Proceedings of the 56th Hawaii International Conference on System Sciences | 2023

PPGN: Physics-Preserved Graph Networks for Real-Time Fault Location in Distribution Systems with Limited Observation and Labels

Wenting Li*
Los Alamos National Laboratory
wenting@lanl.gov

Deepjyoti Deka
Los Alamos National Laboratory
deepjyoti@lanl.gov

Abstract

Electric faults may trigger blackouts or wildfires without timely monitoring and control strategy. Traditional solutions for locating faults in distribution systems are not real-time when network observability is low, while novel black-box machine learning methods are vulnerable to stochastic environments. We propose a novel Physics-Preserved Graph Network (PPGN) architecture to accurately locate faults at the node level with limited observability and labeled training data. PPGN has a unique two-stage graph neural network architecture. The first stage learns the graph embedding to represent the entire network using a few measured nodes. The second stage finds relations between the labeled and unlabeled data samples to further improve the location accuracy. We explain the benefits of the two-stage graph configuration through a random walk equivalence. We numerically validate the proposed method in the IEEE 123-node and 37-node test feeders, demonstrating the superior performance over three baseline classifiers when labeled training data is limited, and loads and topology are allowed to vary.

Keywords: Fault location, Graph neural networks, Limited observation, Low label rates, Distribution systems

1. Introduction

A modern power grid forms a critical infrastructure that delivers electricity for everyday energy consumption of society and the economy. In recent years, the expansion of random, intermittent distributed energy resources (DERs) such as wind and solar energy, particularly in low-voltage power grids, has increased the instability in the grid and resulted in surges, electric line failures, and other grid malfunctions [22].

However, localizing faults in power grids in real-time is faced with several practical challenges: low observability, unreliable estimates of system parameters, and the stochastic ambient environments due to random load variations and topology changes. The real-time observability in power grids has improved due to installing wide-area sensors at grid nodes, called phasor measurement units (PMUs) [29], that collect time-synchronized measurements. This has motivated an interest in data-driven fault localization methods [3, 10, 14, 17, 27] using PMU measurements.

These methods follow three research lines. Traveling-wave-based approaches are accurate and widely applied in the industry. The high-precision and synchronized measurements, however, require the measuring instruments to be installed everywhere, which hinders their extensive application [27]. Another line of work relies on the physical property of data, such as the spatial relations of line impedance and the sparsity of fault currents, but these methods either require the full network observability [14] or high sampling rates (e.g., 10M Hz [10]). The last research line is based on supervised machine learning to locate faults on the bus or line-level [1, 3, 17]. These approaches show superior performance in efficiency and accuracy, especially in large-scale networks with low observability. Unfortunately, the insufficient availability of labeled data and the stochastic environment in practical power systems diminish the performance of such supervised methods.

Physics behind data has been incorporated into

*This work is supported by the Center for Nonlinear Studies (CNLS) and a Laboratory Directed Research & Development (LDRD) Exploratory Research (ER) grant at Los Alamos National Laboratory.

machine learning methods, so-called physics-informed machine learning (PIML), to enhance the interpretability and robustness to imperfect real data [15, 16]. However, such approaches that subtly combine physics with data-driven technology are lacking in the crucial problem of fault localization, for the realistic cases with limited labelled data.

To fill this gap, we analyze whether *we can employ the unique physics of power grids to inform supervised data-driven fault localization algorithms and augment their robustness to challenges associated with realistic grid data.* In this paper, we are thus interested in fault localization in power grids, in the challenging but realistic regime of (a) sparse observations, and (b) system variability, with (c) low fraction of labeled training data.

Contributions: We formulate a unique two-stage graph neural network architecture to locate faults in power grids with low observability and stochastic environments, using a small number of labeled data for training. Precisely, to address the issue of low observability, we inform $\mathcal{G}_{\text{I}}$ (GNN in the first stage) with the structure of the power grid by constructing an adjustable and novel adjacency matrix A. Meanwhile, in the second stage $\mathcal{G}_{\text{II}}$, we use a different adjacency matrix B based on the statistical similarity of labeled and unlabeled datasets. This second GNN improves the localization accuracy when label rates are low. We theoretically interpret the functions of adjacency matrices A and B in stages I and II respectively, through equivalence with random walks. The proposed framework is validated in the IEEE 37 and 123-node test feeders [11], in various scenarios through OpenDSS software [6]. Our approach outperforms the existing algorithms by significant margins in accuracy and robustness to low label rates, load variations, and topology changes.

