University of Nevada, Reno

Towards Automated Machine Learning on Imperfect Data for Situational Awareness in Power System

A dissertation submitted in partial fulfillment
of the requirements for the degree of Doctor of
Philosophy in Computer Science and Engineering

by

Yunchuan Liu

Dr. Lei Yang, Dissertation Advisor

August 2022

© by Yunchuan Liu 2022
All Rights Reserved

The Graduate School

We recommend that the dissertation prepared under our supervision by

Yunchuan Liu

entitled

Towards Automated Machine Learning on Imperfect Data for Situational Awareness in Power System

be accepted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Lei Yang, Ph.D., Advisor

Feng Yan, Ph.D., Committee Member

Dongfang Zhao, Ph.D., Committee Member

Hao Xu, Ph.D., Committee Member

Hanif Livani, Ph.D., Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean, Graduate School

August 2022

Abstract

The increasing penetration of renewable energy sources (such as solar and wind) and incoming widespread electric vehicles charging introduce new challenges in the power system. Due to the variability and uncertainty of these sources, reliable and cost-effective operations of the power system rely on high level of situational awareness. Thanks to the wide deployment of sensors (e.g., phasor measurement units (PMUs) and smart meters) and the emerging smart Internet of Things (IoT) sensing devices in the electric grid, large amounts of data are being collected, which provide golden opportunities to achieve high level of situational awareness for reliable and cost-effective grid operations.

To better utilize the data, this dissertation aims to develop Machine Learning (ML) methods and provide fundamental understanding and systematic exploitation of ML for situational awareness using large amounts of imperfect data collected in power systems, in order to improve the reliability and resilience of power systems.

However, building excellent ML models needs clean, accurate and sufficient training data. The data collected from the real-world power system is of low quality. For example, the data collected from wind farms contains a mixture of ramp and non-ramp as well as the mingle of heterogeneous dynamics data; the data in the transmission grid contains noisy, missing, insufficient and inaccurate timestamp data. Employing ML without considering these distinct features in real-world applications cannot build good ML models. This dissertation aims to address these challenges in two applications, wind generation forecast and power system event classification, by developing ML models in an automated way with less efforts from domain experts, as the cost of processing such large amounts of imperfect data by experts can be prohibitive in practice.

First, we take heterogeneous dynamics into consideration, especially for ramp events. A Drifting Streaming Peaks-over-Threshold (DSPOT) enhanced self-evolving neural networks-based short-term wind farm generation forecast is proposed by utilizing dynamic ramp thresholds to separate the ramp and non-ramp events, based on which different neural networks are trained to learn different dynamics of wind farm generation. As the efficacy of the neural networks relies on the quality of training datasets (i.e., the classification accuracy of ramp and non-ramp events), a Bayesian optimization based approach is developed to optimize the parameters of DSPOT to enhance the quality of the training datasets and the corresponding performance of the neural networks. Experimental results show that compared with other forecast approaches, the proposed forecast approach can substantially improve the forecast accuracy, especially for ramp events.

Next, we address the challenges of event classification due to the low-quality PMU measurements and event logs. A novel machine learning framework is proposed for robust event classification, which consists of three main steps: data preprocessing, fine-grained event data extraction, and feature engineering. Specifically, the data preprocessing step addresses the data quality issues of PMU measurements (e.g., bad data and missing data); in the fine-grained event data extraction step, a model-free event detection method is developed to accurately localize the events from the inaccurate event timestamps in the event logs; and the feature engineering step constructs the event features based on the patterns of different event types, in order to improve the performance and the interpretability of the event classifiers. Moreover, with the small number of good features, we need much less training data to train a good event classifier, which can address the challenge of insufficient and imbalanced training data, and the training time is negligible compared to neural network based approaches. Based on the proposed framework, we developed a workflow for event classification using the real-world PMU data streaming into the system in real time. Using the proposed framework, robust event classifiers can be efficiently trained based on many off-the-shelf lightweight machine learning models. Numerical experiments using the real-world dataset from the Western Interconnection of the U.S power transmission grid show that the event classifiers trained under the proposed framework can achieve high classification accuracy while being robust against low-quality data.

Subsequently, we address the challenge of insufficient training labels. The real-world PMU data is often incomplete and noisy, which can significantly reduce the efficacy of existing machine learning techniques that require high-quality labeled training data. To obtain high-quality event logs for large amounts of PMU measurements, it requires significant efforts from domain experts to maintain the event logs and even hand-label the events, which can be prohibitively costly or impractical in practice. So we develop a weakly supervised machine learning approach that can learn a good event classifier using a few labeled PMU data. The key idea is to learn the labels from unlabeled data using a probabilistic generative model, in order to improve the training of the event classifiers. Experimental results show that even with 95% of unlabeled data, the average accuracy of the proposed method can still achieve 78.4%. This provides a promising way for domain experts to maintain the event logs in a less expensive and automated manner.

Finally, we conclude the dissertation and discuss future directions.

ACKNOWLEDGEMENTS

Words cannot express my gratitude to my advisor Dr. Lei Yang for his invaluable patience, excellent guidance, caring, patience, and engagement throughout the course of my study at University of Nevada, Reno. Thank you for all your intellectual and emotional support throughout this dissertation work especially during hard times. He assisted me financially and encouraged me to pursue my dream to be a professor. I start from knowing nothing in the field of Power System, and I have a chance to present one of the best papers in the top venue conference in Power System recently. I am very fortunate to have him as my advisor.

I would like to express my sincere gratitude to my committee members, Dr. Feng Yan, Dr. Dongfang Zhao, Dr. Hao Xu and Dr. Hanif Livani for their time, generously provided knowledge and valuable intellectual inputs in improving the dissertation work.

I am also grateful to my collaborators: Dr. Amir Ghasemkhani and Dr. Iman Niazazari for the initial work on the system that led to the dissertation. I would also like to thank all the members of the Big Data Lab with whom I had a chance to work in close collaboration who impacted and inspired me in both work and life.

I would like to thank my parents for their endless love and support, without whom I cannot have come this far.

Finally, I am thankful to the National Science Foundation (NSF) and Department of Energy (DOE) for the support provided during my Doctorates studies. The material presented in this thesis is based upon work supported by the NSF under grant numbers NSF-1950485, NSF-1801727 and DOE National Energy Technology Laboratory under Award Number DE-OE0000911. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Department of Energy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

The increasing penetration of renewable energy sources (expected 62% of generated energy will be covered by renewable energy [67] in 2050) and incoming widespread electric vehicles charging introduce new challenges in the power system. Due to the variability and uncertainty of these sources, reliable and cost-effective operations of the power system rely on a high level of situational awareness. As wide deployment of sensors (phasor measurement units (PMUs) smart meters) and the emerging smart Internet of Things (IoT) sensing devices have already been deployed in the electric grid, large amounts of data are being collected, which provide golden opportunities to achieve high level of situational awareness for reliable and cost-effective grid operations.

To better explore those real-world data to improve the reliability and resilience of power system, we develop Machine Learning (ML) methods and provide fundamental understanding and systematic exploitation of ML for situational awareness using large amounts of imperfect data collected in power system. However, building excellent ML models needs clean, accurate and sufficient training data. The data collected from the real-world power system is of low quality and imperfect. For example, the data collected from wind farms contains a mixture of ramp and non-ramp as well as the mingle of heterogeneous dynamics data; the data in the transmission grid contains noisy, missing, insufficient and inaccurate timestamp data.

This dissertation aims to address these challenges in two applications, wind farm generation forecast and power system event classification in an automated and less effortful way. However, employing existing machine learning approaches without considering these distinct features in real-world applications cannot build good ML models. This dissertation aims to address these challenges by considering its unique features in real-world applications.

First, to account for the distinct features of wind farm generation, especially ramps, we proposed Drifting Streaming Peaks-over-Threshold (DSPOT) enhanced self-evolving neural networks based short-term wind farm generation forecasts. Using DSPOT, the proposed method first classifies the wind farm generation data into ramp and non-ramp datasets, where time-varying dynamics is considered by utilizing dynamic ramp thresholds to separate the ramp and non-ramp events, based on which different neural networks are trained to learn different dynamics of wind farm generation. As the efficacy of the neural networks relies on the quality of training datasets (i.e., the classification accuracy of ramp and non-ramp events), a Bayesian optimization based approach is developed to optimize the parameters of DSPOT to enhance the quality of the training datasets and the corresponding performance of the neural networks. Experimental results show that compared with other forecast approaches, the proposed forecast approach can substantially improve the forecast accuracy, especially for ramp events.

Next, we address the challenges of event classification due to the low-quality PMU measurements and event logs. By analyzing the real-world PMU data, we find it is challenging to directly use this dataset for event classifiers due to the low data quality observed in PMU measurements and event logs. To address these challenges, we developed a novel machine learning framework for training robust event classifiers, which consists of three main steps: data preprocessing, fine-grained event data extraction, and feature engineering. Specifically, the data preprocessing step addresses the data quality issues of PMU measurements (e.g., bad data and missing data); in the fine-grained event data extraction step, a model-free event detection method is developed to accurately localize the events from the inaccurate event timestamps in the event logs; and the feature engineering step constructs the event features based on the patterns of different event types, in order to improve the performance and the interpretability of the event classifiers. Based on the proposed framework, we developed a workflow for event classification using the real-world PMU data streaming into the system in real time. Using the proposed framework, robust event classifiers can be efficiently trained based on many off-the-shelf lightweight machine learning models. Numerical experiments using the real-world dataset from the Western Interconnection of the U.S power transmission grid show that the event classifiers trained under the proposed framework can achieve high classification accuracy while being robust against low-quality data.

Subsequently, we address the challenge of insufficient training labels. The real-world PMU data is often incomplete and noisy, which can significantly reduce the efficacy of existing machine learning techniques that require high-quality labeled training data. To obtain high-quality event logs for large amounts of PMU measurements, it requires significant efforts from domain experts to maintain the event logs and even hand-label the events, which can be prohibitively costly or impractical in practice. To address this challenge, we developed a weakly supervised machine learning approach that can learn a good event classifier using a few labeled PMU data. The key idea is to learn the labels from unlabeled data using a probabilistic generative model, in order to improve the training of the event classifier. Experimental results show that even with 95% of unlabeled data, the average accuracy of the proposed method can still achieve 78.4%. This provides a promising way for domain experts to maintain the event logs in a less expensive and automated manner.

1.2 Summary of Contributions

For this section, we briefly discuss the major contributions: (1) adaptive machine learning for wind farm generation forecasting, which addresses the challenges of the non-stationarity and the ramp dynamics of wind farm generation and can greatly facilitate the integration of wind generation in reality; (2) robust event classification using imperfect real-world PMU data, which addresses the challenges of using low quality real-world PMU data to develop event classification for situational awareness in smart grids; and (3) weakly supervised event classification using limited labeled PMU data, which can significantly reduce the efforts from domain experts to maintain the event logs for developing event classifiers.

1.2.1 Drifting Streaming Peaks-Over-Threshold Enhanced Self-Evolving Neural Networks based Short-Term Wind Farm Generation Forecast

Real-world wind farm generation measurements often exhibit distinct features, such as the non-stationarity and the heterogeneous dynamics of ramp and non-ramp events across different classes of wind turbines. Employing existing machine learning approaches without considering these features cannot accurately forecast wind farm generation, especially for ramp events. To account for the distinct features of wind farm generation, we proposed Drifting Streaming Peaks-over-Threshold (DSPOT) enhanced self-evolving neural networks based short-term wind farm generation forecast. Using DSPOT, the proposed method first classifies the wind farm generation data into ramp and non-ramp datasets, where time-varying dynamics is considered by utilizing dynamic ramp thresholds to separate the ramp and non-ramp events, based on which different neural networks are trained to learn different dynamics of wind farm generation. As the efficacy of the neural networks relies on the quality of training datasets (i.e., the classification accuracy of ramp and non-ramp events), a Bayesian optimization based approach is developed to optimize the parameters of DSPOT to enhance the quality of the training datasets and the corresponding performance of the neural networks. Experimental results show that compared with other forecast approaches, the proposed forecast approach can substantially improve the forecast accuracy, especially for ramp events.

1.2.2 Robust Event Classification Using Imperfect Real-world PMU Data

A novel machine learning framework for training event classifiers using large-scale imperfect real-world PMU data is proposed, which consists of three main steps: data preprocessing, fine-grained event data extraction, and feature engineering. The goal is to obtain high-quality labeled PMU training data from the low-quality real PMU data. Specifically, the data preprocessing step addresses the data quality issues of PMU measurements (e.g., bad data and missing data); the fine-grained event data extraction step accurately localizes the events from the inaccurate event timestamps in the event logs by using a model-free event detection method developed based on the low-rank property of PMU data; and the feature engineering step constructs the event features based on the patterns of different event types, in order to improve the performance and the interpretability of the event classifiers. Based on the proposed machine learning framework, we also develop a workflow for event classification using the real PMU data streaming into the system in real time.

One salient merit of the proposed framework is that large-scale real PMU data is reduced to a small set of event features, which can be used to efficiently train many off-the-shelf lightweight machine learning models. As the features are constructed based on the event patterns, the contribution of any single PMU measurement to the features is low, which can effectively mitigate the impact of bad and missing data and improve the robustness of the event classifiers.

Using the two-year real-world PMU data from the Western Interconnection of continental U.S. transmission grid, we evaluate the performance of the proposed machine learning framework. Many off-the-shelf lightweight machine learning models can be efficiently trained in our framework and achieve good performance. For example, the training time of the Random Forest model is less than 2 seconds while the testing accuracy of the Random Forest model is 94%. Moreover, our framework demonstrates a strong robustness against missing data. Even under the missing rate of 50%, the accuracy of the Random Forest model can still achieve 87%. In summary, the proposed machine learning framework provides a promising way to train robust event classifiers with good interpretability and low training cost.

1.2.3 Automatic Labeling Using Imperfect Real-World PMU data with Scarce Labels

We develop a weakly supervised learning based event classification approach that can train good event classifiers using imperfect real-world PMU data with scarce labels. The key idea is to estimate the labels of large amounts of unlabeled PMU data for training event classifiers. Specifically, we first learn a series of labeling functions based on the knowledge of different types of events, which can generate initial estimates of the labels. As these labeling functions are learnt using the same dataset with scarce labels, the estimated labels are often correlated and the estimates can be noisy and biased. Directly using such estimates cannot generate good event classifiers. To enhance the accuracy of the estimated labels, a generative model is developed to characterize the correlations among the estimated labels, based on which the estimated labels from labeling functions are combined in a probabilistic manner to generate the "true" labels. Then, the refined labels from the generative model are used to train event classifiers. It is worth noting that using the proposed weakly supervised learning approach, less efforts are required from domain experts to maintain the event logs for building event classifiers; by examining the classification results, domain experts can further enhance the classification models. To the best of our knowledge, this paper is the first to study weakly supervised event classification in power system. The findings in the paper can shed the light on using imperfect real-world PMU data with scarce labels for event classification.

Using the two-year real-world PMU data from the Western Interconnection of the continental U.S. transmission grid, we evaluate the performance of the proposed weakly supervised event classification approach. The experimental results show that when only 5% of data are labeled, the average accuracy of the estimated labels using our approach can be around 70.9% and the average accuracy of the corresponding event classifier can achieve about 78.4%. This shows a promising way of using weakly supervised learning for developing robust event classification with extremely insufficient labeled data.

1.3 Organization

The remaining chapters are organized as follows. Chapter 2 studies how to address heterogeneous dynamics challenges(i.e. DSPOT). Chapter 3 elaborated how to develop a robust classification ML model under low quality data. Chapter 4 studies how to do auto labeling under extremely insufficient training data, and we propose a weakly supervised learning framework. Finally, Chapter 5 makes a conclusion and discusses the future work.

CHAPTER 2
DRIFTING STREAMING PEAKS-OVER-THRESHOLD ENHANCED SELF-EVOLVING NEURAL NETWORKS BASED SHORT-TERM WIND FARM GENERATION FORECAST

2.1 Introduction

Wind energy constitutes a significant portion of this renewable integration [11]. Due to the high variability of wind energy and high penetration of wind generation make the power system more vulnerable, especially during wind power ramps. This requires accurate forecasts of wind generation to increase the situational awareness. But the imperfect data of wind farm contains the mixture of ramp and non-ramps, different seasonal dynamics, directly applying ML can not achieve a good result (e.g., extreme (or ramp) events are overlooked might lead to poor performance [16]). We aim to develop approach to split the tangled wind farm generation data for better ML models in an automatic way.

Our previous works [22, 61] have shown 1) the non-stationarity and the seasonality of wind farm generation and 2) different dynamics of non-ramp, ramp-up, and ramp-down events of wind farm generation. Therefore, simply applying neural networks without considering ramp events may not obtain the best prediction performance, not to mention the tuning of network topology and hyperparameters, which is often a demanding task. Moreover, as wind farms often consist of different classes of wind turbines, the dynamics of wind generation of different classes of wind turbines can be different, which is observed in our dataset. Motivated by these observations, we aim to develop neural networks based wind generation forecast approaches that consider these unique features of wind farm generation.

In this chapter, we propose seasonal self-evolving neural networks based short-term wind farm generation forecast that accounts for different power output dynamics of different wind turbines under non-ramp, ramp-up, and ramp-down events, in order to achieve a more accurate wind farm generation forecast. The basic idea is as follows: 1) first classify the historical data into non-ramp, ramp-up, and ramp-down datasets, where the non-ramp dataset is further split into 4 datasets for 4 seasons to account for the seasonality; and 2) then develop different neural networks for different power output dynamics of different turbines under non-ramp, ramp-up, and ramp-down events, with the intention of accounting for the unique features of wind farm generation. To account for the non-stationarity as well as reduce the burden of hyperparameter tuning, we leverage NeuroEvolution of Augmenting Topologies (NEAT) [53] to train neural networks, which evolves the neural networks using a genetic algorithm to find the best weighting parameters and network topology. Based on the proposed seasonal self-evolving neural networks, we study both point forecasts and distributional forecasts. Based on wind measurement data from a real-world wind farm, the proposed forecast approach demonstrates significantly improved forecast accuracy, compared with other approaches. Compared to the existing works, this work provides a unified framework that considers 1) the non-stationarity and the seasonality of wind farm power outputs and 2) different dynamics of wind non-ramp, ramp-up, and ramp-down events across different classes of turbines. By using self-evolving neural networks, the proposed approach does not require AI experts to tune the topology of neural networks and the hyperparameters. Thus, the proposed approach can be easily implemented in practice.

2.2 Related Works

There have been many studies on wind generation forecast based on physical models (e.g., numerical weather prediction model [7]) and statistical models (e.g., autoregressive model [43], Markov chains [42]). Recently, advanced artificial intelligence (AI) techniques have been successfully applied to many applications. There have been some studies applying AI techniques for wind generation forecast (e.g., support vector machine (SVM) [31], Artificial Neural Networks (ANN) [17], Wavelet Neural Network (WNN) [10], Adaptive Neuro-Fuzzy Neural Network (ANFIS) [41]). In [51], a long short-term memory (LSTM) prediction model is proposed for short-term wind power forecast. In [21], a two-stage forecasting model based on the error factor and the ensemble method is proposed for multi-step wind power forecast. To deal with the non-stationarity of wind generation, wavelet components decomposition [31], empirical mode decomposition (EMD) based model [58], and variational mode decomposition (VMD) based model [3] are introduced; however, for these works, it is challenging to find appropriate numbers of components or modes.

