Line Faults Classification Using Machine Learning on Three Phase Voltages Extracted from Large Dataset of PMU Measurements

Hussain Otudi[1], Tatjana Dokic[2], Taif Mohamed[2], Mladen Kezunovic[2], Yi Hu[3], Zoran Obradovic[1]

[1] Computer and Information Sciences Department Temple University Philadelphia, PA, U.S.A

[2] Department of Electrical and Computer Engineering Texas A&M University College Station, TX, U.S.A

[3] Quanta Technology

Abstract

An end-to-end supervised learning method was developed to classify transmission line faults in a two-year field-recorded dataset that includes synchronized measurements of three-phase voltages recorded by 38 phasor measurement units (PMUs) sparsely located in the US Western Grid interconnection. Statistical analysis was performed to extract features from this large dataset to train the support vector machine (SVM), random forest (RF), and extreme gradient boosting (XGBoost) classifiers. The training further leverages a simulated dataset from a synthetic grid with 12 PMUs to increase the number of types of faults infrequently seen in the field-recorded dataset. Training the classification models with the combined dataset resulted in a classification accuracy of 98.58%. This is a significant improvement over 86.87% to 87.17% accuracy obtained by relying on the field-recorded dataset alone.

1. Introduction

In recent years, synchrophasor technology has been complementing legacy supervisory control and data acquisition (SCADA) systems [1]. The benefit of using PMUs is their ability to take synchronized phasor measurements at 30 frames per second (fps) or higher. This allows utility operators to monitor the power system with much higher data resolution when compared to the unsynchronized measurement taken by the SCADA scan every few seconds [1, 2]. The goal of real-time automated detection of short-duration events, such as faults, has become more attainable. However, the high data reporting rate coupled with a steady increase in the number of PMUs deployed in the power grid makes fault classification processing of historical or real-time streaming PMU data increasingly challenging. This greatly increases the interest in applying machine learning (ML) technologies to automatically process and analyze the captured PMU datasets for efficient and accurate fault classification.

Several methods of feature extraction and ML techniques have been studied to manage a large volume of data in the past.

Principal component analysis (PCA) was applied to reduce the dimensionality of the dataset collected from PMUs to enable event detection at early stages [3]. The same method has been used to detect complex cascading events [4]. Minimum volume closing ellipsoid (MVEE) was used as the method of feature extraction [5], followed by the agglomerative hierarchical clustering method for event classification. Other event detection methods include detrended fluctuation analysis [6], fast variant of discrete S-transform [7], and signal energy transform [8], among others [9,10]. The normalized value of the wavelet coefficient energy was used as a feature engineering method to detect events [11]. More recently, dynamic programming-based swinging door ending has been used to precisely pinpoint the start time of events [12]. Some studies also investigated the application of event detection in distribution networks using Micro-PMU data, such as in [13], where the performances of SVM, KNN, and decision tree for event detection were compared. A wavelet transform-based feature engineering method was used to generate inputs for the Convolutional Neural Network classification model [14].

We propose an innovative feature engineering and ML approach to automatically classify different types of faults from historical PMU datasets. The main contribution is in the ability to improve the accuracy of the classification by supplementing sparsely field-recorded PMU data with simulated data in cases where

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000913.

the number of events of certain types is insufficiently observed in the field recordings. Our method is applied to signals that have been detected as events by another model, and we classify an event to a line fault type.

The background of the difficulties in using sparse PMU measurements for fault analysis is discussed in Section 2. Insights into the field-recorded and simulated data used to extract the features are given in Section 3. Section 4 elaborates on how datasets have been utilized. Section 5 introduces the ML classifiers and explains the feature extraction and labeling. The experiments and results for the ML classifiers are presented in Section 6. The conclusions are drawn in Section 7, followed by References.

2. Background

There are 11 types of faults that may occur on a transmission line (three phases of the power system are marked as A, B, C, and G stands for ground): A-G, B-G, C-G, AB, AC, BC, AB-G, BC-G, CA-G, ABC, and ABC-G, which can be generalized as P-G, PP, PP-G, 3P, and 3P-G. To classify the line faults, we extract the three-phase voltages from the PMU measurements. By doing so, we encounter two problems.

Problem 1*: It is difficult to separate PP (or 3P) from PP-G (or 3P-G) faults using field-recorded data.* The voltages may be measured at a distance from the fault location because of the sparsity of PMU locations (less than 5% of buses are covered by PMUs), which makes the voltage signal less distinguishable among different fault types. In addition, noise and system imbalance are adding challenges in differentiating these types of faults.

Problem 2*: There are fewer examples of PP, PP-G, 3P, and 3P-G faults compared to P-G faults in the field-recorded dataset.* Owing to the statistical rarity of occurrence, the PMU recordings have an uneven representation of fault types (phase-to-ground faults are much more frequent than any other type).

