oyeniyi Akeem, Khmaies Ouahada, and Adnan M Abu-Mahfouz. 2020. 'A Review of Machine Learning Approaches to Power System Security and Stability.' *IEEE Access.*

Aminifar, Farrokh, Mahmud Fotuhi-Firuzabad, and Amir Safdarian. 2013. 'Optimal PMU Placement Based on Probabilistic Cost/Benefit Analysis.' *IEEE Transactions on Power Systems* 28 (1): 566–567. https://doi.org/10.1109/TPWRS.2012.2198312.

Ayad, Abdelrahman, Hany EZ Farag, Amr Youssef, and Ehab F El-Saadany. 2018. 'Detection of false data injection attacks in smart grids using recurrent neural networks.' In *IEEE Power & Energy Society Innovative Smart Grid Technologies Conference,* 1–5.

Baldwin, Thomas L, Lamine Mili, Monte B Boisen, and Ram Adapa. 1993. 'Power system observability with minimal phasor measurement placement.' *IEEE Transactions on Power systems* 8 (2): 707–715.

Basetti, Vedik, and Ashwani Kumar Chandel. 2016. 'Simultaneous placement of PMUs and communication infrastructure in WAMS using NSGA-II.' *IETE Technical Review* 33 (6): 621–637.

Batista, Gustavo EAPA, and Maria Carolina Monard. 2003. 'An analysis of four missing data treatment methods for supervised learning.' *Applied artificial intelligence* 17 (5-6): 519–533.

Berthelot, David, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. 2019. 'Mixmatch: A holistic approach to semi-supervised learning.' *Advances in Neural Information Processing Systems.*

Bhatt, N., S. Sarawgi, R. O'Keefe, P. Duggan, M. Koenig, M. Leschuk, S. Lee, et al. 2009. 'Assessing vulnerability to cascading outages.' In *2009 IEEE/PES Power Systems Conference and Exposition,* 1–9. https://doi.org/10.1109/PSCE.2009.4840032.

Blasch, Erik, Haoran Li, Zhihao Ma, and Yang Weng. 2021. 'The Powerful Use of AI in the Energy Sector: Intelligent Forecasting.' *arXiv preprint arXiv:2111.02026.*

Blitzer, John, Ryan McDonald, and Fernando Pereira. 2006. 'Domain Adaptation with Structural Correspondence Learning.' In *Conference on Empirical Methods in Natural Language Processing.*

Bottou, Léon. 2012. 'Stochastic gradient descent tricks.' In *Neural networks: Tricks of the trade,* 421–436. Springer.

Brahma, S., R. Kavasseri, H. Cao, N. R. Chaudhuri, T. Alexopoulos, and Y. Cui. 2017. 'Real-time identification of dynamic events in power systems using PMU data, and potential applications, models, promises, and challenges.' *IEEE Transactions on Power Delivery.*

Brunton, Steven L, Joshua L Proctor, and J Nathan Kutz. 2016. 'Discovering governing equations from data by sparse identification of nonlinear dynamical systems.' *Proceedings of the national academy of sciences* 113 (15): 3932–3937.

Camões, Francisco, Julio A. D. Massignan, Vladimiro Miranda, and João B. A. London. 2022. 'Sliding-Priors for Bayesian Information Fusion in SCADA+PMU-based State Estimation.' In *2022 17th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS),* 1–6. https://doi.org/10.1109/PMAPS53380.2022.9810558.

Cao, Bokai, Chun-Ta Lu, Xiaokai Wei, S Yu Philip, and Alex D Leow. 2016. 'Semi-supervised tensor factorization for brain network analysis.' In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer.

Chang, Jaewon, Glauco N Taranto, and Joe H Chow. 1997. 'Dynamic state estimation using a nonlinear observer for optimal series-capacitor switching control.' *International Journal of Electrical Power & Energy Systems* 19 (7): 441–447.

Cheng, Chung-Yuan, Wan-Ling Tseng, Ching-Fen Chang, Chuan-Hsiung Chang, and Susan Shur-Fen Gau. 2020. 'A deep learning approach for missing data imputation of rating scales assessing attention-deficit hyperactivity disorder.' *Frontiers in psychiatry,* https://doi.org/10.3389/fpsyt.2020.00673.

Cheng, Yuanbin, Nanpeng Yu, Brandon Foggo, and Koji Yamashita. 2022. 'Online Power System Event Detection via Bidirectional Generative Adversarial Networks.' *IEEE Transactions on Power Systems.*

Chiu, Pao-Hsiung, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. 2022. 'CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method.' *Computer Methods in Applied Mechanics and Engineering* 395:114909.

Choi, Hyungeun, Seunghyoung Ryu, and Hongseok Kim. 2018. 'Short-Term Load Forecasting based on ResNet and LSTM.' In *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids.*

Cui, M., J. Wang, J. Tan, A. Florita, and Y. Zhang. 2018. 'A novel event detection method using PMU data with high precision.' *IEEE Transactions on Power Systems,* https://doi.org/10.1109/TPWRS.2018.2859323.