Although neural network (NN) based methods may enhance the forecast accuracy to a certain degree, existing NN based approaches may have poor performance during ramp events, simply because the ramp and non-ramp events are not separated when training NNs. It has been shown that NNs may perform poorly if extreme (or ramp) events are overlooked [16]. Our previous studies [22, 61] have revealed that 1) the non-stationary and seasonal dynamics of wind farm generation and 2) heterogeneous dynamics of non-ramp and ramp events. Moreover, as different classes of wind turbines are deployed in wind farms, we observe that the dynamics of wind generation of different classes of wind turbines can be different. Thus, employing NNs without considering these distinct features of wind farm generation cannot accurately forecast wind farm generation, especially for ramp events. In our previous work, seasonal self-evolving neural networks [32] are built for different seasons and ramps are defined using fixed thresholds. However, it is observed that the dynamics of wind ramps may change within each season, and due to this time-varying dynamics of wind ramps, it is challenging to use fixed thresholds to accurately capture the dynamics of wind ramps. To address this challenge, we propose a dynamic threshold based approach that can adapt to the time-varying dynamics of wind ramps.

2.3 Data Description and Key Observations

We use real wind generation data from a large wind farm. The wind farm has a rated capacity of 300.5 MW, where two classes of wind turbines are installed: Mitsubishi and GE turbines. There are 221 Mitsubishi turbines with a rated capacity of 1MW and 53 GE turbines with a rated capacity of 1.5MW. Each class of wind turbines has distinct power curves as well as cut-in and cut-off speed. For each class, a meteorological tower (MET), collocated with a wind turbine, is deployed to collect weather information. The instantaneous power outputs of each turbine together with the weather information are saved every 10 min for the years of 2009 and 2010. In this chapter, we use the power outputs of Mitsubishi turbines $P_{mit}(t)$ and GE turbines $P_{ge}(t)$, the wind speed $W_s(t)$, and the wind direction $W_{dir}(t)$ to develop the proposed NNs.

From the measurements of power outputs, we find 1) the non-stationarity of power measurements and 2) heterogeneous dynamics of wind non-ramp and ramp events across each class of turbines as illustrated in Fig. 2.1, where the Cumulative Distribution Functions (CDFs) of wind power measurements of two classes of turbines over different seasons of a year and different ramp events are presented. In addition, it is shown in Fig. 2.2 that the

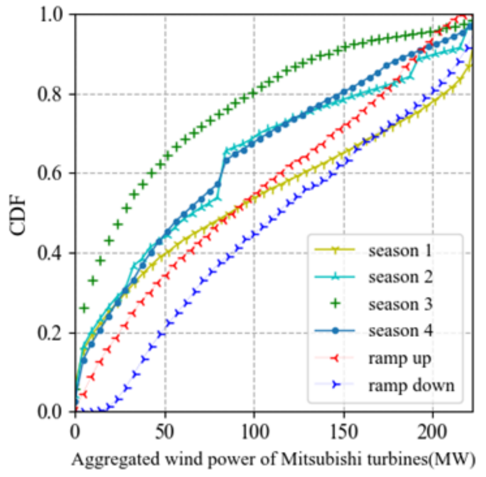
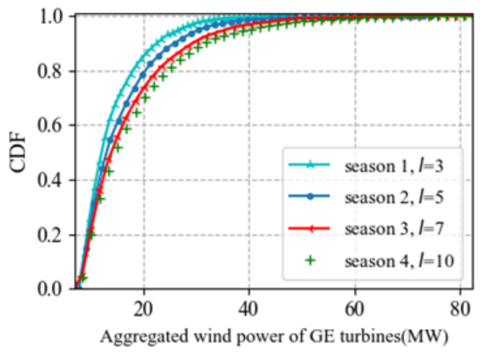
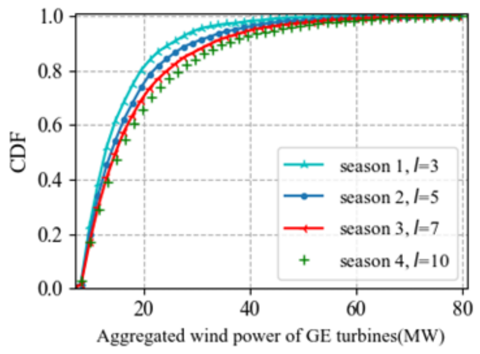


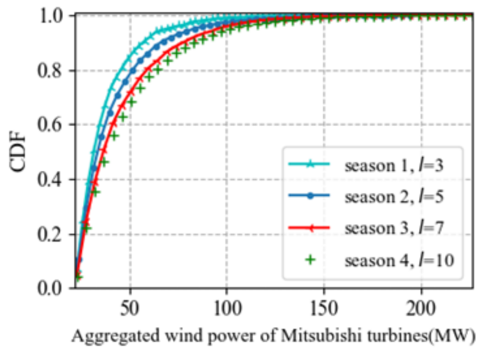
Figure 2.1: Empirical distribution of power outputs of GE and Mitsubishi turbines in 4 seasons and ramp events, where season 4 is from October to December.



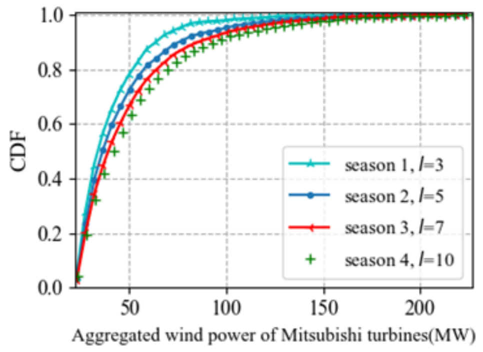
(a) GE ramp up distribution



(b) GE ramp down distribution



(c) Mitsubishi ramp up distribution



(d) Mitsubishi ramp down distribution

Figure 2.2: Empirical ramp distributions of GE and Mitsubishi turbines in different time windows l and different time periods, which follow the generalized Pareto distribution.

distributions of ramps in different time windows l and different time periods are different and follow the generalized Pareto distribution (GPD).

2.4 DSPOT Enhanced Self-Evolving Neural Networks

As observed from Fig. 2.2, fixed thresholds cannot fully capture the dynamics of wind ramp events. To address this challenge, we redefine the ramps by using dynamic thresholds, which change over time based on the dynamics of the ramp events, in order to reduce the forecast error of wind farm generation, especially for ramp events.

Motivated by the observations, we seek to design a short-term forecast of wind farm generation method that accounts for not only heterogeneous dynamics of each class of wind turbines but also the time-varying dynamics of ramp and non-ramp events. Inspired by the success of artificial intelligence (AI) in a wide range of fields, our goal is to use neural networks (NNs) to learn these different dynamics of power outputs. Although there are several attempts along this line (e.g., ANNs [8] and LSTM [51]), these approaches use a single model and overlook the extreme ramp events, which leads to poor forecast performance, especially for ramp events. Additionally, to train good NNs, it is critical to have high quality training datasets (i.e., the ramp and non-ramp datasets should be well separated), which is a challenging task due to the time-varying dynamics of ramp and non-ramp events. Further, when training NNs, it is challenging to find the optimal topology as well as hyperparameters of NNs.

To tackle these challenges, we propose the DSPOT enhanced self-evolving neural networks, namely DSN, for short-term wind farm generation forecast. The idea is to 1) first classify non-ramp and ramp events using DSPOT, which uses dynamic ramp thresholds to account for the time-varying dynamics of non-ramp and ramp events and 2) then train different NNs for each dataset to learn heterogeneous generation dynamics of different classes of wind turbines, where these NNs can self-evolve based on the data, in order to account for

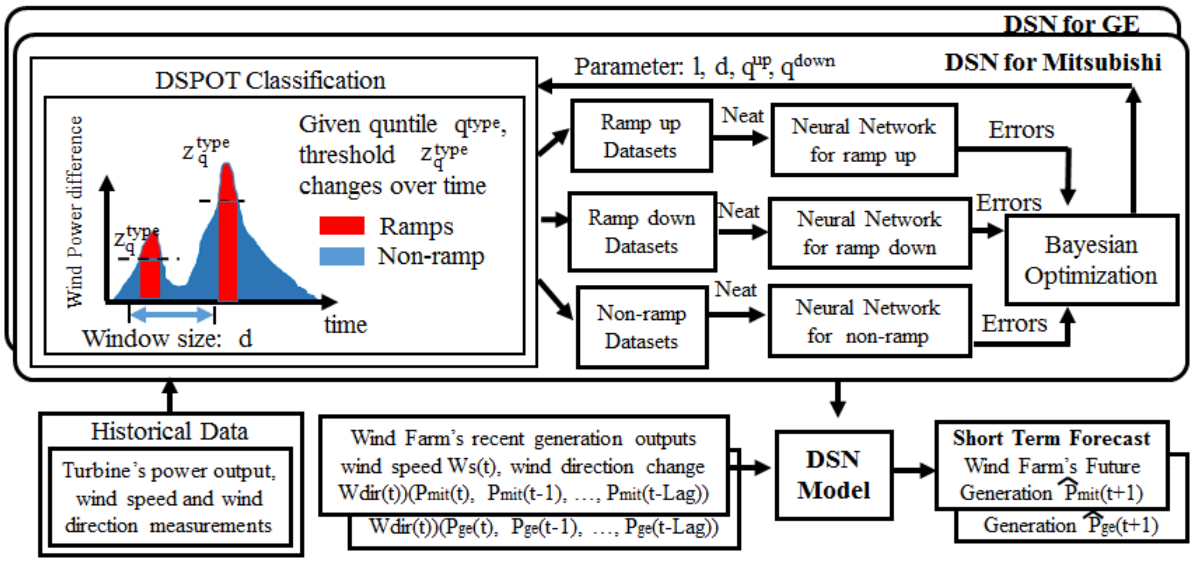

DSN for GE
DSN for Mitsubishi
DSPOT Classification
Wind Power difference
Given quntile q^type, threshold z_q^type changes over time
Ramps
Non-ramp
Window size: d
time
Parameter: l, d, q^up, q^down
Ramp up Datasets
Ramp down Datasets
Non-ramp Datasets
Neat
Neural Network for ramp up
Neural Network for ramp down
Neural Network for non-ramp
Errors
Bayesian Optimization
Historical Data
Turbine's power output, wind speed and wind direction measurements
Wind Farm's recent generation outputs wind speed Ws(t), wind direction change Wdir(t))(Pmit(t), Pmit(t-1), ..., Pmit(t-Lag))
Wdir(t))(Pge(t), Pge(t-1), ..., Pge(t-Lag))
DSN Model
Short Term Forecast Wind Farm's Future Generation Pmit(t+1)
Generation Pge(t+1)

Figure 2.3: Empirical ramp distributions of GE and Mitsubishi turbines in different time windows *l* and different time periods, which follow the generalized Pareto distribution.

the non-stationarity and reduce the overhead of tuning the topology and hyperparameters of NNs.

The design of our model is illustrated in Fig. 2.3. The historical data are first classified into non-ramp, ramp up and ramp down datasets by DSPOT, in which dynamic thresholds are determined based on recent observations in a moving window with size *d*, in order to appropriately define ramp and non-ramp events over time.

Then, we use NeuroEvolution of Augmenting Topologies [53] to train NNs using the classified datasets, in which the NNs evolve based on a Genetic Algorithm to obtain the best topology and hyperparameters of NNs. As a result, 6 NNs, i.e., 3 for Mitsubishi and 3 for GE, are built (see Fig. 2.3). As the efficacy of NNs relies on the quality of training datasets, i.e., how good different ramp events are labeled, a Bayesian optimization based method is proposed to optimize the parameters of DSPOT for enhance the quality of the training datasets and the corresponding performance of the NNs. Ultimately, the proposed DSPOT enhanced self-evolving neural networks form a closed loop of optimizing the performance of wind farm generation forecast purely based on the data. In what follows, the design of each component of the model is described in detail.

2.4.1 DSPOT Based Ramp Classifier

Based on extreme value theory, it is likely that extreme events follow a generalized Pareto distribution (GPD) [20], which is observed in wind power ramps in Fig. 2.2. Thus motivated, we will develop a data fitting technique using the GPD model to determine the dynamic threshold $z_{q^{cat}}(t)$ for different ramp events, where the index $cat \in \{up, down\}$ denotes the category of ramp events and q^{cat} is the quantile of the corresponding ramp event distribution used to determine the threshold $z_{q^{cat}}(t)$. The idea is to first estimate the parameters of the GPD and then use the estimated GPD to find $z_{q^{cat}}(t)$ based on the quantile q^{cat}. To account for the time-varying dynamics of ramp events, the parameters of the GPD will be updated using the recent observed wind power in a moving window with size d.

Specifically, let $P_{class}(t)$ denote the wind power output at time t, where the index $class \in \{GE, Mitsubishi\}$ represents the class of wind turbines. In a specified time period l, ramp up and ramp down events can be separately expressed as:

$$\begin{aligned} P_{class}(t) - P_{class}(t-l) &= \Delta P^{l}_{class}(t) > z_{q^{up}}(t), \\ P_{class}(t) - P_{class}(t-l) &= \Delta P^{l}_{class}(t) < -z_{q^{down}}(t), \end{aligned} \tag{2.1}$$

where l and q^{cat} are parameters to be tuned by BO (see Section 2.4.3) to determine the ramp events.

Based on the above definitions of ramp events, we classify the original dataset into ramp up, ramp down, and non-ramp datasets, i.e., 3 different datasets for each class of wind turbines. Let $\mathcal{X}_i^{class}, i \in \{up, down, non\}$ denote these 3 datasets, where $\mathcal{X}_{up}^{class}$ denotes the ramp up dataset, $\mathcal{X}_{down}^{class}$ the ramp down dataset, and $\mathcal{X}_{non}^{class}$ the non-ramp dataset. These datasets will be used to train NNs in Section 2.4.2. Clearly, the quality of these datasets (i.e., how well different ramp events can be separated) depends on the values of $z_{q^{up}}(t)$ and $z_{q^{down}}(t)$. In this section, we determine $z_{q^{up}}(t)$ and $z_{q^{down}}(t)$ using the GPD model. For ease of presentation, we present how to calculate the dynamic threshold $z_q(t)$ for ramp up events by omitting the index *cat* in the following. Correspondingly, the dynamic threshold for ramp down events can be determined using the same procedure.

2.4.1.1 Calculating $z_q(t)$

We derive the log-likelihood of the GPD using the recent observations $\{\Delta P^l_{class}(t)\}_d$ in a moving window with size d:

$$\begin{aligned} L(\gamma, \xi; \{\Delta P^l_{class}(t)\}_d) = &-d \log \xi \\ &+ (\tfrac{1}{\gamma} - 1) \sum_{i=t-d+1}^{t} \log(1 - \tfrac{\gamma \Delta P^l_{class}(i)}{\xi}), \end{aligned} \tag{2.2}$$

where γ and ξ are the parameters of the GPD ($\gamma \neq 0$). To estimate the parameters of the GPD, we find a solution (γ^*, ξ^*) of L by solving the following two equations:

$$\frac{\partial L(\gamma, \xi; \{\Delta P^l_{class}(t)\}_d)}{\partial \gamma} = 0, \tag{2.3}$$

$$\frac{\partial L(\gamma, \xi; \{\Delta P^l_{class}(t)\}_d)}{\partial \xi} = 0. \tag{2.4}$$

Grimshaw [20] has shown that if a solution (γ^*, ξ^*) is obtained in this equation, the argument $\beta^* = \gamma^*/\xi^*$ is the solution to the scalar equation $u(\beta) \cdot v(\beta) = 1$, where

$$u(\beta) = \frac{1}{|\mathcal{Y}_q|} \sum_{i=1}^{|\mathcal{Y}_q|} \frac{1}{1 + \beta Y_i}, \tag{2.5}$$

$$v(\beta) = 1 + \frac{1}{|\mathcal{Y}_q|} \sum_{i=1}^{|\mathcal{Y}_q|} \log(1 + \beta Y_i). \tag{2.6}$$

Here a set $\mathcal{Y}_q = \{Y_i\}$ is defined for a given quantile q, i.e., $\text{Prob}(\Delta P^l_{class}(i) > P^{th}_q) = q$, where $P^{th}_q > 0$ is the threshold associated with the quantile q. $\mathcal{Y}_q$ contains all $\Delta P^l_{class}(i)$ larger than P^{th}_q with $Y_i = \Delta P^l_{class}(i) - P^{th}_q > 0$. $|\mathcal{Y}_q|$ denotes the cardinality of $\mathcal{Y}_q$. Based on Grimshaw trick [20], ξ^* and γ^* can be obtained using β^* by

$$\gamma^* = v(\beta^*) - 1, \tag{2.7}$$

$$\xi^* = \gamma^*/\beta^*. \tag{2.8}$$

As there are multiple possible solutions of β^*, we need to find all the solutions, in order to best estimate the GPD parameters (γ, ξ) to fit the distribution of ramp events. It is noted that $1 + \beta Y_i$ must be strictly positive. As Y_i is positive, we have $\beta^* \in (-\frac{1}{Y_{\max}}, +\infty)$. Grimshaw also shows an upper-bound $\beta^*_{\max}$:

$$\beta^*_{\max} = 2\frac{\bar{Y} - Y_{\min}}{(Y_{\min})^2}, \tag{2.9}$$

where $\bar{Y}$, $Y_{\max}$, and $Y_{\min}$ are the average amount, the maximum amount, and the minimum amount of $\mathcal{Y}_q$, respectively. Therefore, we can do a numerical root search and find all possible solutions in $(-\frac{1}{Y_{\max}}, \beta^*_{\max})$, in which we choose the solution that maximizes the likelihood L.

Based on the estimated GPD, we can calculate $z_q(t)$ by solving the probability $\text{Prob}(\Delta P^l_{class}(i) > z_q(t))$. Based on [5], we leverage the probability of the exceedances of $\Delta P^l_{class}(i)$ over the threshold P^{th}_q,

$$\begin{aligned}&\text{Prob}\{\Delta P^l_{class}(i) > z_q(t)|\Delta P^l_{class}(i) > P^{th}_q\}\\ &= (1+\hat{\gamma}(\tfrac{z_q(t)-P^{th}_q}{\hat{\xi}}))^{-\frac{1}{\hat{\gamma}}}.\end{aligned} \tag{2.10}$$

As $\text{Prob}(\Delta P^l_{class}(i) > P^{th}_q) = q$, we can solve

$$\text{Prob}(\Delta P^l_{class}(i) > z_q(t)) = q(1+\hat{\gamma}(\tfrac{z_q(t)-P^{th}_q}{\hat{\xi}}))^{-\frac{1}{\hat{\gamma}}} \tag{2.11}$$

based on Bayesian theorem. Using (2.11), we can obtain $z_q(t)$ by

$$z_q(t) = P^{th}_q + \frac{\hat{\xi}}{\hat{\gamma}}\left(\left(\frac{q \cdot d}{|\mathcal{Y}_q|}\right)^{-\hat{\gamma}} - 1\right). \tag{2.12}$$

2.4.1.2 DSPOT Algorithm

Given a quantile q^{cat}, the DSPOT algorithm determines the dynamic threshold $z_{q^{cat}}(t)$ using the recent observations. Based on $z_{q^{cat}}(t)$, wind generation difference $\Delta P^l_{class}(t)$ will be labeled into ramp up, ramp down or non-ramp events, and the wind power of recent measurement $P_{class}(t)$ will be added into the corresponding dataset $\mathcal{X}^{class}_i$. The details of the DSPOT algorithm is provided in Algorithm 1.

Specifically, Algorithm 1 will first initialize the thresholds $z_{q^{up}}(t)$ and $z_{q^{down}}(t)$ using the first $d+l$ wind power measurements. Then, Algorithm 1 will update $z_{q^{up}}(t)$ and $z_{q^{down}}(t)$ using the new wind power measurement in the moving window with size d in an online manner, based on which the new wind power measurement will be added into the corresponding dataset X_i^{class}. Algorithm 1 will be run for wind power measurements of each class of wind turbines.