We further illustrate ***Problem 1*** in Figures 1 and 2, where the three-phase voltage measurements for phase-to-phase (AB) and phase-to-phase-to-ground (AB-G) faults are compared to the corresponding PMU measurements respectively obtained from simulated faults. The measurements were visualized using

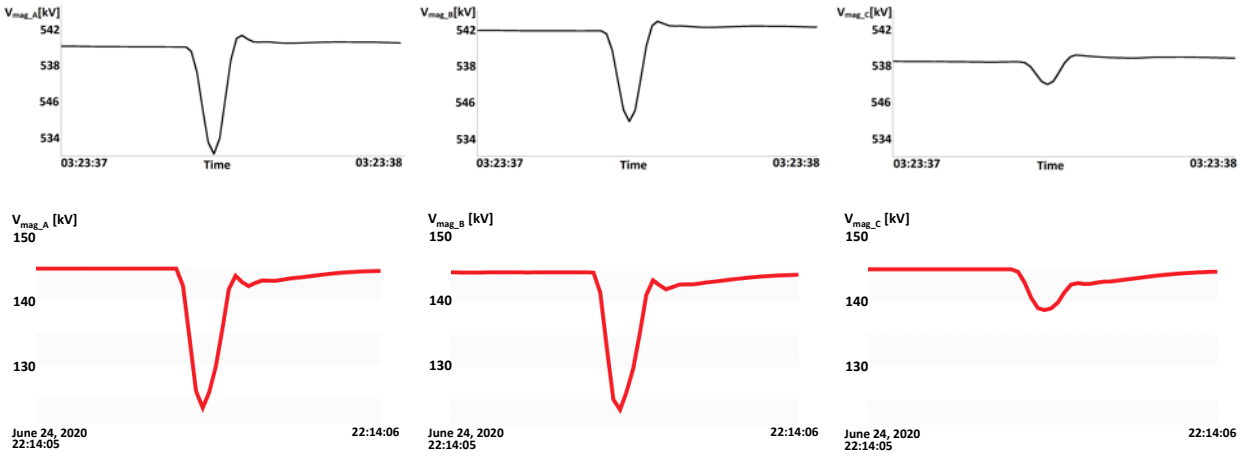

Figure 1. AB fault – Comparison of Field-recorded (top) and simulated (bottom) PMU data (left to right: phases A, B and C)

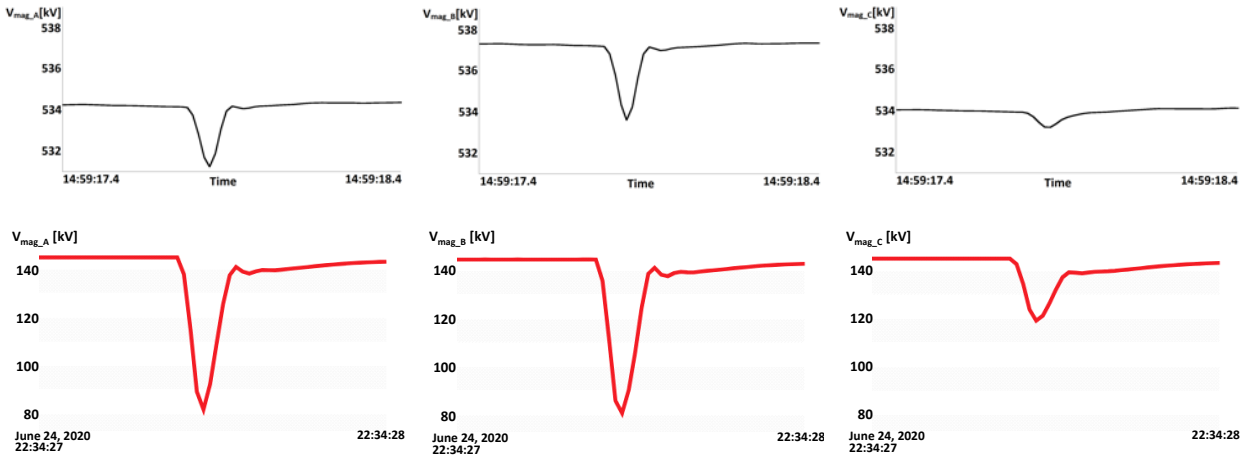

Figure 2. AB-G Fault Events – Comparison of Field-recorded (top) and simulated (bottom) PMU data (left to right: phases A, B and C)

OSIsoft PI Vision software [15]. We make two observations based on Figures 1 and 2:

Observation 1 based on Figure 1: It is not possible to obtain field-recorded fault measurements that match the same event using simulated data because of noise and system imbalance. If we observe Figure 1 with the AB fault, we can see that the simulated example exhibits the same pre-fault voltage level in all three phases, equal voltage drops in phases A and B, and a much smaller voltage drop in phase C compared to phases A and B, which is expected for a perfectly symmetrical system during the AB fault. However, the field-recorded example in Figure 1 demonstrates that there is a difference between both the pre-fault voltage levels and voltage drops between phases A and B during the same type of AB fault.

Observation 2 based on Figures 1 and 2: It is difficult to separate fault AB from AB-G in the field-recorded dataset. The two cases that exhibit the most similarities in field-recorded examples are phase-to-phase and phase-to-phase-to-ground waveforms, as shown in Figures 1 and 2. The bottom three waveforms in each figure were obtained from the simulation. The simulated waveform examples in Figures 1 and 2 demonstrate that it is easy to differentiate fault AB from AB-G because the AB-G fault waveform always has a larger voltage drop in all phases than AB fault. The difference is not clearly detectable in field-recorded signals, as seen in the corresponding top three waveforms in Figures 1 and 2, where it is much more difficult to distinguish between fault AB and AB-G recordings. These two cases (AB and AB-G) need to select very precise thresholds to differentiate them automatically, which becomes challenging considering that in many scenarios, these thresholds might be exceeded due to noise or different proximity of PMUs to the faults.

Problem 2 is illustrated in Figure 3, which shows uneven statistics of different fault types in the field-recorded data collected in the Western Interconnection of the USA for a period of two years. As an example, the fault type P-G has far more recorded cases than the fault types of PP and PP-G combined.