Data Miner 2 — dataminer2.pjm.com. https://dataminer2.pjm.com/feed/load_frcstd_hist. [Accessed 20-Apr-2023].

Day, Oscar, and Taghi M Khoshgoftaar. 2017. 'A Survey on Heterogeneous Transfer Learning.' *Journal of Big Data.*

De Boor, Carl. 1994. 'Polynomial interpolation in several variables.' In *Studies in Computer Science: In Honor of Samuel D. Conte,* 87–109. Springer.

De Yong, David, Sudipto Bhowmik, and Fernando Magnago. 2015. 'An effective power quality classifier using wavelet transform and support vector machines.' *Expert Systems with Applications.*

Dong, Weisheng, Lei Zhang, Rastislav Lukac, and Guangming Shi. 2013. 'Sparse representation based image interpolation with nonlocal autoregressive modeling.' *IEEE Transactions on Image Processing* 22 (4): 1382–1394.

Dorfler, Florian, Fabio Pasqualetti, and Francesco Bullo. 2013. 'Continuous-time distributed observers with discrete communication.' *IEEE Journal of Selected Topics in Signal Processing* 7 (2): 296–304.

Duan, Lixin, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. 2009. 'Domain Adaptation from Multiple Sources via Auxiliary Classifiers.' In *International Conference on Machine Learning.*

Engineering Texas A&M University. 2016a. 'Illinois 200-bus system: ACTIVSg200,' https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg200/.

———. 2016b. 'SouthCarolina 500-bus system: ACTIVSg500,' https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg500/.

Feng, Liang, Yew-Soon Ong, Ivor Wai-Hung Tsang, and Ah-Hwee Tan. 2012. 'An Evolutionary Search Paradigm that Learns with Past Experiences.' In *IEEE Congress on Evolutionary Computation.*

Fukami, Kai, Koji Fukagata, and Kunihiko Taira. 2021. 'Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows.' *Journal of Fluid Mechanics* 909:A9.

Ganjavi, Amin, Edward Christopher, C Mark Johnson, and Jon Clare. 2017. 'A study on probability of distribution loads based on expectation maximization algorithm.' In *IEEE Power & Energy Society Innovative Smart Grid Technologies Conference.*

Gao, Pengzhi, Meng Wang, Scott G Ghiocel, Joe H Chow, Bruce Fardanesh, and George Stefopoulos. 2015. 'Missing data recovery by exploiting low-dimensionality in power system synchrophasor measurements.' *IEEE Transactions on Power Systems* 31 (2): 1006–1013.

Ge, Yinyin, Alexander J Flueck, Dae-Kyeong Kim, Jong-Bo Ahn, Jae-Duck Lee, and Dae-Yun Kwon. 2015. 'Power system real-time event detection and associated data archival reduction based on synchrophasors.' *IEEE Transactions on Smart Grid* 6 (4): 2088–2097.

General Electric Energy Consulting. 2018. 'General Electric Concorda PSLF,' https://www.geenergyconsulting.com/practice-area/software-products/pslf.

Ghahremani, Esmaeil, and Innocent Kamwa. 2011a. 'Dynamic state estimation in power system by applying the extended Kalman filter with unknown inputs to phasor measurements.' *IEEE Transactions on Power Systems* 26 (4): 2556–2566.

———. 2011b. 'Online state estimation of a synchronous generator using unscented Kalman filter from phasor measurements units.' *IEEE Transactions on Energy Conversion* 26 (4): 1099–1108.

Gharavi, Hamid, and Bin Hu. 2017. 'Wireless infrastructure M2M network for distributed power grid monitoring.' *IEEE network* 31 (5): 122–128.

Ghosal, Malini, and Vittal Rao. 2015. 'Fusion of PMU and SCADA data for dynamic state estimation of power system.' In *IEEE North American Power Symposium,* 1–6.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. 2011. 'Deep sparse rectifier neural networks.' In *Proceedings of the fourteenth international conference on artificial intelligence and statistics,* 315–323. JMLR Workshop and Conference Proceedings.

Graves, Alex, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2008. 'A novel connectionist system for unconstrained handwriting recognition.' *IEEE Transactions on pattern analysis and machine intelligence* 31 (5): 855–868.

Gretton, Arthur, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. 'A Kernel Two-sample Test.' *Journal of Machine Learning Research.*

Guillaumin, Matthieu, and Vittorio Ferrari. 2012. 'Large-scale Knowledge Transfer For Object Localization in Imagenet.' In *IEEE Conference on Computer Vision and Pattern Recognition.*

Guo, Minghui, and Yongzhao Du. 2019. 'Classification of Thyroid Ultrasound Standard Plane Images Using ResNet-18 Networks.' In *IEEE International Conference on Anti-counterfeiting, Security, and Identification.*

Gupta, Ankita, Gurunath Gurrala, and PS Sastry. 2018. 'An online power system stability monitoring system using convolutional neural networks.' *IEEE Transactions on Power Systems* 34 (2): 864–872.