The remaining parts are organized as follows. Section 2 introduces the vital physics behind data and formulates the problem for the sake of some practical challenges. Sequentially, we present the two-stage graph learning framework in Section 3. The following section demonstrates the benefits of the constructed adjacency matrices A, B. Section 5 validates the effectiveness and advantages of our approach. Finally, conclusions and future works are discussed in Section 6.

2. Physics of Fault Currents and Problem Formulation

Consider a power grid graph with n nodes in the vertex set $\mathcal{V}$ and l branches/edges in set $\mathcal{E}$, for example, the IEEE 123-node test feeder in Figure. 1. Consider the

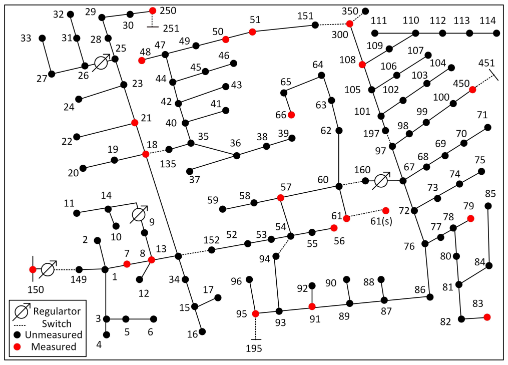


Figure 1. The IEEE 123-node test feeder

three-phase voltages u and currents c at s nodes in the set $\Omega \subset V$. Each data sample corresponds to a fault event, where the location of the fault is defined as its label. Interestingly, the nodal current (termed fault current) variations that arise due to a fault are sparse in nature, where the the nonzero values are closely related to the fault positions or labels [10, 21]. In this section, we will use this sparsity property to explain how the physical laws help reduce the label requirement.

2.1. Sparsity of Fault Currents

When a fault occurs at f (equivalent to a node) on the line between nodes i and j in the grid, we have voltages and currents at the faulted point and the nodes $k \neq f, k \in [1, n]$ denoted as $u_f^{abc}, c_f^{abc}, u_k^{abc}, c_k^{abc} \in C^{3\times 1}$ respectively. Let $Y_{ij} \in C^{3\times 3}$ be the admittance matrix between nodes i and j before the fault, while Y'_{ij} is that during the fault. According to the Kirchhoff's law, we obtain the following equation [17],

$$\underbrace{\begin{bmatrix} Y_{11} & \cdots & \cdots & \cdots & Y_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & Y_{ii} & \cdots & Y_{ij} & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & Y_{ji} & \cdots & Y_{jj} & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Y_{n1} & \cdots & \cdots & \cdots & Y_{nn} \end{bmatrix}}_{Y} \underbrace{\begin{bmatrix} u_1^{abc} \\ \cdots \\ u_i^{abc} \\ \cdots \\ u_j^{abc} \\ \cdots \\ u_n^{abc} \end{bmatrix}}_{U} - \underbrace{\begin{bmatrix} 0 \\ \cdots \\ \delta_i^{abc} \\ \cdots \\ \delta_j^{abc} \\ \cdots \\ 0 \end{bmatrix}}_{\Delta_{ij}} = \underbrace{\begin{bmatrix} c_1^{abc} \\ \cdots \\ c_i^{abc} \\ \cdots \\ c_j^{abc} \\ \cdots \\ c_n^{abc} \end{bmatrix}}_{C}$$

$$\Rightarrow YU - \Delta_{ij} = C \tag{1}$$

where $\delta_i^{abc} = (Y_{ii} - Y'_{ii})u_i^{abc} + (Y_{ij} - Y'_{ij})u_j^{abc}$,

$$\delta_j^{abc} = (Y_{ji} - Y'_{ji})u_i^{abc} + (Y_{jj} - Y'_{jj})u_j^{abc}.$$

Notice that $\Delta_{ij} \in C^{3n}$ is a *sparse vector with the nonzero values corresponding to the two terminals* i, j *of the faulted line.*