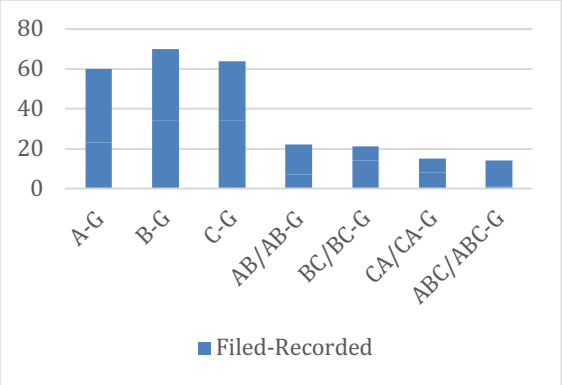

Figure 3. Distribution of labeled faults in field-recorded data

Based on the above observations, our hypothesis is that both problems might be addressed by enhancing training data by carefully selecting PMU recordings of simulated measurements of incorrectly or insufficiently represented fault types in the field-recorded data. The combined field-recorded and simulated PMU measurement data were used to train ML algorithms to validate the hypothesis that the resulting models are more accurate than models trained on field-recorded data alone.

In the next section, we discuss the two sources of data by providing further details.

3. PMU Measurement Data

Field-recorded PMU measurement data provided by electric utilities and simulated data from a synthetic grid were used for this study. After visual inspection of field-recorded current magnitude measurements, it was observed that the current magnitude change was not as prominent as the change observed in the voltage magnitude due to the large distance of PMUs from the fault location for most events in the dataset, so the currents were not used. Both datasets were preprocessed to extract the three-phase measurements of the voltage magnitude. Next, the steps are taken to extract the features based on a statistical analysis of 2-second data windows. The duration of the time window was determined empirically after multiple experiments on the simulated data. This window selection provides high accuracy in terms of line fault-type classification. The simulated data are integrated into the field-recorded data to provide a more balanced training dataset for ML classification algorithms.

3.1. Field-Recorded PMU Data

3.1.1. Data Description. Datasets used include:

- Synchrophasor measurements from 38 PMUs located in the Western Interconnection of the USA were collected over a period of two years. The dataset is anonymized by the provider by removing information about geographical locations and any physical and technological characteristics of the PMUs or the electric grid to which they are connected. The PMUs under study have reported measurements of voltage and current magnitude and angle for each phase, positive sequence voltage and current magnitude and angle, frequency, and rate of change of frequency. For the reasons explained earlier, in this experiment, only the

three-phase voltage magnitudes are used to develop an automatic labeling system according to the phases affected by the fault.

- Historical event logs for a period of two years are stored as a CSV file. The event logs assigned by the data provider have inaccurate timestamps for the event start/stop times because the associated logs came from SCADA and were entered manually, which did not provide sufficient and accurate time-related details. We performed visual inspection to determine the precise start time of each event and create a 2-second time window ensuring that the event is contained within this window. Some of the events have a descriptor field that specifies the phases affected by the event. In some cases, this descriptor was provided in the form of P-G (phase-to-ground) and P-P (phase-to-phase) without naming the phase. In these cases, visual inspection of the three-phase voltage magnitudes was applied to determine the affected phase (A, B, or C).

3.1.2. Data Preparation. While a dataset spanned field-recorded PMU measurements from 43 PMUs over a period of two years (2016-2017), 38 PMUs were selected for feature extraction because of a large fraction of bad data in the remaining five [16]. The data quality issues included missing and duplicate data, an excessive number of outliers, flat 60 Hz frequency recordings, and erroneous time tags. Because the data were stored in the form of Apache Parquet files, Apache Spark [17] was used to retrieve the data. Python [18] was then used to implement the required analysis to prepare the data for the experiment. This stage focuses first on running the raw measurements through calculations to extract six features, as described in Section 4.1. These six features were used to decide which type of fault would be a correct label for an event, and the events were also centered within the 2-second windows to avoid issues with the edges.

3.2. Simulated PMU Data

3.2.1. Data Description. For this research, Quanta Technology's Protection and Control test facility in Raleigh North Carolina was used to create a simulated dataset. The core element of the simulation system was an RTDS NovarCor real-time simulation system with two cores. Twelve PMUs (four actual PMUs, i.e., PMU1 to PMU4, and eight software-emulated PMUs) were placed on a synthetic IEEE 14-Bus Power system (Figure 4) to monitor simulated transmission line fault events. The simulated dataset includes measurements from 1,350 simulated fault events at different locations with multiple combinations of fault resistances, locations, and types. PMU data are streamed to a phasor data concentrator (PDC) (Figure 5) in the IEEE C37.118 protocol, and the data are archived and assembled to create the simulated dataset.