Habermann, Christian, and Fabian Kindermann. 2007. 'Multidimensional spline interpolation: Theory and applications.' *Computational Economics* 30:153–169.

Hafiz, Faizal, Scott Abecrombie, Andrew Eaton, Chirag Naik, and Akshya Swain. 2017. 'Power quality event identification using wavelet packet transform: A comprehensive investigation.' In *TENCON 2017-2017 IEEE Region 10 Conference,* 2978–2983. IEEE.

Hairer, Ernst. 2011. 'Solving differential equations on manifolds.' *Lecture notes.*

Hawkins, Kristen. 2022. *Inside-the-meter intelligence to become the norm,* October. https://utilities.sense.com/inside-the-meter-intelligence-to-become-the-norm/.

He, Haibo, and Edwardo A Garcia. 2009. 'Learning from imbalanced data.' *IEEE Transactions on knowledge and data engineering* 21 (9): 1263–1284.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.' In *Proceedings of the IEEE international conference on computer vision,* 1026–1034.

———. 2016. 'Deep residual learning for image recognition.' In *IEEE Conference on Computer Vision and Pattern Recognition.*

He, Lifang, Chun-Ta Lu, Guixiang Ma, Shen Wang, Linlin Shen, S Yu Philip, and Ann B Ragin. 2017. 'Kernelized support tensor machines.' In *International Conference on Machine Learning.*

He, Xianqiang, Qiang Liu, Jianxing Tang, Qinfeng Ma, Guosong Wang, and Mingshun Liu. 2020. 'Power System Frequency Situation Prediction Method Based on Transfer Learning.' In *IEEE Asia-Pacific Power and Energy Engineering Conference.*

Hillebrecht, Birgit, and Benjamin Unger. 2022. 'Certified machine learning: A posteriori error estimation for physics-informed neural networks.' In *IEEE International Joint Conference on Neural Networks,* 1–8.

Hsu, Chih-Wei, and Chih-Jen Lin. 2002. 'A comparison of methods for multiclass support vector machines.' *IEEE transactions on Neural Networks.*

'IEEE Standard for Synchrophasor Measurements for Power Systems.' 2011. *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005),* 1–61. https://doi.org/10.1109/IEEESTD.2011.6111219.

Illinois Center for a Smarter Electric Grid. 2013. 'IEEE 14-Bus System,' https://icseg.iti.illinois.edu/ieee-14-bus-system/.

Jia, Yongna, Siwei Yu, and Jianwei Ma. 2018. 'Intelligent interpolation by Monte Carlo machine learning.' *Geophysics* 83 (2): V83–V97.

Jiang, Min, Yulong Ding, Ben Goertzel, Zhongqiang Huang, Changle Zhou, and Fei Chao. 2014. 'Improving machine vision via incorporating expectation-maximization into Deep Spatio-Temporal learning.' In *2014 International Joint Conference on Neural Networks (IJCNN),* 1804–1811. https://doi.org/10.1109/IJCNN.2014.6889723.

Jones, Jonathan Spencer. *How next generation smart meters could deliver new use cases — smart-energy.com.* https://www.smart-energy.com/industry-sectors/smart-meters/how-next-generation-smart-meters-could-deliver-new-use-cases/. [Accessed 27-10-2023].

Joos Korstanje. 2021. 'The F1 score,' https://towardsdatascience.com/the-f1-score-bec2bbc38aa6.

Kapoor, Rajiv, Rashmi Gupta, Sudan Jha, Raghvendra Kumar, et al. 2018. 'Boosting performance of power quality event identification with KL Divergence measure and standard deviation.' *Measurement* 126:134–142.

Kim, Cheolmin, Kibaek Kim, Prasanna Balaprakash, and Mihai Anitescu. 2019. 'Graph convolutional neural networks for optimal load shedding under line contingency.' In *IEEE Power & Energy Society General Meeting,* 1–5.

Kim, Do-In. 2021. 'Complementary Feature Extractions for Event Identification in Power Systems Using Multi-Channel Convolutional Neural Network.' *Energies* 14 (15).

Kim, Do-In, Tae Yoon Chun, Sung-Hwa Yoon, Gyul Lee, and Yong-June Shin. 2015. 'Wavelet-based event detection method using PMU data.' *IEEE Transactions on Smart Grid.*

Kingma, Diederik P, and Jimmy Ba. 2017. 'Adam: A method for stochastic optimization.' *arXiv preprint arXiv:1412.6980.*

Kolda, T., and B. Bader. 2009. 'Tensor Decompositions and Applications.' *Journal of Applied Mathematics.*

Kotsiantis, Sotiris B. 2011. 'Cascade generalization with reweighting data for handling imbalanced problems.' *The Computer Journal* 54 (9): 1547–1559.