We also know that U_0, C_0, the voltages and currents on normal conditions, satisfy that $YU_0 = C_0$. Let $\Delta U = U - U_0, \Delta C = C - C_0$ denote the changes in voltage and current respectively due to the fault. Using (1), we acquire the following relation between them:

$$Y\Delta U = \Delta C + \Delta_{ij} \tag{2}$$

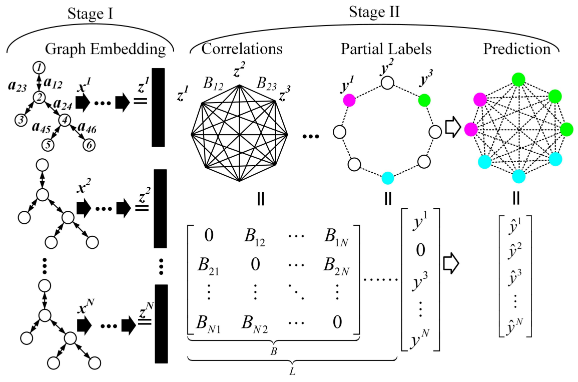
Remarks: The product of Y and ΔU, on the left side of (2), equals the linear combination of the node k's neighbors weighted by the admittance, i.e., $\Sigma_{j\in\mathcal{N}_k} Y_{kj}\Delta u_j^{abc}, k = 1, \cdots, n$, where $\mathcal{N}_k$ denotes the set of nodes connected with k. On the right side of (2), Δ_{ij} only has nonzero values at the nodes i, j connected with the fault point, while ΔC is trivial since loads have a small chance to change dramatically during the fault [20]. When all buses are known, the weighted voltage variations $\Sigma_{j\in\mathcal{N}_k} Y_{kj}\Delta u_j^{abc}$ are significant if k is near the fault. When the observability is low, the measured buses partially include the information of fault location, and a learning strategy benefits by extracting the relations between the partially weighted voltages and the location of faults. Therefore, we formulate the fault location as a learning problem as follows.

2.2. Problem Formulation

We consider a N length data-set of voltage magnitudes and voltage angles in a, b, c three phases from $s < n$ measured nodes in the grid, i.e., $X^p \in R^{n\times 6} = [V^a, \theta^a, V^b, \theta^b, V^c, \theta^c], p = 1, \cdots, N$, only the s entries of X^p corresponding to the measured nodes have nonzero values. Here unmeasured nodes are given a value 0. Additionally, we have partial labels denoting the location of the faults, $y^p \in \{1, \cdots, c\}$ for some datasets $p = 1, \cdots, m$, where $m \ll N$. We target at efficiently predicting the location of unknown faults regardless of fault types and fault impedance. Note that $m \ll N$ implies that only a few number of datasets are labeled with the true fault location while others are not.

Fault location is equivalent to a classification problem [17], but practically this classifier faces more challenges. Though Section 2.1 demonstrates the effect of power system topology in estimating faults via ΔU, the sparse observation of power network and low label rates hinders us from directly applying a conventional graph neural network (GNN) [25] classifier on the the power grid topology. Furthermore, the non-static measurements of power grids demand the classifier to be robust to out of distribution (OOD) data [2], due to changing load dispatch, random fault impedance, and topology changes.

3. Proposed Graph Learning framework



$$\underbrace{\underbrace{\begin{bmatrix} 0 & B_{12} & \cdots & B_{1N} \\ B_{21} & 0 & \cdots & B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & \cdots & 0 \end{bmatrix}}_{B} \cdots\cdots \begin{bmatrix} y^1 \\ 0 \\ y^3 \\ \vdots \\ y^N \end{bmatrix}}_{L} \Rightarrow \begin{bmatrix} \hat{y}^1 \\ \hat{y}^2 \\ \hat{y}^3 \\ \vdots \\ \hat{y}^N \end{bmatrix}$$