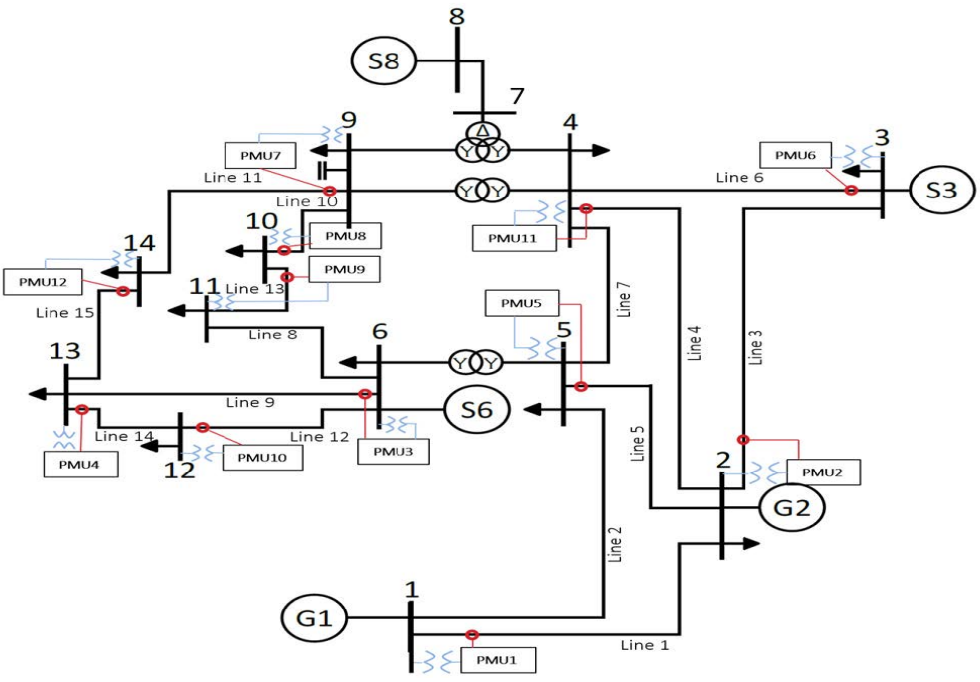

Figure 4. PMU placement in the synthetic IEEE 14-bus power system

The simulated fault events on all lines included also line fault clearing and reclosing sequences. Following the fast fault clearing, automatic line reclosing is simulated for all single-phase-to-ground faults, and manual line switching by human operators was simulated for all multi-phase faults. The simulated dataset has a separate event log file in which information for each simulated fault event (i.e., time, location, resistance, and type) is provided.

3.2.2. Data Preparation. The data in the simulated dataset for each PMU include the magnitude and angle for the positive sequence and three-phase voltage and current synchrophasors, the frequency, and the rate of change of frequency (ROCOF). For the fault classification training, the magnitude of each phase's voltage phasor was extracted for all three phases, namely A, B, and C. Data windowing was performed by splitting the time-series data into consecutive windows of 2 s each.

4.Methodology

In this section, we discuss our proposed method in three subsections: feature extraction, labeling, and integration of field-recorded and simulated data.

4.1. Feature Extraction

After data preparation, six features were extracted based on three-phase voltage measurements: AB_{diff}, BC_{diff}, CA_{diff}, XY_{diff}, YZ_{diff}, and ZX_{diff}. These features were chosen because they characterize the relative voltage drops that occur in each phase owing to a line fault. The features were extracted using the following four steps:

- Determine range of voltage for each PMU
- Aggregate range of voltage for all PMUs
- Calculate difference between each two phases
- Determine ratio of differences between each two phases

Below, each step is discussed in detail.

Step 1- The voltage range is calculated for each of the 38 PMUs (or 12 PMUs in simulations) over a 2-second window, where the minimum voltage measurement is subtracted from the maximum voltage measurement, and then divided by the average magnitude of all data points for each phase Øi, as shown in equation 1.

$$V_{range}(\phi_i) = \frac{\max\left(V_{mag}(\phi_i)\right) - \min\left(V_{mag}(\phi_i)\right)}{Avg(V_{mag}(\phi_i))} \quad (1)$$

where $V_{mag}(\phi_i)$ stands for the voltage magnitude of phase ϕ (A, B, or C) for the i_{th} 2-second window.

Step 2- These ranges are then summed up for all PMUs and divided by the number of PMUs, as in equation 2.

$$SUM\left(V_\phi\right) = \sum_{i=1}^{\text{number of PMUs}} \frac{V_{range}(\phi_i)}{number\ of\ PMUs} \quad (2)$$

Step 3- The difference between each phase AB_{diff},BC_{diff} and CA_{diff} is then calculated, and the signs of these three quantities are considered as the first three features.