Kung, Sun Yuan. 2014. *Kernel methods and machine learning.* Cambridge University Press.

Le, Ngoc Thien, and Watit Benjapolakul. 2018. 'A data imputation model in phasor measurement units based on bagged averaging of multiple linear regression.' *IEEE Access* 6:39324–39333.

Lee, Dongchan, and Deepa Kundur. 2014. 'Cyber attack detection in PMU measurements via the expectation-maximization algorithm.' In *IEEE Global Conference on Signal and Information Processing.*

Li, H., Y. Weng, E. Farantatos, and M. Patel. 2019. 'An Unsupervised Learning Framework for Event Detection, Type Identification and Localization Using PMUs Without Any Historical Labels.' In *IEEE Power Energy Society General Meeting,* 1–5.

Li, Haoran, Zhihao Ma, and Yang Weng. 2022. 'A Transfer Learning Framework for Power System Event Identification.' *IEEE Transactions on Power Systems,* 1–1. https://doi.org/10.1109/TPWRS.2022.3153445.

Li, Haoran, Zhihao Ma, Yang Weng, Erik Blasch, and Surya Santoso. 2022. 'Structural Tensor Learning for Event Identification with Limited Labels.' *IEEE Transactions on Power Systems,* 1–15. https://doi.org/10.1109/TPWRS.2022.3231262.

Li, Haoran, Zhihao Ma, Yang Weng, and Evangelos Farantatos. 2022. 'Transfer Learning for Event-Type Differentiation on Power Systems.' In *2022 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA),* 1–6. IEEE.

Li, Haoran, and Yang Weng. 2021. 'Physical equation discovery using physics-consistent neural network (PCNN) under incomplete observability.' In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining.*

Li, Haoran, Yang Weng, Evangelos Farantatos, and Mahendra Patel. 2019a. '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.*

———. 2019b. 'An unsupervised learning framework for event detection, type identification and localization using PMUs without any historical labels.' In *IEEE Power & Energy Society General Meeting.*

Li, Haoran, Yang Weng, and Hanghang Tong. 2020. 'Heterogeneous Transfer Learning on Power Systems: A Merged Multi-modal Gaussian Graphical Model.' In *IEEE International Conference on Data Mining.*

———. 2022. 'CoNSoLe: Convex Neural Symbolic Learning.' *arXiv preprint arXiv:2206.00257.*

Li, Haoran, Yang Weng, Vijay Vittal, and Erik Blasch. 2023. 'Distribution Grid Topology and Parameter Estimation Using Deep-Shallow Neural Network with Physical Consistency.' *IEEE Transactions on Smart Grid.*

Li, Wenting, Meng Wang, and Joe H Chow. 2018. 'Real-time event identification through low-dimensional subspace characterization of high-dimensional synchrophasor data.' *IEEE Transactions on Power Systems* 33 (5): 4937–4947.

Li, Zikang, Hao Liu, Junbo Zhao, Tianshu Bi, and Qixun Yang. 2021. 'Fast power system event identification using enhanced LSTM network with renewable energy integration.' *IEEE Transactions on Power Systems* 36 (5): 4492–4502.

Liao, Mang, Di Shi, Zhe Yu, Zhehan Yi, Zhiwei Wang, and Yingmeng Xiang. 2018. 'An alternating direction method of multipliers based approach for PMU data recovery.' *IEEE Transactions on Smart Grid.*

Liu, Guolong, Jinjin Gu, Junhua Zhao, Fushuan Wen, and Gaoqi Liang. 2020. 'Super Resolution Perception for Smart Meter Data.' *Information Sciences.*

Liu, Shengyuan, Shutang You, Zhenzhi Lin, Chujie Zeng, Hongyu Li, Weikang Wang, Xuetao Hu, and Yilu Liu. 2021. 'Data-driven event identification in the US power systems based on 2D-OLPP and RUSBoosted trees.' *IEEE Transactions on Power Systems* 37 (1): 94–105.

Liu, Shengyuan, Yuxuan Zhao, Zhenzhi Lin, Yilu Liu, Yi Ding, Li Yang, and Shimin Yi. 2019. 'Data-driven event detection of power systems based on unequal-interval reduction of PMU data and local outlier factor.' *IEEE Transactions on Smart Grid* 11 (2): 1630–1643.

Liu, Yang, and Kai Sun. 2020. 'Solving Power System Differential Algebraic Equations Using Differential Transformation.' *IEEE Transactions on Power Systems* 35 (3): 2289–2299. https://doi.org/10.1109/TPWRS.2019.2945512.