Figure 2. The structure of PPGN

Figure 2 shows our two-stage Physics-Preserved Graph Network (PPGN) learning framework for fault location. In Graph Stage I, named as $\mathcal{G}_{\text{I}}$, we first learn a graph embedding to represent both the node features and topology structure of the power grid by locally aggregating the observed nodes, and then predict the faulty node through the global transformation. The core distinction of $\mathcal{G}_{\text{I}}$ is the *adjustable adjacency matrix* for local aggregation. Such adjacency matrix, different from the conventional GNN [8, 13], handles the challenges of sparse observability and enlarges the capability of location prediction. Following Stage I, we design the Stage II with another graph neural network $\mathcal{G}_{\text{II}}$ using correlations between the available labeled and unlabeled datasets, to further improve the location accuracy. The novel adjacency matrix in $\mathcal{G}_{\text{II}}$ is based on the learnt graph embedding from Stage I. The detailed theoretical analysis of these two adjacency matrices for $\mathcal{G}_{\text{I}}$ and $\mathcal{G}_{\text{II}}$ is presented in Section 4.

3.1. Stage I: Graph Embedding Learning

When the measured nodes of the power grid are sparse, it is difficult to learn informative embedding of the node without measured neighbors. Instead, our key idea is to construct an adjacency matrix $A \in R^{n\times n}$ based on Dijkstra's shortest path [5] between any pair of nodes to ensure that each node is observable. Then we accomplish graph embedding with two major procedures: local aggregation and global transformation.

3.1.1 Local Aggregation K hidden layers map the pth data matrix X^p into hidden variables $H^k \in R^{n\times n_k} = [h_1^k, \cdots, h_n^k], k = 1, \cdots, K$ with the shared weight matrix $W^k \in R^{2n_{k-1}\times n_k}$ among all nodes, and $H^0 = X^p$. The ith row and jth column of A represents the correlation between nodes i, j and is defined as

$$a_{ij} = \begin{cases} \exp(-\frac{d_{i,j}^2}{\delta_i^2}) & \text{if } j \in \mathcal{N}_i^{k_{\mathrm{I}}} \\ 0 & \text{else} \end{cases} \tag{3}$$

where d_{ij} is the shortest path between nodes i, j, $\delta_i = \frac{1}{k_{\mathrm{I}}} \Sigma_{j \in \mathcal{N}_i^{k_{\mathrm{I}}}} d_{ij}$, and $\mathcal{N}_i^{k_{\mathrm{I}}}$ consists of the k_{I} nearest neighbors of node i, where k_{I} is chosen to ensure that each unobserved node has at least one observed neighbor. We symmetries the matrix A by $a_{ij} = \max(a_{ij}, a_{ji})$.

The update rule of the kth layer is:

$$h_i^k = \sigma(\{h_i^{k-1} || \text{Aggregate}_{j \in \mathcal{N}_i}(h_j^{k-1} \tilde{a}_{ij})\} W^k) \tag{4}$$

where $||$ means to concatenate the two vectors, $\text{Aggregate}_{j \in \mathcal{N}_i}(h_j^{k-1} \tilde{a}_{ij}) = \frac{1}{|\mathcal{N}_i|} \Sigma_{j \in \mathcal{N}_i}(h_j^{k-1} \tilde{a}_{ij})$, $\sigma(x) = \max(0, x)$, $\tilde{a}_{ij}$ is the normalized a_{ij} such that $\Sigma_{j \in \mathcal{N}_i} \tilde{a}_{ij} = 1$, and the $\mathcal{N}_i$ is the neighborhood of node i.

3.1.2 Global Transformation Global transformation converts the hidden variables of all the local nodes into prediction probability of the whole data sample as the graph embedding. Firstly, the vectorized hidden variables $\hat{h}^K \in R^{nn_K}$ go through fully connected layers with the trainable weights $W^f \in R^{nn_K \times 2n}, b^f \in R^{2n}, W^o \in R^{2n \times n}, b^o \in R^n$ to become the vector $f \in R^n$. Then the output layer transforms f into graph embedding $z^p \in [0,1]^n$ with $W^o \in R^{2n \times n}, b^o \in R^n$ for the pth data sample as follows,

$$z_i^p = \frac{\exp(f_i)}{\Sigma_{j=1}^n \exp(f_j)}, \quad f = (\hat{h}^K W^f + b^f) W^o + b^o \tag{5}$$

where z_i^p, f_i are the ith entry of z^p and f respectively.