$$A_{to}B = SUM\ (V_A) - SUM\ (V_B) \quad (3)$$

$$B_{to}C = SUM\ (V_B) - SUM\ (V_C) \quad (4)$$

$$C_{to}A = SUM\ (V_C) - SUM\ (V_A) \quad (5)$$

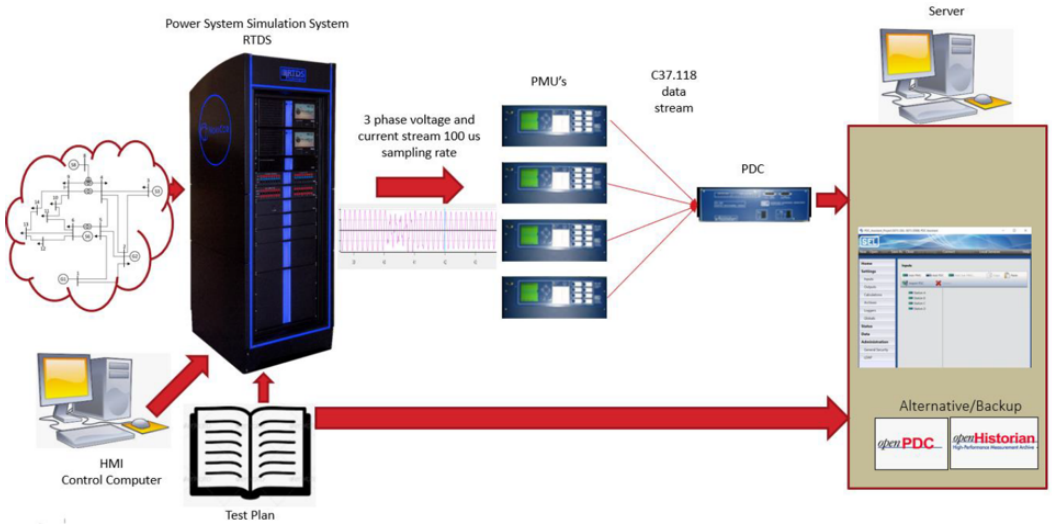

Figure 5. RTDS faults simulation framework

$$AB_{diff} = sign(A_{to}B) \quad (6)$$

$$BC_{diff} = sign(B_{to}C) \quad (7)$$

$$CA_{diff} = sign(C_{to}A) \quad (8)$$

Finally, the ratio between the differences in the voltage range mentioned in the third step is determined such that the larger value is always divided by the smaller value. For example, if abs($A_{to}B$) > abs($B_{to}C$), then $R(AB_{to}BC)$ = abs($A_{to}B$) / abs($B_{to}C$). These ratios are then used to form the remaining three features computed as described in equations (12-14):

$$X = R(AB_{to}BC) \quad (9)$$

$$Y = R(BC_{to}CA) \quad (10)$$

$$Z = R(CA_{to}AB) \quad (11)$$

$$XY_{diff} = sign(X - Y) \quad (12)$$

$$YZ_{diff} = sign(Y - Z) \quad (13)$$

$$ZX_{diff} = sign(Z - X) \quad (14)$$

V_A,Vb and V_c stand for voltage of Phase A, B and C; AB_{diff},BC_{diff} and CA_{diff} stand for the difference between each two phases.
AVG stands for average magnitude of all data points for each phase Øi.
Here, X, Y and Z calculate the ratio between the differences in the voltage by dividing the large value by the small value XY_{diff},YZ_{diff} and ZX_{diff} **,** and then they are compared the difference in ratio between different pair of signals**.**

4.2. Labeling

The Intersecting sub-signal time windows are labeled with an event log provided with the dataset. This step produces a binary label, which indicates whether an event occurred at this sub-signal or if there was a normal operation. The event log also contains a field "Descriptor" that has information on the type of fault that occurred (e.g., A-G, AB, ABC, etc.).

Depending on the values of AB_{diff}, BC_{diff}, CA_{diff}, XY_{diff}, YZ_{diff}, and ZX_{diff} that are extracted as described in Section 4.1, the type of fault can be automatically determined as described below in Table 1. Automatic labeling was then performed for multiclass line faults, as shown in Table 1. Each label represents the occurrence of line fault type. For example, a phase-to-ground fault (i.e., A-G, B-G, or C-G) is a combination of four labels. The labels are assigned as follows: for a phase-to-ground fault, if AB_{diff} = 1 and CA_{diff} = -1 and YZ_{diff} = 1 and ZX_{diff} = -1, then the label will be "A-G." [19].

The other line fault labels are listed in Table 1. The automatic labeling in Table 1 only works well for the separation of the A-G, B-G, and C-G faults. Regarding the PP, PP-G, and 3P faults, the majority of the events were mislabeled. Moreover, the 2-second windows chosen for analysis do not allow for the recognition of faults that might evolve to a different fault within this time window. Rather, the labeling algorithm will only recognize the dominant fault whose effect is greater on the features. Therefore, this automatic labeling system was combined with the machine-learning model described in Section 5.

4.3. Integration of field-recorded and simulated data

Because of the limited number of examples per type of line fault in field-recorded PMU data (Figure 3), as discussed in the background section, simulated data with much more prominent fault types are combined with field-recorded data to generate an integrated training set (Figure 6) aimed at boosting the line fault classification. As shown in Figure 3, some types of faults, such as PP, PP-G, and 3P, are less frequent in the field-recorded data. In the integrated version presented in Figure 6, simulated examples are added

Table 1. Automatic labeling for seven types of line fault

Extracted Features	Type of faults
AB_{diff} = 1 and CA_{diff} = -1 and YZ_{diff} = 1 and ZX_{diff} = -1	"A-G"
AB_{diff} = -1 and BC_{diff} = 1 and XY_{diff} = -1 and ZX_{diff} = 1	"B-G"
BC_{diff} = -1 and CA_{diff} = 1 and XY_{diff} = 1 and YZ_{diff} = -1	"C-G"
BC_{diff} = 1 and CA_{diff} = -1 and XY_{diff} = 1 and YZ_{diff} = -1	"AB/AB-G"
AB_{diff} = -1 and CA_{diff} = 1 and YZ_{diff} = 1 and ZX_{diff} = -1)	"BC/BC-G"
AB_{diff} = 1 and BC_{diff} = -1 and XY_{diff} = -1 and ZX_{diff} = 1	"CA/CA-G"
"ABC/ABC-G" will be assigned if the line fault is not one from all previous combinations	"ABC/ABC-G"