Lock, Eric F. 2018. 'Tensor-on-Tensor Regression.' *Journal of Computational and Graphical Statistics.*

Loh, Po-Ling, and Martin J Wainwright. 2011. 'High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity.' *Advances in neural information processing systems* 24.

Long, Mingsheng, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. 2013. 'Transfer Feature Learning with Joint Distribution Adaptation.' In *IEEE International Conference on Computer Vision.*

Ma, Dazhong, Xuguang Hu, Huaguang Zhang, Qiuye Sun, and Xiangpeng Xie. 2019. 'A hierarchical event detection method based on spectral theory of multidimensional matrix for power system.' *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51 (4): 2173–2186.

Ma, Li, Renjun Shuai, Xuming Ran, Wenjia Liu, and Chao Ye. 2020. 'Combining DC-GAN with ResNet for Blood Cell Image Classification.' *Medical & Biological Engineering & Computing.*

Ma, Rui, Sagnik Basumallik, and Sara Eftekharnejad. 2020. 'A PMU-based data-driven approach for classifying power system events considering cyberattacks.' *IEEE Systems Journal.*

Ma, Zhihao, Haoran Li, Yang Weng, Erik Blasch, and Xiaodong Zheng. 2023. 'Hd-Deep-EM: Deep Expectation Maximization for Dynamic Hidden State Recovery Using Heterogeneous Data.' *IEEE Transactions on Power Systems.*

Machowski, Jan, Zbigniew Lubosny, Janusz W Bialek, and James R Bumby. 2020. *Power system dynamics: stability and control.* John Wiley & Sons.

Madani, V, M Parashar, J Giri, S Durbha, F Rahmatian, D Day, M Adamiak, and G Sheble. 2011. 'PMU placement considerations—A roadmap for optimal PMU placement.' In *2011 IEEE/PES Power Systems Conference and Exposition,* 1–7. IEEE.

Manousakis, Nikolaos M, and George N Korres. 2018. 'A hybrid power system state estimator using synchronized and unsynchronized sensors.' *International Transactions on Electrical Energy Systems* 28 (8): e2580.

McKinley, Sky, and Megan Levine. 1998. 'Cubic spline interpolation.' *College of the Redwoods* 45 (1): 1049–1060.

Mewett, David Trevor, Karen J Reynolds, and Homer Nazeran. 2004. 'Reducing power line interference in digitised electromyogram recordings by spectrum interpolation.' *Medical and Biological Engineering and Computing* 42:524–531.

Min, Zeping, and Cheng Tai. 2022. 'Why pseudo label based algorithm is effective?–from the perspective of pseudo labeled data.' *arXiv preprint arXiv:2211.10039.*

Misyris, George S, Andreas Venzke, and Spyros Chatzivasileiadis. 2020. 'Physics-informed neural networks for power systems.' In *IEEE Power & Energy Society General Meeting,* 1–5.

Mohammadpourfard, Mostafa, Yang Weng, Mykola Pechenizkiy, Mohsen Tajdinian, and Behnam Mohammadi-Ivatloo. 2020. 'Ensuring Cybersecurity of Smart Grid Against Data Integrity Attacks Under Concept Drift.' *International Journal of Electrical Power & Energy Systems.*

Netto, Marcos, and Lamine Mili. 2018. 'A robust data-driven Koopman Kalman filter for power systems dynamic state estimation.' *IEEE Transactions on Power Systems* 33 (6): 7228–7237.

Netto, Marcos, Junbo Zhao, and Lamine Mili. 2016. 'A robust extended Kalman filter for power system dynamic state estimation using PMU measurements.' In *2016 IEEE Power and Energy Society General Meeting (PESGM),* 1–5. IEEE.

Nordhausen, Klaus, and Hannu Oja. 2018. 'Independent component analysis: A statistical perspective.' *Wiley Interdisciplinary Reviews: Computational Statistics* 10 (5): e1440.

North American Electric Reliability Corporation. 2017. 'Real-time Application of Synchrophasors for Improving Reliability,' https://www.naspi.org/sites/default/files/reference_documents/rapir_final_20101017.pdf?fileID=519.

Pan, Sinno Jialin, Ivor W Tsang, James T Kwok, and Qiang Yang. 2010. 'Domain adaptation via Transfer Component Analysis.' *IEEE Transactions on Neural Networks.*

Pan, Sinno Jialin, and Qiang Yang. 2009. 'A Survey on Transfer Learning.' *IEEE Transactions on Knowledge and Data Engineering.*

Pankratov, Aleksey V, Natalia L Batseva, Ekaterina S Polyakova, Alexander S Tavlintsev, Ivan L Lapatin, and Ilya Y Lipnitskiy. 2019. 'Application of expectation maximization algorithm for measurement-based power system load modeling.' In *IEEE International Siberian Conference on Control and Communications.*

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library.' In *Advances in Neural Information Processing Systems 32,* 8024–8035. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. 'Scikit-learn: Machine Learning in Python.' *Journal of Machine Learning Research* 12:2825–2830.