3.1.3 Loss Function of Stage I

$$\mathcal{L}(\Theta_{\mathrm{I}}) = -\Sigma_{p=1}^m y^p \log(z^p) + \lambda_{\mathrm{I}} ||\Theta_{\mathrm{I}}|| \tag{6}$$

The first term of (6) is the cross entropy of y^p and z^p for available labeled data samples, and the second is the regularization term to augment the generalization capability, where $||\Theta_{\mathrm{I}}||$ is the l_2-norm of all the trainable parameters $\Theta^I = \{W^1, \cdots, W^K, W^f, b^f, W^o, b^o\}$ of $\mathcal{G}_{\mathrm{I}}$ with the hyper-parameter λ_{I}. By minimizing (6), the Θ_{I} is automatically learned by back-propagation.

3.1.4 Practical Training Technique Inspired by Yang et.al. [32], we alternatively train $\mathcal{G}_{\mathrm{I}}$ through the local aggregation and global transformation procedures for T_1 and T_2 epochs respectively. Empirically, this alternative training speeds up the convergence and accomplishes higher classification accuracy. The detailed discussion is in Section 5.6.

3.2. Stage II: Label Propagation

To further increase the location accuracy when given a number of unlabeled datasets, we build the correlation matrix $B \in R^{N \times N}$ of the labeled and unlabeled datasets, as the adjacency matrix of the graph $\mathcal{G}_{\mathrm{II}}$. Note that each vertex of $\mathcal{G}_{\mathrm{II}}$ represents one data sample. The intuition is that faults that occur at the same or nearby locations share similar physical characteristics, as the analysis in Section 2.1, resulting in similarity of datasets. Using the learned graph embedding z^p in $\mathcal{G}_{\mathrm{I}}$ of various data samples, we establish the correlation matrix B. Then our graph model $\mathcal{G}_{\mathrm{II}}$ propagates labels to the unlabeled data samples through graph convolutional layers (GCL) [13].

Adjacency Matrix B: The critical purpose of B is to learn useful correlations among data samples while cutting off misleading correlations. For that we use z^p, the output of Stage I. Precisely, we first zero out the entries of z^p that correspond to nodes far beyond the Stage I predicted location $p^* = \arg\max_i z_i^p$, and obtain vector $\hat{z}^p$. Let $\mathcal{S}_{p^*}$ be the set of nodes physically connected with p^* in the original power grid. Then

$$\hat{z}_k^p = \begin{cases} 0 & \text{if } k \notin \mathcal{S}_{p^*} \\ z_k^p & \text{otherwise} \end{cases} \tag{7}$$

Second, we calculate the similarity $s(p,q)$ of embedding between any pair of data samples $\hat{z}^p, \hat{z}^q$ through distance metrics. Here we apply the subspace angle $s(p,q) = \frac{\langle \hat{z}^p, \hat{z}^q \rangle}{||\hat{z}^p||_2 ||\hat{z}^q||_2}$ [26] as distance metric. The entry at the pth row and qth column of B is then defined as

$$B_{pq} = \begin{cases} s(p,q) & \text{if } q, p \text{ are similar to each other} \\ 0 & \text{else} \end{cases} \tag{8}$$

where "q, p are similar to each other " means that the $s(p,q)$ is among the largest k_{II} values of $\{s(p,q'), q' \in [1,N]\}$ or $\{s(p',q), p' \in [1,N]\}$. Significantly, B becomes a sparse matrix with non-zero entries restricted to data-points at close graphical locations. This helps accelerate the training process when $k_{\mathrm{II}} \ll N$. We give a theoretical explanation for B in Section 4.2. Once B is estimated, we use it in a Graph Convolutional Network.