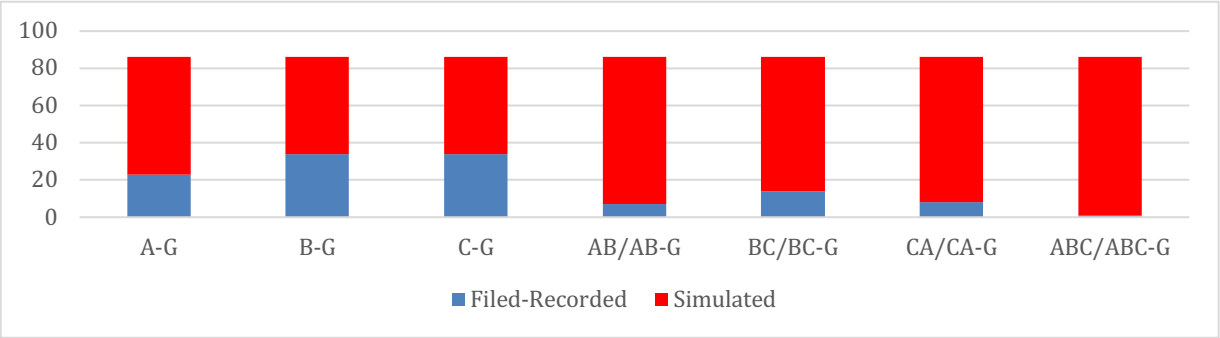

Figure 6. Integrated data distribution

to ensure that each fault type has the same frequency of occurrence in the dataset. The data integration process of the simulated data and field-recorded data is illustrated in Figure 7.

Table 2 shows the increase in the number of each fault type after the simulated data are added to field-recorded data for each line fault type, resulting in much more balanced classes vs. relying on field-recorded training data alone.

4.4. Limitations

The model has some limitations: (1) In a real-time scenario, the classification model will depend on a fault event detection model that can detect fault events out of all types of events in each 2-second window. With the streaming data, the classification will be delayed for the time needed to detect the fault. (2) The proposed method cannot be applied to SCADA measurements because SCADA only takes samples every two seconds or longer. The resolution of the SCADA measurements is too low as compared to the 30-60 samples/sec of PMUs that would not be sufficient for this application.

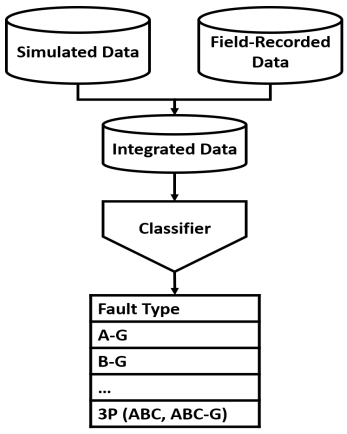

Figure 7. Train a classifier by two sources of data

5. Data Modeling

In this study, three classification models were considered to evaluate the proposed approach, and the performance of each model was compared before and after data integration. These three classifiers are briefly discussed in this section.

5.1. Classifiers

5.1.1. SVM Classifier

The second tested algorithm is the support vector machine (SVM) invented by Vapnik and Chervonenki [20]. In its simplest type, an SVM is applied to binary classification cases [21]. Becuase classifying the types of line faults is a multiclass problem, we broke the problem down into multiple binary classification cases, which is also called one-vs-one. In the one-vs-one approach, each classifier separates cases into two classes, and comprising all one-vs-one classifiers leads to a multiclass classifier [22]. This required training twenty-one models for seven classes of faults. Each binary classification model predicts one class label and selects the model with the most votes per a pair of classes.

Table 2. The number of faults before and after data integration

Line fault type	Before integration	After integration
A-G	60	86
B-G	70	86
C-G	64	86
AB/AB-G	22	86
BC/BC-G	21	86
CA/CA-G	15	86
ABC/ABC-G	14	86

5.1.2. RF Classifier

The last ML algorithm considered is the random forest classifier. Random forest is one of the most commonly used ensembles learning algorithms, and it is implemented [23] in the scikit-learn library (version 0.22.1) for Python, where default parameters are utilized.

It consists of multiple classification decision trees. Each new event is classified separately by each of the decision trees, where each tree of the forest gives a unit vote, assigning each input to the most probable class label. The final rule on the class was selected based on the maximum votes [24]. The number of trees in the forest and the depth of the trees reflecting hyperparameters should be chosen for each specific problem. For instance, one of the hyperparameters used in the RF classifier is max features that used to determine the maximum number of features to consider while looking for a split. Cross-entropy was used as a loss function that measures the probabilities of binary cross-entropy for each class separately. Cross-entropy is considered more popular in multiclass problems because it minimizes the difference between the two probabilities for each pair of classes. The reason for utilizing RF in our experiment is that RF is a fast method, robust to noise, and it is an ensemble that can successfully identify nonlinear patterns in the data.

5.1.3. XGBoost Classifier

The first algorithm used in our experiment is eXtreme gradient boosting, which is known as XGBoost [25]. The XGBoost classifier implemented in the scikit-learn library (version 0.18.1) for Python with default parameters was utilized. XGBoost is a decision-tree-based ensemble machine learning algorithm that uses a gradient-boosting framework [26]. XGBoost learning parameter was configured to multiclass soft probability to deal with multi-class classification tasks. The output is the predicted probability of each data point belonging to each class.XGBoost consists of multiple classification decision trees.Each line fault is classified separately by each of the decision trees, where each tree puts the classified line fault in one of the classes, as described in Table1.