Algorithm 1 DSPOT

```
**Input:** $\{P_{class}(t)\}$, $d$, $l$, $q^{up}$, and $q^{down}$.
**Output:** $X_{up}^{class}$, $X_{down}^{class}$, and $X_{non}^{class}$.
**Initialization:**
1) Calculate initial thresholds $z_{q^{up}}$, $z_{q^{down}}$ based on Section 2.4.1.1 using $\{P_{class}(t)|t = 1, ..., d + l\}$.
2) Initialize $X_{up}^{class}$, $X_{down}^{class}$, and $X_{non}^{class}$ based on $z_{q^{up}}$ and $z_{q^{down}}$.
**End Initialization**
**For every** $t > d + l$ **in** $\{P_{class}(t)\}$
1) Update $z_{q^{up}}(t)$ and $z_{q^{down}}(t)$ based on Section 2.4.1.1 using the recent observations $\{\Delta P_{class}^{l}(t)\}_d$.
2) Classify $\Delta P_{class}^{l}(t)$ based on $z_{q^{up}}(t)$ and $z_{q^{down}}(t)$, and add $P_{class}(t)$ into the corresponding dataset $X_i^{class}$.
```

2.4.2 Self-Evolving Neural Network

A self-evolving neural network (SEN) will be built for each dataset X_{up}^{class}, X_{down}^{class}, and X_{non}^{class}. When training the neural networks (NNs), each element $P_{class}(t + 1)$ in X_i^{class} is treated as the label and the corresponding features contain the wind speed $W_s(t)$, the change of wind direction degree $W_{dir}(t)$, and current power measurements $\{P_{class}(t), P_{class}(t-1), ..., P_{class}(t-Lag)\}$, where Lag depends on the measurements (see the discussion in Section 2.5.1). As demonstrated in Fig. 2.4, NEAT [53] is used to train a NN. NEAT leverages a genetic algorithm (GA) to evolve the NN. It obtains the best network topology and the best weighting parameters by minimizing the forecast error, i.e., min $\sum_t(\hat{P}_{class}(t) - P_{class}(t))^2$, where $\hat{P}_{class}(t)$ denotes the forecast from the NN.

As demonstrated in Fig. 2.4, the workflow of NEAT contains random population generation, crossover, mutation, speciation, and evaluation by the fitness function. In this chapter,

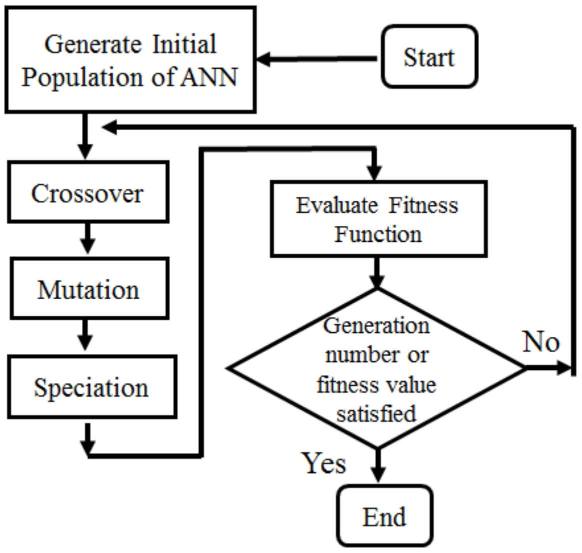

Generate Initial Population of ANN
Start
Crossover
Mutation
Speciation
Evaluate Fitness Function
Generation number or fitness value satisfied
No
Yes
End

Figure 2.4: Workflow of NEAT.

the fitness function is defined using the forecast accuracy:

$$Fit = -\sum_{t}(\hat{P}_{class}(t) - P_{class}(t))^2. \tag{2.13}$$

Each gene in the population set corresponds to a neural network. We aim to find the best gene with the largest fitness value (i.e., the lowest prediction error). In NEAT, the topology of a NN is directly encoded into the gene by a direct encoding scheme [35], in order to avoid Permutations Problem [45] and Competing Conventions Problem [36]. Specifically, connection and node (list of inputs, hidden nodes, and outputs) are encoded. Every unit of connection gene describes the connection weight (W), output node (O), input node (I), enable gate (E), and the number of innovation (N) that corresponds to a consecutive arrangement of new generated node. The workflow of NEAT will be elaborated in the following.

First, initial population (i.e., a set of genes) is generated randomly. Each gene represents a NN. Note that under this random generation, a neural network might contain no route from inputs to outputs, and we will remove these NNs from the initial population. For example,

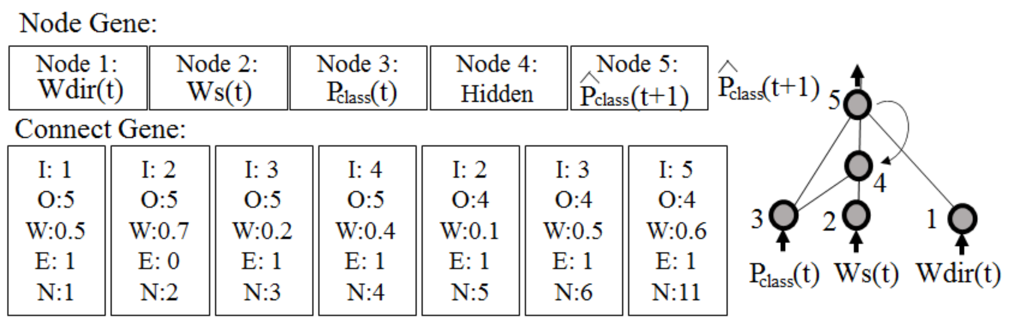


Figure 2.5: Encoding of a NN with 1 output and 3 inputs.

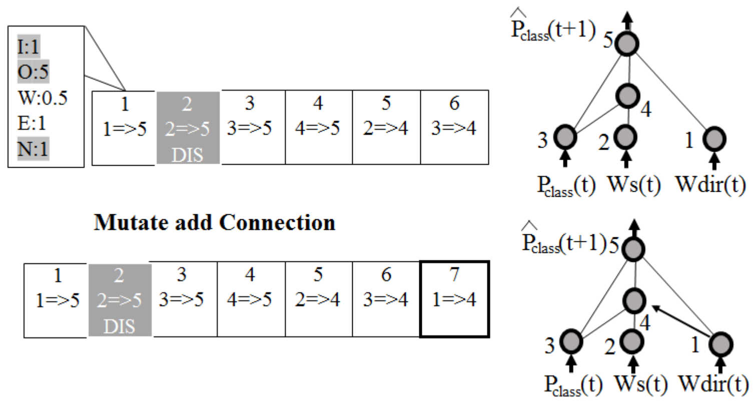


Figure 2.6: Mutation by appending a connection, where the link from Node 1 to Node 4 is inserted.

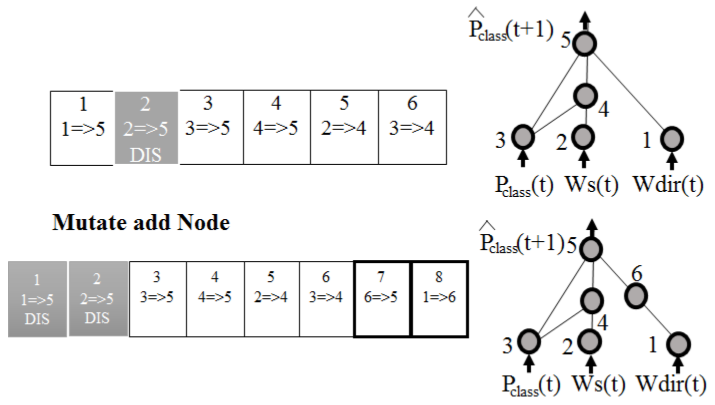


Figure 2.7: Mutation by appending a node, where Node 6 is inserted between Node 1 and Node 5.

Fig. 2.5 shows a NN containing 3 inputs ($P_{class}(t)$, $W_s(t)$, $W_{dir}(t)$) and 1 output ($\hat{P}_{class}(t+1)$), where in the first unit of connect gene, I:1 O:5 W:0.5 indicates connection from Node 1 to Node 5 with weight of 0.5, and E:1 means that this is an enabled connection.

After generating the initial population, NEAT iteratively optimizes the topology and connection weights of NNs using crossover and mutation. Specifically, nodes and connections of NNs are inserted or removed randomly based on the Poisson distribution [53]. For example, Figs. 2.6 and 2.7 show possible mutations by appending a connection and a node to a neural network, respectively. After crossover and mutation, topologically homogeneous genes are classified as one speciation determined by compatibility distance [53].

Then the fitness of species will be evaluated. If the highest fitness of species does not increase or the number of generation is achieved, NEAT will output the species with high fitness value, which will be used for wind generation forecast.

2.4.3 Bayesian Optimization based Parameter Search

The performance of NNs depends on the quality of datasets X_{up}^{class}, X_{down}^{class}, and X_{non}^{class} obtained by the DSPOT based ramp classifier in Section 2.4.1. As the performance of the DSPOT based ramp classifier relies on the parameters $\mathbf{b} = (l, d, q^{up}, q^{down})$, we develop a Bayesian optimization based approach that can efficiently find the best parameters $\mathbf{b}^*$. The idea is to model the unknown function between the parameters and the training errors as a multivariate Gaussian distribution, and then use a computationally cheap acquisition function to guide the search for the best parameters.

Specifically, we introduce an acquisition function $\zeta(\cdot)$ as the optimization objective, which characterizes the expected training error improvement under $\mathbf{b}$,

$$\zeta(\mathbf{b}) = \mathbb{E}\left[\sum_{class}\sum_{i}(F_i^{class}(\mathbf{b}^*) - F_i^{class}(\mathbf{b}))^+\right], \tag{2.14}$$

where $F_i^{class}(\mathbf{b})$ denotes the training error of the NN trained under $\mathbf{b}$ using X_i^{class} described in Section 2.4.2 and $F_i^{class}(\mathbf{b}^*)$ is the lowest error that has been obtained so far. It is assumed that the training errors $\{F_i^{class}(\mathbf{b})|i \in \{up, down, non\}, class \in \{GE, Mitsubishi\}\}$ are random variables following the multivariate Gaussian distribution $\mathcal{G} \sim \mathcal{N}(m(\mathbf{b}), \Sigma(\mathbf{b}))$ with mean $m(\mathbf{b})$ and covariance $\Sigma(\mathbf{b})$. In each attempt, we find $\mathbf{b}$ that maximizes the acquisition function $\zeta(\mathbf{b})$. Then, we use this $\mathbf{b}$ as the input of Algorithm 1 to determine X_{up}^{class}, X_{down}^{class}, and X_{non}^{class}, based on which we evolve the NNs. Then, $\{F_i^{class}(\mathbf{b})|i \in \{up, down, non\}, class \in \{GE, Mitsubishi\}\}$ will be added into a sample set $\mathcal{S}$, and the mean $m(\mathbf{b})$ and covariance $\Sigma(\mathbf{b})$ of $\mathcal{G}$ will be updated based on Bayesian Optimization [52]. The details of the Bayesian optimization based parameter search are given in Algorithm 3.

Algorithm 2 Bayesian optimization based parameter search

```
**Initialization:** Initialize $\mathcal{S} = \{(\mathbf{b}, \{F_i^{class}(\mathbf{b})\})\}$.
**For each attempt:**
1) Find the parameter vector $\hat{\mathbf{b}}$ that maximizes $\zeta$, i.e., $\hat{\mathbf{b}} = \arg\max_{(\mathbf{b},\{F_i^{class}(\mathbf{b})\})\in\mathcal{S}} \zeta(\mathbf{b})$.
2) Generate $X_{up}^{class}$, $X_{down}^{class}$, and $X_{non}^{class}$ based on Algorithm 1 using $\hat{\mathbf{b}}$, and evolve the NNs accordingly.
3) Add the current training errors $\{F_i^{class}(\hat{\mathbf{b}})\}$ into the sample set $\mathcal{S} = \mathcal{S} \cup (\hat{\mathbf{b}}, \{F_i^{class}(\hat{\mathbf{b}})\})$, and update the parameters of $m(\mathbf{b})$ and $\Sigma(\mathbf{b})$ using $\mathcal{S}$.
```

2.4.4 Short-Term Wind Farm Generation Forecast

The proposed DSPOT enhanced self-evolving neural networks (DSN) will train multiple NNs, which capture different dynamics of wind farm generation. When forecasting wind farm generation, we will first leverage the DSPOT based ramp classifier to determine whether the current state of wind farm generation is in ramp up, ramp down, or non-ramp. Based on the classified state, we choose the corresponding NNs to forecast the wind farm generation.

Specifically, let the function $H_{\theta_i}^{class}(\cdot)$ represents the neural network with parameters θ_i (i.e., the best gene) trained using the datasets: $X_i^{class}(t) = \{W_s(t), W_{dir}(t), P_{class}(t), P_{class}(t-1), ..., P_{class}(t-Lag)\}$, the output of the neural network is

$$\hat{P}_{class}(t+1) = H_{\theta_i}^{class}(X_i^{class}(t)). \tag{2.15}$$

Based on the results of the ramp classifier, we pick the corresponding NNs (i.e., the best gene) for each class of wind turbines. Therefore, the wind farm generation forecast $\hat{P}_{ag}(t+1)$ can be achieved by:

$$\hat{P}_{ag}(t+1) = \hat{P}_{mit}(t+1) + \hat{P}_{ge}(t+1). \tag{2.16}$$

Eq. (2.16) is the point forecast of wind farm generation.

To efficiently integrate the wind generation, distributional forecasts are often needed to manage the uncertainty [60]. To this end, we leverage the collection of genes generated in NEAT and use the forecasts by these genes to develop distributional forecasts. Let $\{\hat{P}_{ag}^{(j)}(t)\}$ represent the set of forecasts offered by each gene j. It is assumed that the forecast error of the point forecasts follows the standard normal distribution with the mean μ_t and the variance σ_t^2 as follows:

$$\mu_t = \frac{1}{J}\sum_{j=1}^{J} \hat{P}_{ag}^{(j)}(t), \tag{2.17}$$

$$\sigma_t^2 = \frac{1}{J}\sum_{j=1}^{J} (\hat{P}_{ag}^{(j)}(t) - \mu_t)^2, \tag{2.18}$$

where J is the number of genes. Under such assumption, we calculate the $(1-\alpha)$ confidence interval of the point forecasts (2.16) as follows:

$$[\hat{P}_{ag}(t+1) - Z(1-\frac{\alpha}{2})\sigma_{t+1}, \hat{P}_{ag}(t+1) + Z(1-\frac{\alpha}{2})\sigma_{t+1}], \tag{2.19}$$

where $Z(1-\frac{\alpha}{2})$ represents the point where the cumulative distribution function of the standard normal distribution is equivalent to $1-\frac{\alpha}{2}$.

Remarks. The proposed SENs can be trained offline. As the learning process of each SEN is based on different datasets, we can train these SENs on parallel. This can significantly reduce the training time of these SENs. Furthermore, the learning of SENs needs no AI experts to manually tune the topology and the hyperparameters; SENs can automatically adapt to the changing dynamics of wind farm generation purely based on the data. This can greatly facilitate the implementation of the proposed method in reality.

2.5 Experimental Evaluation

2.5.1 Experimental Setup

2.5.1.1 Data

The data used in case studies are described in Section 3.3.1. Specifically, we use the data of year 2009 to train the proposed SENs, and the data of year 2010 to validate the forecast performance of the proposed approach.

2.5.1.2 Evaluation Metrics

Mean absolute error (MAE) and root-mean square error (RMSE) are employed to evaluate the forecast performance, i.e.,

$$MAE = \frac{1}{N_t} \sum_t \left| \hat{P}_{ag}(t) - P_{ag}(t) \right|,$$

$$RMSE = \sqrt{\frac{1}{N_t} \sum_t \left| \hat{P}_{ag}(t) - P_{ag}(t) \right|^2}.$$

where N_t is the number of data points in the test dataset.

2.5.1.3 Parameter Tuning

The forecast performance of NNs greatly depends on the quality of training datasets, which hinges on the parameters (l, d, q^{up}, q^{down}) and Lag. To find the best (l, d, q^{up}, q^{down}), Algorithm 3 is run with 200 attempts. To optimize Lag, we evaluate MAE under different values of Lag (see Fig. 2.8) and pick the one with the lowest MAE. It is observed that the lowest MAE is achieved when the feature dimension is 9 (i.e., $Lag = 7$). Worth noting is that 1) for larger Lag, bad features would be incorporated because the non-stationary nature of wind generation (see the increase of MAE when the feature dimension is larger than 9); and 2) the evolution process under larger Lag becomes substantially complex, which makes the running time increase significantly.

2.5.1.4 Benchmark

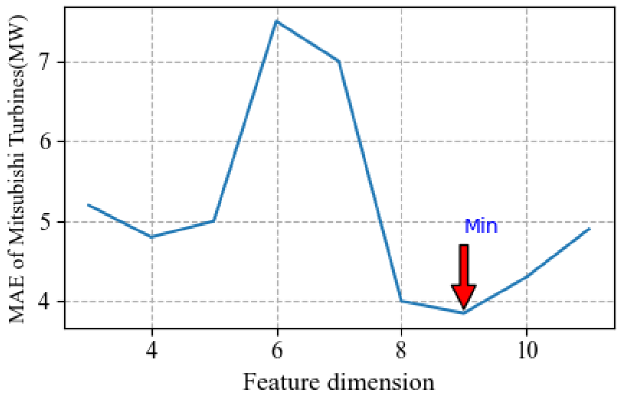


Figure 2.8: MAE versus feature dimension size.

We compare the forecast performance of the proposed approach with the following benchmarks:

- The adaptive AR model [61],
- The Markov-chain-based (MC) model [22],
- The SVM enhanced Markov (SVM-MC) model [61],
- The Seasonal NEAT (SNEAT) model trained by different season data without splitting ramp events,
- The NEAT model trained by the entire year data without splitting ramp events,
- The Long Short-Term Memory (LSTM) model trained by the entire year data,
- The Seasonal Self-evolving Neural Networks (SSEN) model [32].

The Seasonal NEAT model considers four seasons; but it does not split ramp and non-ramp events in the training process, which would lead to poor performance when ramp events occur. We use a prevailing structure of 3 layers to build the LSTM with the same configuration in our previous work [32].

2.5.2 Experimental Results

2.5.2.1 10-min Ahead Forecast

In Tables 2.1 and 2.2, we compare 10-min ahead forecast under different models for the whole year data and ramp events in year 2010, respectively. The forecast results in terms of MAE and RMSE are normalized using the nominal capacity 300.5MW.

Table 2.1
FORECAST UNDER DIFFERENT MODELS OVER THE WHOLE YEAR 2010

Error	AR	MC	SVM-MC	NEAT	SNEAT	LSTM	SSEN	DSN
MAE(%)	2.441	2.413	2.214	1.734	1.778	1.799	1.704	**1.661**
RMSE(%)	3.974	3.524	3.342	3.030	3.074	3.072	3.023	2.996

Table 2.2
FORECAST UNDER DIFFERENT MODELS OVER ALL RAMPS OF THE YEAR 2010

Error	AR	MC	SVM-MC	NEAT	SNEAT	LSTM	SSEN	DSN
MAE(%)	2.945	2.856	2.657	2.363	2.416	2.469	2.320	**2.288**
RMSE(%)	4.403	3.837	3.654	3.593	3.667	3.679	3.534	**3.518**

From Tables 2.1 and 2.2, we observe that the proposed approach DSN outperforms the benchmarks. Compared with the non-NN based benchmarks (AR, MC, and SVM-MC), the proposed approach improves the MAE at least 24.9% for the whole year data and at least 13.8% for ramp events, respectively. Compared with the NN based benchmarks, the improvement of the proposed approach(DSN) in terms of MAE is at least 2.5% for the whole year and at least 1.3% for ramp events. Such improvements are because of the splitting of non-ramp and ramp events, which enables SEN to more effectively learn different dynamics of GE and Mitsubishi turbines measurements under non-ramp and ramp events.