3.2.1 Graph Convolutional Layers Reshape all the raw data samples $X^p, p = 1, \cdots, N$ to form matrix $C^0 \in R^{N \times 6n}$ as the input of the Graph Convolutional Layers (GCL).

$$C^l = \sigma(D^{-\frac{1}{2}} \hat{B} D^{-\frac{1}{2}} C^{l-1} W_{\mathrm{II}}^l) \tag{9}$$

where $C^l, l = 1, \cdots, L$ is the lth output of the hidden layer with weights $W_{\mathrm{II}}^l \in R^{n_{l-1} \times n_l}$, $\hat{B} = I_N + B$

and D is the degree matrix corresponding to $\hat{B}$ for normalization.

3.2.2 Output Layer The output layer performs the linear regression on the c^p, the pth row of C^L, with the weight $W_{\text{II}}^o \in R^{n_L \times n}$ and the bias $b_{\text{II}}^o \in R^n$ to obtain $g^p \in R^n$, and then converts g^p to be the prediction probability $\hat{y}^p \in R^n$ through a softmax function as follows,

$$\hat{y}_i^p = \frac{\exp(g_i^p)}{\Sigma_{j=1}^n \exp(g_j^p)}, \quad g^p = c^p W_{\text{II}}^o + b_{\text{II}}^o \tag{10}$$

where $\hat{y}_i^p, g_i^p$ are the ith entry of $\hat{y}^p$ and g^p respectively.

3.2.3 Loss function of Stage II We use the regularized cross entropy loss function as

$$\mathcal{L}'(\Theta_{\text{II}}) = -\Sigma_{p=1}^m y^p \log(\hat{y}^p) + \lambda_{\text{II}} \|\Theta_{\text{II}}\| \tag{11}$$

where $\Theta_{\text{II}} = \{W_{\text{II}}^1, \cdots, W_{\text{II}}^L, W_{\text{II}}^o, b^o\}$ includes all the trainable parameters of $\mathcal{G}_{\text{II}}$.

4. Theoretical Interpretations

We interpret the graph learning in the two stages through the random walk equivalence [30, 31]. Here *a walk* steps from one node randomly into its neighbor defined by the adjacency matrix of the graph. From the point of this view, we demonstrate the advantages of our constructed A and B in augmenting visibility and location accuracy.

4.1. Construct A to Improve Visibility

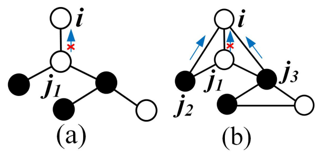


Figure 3. Example: Filled nodes are measured, and void nodes are unmeasured. When $k_{\text{I}} = 2, K = 1$, node i only has one connected node j_1 in (a), but i has three neighbors $j_k, k = 1, 2, 3$ in (b) defined by our A, which ensures that there are some paths from the unmeasured nodes to the measured nodes.

The construction of A determines the learning path in the stage I. We rigorously derive the information flow along the path through node influence.

Node Influence of $\mathcal{G}_{\text{I}}$: The *node influence*, denoted as $I(h_i^k, h_j^0; k)$, measures the influences of the node j's input, h_j^0, on node i's learned hidden variable, h_i^k after k layers [31], which quantifies the variations of h_i^k *when* h_j^0 *changes,*

$$\begin{aligned} I(h_i^k, h_j^0; k) &= \partial h_i^k / \partial h_j^0 \\ &= \Pr(\text{Walk from } i \text{ to } j \text{ with } k \text{ steps}) \\ &= \Sigma_{\mathcal{P} \in \mathcal{P}_k^{i \to j}} \prod_{e \in \mathcal{P}} \tilde{a}_e \end{aligned} \tag{12}$$

where Pr is the abbreviation of probability, $\mathcal{P}_k^{i \to j}$ represents the path from i to j with k steps, a_e is the weight of the edge e defined by A that $\tilde{a}_e = \tilde{a}_{i'j'}$ when $e = (i', j')$. As the learning process of each layer is independent, the probability of choosing one path $\mathcal{P}$ is the product of $\tilde{a}_e, e \in \mathcal{P}$.