Pérez, Enrique, and Julio Barros. 2008. 'A proposal for on-line detection and classification of voltage events in power systems.' *IEEE Transactions on Power Delivery.*

Plathottam, Siby Jose, Hossein Salehfar, and Prakash Ranganathan. 2017. 'Convolutional Neural Networks (CNNs) for power system big data analysis.' In *IEEE North American Power Symposium,* 1–6.

Qin, Jinlei, Zheng Li, and Youchan Zhu. 2021. 'Protection Mechanism in Reliability Evaluation Approach to Multistate System with Common Cause Failure.' *Complexity.*

Qu, Bogang, Zidong Wang, and Bo Shen. 2021. 'Fusion estimation for a class of multi-rate power systems with randomly occurring SCADA measurement delays.' *Automatica* 125:109408.

Radchenko, Danylo, and Maryna Viazovska. 2019. 'Fourier interpolation on the real line.' *Publications mathématiques de l'IHÉS* 129:51–81.

Radford, Alec, Luke Metz, and Soumith Chintala. 2015. 'Unsupervised representation learning with deep convolutional generative adversarial networks.' *arXiv preprint arXiv:1511.06434.*

Rafferty, Mark, Xueqin Liu, David M Laverty, and Sean McLoone. 2016. 'Real-time multiple event detection and classification using moving window PCA.' *IEEE Transactions on Smart Grid* 7 (5): 2537–2548.

Raissi, Maziar, Paris Perdikaris, and George E Karniadakis. 2019. 'Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.' *Journal of Computational physics* 378:686–707.

Rasmus, Antti, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. 2015. 'Semi-supervised learning with ladder networks.' *arXiv.*

Ren, Chao, and Yan Xu. 2019. 'Transfer learning-based power system online dynamic security assessment: using one model to assess many unlearned faults.' *IEEE Transactions on Power Systems.*

Ren, Mengye, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. 'Learning to reweight examples for robust deep learning.' In *International conference on machine learning,* 4334–4343. PMLR.

Rosenstein, Michael T, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. 2005. 'To Transfer or Not to Transfer.' In *Neural Information Processing Systems Workshop on Transfer Learning.*

Sánchez A, V David. 2003. 'Advanced support vector machines and kernel methods.' *Neurocomputing* 55 (1-2): 5–20.

Sayed, Khairy, and Hossam A Gabbar. 2017. 'SCADA and smart energy grid control automation.' In *Smart energy grid engineering,* 481–514. Elsevier.

Schneider, Kevin P, BA Mather, Bikash Chandra Pal, C-W Ten, Greg J Shirek, Hao Zhu, Jason C Fuller, Jose Luiz R Pereira, Luis F Ochoa, Leandro Ramos de Araujo, et al. 2017. 'Analytic considerations and design basis for the IEEE distribution test feeders.' *IEEE Transactions on power systems* 33 (3): 3181–3188.

Schölkopf, Bernhard, Ralf Herbrich, and Alex J Smola. 2001. 'A generalized representer theorem.' In *International conference on computational learning theory.* Springer.

Scholtz, Ernst. 2004. 'Observer-based monitors and distributed wave controllers for electromechanical disturbances in power systems.' PhD diss., Massachusetts Institute of Technology.

Shao, Mingjie, Wing-Kin Ma, Junbin Liu, and Zihao Huang. 2022. 'Accelerated and Deep Expectation Maximization for One-Bit MIMO-OFDM Detection.' *arXiv preprint arXiv:2210.03888.*

Shaw, Priyabrata, and Manas Kumar Jena. 2020. 'A novel event detection and classification scheme using wide-area frequency measurements.' *IEEE Transactions on Smart Grid* 12 (3): 2320–2330.

Shen, Wei, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. 2015. 'Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection.' In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 3982–3991.

Shi, Jie, Brandon Foggo, and Nanpeng Yu. 2021. 'Power system event identification based on deep neural network with information loading.' *IEEE Transactions on Power Systems.*

Shih, Kuang-Rong, and Shyh-Jier Huang. 2002. 'Application of a robust algorithm for dynamic state estimation of a power system.' *IEEE Transactions on Power Systems* 17 (1): 141–147. https://doi.org/10.1109/59.982205.

Shin, Yeonjong, Jerome Darbon, and George Em Karniadakis. 2020. 'On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs.' *arXiv preprint arXiv:2004.01806.*

Sidiropoulos, Nicholas D, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. 2017. 'Tensor decomposition for signal processing and machine learning.' *IEEE Transactions on Signal Processing.*

Simões Costa, Antonio, André Albuquerque, and Daniel Bez. 2013. 'An estimation fusion method for including phasor measurements into power system real-time modeling.' *IEEE Transactions on Power Systems* 28 (2): 1910–1920. https://doi.org/10.1109/TPWRS.2012.2232315.