Figs. 2.9, 2.10, and 2.11 illustrates the prediction intervals for 3 representative ramp

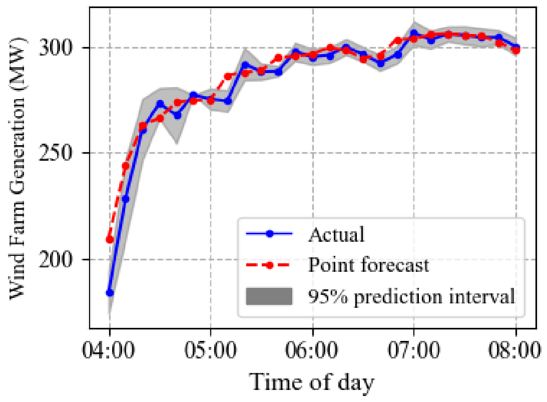


Figure 2.9: January 5, 2010.

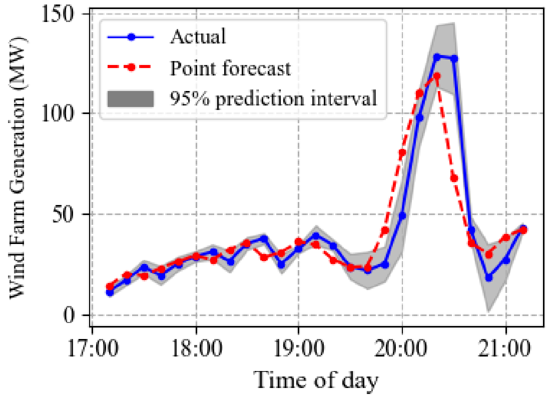


Figure 2.10: March 19, 2010.

events. January 5, 2010 is chosen because there is a wind power ramp up event from 4AM to 5AM with a ramp up rate of 85 Megawatt per hour (MW/H). March 19, 2010 is chosen because of the significant wind power fluctuation from 7PM to 9PM with both ramp up and ramp down events of an average ramp rate around 100 MW/H. October 9, 2010 is chosen because of a remarkable ramp down event from 3AM to 5AM with an average ramp rate of 66.5 MW/H. As demonstrated in those pictures, the actual wind farm generation is mostly confined in the prediction interval achieved from (2.19), regardless of the sharp ramps.

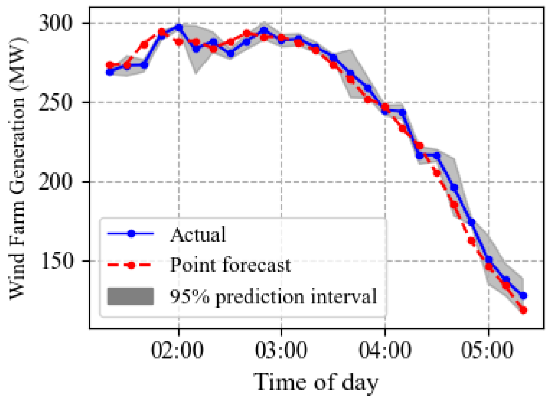


Figure 2.11: October 9, 2010.

2.5.2.2 Other Forecasting Horizons

In Tables 2.3 and 2.4, we compare forecast of different models under different horizons using the whole year data and ramp events in year 2010, respectively. From Tables 2.3 and 2.4, we observe that the proposed approach outstrips the benchmarks under these forecasting horizons. It is observed in most cases that seasonal NEAT do not outperforms NEAT which is trained by entire year data. It is due to the data of one season might not enough to train a good NN compared to the while year data.

Table 2.3
MAE OF DIFFERENT MODELS AT DIFFERENT FORECASTING HORIZONS OVER THE WHOLE YEAR 2010

Model	30min	40min	50min	60min
AR	4.837	6.516	8.160	9.624
MC	4.733	6.233	7.551	8.727
SVM-MC	4.733	6.233	7.550	8.727
NEAT	4.804	5.851	6.939	7.640
SNEAT	5.064	6.322	7.681	8.277
LSTM	4.664	6.517	7.681	8.257
SSEN	4.852	5.970	7.095	7.862
DSN	**3.755**	**4.220**	**4.746**	**5.069**

Table 2.4
MAE OF DIFFERENT MODELS AT DIFFERENT FORECASTING HORIZONS OVER RAMP EVENTS OF THE YEAR 2010

Model	30min	40min	50min	60min
AR	6.991	8.871	11.883	11.996
MC	6.592	8.426	10.654	11.091
SVM-MC	6.591	8.425	10.654	11.091
NEAT	7.255	8.379	10.366	10.274
SNEAT	7.427	8.612	10.471	10.385
LSTM	7.025	9.255	11.558	10.915
SSEN	7.092	8.182	9.727	9.849
DSN	**5.420**	**5.595**	**7.023**	**6.197**

For the 30-min ahead forecast, compared with the non-NN based benchmarks (AR, MC, and SVM-MC), the proposed approach improves the MAE at least 20.6% for the whole year data and at least 17.7% for ramp events, respectively; compared with the NN-based benchmarks (NEAT, LSTM and SSEN), the enhancement of the proposed approach by MAE is no less than 19.4% for the whole year data and at least 22.8% for ramp events.

For the 40-min ahead forecast, compared with the non-NN based benchmarks (AR, MC, and SVM-MC), the proposed approach improves the MAE at least 32.2% for the whole year data and at least 33.5% for ramp events, respectively; compared with the NN-based benchmarks (NEAT, LSTM and SSEN), the enhancement of the proposed approach by MAE is no less than 27.8% for the whole year data and at least 31.6% for ramp events.

For the 50-min ahead forecast, compared with the non-NN based benchmarks (AR, MC, and SVM-MC), the proposed approach improves the MAE at least 37.1% for the whole year data and at least 34% for ramp events, respectively; compared with the NN-based benchmarks (NEAT, LSTM and SSEN), the enhancement of the proposed approach by MAE is no less than 31.6% for the whole year data and at least 27.7% for ramp events.

For the 60-min ahead forecast, compared with the non-NN based benchmarks (AR, MC, and SVM-MC), the proposed approach improves the MAE at least 41.9% for the whole year data and at least 44.1% for ramp events, respectively; compared with the NN-based benchmarks (NEAT, LSTM and SSEN), the enhancement of the proposed approach by MAE is no less than 33.6% for the whole year data and at least 37% for ramp events.

2.6 Conclusion

We develop a method to enhance situational awareness in short-term wind power forecasts by automatically splitting different dynamic data. Specifically, the proposed approach initially classifies the wind farm generation data into ramp and non-ramp datasets using DSPOT, which leverages dynamic ramp thresholds to account for the time-varying dynamics of ramp and non-ramp events. We then train different NNs based on each dataset to learn different dynamics of wind farm generation by NEAT, which are able to obtain the best network topology and weighting parameters. As the efficacy of the neural networks relies on the quality of training datasets (i.e., the classification accuracy of ramp and non-ramp events), a Bayesian optimization based approach is developed to optimize the parameters of DSPOT to enhance the quality of the training datasets and the corresponding performance of the neural networks. Experimental results show the proposed approach is a promising way to enhance situational awareness.

CHAPTER 3
A MACHINE LEARNING FRAMEWORK FOR ROBUST EVENT CLASSIFICATION USING IMPERFECT REAL-WORLD PMU DATA

3.1 Introduction

Growing miscellaneous faults (e.g., bad weather,time-worn power lines, wildfire and animal activities) in transmission make the complex grid more vulnerable. Recent years have witnessed the booming deployment of phasor measurement units (PMUs) [2]. More than 2500 PMUs are installed in the North American power system. Compared to traditional supervisory control and data acquisition (SCADA) systems, PMUs are of much higher sampling rates (e.g., 30 or 60 samples per second in the U.S.), which provides golden opportunities to achieve high level of situational awareness (e.g., real-time event detection and classification), in order to prevent large-scale blackouts (e.g., [1, 57])

However, large volumes of streaming data (i.e., 100 PMUs @60Hz could generate 600 GB raw data in one day) make detection and identification manually impossible, this calls for automatic event classification. On the other hand, the real-world data is imperfect (e.g., bad data and missing data) and insufficient, directly using such data to build machine learning (ML) models can hardly get good results. Further, the raw data is in an overwhelmingly large volume, so it can hardly use it directly as inputs of any ML models.

In this chapter, we proposed a framework to enhance the situational awareness on transmission grid in terms of event classification that could work in imperfect real-world data.

3.2 Related Works

Much effort has been made on the development of PMU based event detection and classification. For PMU based event detection, many methods have been developed (e.g., [12, 29, 30, 38, 48, 49, 59, 65]). Compared to PMU based event detection, large amounts of high-quality labeled PMU datasets are critical for the development of PMU based event classification, especially for the development of neural network based classifiers. However, based on our observations from large amounts of real-world PMU data (see Section 3.3), high-quality labeled PMU datasets are not available in practice. In fact, many existing works on PMU based event classification consider a small amount of PMU data with a few labeled events. For example, in [28], the dataset consists of only 32 labeled events; in [14], only 4 PMUs are used in case studies; in [39], only 57 labeled line events are used to train an event classifier; in [56], hundreds of labeled frequency events from the FNET/GridEye system are used to train a Convolutional Neural Network (CNN) based frequency event detector. The generalization of event classifiers trained using a small dataset can be poor.

To address the challenge of insufficient PMU data, some studies leverage synthetic data generated by simulation or neural networks. For instance, in [30], simulated data with man-made noises are used; in [63], Generative Adversarial Networks (GAN) are used to generate synthetic event data. While synthetic data can increase the amounts of data for training event classifiers, the events and grid characteristics hidden in the real PMU data can hardly be represented by synthetic data. Thus, the generalization of event classifiers trained using synthetic data can still be poor.

We leverage large amounts of real-world PMU data from the Western Interconnection of continental U.S. transmission grid (see the data description in Section 3.3.1) for the development of event classifiers. By analyzing this large dataset, we find that it is challenging to directly use this dataset to train event classifiers due to the low data quality. Specifically, the real PMU data are noisy and contain bad data, dropouts, and timestamp errors. The timestamps of labeled events provided in the event logs are inaccurate. Though the entire dataset contains measurements from many PMUs over a two-year period, the total number of labeled events is only a few thousands and the distribution of different event types is highly imbalanced. Due to these issues, the performance of off-the-shelf machine learning models directly trained using such dataset can be significantly degraded.

Recently, there have been several attempts using large-scale real PMU data to develop event classifiers based on neural networks (e.g., auto-encoder model [24], Convolutional Neural Network (CNN) [40], spatial pyramid pooling (SPP)-aided CNN [62], and information loading enhanced Deep Neural Network (DNN) [50]). However, the hyperparameter tuning of the neural network based approaches is challenging and the training time is long. Though good classification results using neural networks are reported [24, 40, 50, 62], the interpretability of neural networks is low, not to mention the adversarial vulnerability of neural networks.

3.2.1 Main Contributions

To address these challenges, we develop a novel machine learning framework for training event classifiers using large-scale imperfect real-world PMU data, which consists of three main steps: data preprocessing, fine-grained event data extraction, and feature engineering. The goal is to obtain high-quality labeled PMU training data from the low-quality real PMU data. Specifically, the data preprocessing step addresses the data quality issues of PMU measurements (e.g., bad data and missing data); the fine-grained event data extraction step accurately localizes the events from the inaccurate event timestamps in the event logs by using a model-free event detection method developed based on the low-rank property of PMU data; and the feature engineering step constructs the event features based on the patterns of different event types, in order to improve the performance and the interpretability of the event classifiers. Based on the proposed machine learning framework, we also develop a workflow for event classification using the real PMU data streaming into the system in real time. One salient merit of the proposed framework is that large-scale real PMU data is reduced to a small set of event features, which can be used to efficiently train many off-the-shelf lightweight machine learning models. As the features are constructed based on the event patterns, the contribution of any single PMU measurement to the features is low, which can effectively mitigate the impact of bad and missing data and improve the robustness of the event classifiers.

Using the two-year real-world PMU data from the Western Interconnection of continental U.S. transmission grid, we evaluate the performance of the proposed machine learning framework. Many off-the-shelf lightweight machine learning models can be efficiently trained in our framework and achieve good performance. For example, the training time of the Random Forest model is less than 2 seconds while the testing accuracy of the Random Forest model is 94%. Moreover, our framework demonstrates a strong robustness against missing data. Even under the missing rate of 50%, the accuracy of the Random Forest model can still achieve 87%. In summary, the proposed machine learning framework provides a promising way to train robust event classifiers with good interpretability and low training cost.

3.3 Data Description and Key challenges

In this section, we introduce the real-world PMU data used in this study and identify the key challenges of using this dataset for the development of event classifiers.

3.3.1 Data Description

This paper uses real-world PMU data from the Western Interconnection of continental U.S. transmission grid. The dataset is complied by the Pacific Northwest National Laboratory (PNNL) to anonymize the data such that proprietary information (e.g., PMU locations, event locations, and the system topology) is unavailable. The dataset contains measurements from 43 PMUs over a two-year period (2016–2017). The sampling rates of PMUs are either 30 or 60 frames per second. The size of the dataset is about 5 TB (stored in Parquet format), which contains over 93 billion records. In each record, the measurements contain: 1) coordinated universal time (UTC), 2) voltage magnitude of positive sequence, A phase, B phase, and C phase, 3) voltage angle of positive sequence, A phase, B phase, and C phase, 4) current magnitude of positive sequence, A phase, B phase, and C phase, 5) current angle of positive sequence, A phase, B phase, and C phase, 6) frequency, 7) rate of change of frequency (ROCOF), 8) PMU status flag, and 9) anonymized PMU ID.

Besides the raw PMU measurements, event logs are provided for the development of event classifiers. In the event logs, four types of events (i.e., line outage, transformer outage, frequency event, and oscillation event) are recorded. For each event, start timestamp, end timestamp, event type, event cause, and event description are provided. The total number of events in the event logs is 4,854, including 3,667 line outages, 621 transformer

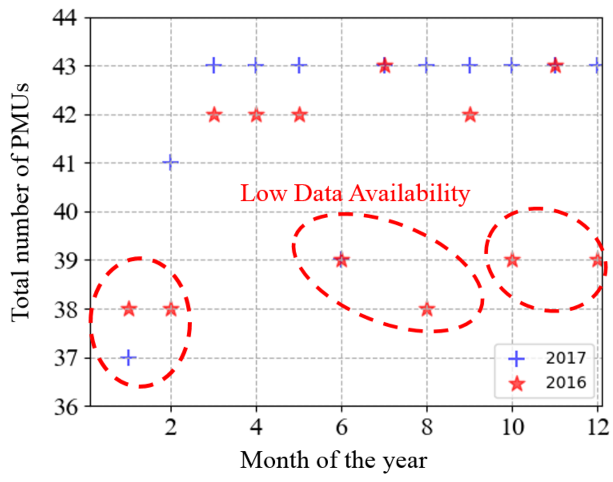


Figure 3.1: PMU data availability.

outages, 465 frequency events, and 100 oscillation events.

3.3.2 Key Challenges

By analyzing the real-world PMU measurements and the event logs, we find that it is challenging to directly use this dataset to develop event classifiers as the data quality is low and off-the-shelf machine learning approaches require high-quality training data. Specifically, we face the following major challenges of using this dataset for the development of event classifiers.

3.3.2.1 Incomplete and Noisy PMU Measurements

Fig. 3.1 illustrates the availability of PMU data, where the data from some PMUs are completely missing in certain months. Moreover, the current magnitude of A phase, B phase and C phase and the current angle of A phase, B phase and C phase are unavailable

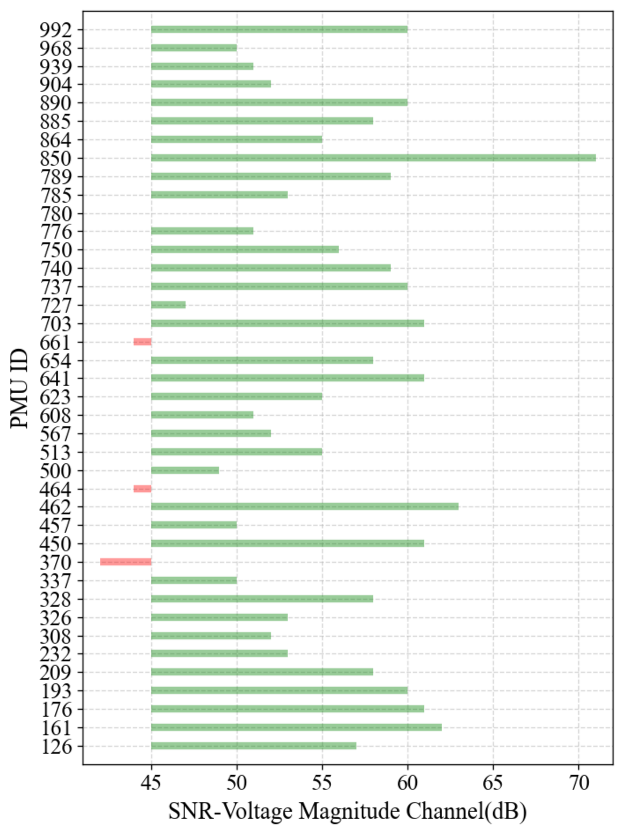
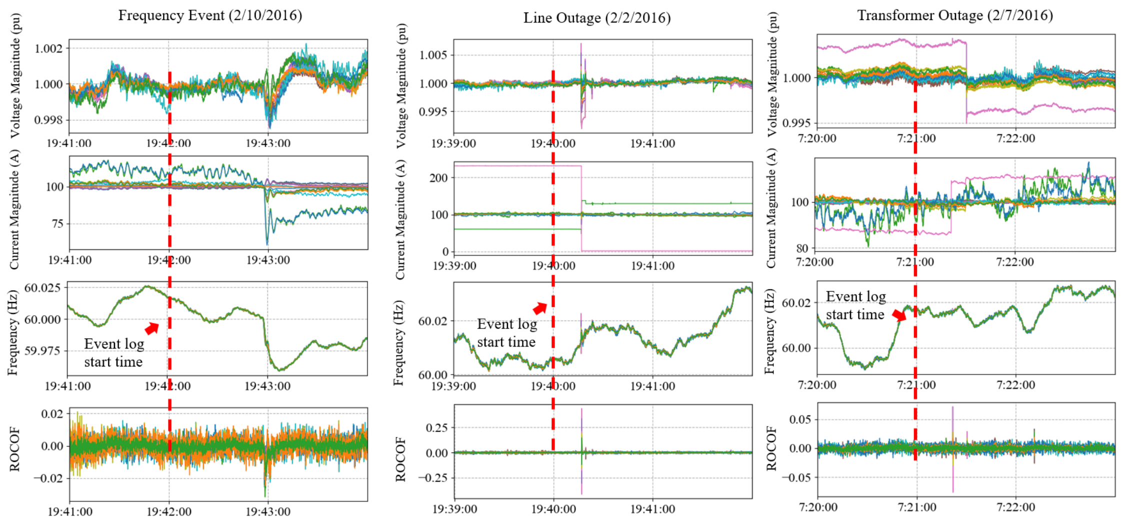


Figure 3.2: SNR of voltage magnitude of positive sequence of different PMUs.

in most cases. Fig. 3.2 illustrates the signal-to-noise ratio (SNR) of PMUs. Based on [6], 45 dB can be used as a threshold to indicate whether the PMU measures are noisy. As shown in Fig. 3.2, three PMUs are below 45 dB. Therefore, the quality of PMU measurements needs to be accounted for when preparing for the training dataset.

3.3.2.2 Inaccurate Event Timestamps of Event Logs

By analyzing the event logs, we observe that the timestamps of events provided in the event logs are inaccurate. Fig. 3.3 illustrates the PMU measurements during different events, where the red vertical line indicates the start time of the event provided in the event logs. For example, in Fig. 3.3 (a), the actual start time of the frequency event is around 19:43:00, while the start time provided in the event logs is 19:42:00. Clearly, the timestamps in the event logs are not accurate. If such timestamps are used for event extraction, with high probability, the event features would be missing and therefore the performance of event classification would be degraded.



(a) Example of a frequency event. (b) Example of a line event (c) Example of a transformer event

Figure 3.3: Example of different events.