The information collected by each node via $\mathcal{G}_{\text{I}}$ is implied by the *total node influence* $I(h_i^K)$ of the node i after K layers,

$$\begin{aligned} I(h_i^K) &= \Sigma_{j \in \mathcal{N}_i^K, j \in \Omega} I(h_i^K, h_j^0; K) \\ &= \Sigma_{j \in \mathcal{N}_i^K, j \in \Omega} \Sigma_{\mathcal{P} \in \mathcal{P}_K^{i \to j}} \prod_{e \in \mathcal{P}} \tilde{a}_e \end{aligned} \tag{13}$$

where $\mathcal{N}_i^K$ denotes the K-hop neighborhood of node i, Ω is the set of measured nodes. Therefore, we conclude that the information obtained at node i is richer if more paths from the observed nodes to node i. Following this principle, we construct A to ensure that each unobserved node has some path from the nearby observed nodes even though its immediate neighbors are unmeasured. Figure 3 (a) shows one simple example to illustrate the distinctive learning paths when using the physical topology in (a) and that using our A in (b).

4.2. Construct B to Enhance the Exact Prediction Probability

B characterizes the correlations of labeled and unlabeled data samples to improve the prediction probability. We first show the random-walk interpretation of the node influence in $\mathcal{G}_{\text{II}}$, and then introduce how we increase correct prediction probability by cutting off misleading correlations. The effects of B are also validated by experiments in Section 5.4.

Node Influence of $\mathcal{G}_{\text{II}}$: In $\mathcal{G}_{\text{II}}$, the pth node corresponds to the data sample X^p, and we care about the influence of the known labeled data samples on the unlabeled ones. The node influence of the labeled data sample q on the unlabeled data sample p after l layers is

$I'(C_p^l, C_q^0; l)$.

$$I'(C_p^l, C_q^0; l) = \partial C_p^l / \partial C_q^0$$
$$= \text{Pr (Walk from } p \text{ to } q \text{ with } l \text{ steps)}$$
$$= \Sigma_{\mathcal{P} \in \mathcal{P}_l^{p \to q}} \prod_{e \in \mathcal{P}} \bar{B}_e \quad (14)$$

where $\bar{B}_e = \bar{B}_{p'q'}$, $e = (p', q')$ is the edge e between p' and q', and $\bar{B}_{p'q'}$ is the normalized weight that $\Sigma_{q'=1}^N \bar{B}_{p'q'} = 1$.

The expectation of the total influence of node q on p after L hidden layers is defined as (by Theory 1 in [31],

$$I_p'(q; L) = \Sigma_q I'(C_p^L, C_q^0; L) / \Sigma_r I'(C_p^L, C_r^0; L)$$

is equivalent to the probability that data sample p has the same label with q after L-step random walk, i.e.,

$$\text{Pr}(\hat{y}^p = y^p) = E(\frac{\Sigma_{q:y^q=y^p} I'(C_p^L, C_q^0; L)}{\Sigma_{r:y^r \in [1,c]} I'(C_p^L, C_r^0; L)}) \quad (15)$$
$$= E(\frac{\Sigma_{q:y^q=y^p} \Sigma_{\mathcal{P} \in \mathcal{P}_L^{p \to q}} \prod_{e \in \mathcal{P}} \bar{B}_e}{\Sigma_{r:y^r \in [1,c]} \Sigma_{\mathcal{P} \in \mathcal{P}_L^{p \to r}} \prod_{e \in \mathcal{P}} \bar{B}_e}) \quad (16)$$

where $q : y^q = y^p$ denotes those data samples q have the same labels with y^p, and $r : y^r \in [1, c]$ represents any data sample r. Hence, the correct prediction of the unlabeled data sample p has high probability if (1) The paths from the data samples with the same labels y^p are more than those with other labels; (2) the number of labeled data samples with y^p is significant.