Table 3. Fault classification performance across multiple evaluation metrics

Field-Recorded Data			
Models	**Weighted Precision**	**Weighted Recall**	**F1-score**
SVM	83.25%	91.03%	86.87.%
RF	83.31%	91.03%	86.89%
XGBoost	84.13%	91.03%	87.17%
Micro_average_of_precision_recall			94.90%
Integrated Data			
SVM	**98.69%**	98.62%	**98.58**%*
RF	98.08%	97.93%	97.83%
XGBoost	98.25%	97.93%	97.88%
Micro_average_of_precision_recall			99.20%
The largest metric values are bolded, while '*' indicates SVM performance are relatively high compared to RF and XGBoost			

6. Classifier Evaluation

We compare the results of evaluating the SVM, RF, and XGBoost classifiers trained on field-recorded and integrated field and simulated data. In the field-recorded data, the data were temporally split. We trained the models using an integrated set consisting of field-recorded data from 2016 and simulated data, which were evaluated only on data from 2017, which is "unseen" data. The evaluation results for the out-of-sample field-recoreded data are shown in Table 3.

The performance of the fault-type classification models on out-of-sample filed-recorded data was measured using weighted precision, weighted recall, and F1_score, which are suitable measures for multiclass classification problems [27]. The obtained results provide evidence that training any model on integrated data significantly improves the test accuracy. The difference in accuracy among SVM, RF, and XGBoost models was minimal when relying on integrated training data, whereas this was not the case when relying only on filed-recorded data alone (XGBoost was less accurate from SV and RF).

We show the SVM results in more detail for both cases in Figure 8 (i.e., training on filed-recorded data) and Figure 9 (i.e., using integrated training data). These results provide additional evidence that the accuracy using field-recorded PMU data alone was lower when compared to relying on combined field-recorded data enhanced by simulated faults to compensate for the types least present in the field-recorded data.

For a multiclass classification problem, precision-recall measurement was utilized on the SVM classifier for each type of fault using filed-recorded data and integrated data, respectively. The micro-average precision results in Table 3 show an increase from 94.90% to 99.20% when using integrated data compared to the overall performance for a seven-class problem). The micro-average precision scores are computed as the sum of true positives for all the classes divided by all positive predictions [28][29], as in equation 15:

$$\text{Micro\_avg\_of\_precision\_recall} = \frac{\text{TPsum}}{\text{TPsum} + \text{FPsum}} \quad (15)$$

As a result, the performance was significantly improved when training the model on integrated data for types of faults insufficiently present in field-recorded training data because the data deficiencies were reduced by including simulated cases of these types.

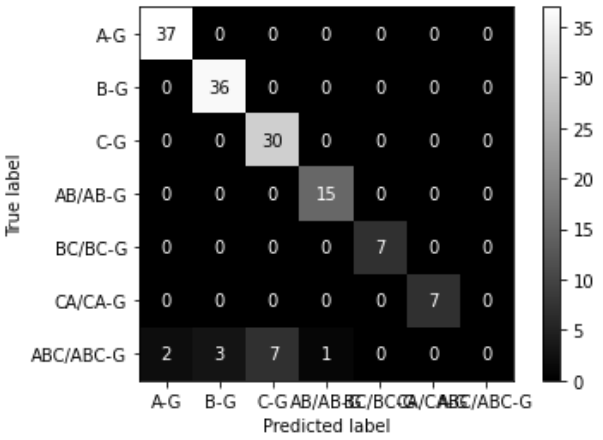

Figure 8. Confusion matrix for SVM trained on field-recorded data

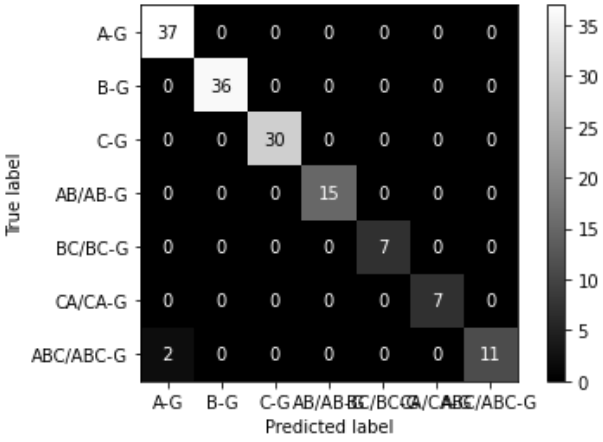

Figure 9. Confusion matrix for SVM trained on integrated data

7. Conclusions

In this study, we propose a novel method for line fault classification using machine learning on a large dataset of PMU measurements. The uniqueness and benefits of our approach are as follows:

- Our approach integrates simulated and field-recorded three-phase voltage data to classify faults in the electric grid. This data enhancement has helped train more accurate ML models to classify faults in the 2-second windows.
- It has been demonstrated that it is beneficial for train classification models using integrated field-recorded PMU data and simulations vs. relying on filed-recorded data alone when certain types of events of interest are insufficiently represented in field-recorded data over the training period.
- An innovative feature engineering tool and ML approach were developed to automatically classify different types of faults from historical PMU datasets.
- Data integration created a valuable dataset that increased line fault type classification accuracy with rare examples in the field recorded data.
- Extracted features and automatic labeling have shown patterns for all seven-line faults that have aided ML models to learn these patterns and generalized them to unseen data.

Disclaimer

References

[1] M. Kezunovic,S.Meliopoulos, S. Venkatasubramanian, V. Vittal, “Application of Time-Synchronized Measurements in Power System Transmission Networks,” Springer, ISBN 978-3-319-06218-1, 2014.

[2] M. Patel, S. Aivaliotis, E. Ellen et al., "Real-time application of synchrophasors for improving reliability", *NERC Report*, Oct 2010.