3.3.2.3 Insufficient and Imbalanced Training Data

As shown in Fig.3.4 total number of events in the event logs is only less than five thousands(i.e., total number of events in the event logs is 4,854, including 3,667 line outages, 621 transformer outages, 465 frequency events, and 100 oscillation events) and 75% of the events are line outages. Directly applying off-the-shelf machine learning models on such dataset can lead to overfitting issues, especially for neural network based models, where the parameters of neural networks can be much larger than the number of events in the training dataset. To address this challenge, recent works on neural networks (e.g., [50, 62]) leverage data augmentation and report good classification results; however, the training time of neural networks is long and the interpretability of neural networks is low.

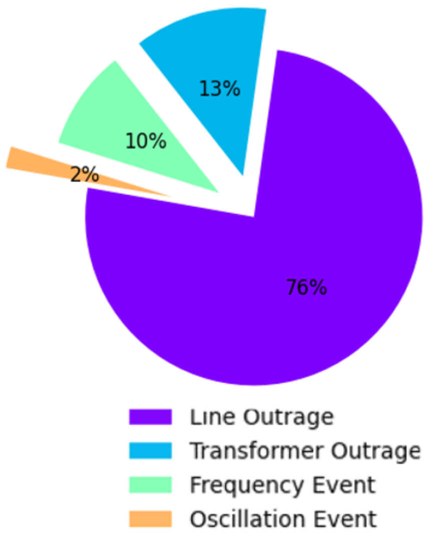


Figure 3.4: Insufficient and imbalanced training data.

3.3.2.4 Incomplete Labeled PMU Data

As shown in Fig.3.5, about twenty-five hundred events in 2016 are covered in the event logs, however we found there are at least one thousand events not been reported. In 2017, about four thousand events need to be added. Also, those event logs are maintained manually, so here is a chance to contain some errors.

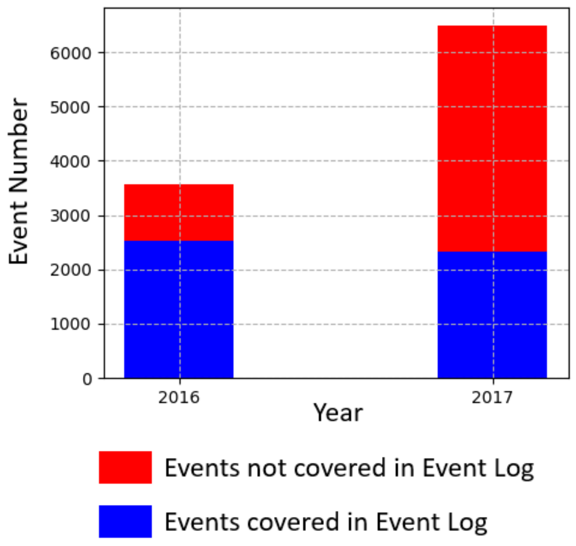


Figure 3.5: The event logs are incomplete.

As shown in the next section, different types of the events exhibit distinct features, based on which good machine learning models with better interpretability and low training cost can be built.

3.4 A Machine Learning Framework for Robust Event Classification

3.4.1 The Machine Learning Framework for Real-world PMU Data

To address the challenges of using the real-world PMU data for the development of event classifiers, we propose a novel machine learning framework to handle these challenges, as illustrated in Fig. 3.6. Specifically, we first do data preprocessing: 1) we coarsely localize the events based on the inaccurate event timestamps in the event logs, where a large time window (e.g., 10 minutes) around the event timestamp in the event logs is used to ensure that the event features are included; 2) we then do data quality assessment to filter out bad readings, which are treated as "missing" data; and 3) we complete all the missing data based on our prior work [19]. Then, we do fine-grained event data extraction to accurately localize the events using event detection, where a much smaller time window (e.g., 5 seconds) will be used to ensure that only the event data is included. Based on the fine-grained event data extraction, more distinct features can be constructed for each event type. These features will be used to train event classifiers. In the following, we discuss the details of each component in the proposed machine learning framework.

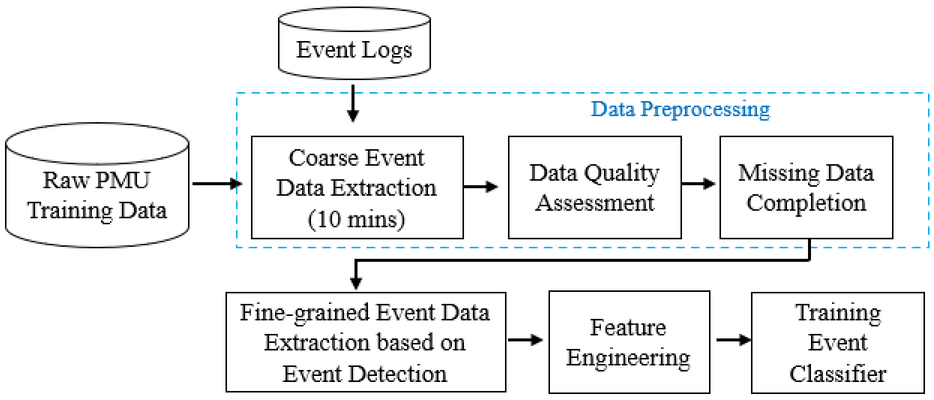

Event Logs
Data Preprocessing
Raw PMU Training Data
Coarse Event Data Extraction (10 mins)
Data Quality Assessment
Missing Data Completion
Fine-grained Event Data Extraction based on Event Detection
Feature Engineering
Training Event Classifier

Figure 3.6: The machine learning framework for robust event classification.

3.4.2 Data Preprocessing

As the event timestamps in the event logs are inaccurate, we extract the data in a large time window (i.e., 10 minutes) centered at the start time of an event provided in the event logs (i.e, 5 minutes before and 5 minutes after the start time), in order to preserve the event features. In the raw PMU dataset, the measurements of all PMUs in each day are stored in one Parquet file and these measurements are not ordered according to the timestamps. When doing the event data extraction, we partition the data based on PMU ID and sort the corresponding measurements based on the timestamps. As mentioned in Section 3.3, certain signals (e.g., the current magnitude of A, B and C phases) are missing in most cases. In the event data extraction, we extract the voltage magnitude and the current magnitude of positive sequence and ROCOF for the development of event classifiers.

Then, we do data quality assessment to filter out bad data. As the PMU status flags are provided, we first remove the measurements when PMUs were malfunctioning or in test mode. After the status check, we further remove the bad data based on the following criteria:

- Remove the measurements out of physical bound, i.e., the angle measurement larger than 180 or less than zero and the current or voltage magnitude less than zero.
- Remove the measurements that are 3 times greater than the standard deviation based on the empirical distribution of the measurements.

The bad data are treated as "missing" data, which are completed based on our regularized tensor completion approach [19].

3.4.3 Event Detection Based Fine-grained Data Extraction

The goal of fine-grained data extraction is to accurately extract the event data for better constructing event features, as the event timestamps provided in the event logs are not accurate. To this end, we develop a model-free event detection method based on the low-rank property of PMU data to accurately localize the events. We observe that when the disturbance occurs in the system, the low-rankness of PMU data will change (see Fig. 3.7), which can be quantified using the singular values of PMU measurement matrices.

Specifically, let $M_s^w(t) \in \mathbb{C}^{w \times n}$ be a PMU measurement matrix for an extracted signal $s \in \mathcal{S}$, which embraces the past w measurements before the timestamp t from n PMUs. Here $\mathcal{S}$ denotes the set of extracted signals including the voltage magnitude and the current magnitude of positive sequence and ROCOF. For $M_s^w(t)$, we do the singular value decomposition (SVD) and compute the ratio of the largest σ_1 and the second largest singular σ_2 values, i.e., $\eta_t = \frac{\sigma_2}{\sigma_1}$. Then, the average relative change of this ratio in the time window w is calculated as

$$\xi_s^w(t) = \frac{\eta_t - \eta_{t-w}}{\eta_{t-w} \cdot w}. \tag{3.1}$$

Based on these $\xi_s^w(t)$ from different signals, we use a threshold-based OR rule to determine whether there is an event. In other words, an event is detected, if one of these $\xi_s^w(t)$ is greater than a pre-determined threshold θ_s. Fig. 3.7 gives an example of detecting a frequency event using the ROCOF signal, where $\xi_f^w(t)$ changes significantly when the frequency event occurs. When an event is detected, we extract the event data in a smaller time window W (e.g., 5 seconds) centered at the detected event start time.

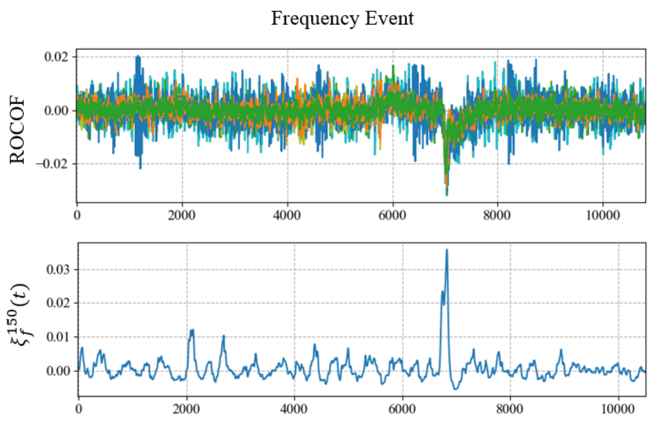


Figure 3.7: Frequency event detection based on the low-rankness of PMU data, where $\xi_f^{150}(t)$ of the ROCOF signal is shown in the bottom and $w = 150$ is used.

The proposed event detection approach is model-free with the parameters of the window size w and the detection thresholds $\boldsymbol{\theta} = \{\theta_s\}$ of extracted signals $\mathcal{S}$. To optimize the detection performance, we tune the parameters by a Baysian optimization algorithm, which can efficiently search for the best parameters. The idea of Baysian optimization is to model the unknown function between the parameters and the detection errors using a multivariate Gaussian distribution, and then use a computationally cheap acquisition function to guide the search. We introduce an acquisition function $\beta(\cdot)$ as the optimization objective, which characterizes the expected detection error improvement under $(\boldsymbol{\theta}, w)$:

$$\beta(\boldsymbol{\theta}, w) = \mathbb{E}[(e(\boldsymbol{\theta}^*, w^*) - e(\boldsymbol{\theta}, w))^+], \tag{3.2}$$

where $e(\boldsymbol{\theta}, w)$ denotes the detection error under $(\boldsymbol{\theta}, w)$, and $e(\boldsymbol{\theta}^*, w^*)$ denotes the lowest detection error that has been obtained so far. It is assumed that the detection errors are random variables following the multivariate Gaussian distribution: $\mathcal{G} \sim \mathcal{N}(m(\boldsymbol{\theta}, w), Cov(\boldsymbol{\theta}, w))$ with mean $m(\boldsymbol{\theta}, w)$ and covariance $Cov(\boldsymbol{\theta}, w)$. In each iteration, we find $(\boldsymbol{\theta}, w)$ that maximizes the acquisition function $\beta(\boldsymbol{\theta}, w)$. Then, $(\boldsymbol{\theta}, w)$ and the corresponding $e(\boldsymbol{\theta}, w)$ will be added into a sample set $\mathcal{S}$, and the mean $m(\boldsymbol{\theta}, w)$ and covariance $Cov(\boldsymbol{\theta}, w)$ of $\mathcal{G}$ will be updated accordingly [52]. The details of the Bayesian optimization based parameter search are given in Algorithm 3.

Algorithm 3 Bayesian optimization based parameter search

```
  **Initialization:** Initialize $\mathcal{S} = \{((\boldsymbol{\theta}, w), e(\boldsymbol{\theta}, w))\}$.
  **For each iteration:**
  1) Find the parameters $(\hat{\boldsymbol{\theta}}, \hat{w})$ that maximize $\beta$, i.e., $(\hat{\boldsymbol{\theta}}, \hat{w}) = \arg\max_{((\boldsymbol{\theta}, w), e(\boldsymbol{\theta}, w)) \in \mathcal{S}} \beta(\boldsymbol{\theta}, w)$.
  2) Use $(\hat{\boldsymbol{\theta}}, \hat{w})$ for event detection and compute the corresponding detection error $e(\hat{\boldsymbol{\theta}}, \hat{w})$.
  3) Add $((\hat{\boldsymbol{\theta}}, \hat{w}), e(\hat{\boldsymbol{\theta}}, \hat{w}))$ into the sample set $\mathcal{S} = \mathcal{S} \cup ((\hat{\boldsymbol{\theta}}, \hat{w}), e(\hat{\boldsymbol{\theta}}, \hat{w}))$, and update the parameters of $m(\boldsymbol{\theta}, w)$ and $Cov(\boldsymbol{\theta}, w)$ using $\mathcal{S}$.
```

3.4.4 Feature Engineering

Using the event data from fine-grained data extraction, we construct the event features based on the patterns of different event types. By carefully analyzing the PMU data during events, we find that the shape and the duration of the signals under different event types are distinct:

- The ROCOF measurements of frequency events tend to have a deeper and wider dip, compared to a narrow spark of line and transformer outages (see Fig. 3.8(a)).
- The voltage magnitude measurements of transformer outages tend to have a cliff-like drop with a longer duration time, compared with a narrow spark of line outages (see Fig. 3.8(b)).
- The signal similarity among different PMUs under different event types is different. Frequency events can be observed by all PMUs, while only a few PMUs can capture line and transformer outages.

Based on these observations, we construct the event features.

Specifically, let $X_i^s(t)$ denote the measurement of the ith PMU's signal $s \in \mathcal{S}$ at time t. In the event detection based fine-grained data extraction, we extract the event data in a small time window W. Let $\mathcal{W}$ denote the set of timestamps in this time window W. Using the measurements $\{X_i^s(t), t \in \mathcal{W}\}$ of each PMU's signal in this window, we construct the following six event features as illustrated in Fig. 3.9:

- Amplitude above the average:

$$F_1^s(i) = \max_{t \in \mathcal{W}} \{X_i^s(t) - \overline{X_i^s}\} \tag{3.3}$$

- Amplitude below the average:

$$F_2^s(i) = \max_{t \in \mathcal{W}} \{\overline{X_i^s} - X_i^s(t)\} \tag{3.4}$$

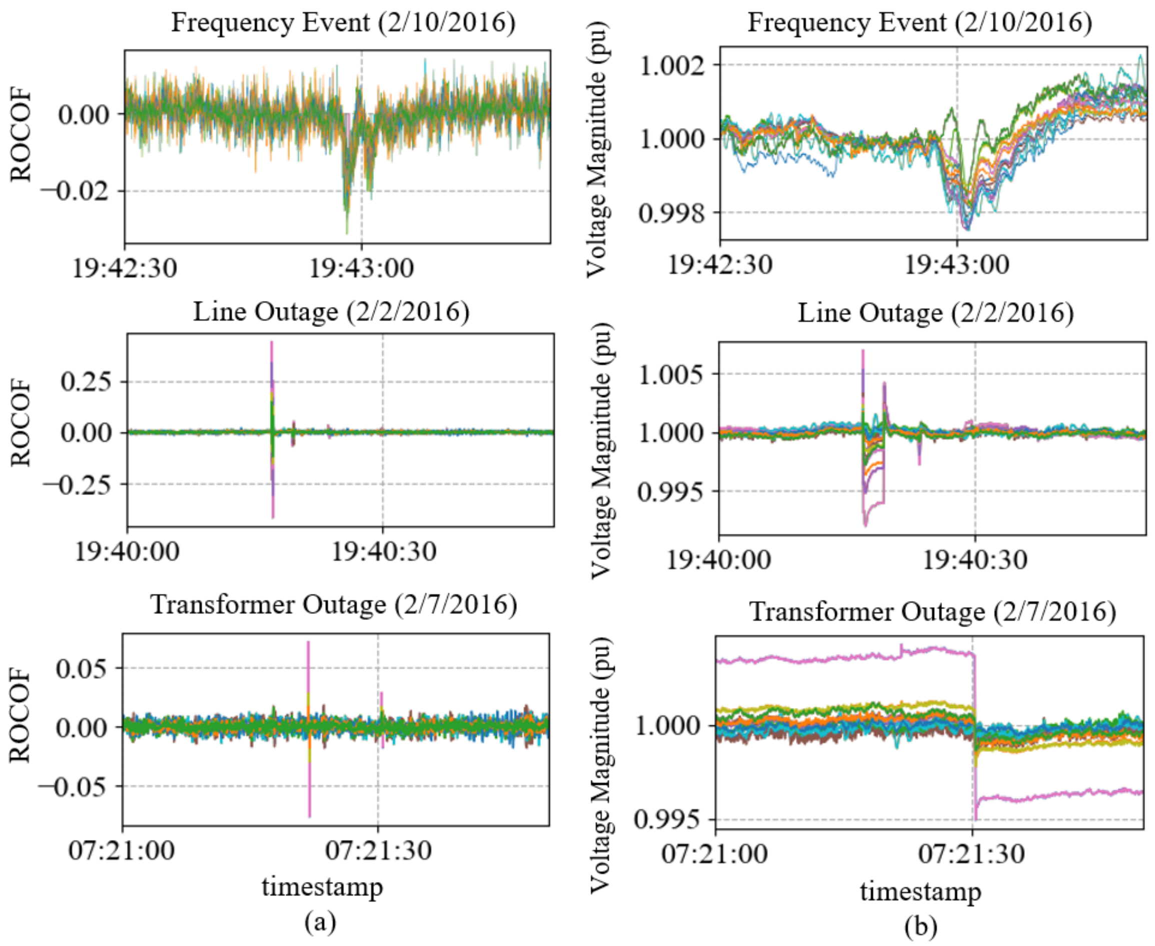


Figure 3.8: Comparison of ROCOF and voltage magnitude in different events.

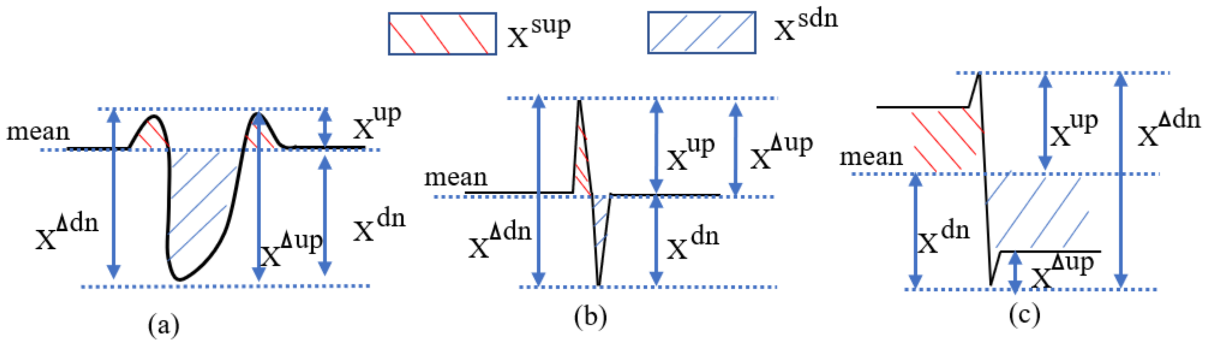


Figure 3.9: Example of created features.

- Ramp-up rate:

$$F_3^s(i) = \max_{t_1>t_2, t_1,t_2\in W} \{X_i^s(t_1) - X_i^s(t_2)\} \tag{3.5}$$

- Ramp-down rate:

$$F_4^s(i) = \max_{t_1<t_2, t_1, t_2 \in \mathcal{W}} \{X_i^s(t_1) - X_i^s(t_2)\} \tag{3.6}$$

- Area above the average:

$$F_5^s(i) = \sum_{\{t | X_i^s(t) > \overline{X_i^s}, t \in \mathcal{W}\}} (X_i^s(t) - \overline{X_i^s}) \tag{3.7}$$

- Area below the average:

$$F_6^s(i) = \sum_{\{t | X_i^s(t) < \overline{X_i^s}, t \in \mathcal{W}\}} (\overline{X_i^s} - X_i^s(t)) \tag{3.8}$$

where $\overline{X_i^s}$ is the average of the ith PMU's signal s, i.e., $\overline{X_i^s} = \frac{1}{W} \sum_{t \in \mathcal{W}} X_i^s(t)$. These features aim to capture the shape features of different event types shown in Figs. 3.9 and 3.8(b). Fig. 3.9 illustrates the constructed features for each event type, where Fig. 3.9(a) corresponds to the typical shape of ROCOF in a frequency event, Fig. 3.9(b) corresponds to the typical shape of voltage/current magnitude in a transformer outage, and Fig. 3.9(c) corresponds to the typical shape of ROCOF in a transformer/line outage.