Sohn, Kihyuk, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. 'Fixmatch: Simplifying semi-supervised learning with consistency and confidence.' *Advances in Neural Information Processing Systems.*

Song, Qinbao, and Martin Shepperd. 2007. 'Missing data imputation techniques.' *International journal of business intelligence and data mining* 2 (3): 261–291.

Stanković, Alex M, Aleksandar A Sarić, Andrija T Sarić, and Mark K Transtrum. 2020. 'Data-driven symbolic regression for identification of nonlinear dynamics in power systems.' In *IEEE Power & Energy Society General Meeting),* 1–5.

Stegner, Christoph, Josef Bogenrieder, Philipp Luchscheider, and Christoph J Brabec. 2016. 'First Year of Smart Metering with A High Time Resolution—realistic Self-sufficiency Rates for Households with Solar Batteries.' *Energy Procedia.*

Surana, Amit. 2016. 'Koopman operator based observer synthesis for control-affine nonlinear systems.' In *2016 IEEE 55th Conference on Decision and Control (CDC),* 6492–6499. IEEE.

———. 2020. 'Koopman operator framework for time series modeling and analysis.' *Journal of Nonlinear Science* 30:1973–2006.

Tang, Yichuan. 2015. 'Deep learning using linear support vector machines.' *arXiv preprint arXiv:1306.0239.*

Tang, Yuchun, Yan-Qing Zhang, Nitesh V. Chawla, and Sven Krasser. 2009. 'SVMs Modeling for Highly Imbalanced Classification.' *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39 (1): 281–288. https://doi.org/10.1109/TSMCB.2008.2002909.

Thomas, Mini S, and John Douglas McDonald. 2017. *Power system SCADA and smart grids.* CRC press.

Tsai, Yao-Hung Hubert, Yi-Ren Yeh, and Yu-Chiang Frank Wang. 2016. 'Learning Cross-domain Landmarks for Heterogeneous Domain Adaptation.' In *IEEE Conference on Computer Vision and Pattern Recognition.*

Tucker, Warwick. 2002. 'A rigorous ODE solver and Smale's 14th problem.' *Foundations of Computational Mathematics* 2:53–117.

Usman, Muhammad Usama, and M Omar Faruque. 2019. 'Applications of Synchrophasor Technologies in Power Systems.' *Journal of Modern Power Systems and Clean Energy.*

Valverde, Gustavo, and Vladimir Terzija. 2011. 'Unscented Kalman filter for power system dynamic state estimation.' *IET generation, transmission & distribution* 5 (1): 29–37.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. 'Attention is all you need.' *Advances in neural information processing systems* 30.

Vermaak, J, and EC Botha. 1998. 'Recurrent neural networks for short-term load forecasting.' *IEEE Transactions on Power Systems* 13 (1): 126–132.

Wahba, Grace. 1981. 'Spline interpolation and smoothing on the sphere.' *SIAM Journal on Scientific and Statistical Computing* 2 (1): 5–16.

Wang, Bin, Yang Liu, and Kai Sun. 2015. 'Power system differential-algebraic equations.' *arXiv preprint arXiv:1512.05185.*

Wang, Keyou, and Mariesa L Crow. 2011. 'Numerical simulation of stochastic differential algebraic equations for power system transient stability with random loads.' In *IEEE Power and Energy Society General Meeting,* 1–8.

Wang, Weikang, He Yin, Chang Chen, Abigail Till, Wenxuan Yao, Xianda Deng, and Yilu Liu. 2020. 'Frequency disturbance event detection based on synchrophasors and deep learning.' *IEEE Transactions on Smart Grid* 11 (4): 3593–3605.

Wang, Wentao, Han Xu, Xiaorui Liu, Yaxin Li, Bhavani Thuraisingham, and Jiliang Tang. 2021. 'Imbalanced adversarial training with reweighting.' *arXiv preprint arXiv:2107.13639.*

Weng, Yang, Rohit Negi, and Marija D. Ilić. 2019. 'Probabilistic Joint State Estimation for Operational Planning.' *IEEE Transactions on Smart Grid.*

Wu, CF Jeff. 1983. 'On the convergence properties of the EM algorithm.' *The Annals of statistics,* 95–103.

Xiang, Yuwei, Tong Wang, and Zengping Wang. 2019. 'Improved Gaussian mixture model based probabilistic power flow of wind integrated power system.' In *IEEE Power & Energy Society General Meeting.*

Yadav, Ravi, Ashok Kumar Pradhan, and Innocent Kamwa. 2018. 'Real-time multiple event detection and classification in power system using signal energy transformations.' *IEEE Transactions on Industrial Informatics* 15 (3): 1521–1531.