[3] L. Xie, Y. Chen, P. R. Kumar, “Dimensionality Reduction of Synchrophasor Data for Early Event Detection: Linearized Analysis,” IEEE Trans. Power Systems, vol. 29, no. 6, pp. 2784-2794, Nov. 2014.

[4] M. Rafferty, X. Liu, D. M. Laverty, S. McLoone, “Real-Time Multiple Event Detection and Classification Using Moving Window PCA,” IEEE Trans. Smart Grid, vol. 7, no. 5, pp. 2537-2548, Sep 2016.

[5] O. P. Dahal, S. M. Brahma, H. Cao, “Comprehensive Clustering of Disturbance Events Recorded by Phasor Measurement Units,” *IEEE Trans. Power Delivery*, vol. 29, no. 3, pp. 1390-1397, Jun. 2014.

[6] M. Khan, P. M. Ashton, M. Li, G. A. Taylor, I. Pisica, J. Liu, “Parallel Detrended Fluctuation Analysis for Fast Event Detection on Massive PMU Data,” IEEE Trans. Smart Grid, vol. 6, no. 1, pp. 360-368, Jan 2015.

[7] M. Biswal, S. M. Brahma, H. Cao, “Supervisory Protection and Automated Event Diagnosis Using PMU Data,” IEEE Trans. Power Delivery, vol. 31, no. 4, pp. 1855-1863, Aug. 2016.

[8] R. Yadav, A. K. Pradhan, I. Kamwa, “Real-Time Multiple Event Detection and Classification in Power System using Signal Energy Transformations,” IEEE Trans. Industrial Informatics, vol. 15, no. 3, Mar. 2019, pp. 1521-1531.

[9] M. Biswal, Y. Hao, P. Chen, S. Brahma, H. Cao, and P. DeLeon, “Signal features for classification of power system disturbances using PMU data,” Proc. Power Syst. Comput. Conf., Jun. 2016, pp. 1–7.

[10] S. Brahma, R. Kavasseri, H. Cao, N. R. Chaudhuri, T. Alexopoulos, Y. Cui, “Real-Time Identification of Dynamic Events in Power Systems Using PMU Data, and Potential Applications-Models, Promises, and Challenges,” IEEE Trans. Power Delivery, vol.32, no. 1, Feb. 2017, pp. 294-301.

[11] D.-I. Kim, T. Y. Chun, S.-H. Yoon, G. Lee, Y.-J. Shin, “Wavelet-Based Event Detection Method Using PMU Data,” IEEE Trans. Smart Grid, vol. 8, no. 3, May. 2017, pp. 1154-1162.

[12] M. Cui, J. Wang, J. Tan, A. R. Florita, Y. Zhang, “A Novel Event Detection Method Using PMU Data With High Precision,” IEEE Trans. Power Systems, vol. 34, no. 1, Jan. 2019, pp. 454-466.

[13] A. Shahsavari, M. Farajollahi, E. Stewart, E. Cortez, H. Mohsenian-Rad, “Situational Awareness in Distribution Grid Using Micro-PMU Data: A Machine Learning Approach,” IEEE Trans. Smart Grid, vol. 10, no. 6, Nov. 2019, pp. 6167-6177.

[14] S. Wang, P. Dehghanian, L. Li, “Power Grid Online Surveillance through PMU-Embedded Convolutional Neural Networks,” IEEE Trans. Industrial Applications, vol. 56, no. 2, Mar.-Apr. 2020, pp. 1146-1155.

[15] OSlsoftPIVision,[Online]Available: https://www.osisoft.com/pi-system/pi-core/visualization

[16] NASPI PMU Applications Requirements Task Force, "PMU Data Quality: A Framework for the Attributes of PMU Data Quality and a Methodology for Examining Data Quality Impacts to Synchrophasor Applications," NASPI-2017-TR-002, March 2017.

[17] ApacheSpark,[Online]Available: https://spark.apache.org/

[18] Python, [Online] Available: https://www.python.org/Brownlee, Jason. “How to Develop Your First

[19] Kezunovic, M., et al. "An expert system for transmission substation event analysis." *IEEE Transactions on power delivery* 8.4 (1993): 1942-1949.

[20] Prudhvi, Poorna. “Support Vector Machines.” Medium, Medium, 13 Oct. 2017, medium.com/@poornaprudhvi/support-vector-machines-bf0242ade522.

[21] “Sklearn.multiclass.OneVsOneClassifier.” Scikit, scikitlearn.org/stable/modules/generated/sklearn.multiclass. OneVsOneClassifier.html.

[22] J. Browniee, “One-vs-Rest and One-vs-One for Multi-Class Classification,” April 13, 2020. [Online].Available: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[23] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, 2001.

[24] T. Yiu, “Understanding Random Forest,” Medium, Towards Data Science, June 12, 2019. [Online] towardsdatascience.com/understanding-random-forest-58381e0602d2.

[25] XGBoost Model in Python.” Machine Learning Mastery, 18 Jan. 2021, machinelearningmastery.com/develop-first- xgboost-model-python-scikit-learn/.

[26] “Multiclass & Multilabel Classification with XGBoost [Online].Available:https://gabrielziegler3.medium.com/multiclass-multilabel-classification-with-xgboost.

[27] J. Brownlee, “Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning,” Machine Learning Mastery, 2020.

[28] J. Mohajon, “Confusion Matrix for Your Multi-Class Machine Learning Model,” May 29, 2020. [Online] Available: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

[29] A. Kumar, “Micro-Average & Macro-Average Scoring Metrics - Python.” Data Analytics, 4 Sept. 2020, vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python.