To capture the signal similarity among different PMUs, we compute the maximum $F_j^{s,max}$, the minimum $F_j^{s,min}$, and the mean $\overline{F_j^s}$ values of each feature $j = 1, ..., 6$ based on n PMUs' features $\{F_j^s(i), i = 1, ..., n\}$. As the voltage magnitude, the current magnitude, and ROCOF are considered in the extracted event data, the number of features constructed for each event is 54 ($6 \times 3 \times 3$).

Based on our experiments, we find that the auxiliary ratio features $\{\overline{F_1^s}/\overline{F_2^s}, \overline{F_5^s}/\overline{F_6^s}, \overline{F_5^s}/(\overline{F_5^s}+$

$\overline{F^s_6}$)} for ROCOF constructed based on $\overline{F^s_j}$ can better capture the signal shape. Therefore, in our experiments, we use these 57 (54+3) features of each event to build event classifiers.

Remarks. The proposed features are constructed based on the patterns of different event types observed in real-world PMU data. Compared to neural network based approaches, the number of features in our approach is much less than the large number of automatically generated features (e.g., CNN), and the interpretability of our approach is much better. Moreover, with the small number of good features, we need much less number of training data to train a good event classifier, which can address the challenge of insufficient and imbalanced training data, and the training time is negligible compared to neural network based approaches.

3.4.5 Classification Model

In our machine learning framework, we address the challenges of using real-world PMU data through data preprocessing, fine-grained event extraction, and feature engineering. The proposed machine learning framework can significantly reduce the dimensionality of the training data to a few number of good features, which can facilitate the use of many off-the-shelf lightweight models. To train an event classification model, we use the constructed features as input and the corresponding event type in the event logs as label. To find the best model, we train different machine learning models and compare their performance. In our experiments (see Section 4.4), we examine the performance of many off-the-shelf models, in which the Random Forest model performs best.

3.4.6 Event Classification Using Real-world PMU Data

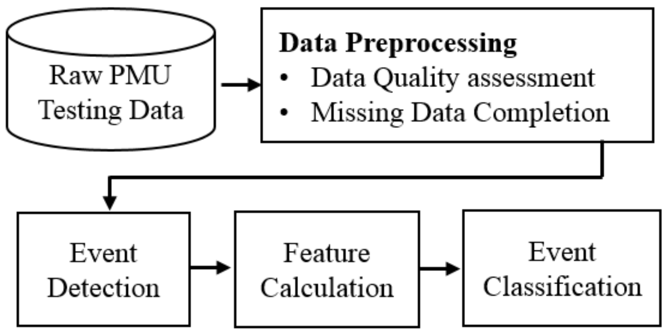


Figure 3.10: The workflow for event classification using real-world PMU data.

To apply the trained event classifier in practice, we still need to deal with the incomplete and noisy PMU measurements, which are streaming into the system in real time. Based on the proposed machine learning framework, we develop a workflow for event classification using real-world PMU data, as illustrated in Fig. 3.10. Specifically, the raw PMU data are streaming into the system. The data preprocessing (i.e., data quality assessment and missing data completion) developed in Section 3.4.2 is first applied on the raw PMU data to fix the bad and missing data. Then, the event detection is used to determine whether there is an event. If an event occurs, the PMU data in the small window W will be used to calculate the features based on Section 3.4.4. The calculated features will be input into the event classifier to determine the event type.

3.5 Experimental Evaluation

3.5.1 Experimental Setup

3.5.1.1 Data

The data used in case studies are described in Section 3.3.1. For the 2-year PMU data, we randomly split the PMU data in each month into 80% for training and 20% for testing, so that the events across the entire year are fairly distributed between the training set and the testing set. The event types provided in the event logs are used as the labels for classification, except the line trip and the oscillation events, because the recorded line trip cannot be determined as faults based on the current high-level description of the event logs and oscillation events can be detected/classified by many existing approaches (e.g., [34, 37]), which can be easily integrated into the proposed machine learning framework. Similar to the related work [62], this paper focuses on the classification of line outages, transformer outages, and frequency events.

3.5.1.2 Evaluation Metrics

Four metrics are used to evaluate the classification performance, i.e., accuracy (ACC), precision (PRE), recall (REC), and F1 score, which are defined as follows:

$$\begin{aligned}
\text{ACC} &= (\text{TP+FN})/(\text{TP+TN+FP+FN}),\\
\text{PRE} &= \text{TP}/(\text{TP} + \text{FP}),\\
\text{REC} &= \text{TP}/(\text{TP} + \text{FN}),\\
\text{F1} &= 2 \times (\text{PRE} \times \text{REC})/(\text{PRE} + \text{REC}),
\end{aligned}$$

where TP (i.e., True Positive) and TN (i.e., True Negative) denote the number of positive and negative instances that are correctly classified, respectively. FP (i.e., False Positive) and FN (i.e., False Negative) denote the number of misclassified negative and positive instances, respectively. The precision represents how good the proposed model is at excluding false alarms. The recall represents how good the proposed model is at not missing true events.

3.5.1.3 Parameter Tuning

In the proposed machine learning framework, there are two key parameters, i.e., the number of measurements w for event detection and the window size W for feature calculation. Based on Algorithm 3, we find the best w is 120 (detection accuracy: 99.3%), which is used throughout the experiments. To show the impact of w on the event detection, Fig. 3.11 illustrates the event detection accuracy obtained under different w.

For the window size W, we optimize W based on the classification accuracy. In Fig.

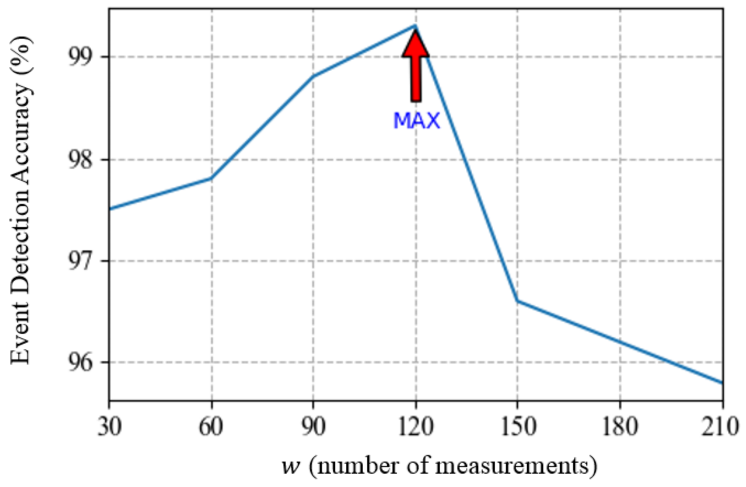


Figure 3.11: Event detection accuracy versus w.

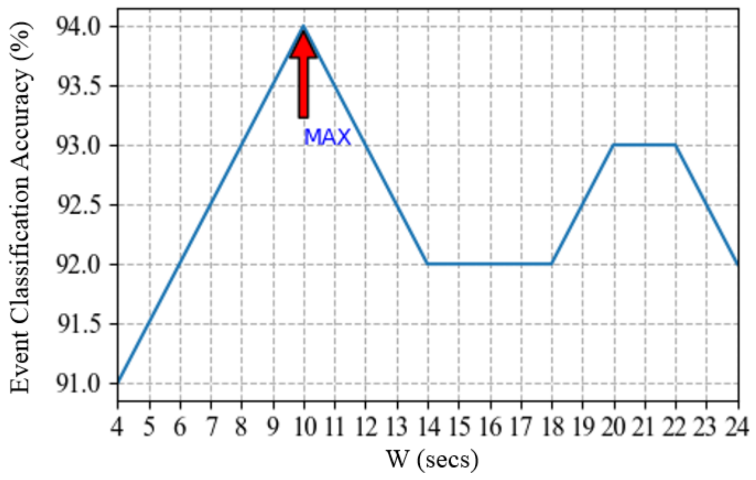


Figure 3.12: Event classification accuracy versus W.

3.12, we evaluate the classification accuracy under different W and find the best accuracy is achieved when W is 10 seconds, where the Random Forest model is used.

3.5.1.4 Benchmark

We evaluate the performance of different classification models under the proposed machine learning framework:

- The Random Forest (RF) model,
- The Gradient Boosting Decision Tree (GBDT) model,
- The K-Nearest Neighbor (KNN) model,
- The Logistic Regression (LR) model,
- The Support Vector Machine (SVM) model,
- The Decision Tree (DT) model.

We use the radial basis function (RBF) as the kernel of SVM. We fine-tune the hyperparameters of all these models via grid search. We also compare the performance of these models with neural network models, i.e., the Long Short-Term Memory (LSTM) model and the Convolutional Neural Networks (CNN) model, where the features are automatically generated based on the input signals. Specifically, for LSTM and CNN, 3-minute measurements of each event (1 minute before and 2 minutes after the start timestamp of a detected event) are used as input to train the classifiers. When training these models, 5-fold cross-validation is used. The trained models are then evaluated using the testing dataset.

3.5.2 Classification Results

Table 3.1
EVENT CLASSIFICATION PERFORMANCE UNDER DIFFERENT MODELS.

Model	ACC(%)	PRE(%)	REC(%)	F1(%)
RF	**94**	**95**	**88**	**91**
GBDT	93	94	**88**	**91**
KNN	85	78	63	66
LR	93	81	66	69
DT	81	78	81	79
SVM	92	86	74	79
CNN	69	23	33	27
LSTM	69	23	33	27

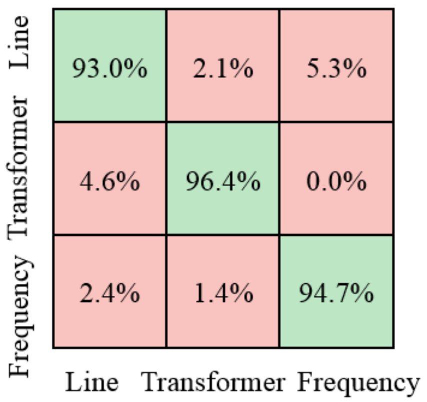


Figure 3.13: Confusion matrix of the RF model for the testing data.

Table 3.1 compares the performance of different classification models using the testing data. The performance of the RF model outperforms the other models in all evaluation metrics. From Table 3.1, we can observe that the performance of non-neural network models (i.e., RF, GBDT, KNN, LR, SVM, DT) trained using the proposed machine learning framework is better than the neural network models (i.e., CNN, LSTM) trained directly using the PMU data. For the non-neural network models, the performance of RF and GBDT are close, and both performs much better than KNN, LR, SVM, and DT. Compared to KNN, LR, SVM, and DT, RF and GBDT are ensemble methods, which consist of a pool of trees that can better capture the features of different types of events and deal with the imbalanced training data.

Fig. 3.13 illustrates the confusion matrix of the RF model based on the testing data, where the rows represent the estimated event type, the columns represent the true event type, and the value of each cell represents the classification accuracy of the corresponding event type. The diagonal and off-diagonal cells in Fig. 3.13 represent the events that are correctly and incorrectly classified, respectively. From Fig. 3.13, it is observed that most of the events (i.e., 94% on average) can be classified correctly. For the misclassified events, 4.6% of line outages are misclassified as transformer outages, and 5.3% of frequency events are misclassified as line outages. By analyzing these misclassified events, we find that the reasons for misclassifying these events are 1) the patterns of some line and transformer outages are similar and 2) the patterns in the ROCOF signal of some frequency events are buried by noise due to the low SNR.

For the neural network models, the classification results in Table 3.1 agree with the findings in [50, 62] that directly using the PMU data to train neural networks cannot achieve good classification results. This is because 1) directly applying such imbalanced data can cause severe overfitting problems; 2) the number of labeled events is only a few thousands, which is not enough for training a good neural network; and 3) neural network based automatic feature generation using the PMU data cannot well capture the event patterns. Hence, the performance of off-the-shelf neural network models directly trained using such a dataset can be significantly poor.

Table 3.2
TRAINING TIME OF DIFFERENT NON-NEURAL NETWORK MODELS.

Model	RF	GBDT	KNN	LR	DT	SVM
Training time(secs)	1.59	25.39	0.01	0.37	0.21	3.48

Table 4.4 compares the training time of different non-neural network models in a server with dual-sockets Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz and 64 GB of memory. The training of all these models takes only a few seconds, compared with hours of training time for neural networks reported in [50, 62]. Specifically, the RF model, which achieves the best classification performance, takes only 1.59 seconds for training. In comparison, the training time of the GBDT model is about 25 seconds. These results show that the proposed machine learning framework provides a promising ways to train good event classifiers with good interpretability and low training cost.

3.5.3 Robustness Analysis

Event classifiers trained under the proposed machine learning framework are robust against the low-quality PMU data (e.g., bad and missing data described in Section 3.4.2). To evaluate the robustness, we compare the classification performance under different data missing rates in the testing dataset. Specifically, the PMU data are assumed to be missing randomly with a probability. The missing data will be first recovered in our framework using our regularized tensor completion approach [19]. Then, the recovered PMU data will be used to calculate the features for event classification.

Table 3.3
EVENT CLASSIFICATION PERFORMANCE OF THE RF MODEL UNDER DIFFERENT MISSING RATES

Missing Rate(%)	ACC(%)	PRE(%)	REC(%)	F1(%)
10	94	94	88	91
20	87	93	87	85
30	87	89	63	69
40	88	91	65	71
50	87	91	64	69

Table 3.3 compares the classification performance using the RF model under different missing rates. It is observed that the classification performance under 10% missing rate is almost the same as the results in Table 3.1. As the missing rate increases, the classification performance slightly drops but still remains at a high level, while the classification performance of the neural network model in [62] drops significantly. This is because the proposed machine learning framework leverages the features constructed based on the event patterns and does not depend on the specific PMU measurements for event classification. Therefore, even with high missing rates, our approach can still perform well as long as the patterns are preserved.

3.6 Conclusion

We have developed a machine learning framework for training robust event classifiers to enhance the situational awareness of power systems using imperfect PMU data. Our framework has addressed the challenges of using real-world PMU data, such as incomplete and noisy PMU measurements, inaccurate event timestamps in the event logs, and insufficient and imbalanced training data. One salient merit of the proposed framework is that large-scale real-world PMU data is reduced to a small set of event features, which can be used to efficiently train many off-the-shelf lightweight machine learning models. As the features are constructed based on the event patterns, the contribution of any single PMU measurement to the features is low, which can effectively mitigate the impact of bad and missing data and improve the robustness of the event classifiers. Numerical experiments using the real-world dataset from the Western Interconnection of the U.S power transmission grid show that the event classifiers trained under the proposed framework can achieve high classification accuracy while being robust against low-quality data. Experimental results show the proposed approach is a promising way to enhance situational awareness with good interpretability and low training cost.

CHAPTER 4
AUTOMATIC LABELING REAL-WORLD PMU DATA

4.1 Introduction

As there are more than 2500 phasor measurement units (PMUs) deployed in North America with high sampling rates (e.g., 30 or 60 samples per second in the U.S.), it provides golden opportunities to achieve high level of situational awareness, but it also trigger the alarm that makes the event labeling works prohibitively costly. Because it requires significant efforts from domain experts to maintain the event logs (i.e., labels) and even hand-label the events. And high volume data makes the event labeling works impossible (e.g., 100 PMUs @ 60Hz sampling rate incessantly generate 600GB raw data in 1 day). On the other hand, the real-world data is imperfect (e.g., bad data and missing data). Therefore, one fundamental open question is how to deal with automatic labeling work using imperfect real-world PMU data.

In this chapter, we focus on events auto labeling. Basically, we consider the scenario with very limited labeled data.

4.2 Related Works

One key task for the automatic data labeling is to build a better event classifier, however, it usually requires sufficient labeled PMU data.

To address the challenge of insufficient labeled PMU data, some studies leverage synthetic data by simulation [30] or Generative Adversarial Neural Networks [63] to train event classification models. Although these approaches can increase the number of labeled data for training event classifiers, the event and grid characteristics hidden in the real-world PMU data can hardly be represented by synthetic data. Thus, the generalization of event classifiers trained using synthetic data can be poor.

In order to leverage the large amount of unlabeled PMU data, semi-supervised learning (SSL) based methods (e.g., self-training [13, 25], hidden structure semi-supervised machine [64], and adversarial SSL [18]) have been proposed, which leverage both labeled and unlabeled data. The idea of SSL is to use limited labeled data to guide the labeling process for the unlabeled data, which can be used for training event classifiers. However, these SSL based methods rely on well-developed and non-customizable models to label the unlabeled data, and the performance of these models largely depends on the amount of available labeled data. With scarce labeled data, the performance of SSL based methods could be poor. Moreover, the SSL based methods cannot easily incorporate the domain knowledge on different types of events, which can provide useful information for event classification.

Different unsupervised learning based labeling approaches have been proposed to deal with no labeled data scenario, e.g., clustering techniques [15], low-rank techniques [27], PCA based techniques [9, 26], and convolutive dictionary based techniques [48]. One key challenge is how to determine the correct number of clusters (i.e., the number of event types). As no labeled PMU data are utilized, it is challenging for the unsupervised learning approaches to accurately capture the features of different types of events. Thus, it is highly possible that different types of events may be identified as the same type or normal operations could be considered as events, which would introduce modeling error in auto labeling results.

To address these challenges, we proposed auto labeling framework both limited labeled data and no labeled PMU data, which can use noisy and low-quality PMU data [66]. For limited labeled data scenario, we use Snorkel [46], an open-source system for quickly assembling training data through weak supervision, to employ the central principles of the data programming paradigm [47], in which developers create labeling functions to label the data and employ supervised learning techniques to assess the accuracy of these labeling functions. For no labeled data scenario, we use K-means algorithm for different labeling function learning, then we use the Snorkel to refine the estimated labels. Compare to the recent framework in 2 scenarios, our method can easily incorporate the domain knowledge and potentially use low-quality unlabeled PMU data to achieve auto labeling task.

4.2.1 Main Contributions

We develop an auto labeling framework for both limited labeled and no labeled PMU data using imperfect real-world PMU data. The key idea is to estimate the labels of large amounts of unlabeled PMU data.

Specifically, for a limited labeled data scenario, we first learn a series of labeling functions based on the knowledge of different types of events, which can generate initial estimates of the labels. As these labeling functions are learnt using the same dataset with scarce labels, the estimated labels are often correlated and the estimates can be noisy and biased. Directly using such estimates cannot generate good event classifiers. To enhance the accuracy of the estimated labels, a generative model(i.e., snorkel and T-cherry junction tree) is developed to characterize the correlations among the estimated labels, based on which the estimated labels from labeling functions are combined in a probabilistic manner to generate the "true" labels. Then, the refined labels from the generative model are used to train event classifiers. It is worth noting that using such an approach, less efforts are required from domain experts to maintain the event logs for building event classifiers; by examining the labeling performances, domain experts can further enhance the classification models. The findings in our work can shed the light on using imperfect real-world PMU data with scarce labels for event classification.

Using the two-year real-world PMU data from the Western Interconnection of the continental U.S. transmission grid, we evaluate the performance of the proposed auto labeling framework. The experimental results show that in limited labeled data scenario (i.e., only 5% labeled data), the average accuracy of the estimated labels using our approach can be around 70.9%, 73.8% and the average accuracy of the corresponding event classifier can achieve about 78.4%, 81.2% for Snorkel and T-cherry junction tree respectively.