Yan, Yuguang, Wen Li, Hanrui Wu, Huaqing Min, Mingkui Tan, and Qingyao Wu. 2018. 'Semi-Supervised Optimal Transport for Heterogeneous Domain Adaptation.' In *International Joint Conference on Artificial Intelligence.*

Young, Marcus, and Alison Silverstein. 2014. 'Factors affecting pmu installation costs.' *US Department of Energy-Office of Electricity Delivery and Energy Reliability.*

Yu, Jiafan, Yang Weng, and Ram Rajagopal. 2017. 'PaToPa: A data-driven parameter and topology joint estimation framework in distribution grids.' *IEEE Transactions on Power Systems.*

Yuan, Ye, Steven Low, Omid Ardakanian, and Claire Tomlin. 2016. 'Inverse Power Flow Problem.' *ArXiv.*

Yuan, Yuxuan, Yifei Guo, Kaveh Dehghanpour, Zhaoyu Wang, and Yanchao Wang. 2021a. 'Learning-Based real-time event identification using rich real PMU data.' *IEEE Transactions on Power Systems.*

———. 2021b. 'Learning-Based Real-Time Event Identification Using Rich Real PMU Data.' *IEEE Transactions on Power Systems* 36 (6): 5044–5055.

Yuan, Yuxuan, Zhaoyu Wang, and Yanchao Wang. 2020. 'Learning Latent Interactions for Event Identification via Graph Neural Networks and PMU Data.' *arXiv preprint arXiv:2010.01616.*

Zadrozny, Bianca, John Langford, and Naoki Abe. 2003. 'Cost-sensitive learning by cost-proportionate example weighting.' In *Third IEEE international conference on data mining,* 435–442. IEEE.

Zhang, S., Y. Wang, M. Liu, and Z. Bao. 2018. 'Data-Based line trip fault prediction in power systems using LSTM networks and SVM.' *IEEE Access,* https://doi.org/10.1109/ACCESS.2017.2785763.

Zhang, Yiqing, Jianyong Sun, Jiang Xue, Geoffrey Ye Li, and Zongben Xu. 2022. 'Deep Expectation-Maximization for Joint MIMO Channel Estimation and Signal Detection.' *IEEE Transactions on Signal Processing* 70:4483–4497. https://doi.org/10.1109/TSP.2022.3205478.

Zhao, Junbo, Marcos Netto, and Lamine Mili. 2016. 'A robust iterated extended Kalman filter for power system dynamic state estimation.' *IEEE Transactions on Power Systems* 32 (4): 3205–3216.

Zhao, Tuo, Mo Yu, Yiming Wang, Raman Arora, and Han Liu. 2014. 'Accelerated mini-batch randomized block coordinate descent method.' *Advances in Neural Information Processing Systems.*

Zhou, Joey Tianyi, Ivor W Tsang, Sinno Jialin Pan, and Mingkui Tan. 2014. 'Heterogeneous Domain Adaptation for Multiple Classes.' In *Artificial Intelligence and Statistics.*

Zhu, Chenfeng, and Andreas Reinhardt. 2019. 'Reliable Streaming and Synchronization of Smart Meter Data over Intermittent Data Connections.' In *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids.*

Zhu, Qiaomu, Jinfu Chen, Lin Zhu, Dongyuan Shi, Xiang Bai, Xianzhong Duan, and Yilu Liu. 2018. 'A deep end-to-end model for transient stability assessment with PMU data.' *IEEE Access* 6:65474–65487.

Zhu, Yin, Yuqiang Chen, Zhongqi Lu, Sinno Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. 2011. 'Heterogeneous Transfer Learning For Image Classification.' In *AAAI Conference on Artificial Intelligence.*

APPENDIX A

APPENDIX

Chapter 3 is based on the paper "Hd-Deep−EM: Deep Expectation Maximization for Dynamic Hidden State Recovery Using Heterogeneous Data", Z. Ma, H. Li, Y. Weng, E. Blasch, and X. Zheng, IEEE Transactions on Power Systems, pp. 1-10, Jun 2023.

Chapter 4 is based on the paper "A Transfer Learning Framework for Power System Event Identification", H. Li, Z. Ma, and Y. Weng IEEE Transactions on Power Systems, vol. 37, no. 6, pp. 4424-4435, Nov 2022.

Chapter 5 is based on the paper "Structural Tensor Learning for Event Identification with Limited Labels", H. Li, Z. Ma, Y. Weng, E. Blasch, and S. Santoso, IEEE Transactions on Power Systems, vol. 38, no. 6, pp. 5314-5328, Nov 2023.

I affirm that all co-authors of the papers that have been incorporated into this thesis have granted their explicit permission for the use of these works. This permission encompasses the inclusion of the papers in their entirety or in part within the context of this thesis, which is being submitted to Arizona State University as part of the requirements for the Doctor of Philosophy degree.

Each co-author has been informed about the content of the thesis and its purpose and has consented to the use of the published material for academic purposes, specifically as part of this thesis.