This shows a promising way for auto labeling under extremely insufficient labeled data.

4.3 Auto Labeling by Limited Labeled Data

4.3.1 Learning Framework

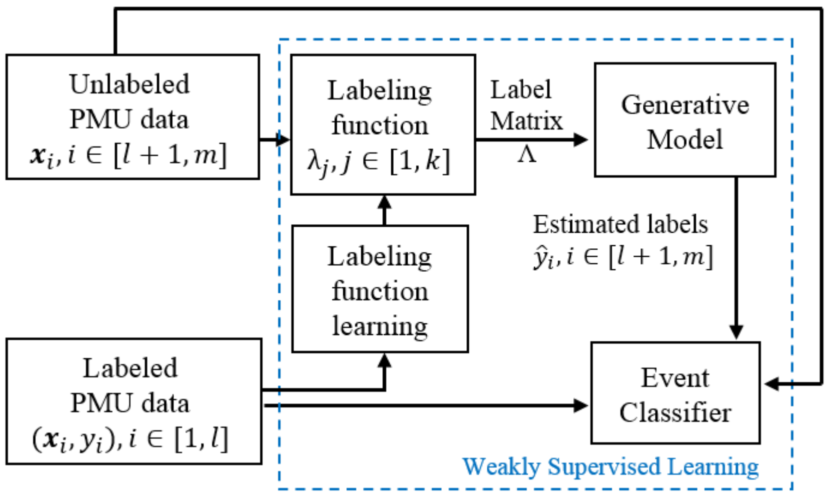

Unlabeled PMU data $x_i, i \in [l+1, m]$
Labeling function $\lambda_j, j \in [1, k]$
Label Matrix Λ
Generative Model
Labeling function learning
Estimated labels $\hat{y}_i, i \in [l+1, m]$
Labeled PMU data $(x_i, y_i), i \in [1, l]$
Event Classifier
Weakly Supervised Learning

Figure 4.1: Learning framework of limited label data.

We propose a learning framework for limited labeled data(see Fig. 4.1) to estimate the labels of large amounts of unlabeled PMU data for training event classifiers. As the raw PMU data are often of low quality (e.g., bad data, dropouts, and timestamp errors), we first fix the data quality issues before using the proposed framework to estimate the labels, where the data preprocessing proposed in our recent work [33] is carried out and construct event features from the PMU data. Then, we estimate the labels of unlabeled PMU data in two main steps: labeling function learning and generative model based label estimation.

- **Labeling function learning.** Labeling functions (LFs) can be treated as event classifiers. Given the input PMU data, a LF outputs a label (i.e., event type), e.g., line outage, transformer outage, or frequency event. In the proposed framework, multiple LFs are learnt to characterize the features of different event types. Due to the limited labeled PMU data, the learnt LFs can be noisy and biased. Therefore, directly using the estimates from the LFs cannot generate good event classifiers.

- **Generative model.** To enhance the accuracy of the estimated labels from the LFs, a generative model is developed. As LFs are learnt using the same dataset, the estimated labels from the LFs are correlated. The goal of the generative model is to characterize the dependency structure of the LFs, based on which we can better combine the estimated labels from the LFs to generate better estimates without knowing the ground truth.

Using the estimated labels from the generative model together with the limited labeled data, we train event classifiers, where different off-the-shelf machine learning models (e.g., random forest) can be used. In the following, we discuss the details of each main step in the proposed framework.

4.3.2 Labeling Function Learning

From our previous work [33], we find that the patterns of the PMU measurement signals under different event types are distinct, as illustrated in Fig. 4.2. For example, frequency events have a deep and wide V-shape; line outages have a narrow spark; and transformer outages are similar to a step function. Based on these event characteristics, we construct multiple event features F_i^s for each measurement signal s (see Fig. 4.2) to train event classifiers, which are more robust and accurate than event classifiers trained using the PMU measurements directly. Therefore, we learn multiple LFs based on these domain specific knowledge of different types of events.

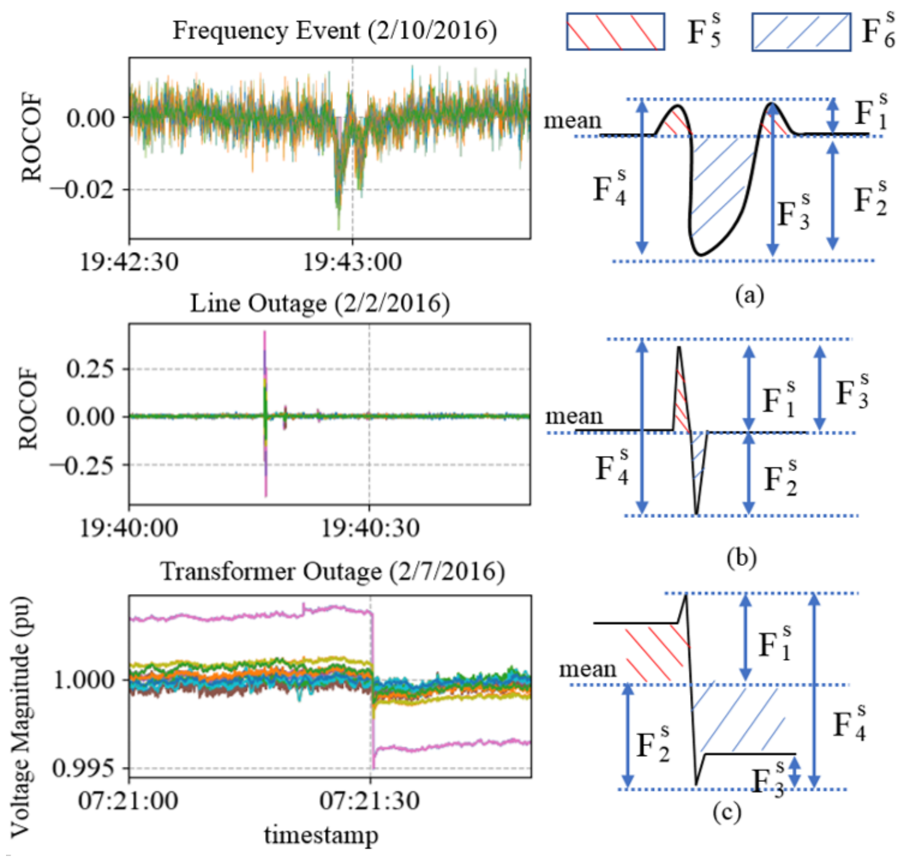


Figure 4.2: Illustration of constructed features for different event types.

Let $C = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l), \boldsymbol{x}_{l+1}, \boldsymbol{x}_{l+2}, ..., \boldsymbol{x}_m\}$ denote the training dataset with m training samples, where $\boldsymbol{x}_i$ denotes the set of constructed features F^s_i of the training sample (event) based on our previous work [33]. In the training dataset C, only l samples are labeled, where y_i denotes the event label for $i = 1, ..., l$. In practice, the number of labeled data is much smaller than the total number of data, i.e., $l \ll m$.

Using these l labeled samples, we train multiple LFs using different machine learning models, e.g., Random Forest (RF), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Gradient Boosting Decision Tree (GBDT). Specifically, we use event features constructed from different measurement signals (including voltage magnitude, current magnitude, and rate of change of frequency (ROCOF)) and their combinations to train different machine learning models. Table 4.1 shows the possible LFs that we can build using different features and models, where each LF is denoted as λ_j.

Table 4.1
LFs BUILT BY DIFFERENT FEATURES AND CLASSIFIERS

FeatureClassifier	RF	LR	KNN	GBDT
Voltage magnitude	λ_1	λ_2	λ_3	λ_4
Current magnitude	λ_5	λ_6	λ_7	λ_8
Rocof	λ_9	λ_{10}	λ_{11}	λ_{12}

Suppose k LFs (i.e., $\{\lambda_j, j = 1, 2, ..., k\}$) are learnt using the l labeled samples. The output of each LF, i.e., $\lambda_j(\boldsymbol{x}_i)$ for $i = l + 1, ..., m$, is the estimate of the event label of the unlabeled data. Let Λ be the label matrix generated by the k LFs for the unlabeled data, where $\Lambda_{i,j} = \lambda_j(\boldsymbol{x}_i)$ for $i \in [l + 1, m]$ and $j \in [1, k]$. For a given unlabeled data $\boldsymbol{x}_i$, the outputs from k LFs (i.e., $\Lambda_{i,j}$ for $j \in [1, k]$) can be treated as noisy votes. From these noisy votes, we aim to estimate the "true" label without knowing the ground truth.

As LFs are learnt using the same dataset with scarce labels, the estimated labels (i.e., $\Lambda_{i,j}$) are often correlated and the estimates can be noisy and biased. Simply using the majority vote to combine $\Lambda_{i,j}$ for $j \in [1, k]$ cannot obtain good estimate, as the majority vote assumes that the LFs are independent. To enhance the accuracy of the estimated labels, it is of paramount importance to characterize the correlations among the LFs, based on which $\Lambda_{i,j}$ for $j \in [1, k]$ can be better combined.

4.3.3 Generative Model

4.3.3.1 Snorkel

To better estimate the true labels Y using Λ, we assume the outcome of different labeling function can form a conditionally independent graph. Therefor, a conditionally independent model is leveraged, where the output of each LF, i.e., $\Lambda_{i,j}$, is modeled as a random variable. Using the observation matrix $\bar{\Lambda}$ (i.e., the outputs of Λ using the unlabeled data), we estimate the joint distribution $\text{Prob}(\Lambda, Y)$ of this generative model. $\text{Prob}(\Lambda, Y)$ will be used to estimate the accuracy of each LF, based on which we can better combine Λ to estimate Y. Specifically, the joint distribution can be specified as:

$$\text{Prob}(\Lambda, Y) \propto \exp\left(\sum_{i=l+1}^{m} \sum_{d \in \mathcal{D}} \theta_d \phi_d(\Lambda_i, y_i)\right), \tag{4.1}$$

where $\Lambda_i = (\Lambda_{i,1}, ...\Lambda_{i,k})$ denotes the vector of the estimated labels from k LFs for the training sample i. $\mathcal{D}$ is a set of tuples that describe the dependencies between LFs, denoted by $\phi_d(\Lambda_i, y_i)$, and θ_d denotes how correlated the LFs are. For example, if d refers to the dependency between two LFs j_1 and j_2, we have $\phi_d(\Lambda_i, y_i) = \phi_{j_1,j_2}(\Lambda_i, y_i) = 1\{\Lambda_{i,j_1}, \Lambda_{i,j_2}\}$, where the indication function $1\{\Lambda_{i,j_1}, \Lambda_{i,j_2}\}$ represents whether LFs j_1 and j_2 depend on each other.

As Y is unknown for the unlabeled data, we therefore estimate the distribution $\text{Prob}(\Lambda)$. As the number of possible dependencies increases at least quadratically with the number of LFs, we leverage the approach in [4] to efficiently learn the dependencies by changing the optimization objective to the log marginal pseudolikelihood of output of a single LF λ_j conditioned on the outputs of the other LFs $\lambda_{\setminus j}$ using L_1 regularization, i.e.,

$$\begin{aligned} &\underset{\theta=\{\theta_d, \forall d \in \mathcal{D}\}}{\arg\min}\, (-\log\ \text{Prob}(\bar{\Lambda}_j | \bar{\Lambda}_{\setminus j}) + \epsilon \|\theta\|_1) \\ &= \underset{\theta}{\arg\min}(-\sum_{i=l+1}^{m} \log \sum_{y_i} \text{Prob}(\bar{\Lambda}_{i,j}, y_i | \bar{\Lambda}_{i \setminus j}) + \epsilon \|\theta\|_1), \end{aligned} \tag{4.2}$$

where $\epsilon > 0$ is a hyperparameter. This problem can be solved efficiently using the stochastic gradient descent method [4].

After solving (4.2), we select the dependencies based on θ that have a sufficiently large magnitude to estimate Prob(Λ, Y). For each unlabeled sample i, the estimated label $\hat{y}_i$ can be obtained by $\hat{y}_i = \arg\max_{y_i} \text{Prob}(y_i|\bar{\Lambda}_i)$ for $i = l + 1, ..., m$.

4.3.3.2 T-Cherry Junction Tree

T-cherry junction tree is another way to determine the joint distribution. The reason why we want to use the T-cherry junction tree because Snorkel is one solution and it do not necessarily be the optimal approximation; it has been proved [54] that such T-cherry junction tree gives the best possible approximation (in the sense of Kullback-Leibler) of all approximations. In other words, the best possible approximation is contained is the junction tree sets, therefore, it might be able to get more accurate estimated labels by this measure. Then, we can build a Bayesian Network [23] to train such a classifier.

We will discuss how we introduce the T-cherry junction tree in our task.

We use the same notation as in section 4.3.3.1, we aim to estimate the "true" label without knowing the ground truth.

As $\Lambda_i = (\Lambda_{i,1}, ..., \Lambda_{i,k})$ denotes the vector of the estimated labels from k LFs for the training sample i. Our goal is to find joint distribution of $Prob(\Lambda_1, ..., \Lambda_k)$. But to solve it directly generally lead to a large overhead. So we switch to calculate the marginal distributions by using a small subset from $\Lambda_1, ..., \Lambda_k$. This is also known as junction tree.

We define a junction tree as a tree structure over different LF with each node is the outcome of the LF. It contains cluster and separator [44]:

- Each of the node of the junction tree is subset of the outcome of the LF, also known as a cluster denoted as Λ_C with the distribution $Prob(\Lambda_C)$.
- A separator is a edge that contains two clusters, it can be denoted as $\Lambda_S = \Lambda_{C1} \cap \Lambda_{C2}$. The separator is also a subset of the outcome of the LF and its distribution can be defined as $Prob(\Lambda_S)$.
- The union of all clusters is the entire set: $\Lambda_1, ...\Lambda_k$

So the joint distribution can be as:

$$Prob(\Lambda_1, ..., \Lambda_k) = \frac{\prod_{C \in \mathcal{C}} Prob(\Lambda_C)}{\prod_{S \in \mathcal{S}} (Prob(\Lambda_S))^{v_S - 1}}, \tag{4.3}$$

where $\mathcal{C}$ is the set of cliques and $\mathcal{S}$ is the set of separators of the junction tree. v_S is the number of those clusters which contain all of the variables involved in separator $\mathcal{S}$. An example of 4 nodes corresponding to the outcome of 4 LFs is shown in Fig 4.3, so the estimation of the joint distribution junction is:

$$Prob(\Lambda_2, \Lambda_3, \Lambda_4; Y) = \frac{Prob(\Lambda_2, \Lambda_3, Y) Prob(\Lambda_3, Y, \Lambda_4)}{Prob(\Lambda_3, Y)}$$

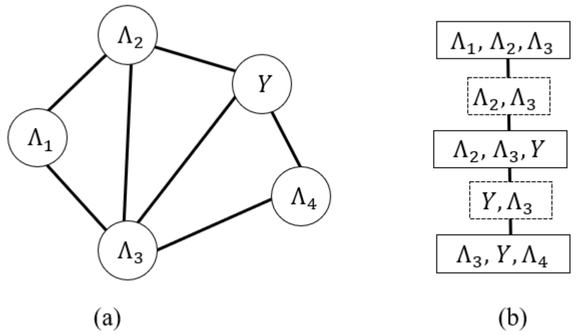


Figure 4.3: Illustration of Junction Tree:(a) triangulated graph (b) the corresponding junction tree.

However, we can find lots of combination to form junction trees.Our goal is to find the minimum difference between approximation and the actual distribution.

We denote the entropy of such vector as $H(\Lambda)$. Let the information content be defined as:

$$I(\Lambda_i) = \sum_{j=1}^{k} prob(\Lambda_{i,j}) log \left(\frac{prob(\Lambda_{i,j})}{prob(\Lambda_{i,1})...prob(\Lambda_{i,k})} \right) \tag{4.4}$$

The optimal solution is achieved, when the Kullback-Leibler divergence between a junction tree approximation and the actual distribution reaches the minimum value [54]. It can be denoted as:

$$KL(prob(\Lambda), prob_{tree}(\Lambda)) = -H(\Lambda) - \sum_{C \in \mathcal{C}} I(\Lambda_C) + \sum_{S \in \mathcal{S}} (v_S - 1) I(\Lambda_S) + \sum_i H(\Lambda_i) \tag{4.5}$$

The first term and last term are not determined by the junction tree. So it is equivalent to maximize the weight W of the junction tree:

$$W = \sum_{C \in \mathcal{C}} I(\Lambda_C) - \sum_{S \in \mathcal{S}} (v_S - 1) I(\Lambda_S) \tag{4.6}$$

But to find the tree that maximizes W is an NP hard problem [55]. T-cherry junction trees are guaranteed to contain the maximum weight junction tree. We define a b-order T-cherry junction tree that satisfies: each cluster contains b variables and each separator contains b-1 variables. Next we want to construct such b-order T-cherry junction trees (i.e., via $b\binom{k}{b}$ in total possible cluster-separator pairs).

We use a Greedy Puzzling Algorithm [55] for searching. Once the searching is done, the outcome of the set of clusters C and the set of separators $\mathcal{S}$. After that we can form a Bayesian Network [23] with the T-cherry tree as a moral graph.

4.4 Experimental Evaluation

4.4.1 Experimental Setup

4.4.1.1 Data

We use real-world PMU data from the Western Interconnection of continental U.S. transmission grid. The measurement data is collected from 23 PMU streams over a two-year period (2016–2017). The sampling rate of PMUs is 60 samples per second. The measurements used in the experiments are voltage magnitude of positive sequence, current magnitude of positive sequence, frequency, and rate of change of frequency (ROCOF). Besides, event logs for the period of two years are provided, where start timestamp, end timestamp, event type, event cause, and event description for each event are provided.

4.4.1.2 Evaluation Metrics

Four metrics are used to evaluate the classification performance, i.e., accuracy (ACC), precision (PRE), recall (REC), and F1 score, which are defined as follows:

$$
\begin{aligned}
\text{ACC} &= (\text{TP}+\text{FN})/(\text{TP}+\text{TN}+\text{FP}+\text{FN}),\\
\text{PRE} &= \text{TP}/(\text{TP} + \text{FP}),\\
\text{REC} &= \text{TP}/(\text{TP} + \text{FN}),\\
\text{F1} &= 2 \times (\text{PRE} \times \text{REC})/(\text{PRE} + \text{REC}),
\end{aligned}
$$

where TP (i.e., True Positive) and TN (i.e., True Negative) denote the number of positive and negative instances that are correctly classified, respectively. FP (i.e., False Positive) and FN (i.e., False Negative) denote the number of misclassified negative and positive instances, respectively.

4.4.2 Performance of Auto Labeling by Limited Labels

To evaluate the learning model, we use 3,877 events in the event logs, including 2,507 line events, 921 transformer events, and 449 frequency events. For these events, we randomly split these events in each month into 80% for training (i.e., 3098 events) and 20% for testing (i.e., 779 events), in order to ensure the events across the entire year are fairly distributed between the training set and the testing set. For the training data, we preserve 5% data as labeled data (i.e., 154 events) and remove the labels for the remaining training data, in order to evaluate the performance of the proposed event classification. We test the event classification (Snorkel) performance over 20 runs, and in each run, the data are randomly selected.

Table 4.2
Performance of event classification(Snorkel) under different models using the testing data.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
RF	78.4±1.0	81.2±3.0	63.8±2.8	63.9±1.3
GBDT	77.5±3.3	77.2 ±7.5	62.6 ±10.2	64.8 ±8.5
SVM	71.1±0.1	35.7±21.4	33.63±0.9	28.3±2.0
LR	71.6±3.4	58.9±26.07	38.4±4.7	36.0±6.8
KNN	74.1±2.0	69.2±6.0	54.7±8.4	56.0±7.9
DT	74.9±3.3	70.6±7.3	62.1±8.4	64.3±6.8

Table 4.2 compares the average performance of different machine learning models over 20 runs, including Random Forest (RF), Gradient Boosting Decision Tree (GBDT), Support Vector Machine (SVM), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Decision Tree (DT). Each model is trained using the training data with the estimated labels obtained using the weakly supervised learning and then tested using the testing data. It is observed that RF outperforms the other models in terms of accuracy, precision, and recall, and F1 score of RF is close to GBDT but with less variation. Thus, RF is used to train the weakly supervised event classifier.

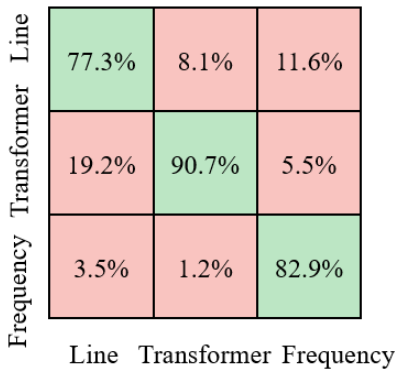


Figure 4.4: Confusion matrix of the weakly supervised RF model for the testing data.

Fig. 4.4 illustrates the confusion matrix of the weakly supervised event classifier using RF, where the rows represent the estimated event type, the columns represent the true event type. The diagonal and off-diagonal cells in Fig. 4.4 represent the events that are correctly and incorrectly classified, respectively. From Fig. 4.4, it is observed that with only 5% labeled data, the proposed weakly supervised event classification can correctly classify the majority of the events (i.e., 78.4% on average). The misclassified events are due to the facts that 1) the patterns of line events and transformer events are similar and 2) the patterns in the ROCOF signal of some frequency events are buried by noise due to low signal-to-noise ratio (SNR). With scarce labeled data, it is challenging for the event classifier to correctly classify these events. We believe domain experts can further enhance the classification performance by examining the classification results, e.g., adding additional LFs using domain knowledge.

4.4.2.1 Comparison with Semi-Supervised Learning

We compare the performance of the proposed event classification Snorkel and T-cherry Junction Tree(order number b =3) with the semi-supervised learning based event classification approaches in [25]. For the semi-supervised learning, we compare the self-training under different models in [25] (i.e., LF, KNN, SVM, DT, Naive Bayes (NB), and the majority vote (Maj) based on these 5 models).

Table 4.3
The accuracy of the estimated labels under different models for the 95% unlabeled training data.

Model	Label Accuracy(%)
T-cherry	73.8±0.3
Snorkel	70.9±0.6
Semi-DT	62.1±1.2
Semi-NB	64.7±0.3
Semi-SVM	64.7±0.1
Semi-LR	64.2±0.3
Semi-KNN	65.7±2.3
Semi-Maj	64.7±0.8

Table 4.3 compares the accuracy of the estimated labels using the proposed weakly supervised learning and the semi-supervised self-training under different models over 20 runs. It is observed that our method outperforms the semi-supervised models. Compared with the semi-supervised models, the proposed weakly supervised learning can easily incorporate the domain knowledge using different LFs, and thereby enhance the accuracy of the estimated labels.

Table 4.4
Testing performance under different models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
Fully supervised	94.3±0.1	95.1±0.3	87.9±0.2	91.0±0.2
T-cherry	81.2±1.2	86.0±2.5	62.3±2.2	68.0±0.6
Snorkel	78.4±1.0	81.2±3.0	63.8±2.8	63.9±1.3
Semi-DT	69.9±1.4	74.3±1.2	56.0±2.5	48.9±2.5
Semi-NB	71.6±0.1	58.2±19.1	34.1±1.3	29.2±6.2
Semi-SVM	71.6±0.1	57.2±0.1	34.0±0.1	29.1±0.1
Semi-LR	71.3±1.5	54.4±25.7	35.2±9.9	31.3±13.8
Semi-KNN	73.3±0.5	79.5±5.9	42.8±10.8	43.6±13.7
Semi-Maj	71.6±0.3	58.9±31.7	34.1±1.4	29.2±2.7

Table 4.4 compares the average performance of the weakly supervised event classification using RF and different semi-supervised approaches. The fully supervised learning case in our recent work [33] is also provided, where all the labels are provided in the training data. From Table 4.4, the proposed weakly supervised event classification outperforms the other semi-supervised models in all metrics by at least 5.1%, 1.7%, 7.8%, and 15% improvement in accuracy, precision, recall and F1 score, respectively.

Compared to the fully supervised case, the weakly supervised event classification can obtain satisfactory results using only 5% labeled training data. Due to the scarce labeled data, the label estimation errors would degrade the performance of the event classification models. In the future, we will explore the use of other LFs developed using domain knowledge to enhance the classification performance.

4.5 Conclusion

We have developed a weakly supervised learning framework for training event classifiers using imperfect real-world PMU data with scarce labels to enhance the situational awareness. One salient merit of the proposed framework is easy to incorporate the domain knowledge by adding labeling functions so that domain experts can further enhance the classification models. Another advantage is that less efforts are required from domain experts to maintain the event logs for building event classifiers. Numerical experiments using the real-world PMU data from the Western Interconnection of the U.S power transmission grid show that the event classifiers trained under the proposed framework when the training data has only 5% labeled data, a satisfactory classification accuracy of 78.4% and 81.2% are still achieved under Snorkel and T-cherry junction tree model. In conclusion, the proposed framework offers a promising way to train event classifiers using scarce and low-quality labeled PMU data.

CHAPTER 5
CONCLUSION AND FUTURE WORK

This dissertation aims to develop Machine Learning (ML) methods and provide fundamental understanding and systematic exploitation of ML for situational awareness using large amounts of imperfect data collected in power systems, in order to improve the reliability and resilience of power systems. Specifically, we address challenges of heterogeneous dynamics, low data quality and insufficient and imbalanced training data in real-world application. The main contributions are summarized below:

- First, we propose Drifting Streaming Peaks-over-Threshold (DSPOT) enhanced self-evolving neural networks based short-term wind farm generation forecast, adaptive machine learning for wind farm generation forecasting, which addresses the challenges of the non-stationarity and the ramp dynamics of wind farm generation and can greatly facilitate the integration of wind generation in reality. Based on the proposed neural networks, both distributional and point forecasts are developed. Experimental results corroborate the superior performance of the proposed approach over other forecast approaches.
- Then, we propose robust event classification using imperfect real-world PMU data, which addresses the challenges of low-quality real-world PMU data, such as incomplete and noisy PMU measurements, inaccurate event timestamps in the event logs, and insufficient and imbalanced training data. One salient merit of the proposed framework is that large-scale real-world PMU data is reduced to a small set of event

features, which can be used to efficiently train many off-the-shelf lightweight machine learning models. As the features are constructed based on the event patterns, the contribution of any single PMU measurement to the features is low, which can effectively mitigate the impact of bad and missing data and improve the robustness of the event classifiers. Numerical experiments using the real-world dataset from the Western Interconnection of the U.S power transmission grid show that the event classifiers trained under the proposed framework can achieve high classification accuracy while being robust against low-quality data. The proposed machine learning framework provides a promising way to train robust event classifiers with good interpretability and low training cost.

- Finally, automatic labeling real-world PMU data is studied. Automatic labeling can significantly reduce the efforts from domain experts to maintain the event logs for developing event classifiers. One salient merit of the proposed framework is that we can easily incorporate the domain knowledge by adding labeling functions so that domain experts can further enhance the classification models. Another advantage is that less efforts are required from domain experts to maintain the event logs for building event classifiers. Numerical experiments using the real-world PMU data from the Western Interconnection of the U.S power transmission grid show that the event classifiers trained under the proposed framework when the training data has only 5% labeled data, a satisfactory classification accuracy of 78.4% using Snorkel and 81.2% using T-cherry junction tree method are achieved. In conclusion, the proposed framework offers a promising way to train event classifiers using scarce and low-quality labeled PMU data.

In the future, we will study PMU data labeling without any labeled PMU data, as it is observed that real-world PMU data from the Texas Interconnection of continental U.S.

transmission grid do not provide any labeled PMU data. Along this line, we will explore unsupervised learning methods. The idea is to first cluster the PMU data using the features extracted from the PMU data. Clearly, these clusters would not be pure such that different types of events may be clustered together under the unsupervised learning methods. Using such noisy clusters, we will then train a series of labeling functions, which would generate noisy labels. Using the generative model, we will further enhance the labeling accuracy. We will explore different unsupervised learning methods and feature extraction methods, in order to achieve the best labeling accuracy.

BIBLIOGRAPHY

[1] Ice storm makes christmas a dark day for tens of thousands. `https://www.cbc.ca/news/canada/ice-storm-means-dark-christmas-for-tens-of-thousands-1.2476164`, 2013.

[2] Synchrophasor technology fact sheet. `https://www.naspi.org/sites/default/files/reference_documents/33.pdf?fileID=1326`, 2014.

[3] Ali Akbar Abdoos. A new intelligent method based on combination of vmd and elm for short term wind power forecasting. *Neurocomputing*, 203:111–120, 2016.

[4] Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*, pages 273–282. PMLR, 2017.

[5] Esther Bommier. Peaks-over-threshold modelling of environmental data, 2014.

[6] Michael Brown, Milan Biswal, Sukumar Brahma, Satish J Ranade, and Huiping Cao. Characterizing and quantifying noise in pmu data. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5, 2016.

[7] Federico Cassola and Massimiliano Burlando. Wind speed and wind energy forecast through kalman filtering of numerical weather prediction model output. *Applied energy*, 99:154–166, 2012.

[8] GW Chang, HJ Lu, YR Chang, and YD Lee. An improved neural network-based approach for short-term wind speed and power forecast. *Renewable energy*, 105:301–311, 2017.

[9] Yang Chen, Le Xie, and Panganamala Ramana Kumar. Power system event classification via dimensionality reduction of synchrophasor data. In *2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 57–60. IEEE, 2014.

[10] Hamed Chitsaz, Nima Amjady, and Hamidreza Zareipour. Wind power forecast using wavelet neural network trained by improved clonal selection algorithm. *Energy conversion and Management*, 89:588–598, 2015.

[11] Karlynn S Cory and Blair G Swezey. Renewable portfolio standards in the states: Balancing goals and implementation strategies. Technical report, National Renewable Energy Lab (NREL), 2007.

[12] Mingjian Cui, Jianhui Wang, Jin Tan, Anthony R Florita, and Yingchen Zhang. A novel event detection method using pmu data with high precision. *IEEE Transactions on Power Systems*, 34(1):454–466, 2018.

[13] Qiushi Cui and Yang Weng. Enhance high impedance fault detection and location accuracy via micro pmus. *IEEE Transactions on Smart Grid*, 11(1):797–809, 2019.

[14] Om P Dahal and Sukumar M Brahma. Preliminary work to classify the disturbance events recorded by phasor measurement units. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–8. IEEE, 2012.

[15] Om P Dahal, Sukumar M Brahma, and Huiping Cao. Comprehensive clustering of disturbance events recorded by phasor measurement units. *IEEE Transactions on Power Delivery*, 29(3):1390–1397, 2013.

[16] Daizong Ding, Mi Zhang, Xudong Pan, Min Yang, and Xiangnan He. Modeling extreme events in time series prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1114–1122, 2019.

[17] Gao-feng Fan, Wei-sheng Wang, Chun Liu, and Hui-zhu DAI. Wind power prediction based on artificial neural network. *Proceedings of the CSEE*, 28(34):118–123, 2008.

[18] Maryam Farajzadeh-Zanjani, Ehsan Hallaji, Roozbeh Razavi-Far, Mehrdad Saif, and Masood Parvania. Adversarial semi-supervised learning for diagnosing faults and attacks in power grids. *IEEE Transactions on Smart Grid*, 2021.

[19] Amir Ghasemkhani, Iman Niazazari, Yunchuan Liu, Hanif Livani, Virgilio A Centeno, and Lei Yang. A regularized tensor completion approach for pmu data recovery. *IEEE Transactions on Smart Grid*, 12(2):1519–1528, 2020.

[20] Scott D Grimshaw. Computing maximum likelihood estimates for the generalized pareto distribution. *Technometrics*, 35(2):185–191, 1993.

[21] Yan Hao and Chengshi Tian. A novel two-stage forecasting model based on error factor and ensemble method for multi-step wind power forecasting. *Applied energy*, 238:368–383, 2019.

[22] Miao He, Lei Yang, Junshan Zhang, and Vijay Vittal. A spatio-temporal analysis approach for short-term forecast of wind farm generation. *IEEE Transactions on Power Systems*, 29(4):1611–1622, 2014.

[23] Paul Helman, Robert Veroff, Susan R Atlas, and Cheryl Willman. A bayesian network classification methodology for gene expression data. *Journal of computational biology*, 11(4):581–615, 2004.

[24] Bruno P Leao, Dmitriy Fradkin, Yubo Wang, and Sindhu Suresh. Big data processing for power grid event detection. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4128–4136. IEEE, 2020.

[25] Haoran Li, Yang Weng, Evangelos Farantatos, and Mahendra Patel. A hybrid machine learning framework for enhancing pmu-based event identification with limited labels. In *2019 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, pages 1–8. IEEE, 2019.

[26] Haoran Li, Yang Weng, Evangelos Farantatos, and Mahendra Patel. An unsupervised learning framework for event detection, type identification and localization using pmus without any historical labels. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2019.

[27] Wenting Li, Meng Wang, and Joe H Chow. Fast event identification through subspace characterization of pmu data in power systems. In *2017 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, 2017.

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

[29] Zenan Ling, Robert Caiming Qiu, Xing He, and Lei Chu. A new approach of exploiting self-adjoint matrix polynomials of large random matrices for anomaly detection and fault location. *IEEE Transactions on Big Data*, 2019.

[30] Shengyuan Liu, Yuxuan Zhao, Zhenzhi Lin, Yilu Liu, Yi Ding, Li Yang, and Shimin Yi. 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, 2019.

[31] Yongqian Liu, Jie Shi, Yongping Yang, and Wei-Jen Lee. Short-term wind-power prediction based on wavelet transform–support vector machine and statistic-characteristics analysis. *IEEE Transactions on Industry Applications*, 48(4):1136–1141, 2012.

[32] Yunchuan Liu, Amir Ghasemkhani, Lei Yang, Jun Zhao, Junshan Zhang, and Vijay Vittal. Seasonal self-evolving neural networks based short-term wind farm generation forecast. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE, 2020.

[33] Yunchuan Liu, Lei Yang, Amir Ghasemkhani, Hanif Livani, Virgilio A. Centeno, Pin-Yu Chen, and Junshan Zhang. Robust event classification using imperfect real-world pmu data. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[34] Yue Ma, Qi Huang, Zhenyuan Zhang, and Dongsheng Cai. Application of multisynchrosqueezing transform for subsynchronous oscillation detection using pmu data. *IEEE Transactions on Industry Applications*, 57(3):2006–2013, 2021.

[35] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[36] David J Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.

[37] Seyedbehzad Nabavi, Jianhua Zhang, and Aranya Chakrabortty. Distributed optimization algorithms for wide-area oscillation monitoring in power systems using interregional pmu-pdc architectures. *IEEE Transactions on Smart Grid*, 6(5):2529–2538, 2015.

[38] Sanjay Singh Negi, Nand Kishor, Kjetil Uhlen, and Richa Negi. Event detection and its signal characterization in pmu data stream. *IEEE Transactions on Industrial Informatics*, 13(6):3108–3118, 2017.

[39] Duc Nguyen, Richard Barella, Scott A Wallace, Xinghui Zhao, and Xiaodong Liang. Smart grid line event classification using supervised learning over pmu data streams. In *2015 Sixth International Green and Sustainable Computing Conference (IGSC)*, pages 1–8. IEEE, 2015.

[40] Iman Niazazari, Yunchuan Liu, Amir Ghasenikhani, Shuchismita Biswas, Hanif Livani, Lei Yang, and Virgilio A Centeno. Pmu-data-driven event classification in power transmission grids. In *2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2021.

[41] GJ Osório, JCO Matias, and JPS Catalão. Short-term wind power forecasting using adaptive neuro-fuzzy inference system combined with evolutionary particle swarm optimization, wavelet transform and mutual information. *Renewable Energy*, 75:301–307, 2015.

[42] George Papaefthymiou and Bernd Klockl. Mcmc for wind power simulation. *IEEE transactions on energy conversion*, 23(1):234–240, 2008.

[43] Pierre Pinson and Henrik Madsen. Adaptive modelling and forecasting of offshore wind power fluctuations with markov-switching autoregressive models. *Journal of forecasting*, 31(4):281–313, 2012.

[44] Brian Proulx and Junshan Zhang. Modeling social network relationships via t-cherry junction trees. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2229–2237. IEEE, 2014.

[45] Nicholas J Radcliffe. Genetic set recombination and its application to neural network topology optimisation. *Neural Computing & Applications*, 1(1):67–90, 1993.

[46] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[47] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575, 2016.

[48] Dilan Senaratne, Jinsub Kim, and Eduardo Cotilla-Sanchez. Spatio-temporal frequency domain analysis of pmu data for unsupervised event detection. In *2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2021.

[49] Jie Shi, Brandon Foggo, Xianghao Kong, Yuanbin Cheng, Nanpeng Yu, and Koji Yamashita. Online event detection in synchrophasor data with graph signal processing. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7. IEEE, 2020.

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

[51] Xiaoyu Shi, Xuewen Lei, Qiang Huang, Shengzhi Huang, Kun Ren, and Yuanyuan Hu. Hourly day-ahead wind power prediction using the hybrid model of variational model decomposition and long short-term memory. *Energies*, 11(11):3227, 2018.

[52] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimiza-

tion of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[53] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[54] Tamás Szántai and Edith Kovács. Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. *Annals of Operations Research*, 193(1):71–90, 2012.

[55] Tamás Szántai and Edith Kovács. Discovering a junction tree behind a markov network by a greedy algorithm. *Optimization and Engineering*, 14(4):503–518, 2013.

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

[57] David White, Amy Roschelle, Paul Peterson, David Schlissel, Bruce Biewald, and William Steinhurst. The 2003 blackout: solutions that won't cost a fortune. *The Electricity Journal*, 16(9):43–53, 2003.

[58] Qunli Wu and Chenyang Peng. Wind power generation forecasting using least squares support vector machine combined with ensemble empirical mode decomposition, principal component analysis and a bat algorithm. *Energies*, 9(4):261, 2016.

[59] Le Xie, Yang Chen, and PR Kumar. Dimensionality reduction of synchrophasor data for early event detection: Linearized analysis. *IEEE Transactions on Power Systems*, 29(6):2784–2794, 2014.

[60] L. Yang, M. He, V. Vittal, and J. Zhang. Stochastic optimization-based economic dispatch and interruptible load management with increased wind penetration. *IEEE Transactions on Smart Grid*, 7(2):730–739, 2016.

[61] Lei Yang, Miao He, Junshan Zhang, and Vijay Vittal. Support-vector-machine-enhanced markov model for short-term wind power forecast. *IEEE Transactions on Sustainable Energy*, 6(3):791–799, 2015.

[62] Yuxuan Yuan, Yifei Guo, Kaveh Dehghanpour, Zhaoyu Wang, and Yanchao Wang. Learning-based real-time event identification using rich real pmu data. *IEEE Transactions on Power Systems*, 2021.

[63] Xiangtian Zheng, Bin Wang, Dileep Kalathil, and Le Xie. Generative adversarial

networks-based synthetic pmu data creation for improved event classification. *IEEE Open Access Journal of Power and Energy*, 8:68–76, 2021.

[64] Yuxun Zhou, Reza Arghandeh, and Costas J Spanos. Partial knowledge data-driven event detection for power distribution networks. *IEEE Transactions on Smart Grid*, 9(5):5152–5162, 2017.

[65] Yuxun Zhou, Reza Arghandeh, Han Zou, and Costas J Spanos. Nonparametric event detection in multiple time series for power distribution networks. *IEEE Transactions on Industrial Electronics*, 66(2):1619–1628, 2018.

[66] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

[67] Radoslav Štompf. 7 major challenges of a power grid and their solutions, 2020.