Cut off the Unrelated Paths through B: We construct B via $\hat{z}^p, p = 1, \cdots, N$ rather than the raw datasets X^p to produce more zero entries that cut off the paths from mismatching labels, resulting in an increase in the accurate prediction probability. If we calculate B with X^p, the correct prediction probability $\text{Pr}(\hat{y}^p = y^p)$ is given by (15). However, if we use $\hat{z}^p$ as input, it is clear from (7) and (8) that $s(p, q)$ for data-sets p, q is nonzero only if the prediction q^* and p^* are within two-hops of each other. Thus, the physical distance between the true labels of data-set p and q should be within four hops if $s(p, q)$ is not zero. The correct prediction probability, in that case, is thus larger as the pth data sample only has the access to partial data samples r whose true labels $y^r \in \mathcal{N}_{y^p}^{4L}$, i.e.,

$$P'(\hat{y}^p = y^p) = E(\frac{\Sigma_{q:y^q=y^p} I'(X^p, X^q; L)}{\Sigma_{r:y^r \in \mathcal{N}_{y^p}^{4L}} I'(X^p, X^r; L)}) \quad (17)$$
$$\geq \text{Pr}(\hat{y}^p = y^p)$$

As we also control the total number of nonzero values of each row of B to be no more than $2k_{\text{II}}$, thus $|\mathcal{N}_{y^p}^{4L}| \leq 2k_{\text{II}} < |\mathcal{V}|$, where $|\cdot|$ denotes the size of a set. Therefore, we improve the correct prediction probability by constructing B using the learned graph embedding.

5. Numerical Experiments

We implement the proposed framework in the 123-node test feeder [11], simulated by OpenDSS [6]. This test feeder is typically composed of grid components such as voltage regulators, overhead/underground lines, switch shunts, and unbalancing loads. Our approach shows high performance for various types of faults at low label rates, outperforming three well-known baselines by significant margins. Moreover, we validate the robustness to topology changes and load variations, and analyze the effects of different stages. In addition, our graph learning framework can easily adapt to another system, the IEEE 37-node test feeder, where we demonstrate the superior location performance using the proposed training strategy in Section 3.1. The codes and datasets are available at `https://github.com/Wendy0601/PPGN-Physics-Preserved-Graph-Networks`.

5.1. Implementation Details

The graphical structure our testing system is shown in Figure 1, where 21 out of the 128 nodes are measured marked as red. Nine pairs of nodes are connected by switches or regulators at the same locations. Thus a total of 119 possible fault positions exist, which are represented by the labels $y^p \in \{1, \cdots, c\}, c = 119$. We simulate $N = 24480$ data samples including single phase to ground (SPG) faults, phase to phase (PP) faults, and Double-phase to ground (DPG) faults at all the three phase nodes with fault impedance varying from $0.05\Omega \sim 20\Omega$. In the industrial practice, training data samples can be acquired either from historical datasets or by some advanced data generation techniques, such as [4].

We implement three best baseline classifiers: fully connected neural networks (NN), convolutional neural networks (CNN), and graph convolutional neural networks (GCN). Our NN has two rectified linear unit (ReLU) layers, each of which reduces the dimension of the input by one half; CNN has four ReLU convolution layers with filters of size 2×2 and depth of $8, 8, 16, 16$. Each convolution layer is followed by batch normalization and maximum pooling layers; GCN has three convolutional graph layers with filters of size 32. The baseline classifiers all apply the cross entropy loss function with l_2 norm regularization. We apply Adam, a stochastic gradient descent based optimizer [12], with learning rate being 0.001 to train the classifiers.

The loads for each data sample are random, following a typical load shape, and the expectation of load variations at one node is 0.53 per unit (p.u.). Each data sample is a matrix $X^p \in R^{128\times 6}$, where only measured nodes have nonzero values. To validate the robustness to OOD data, we simulate another nine sets of data samples covering various system topology or load variations as shown in Section 5.5, and each set includes 12240 faults of all types and locations. We normalize each data sample X^p by subtracting the mean values and scaling with the standard deviation of all datasets.

Structures of $\mathcal{G}_{\mathbf{I}}, \mathcal{G}_{\mathbf{II}}$ $\mathcal{G}_{\mathrm{I}}$ has three hidden layers with $W^k \in R^{32\times 32}, n_k = 32, k = 1, \cdots, K = 3$. We set $k_{\mathrm{I}} = 3 \ll n$ considering the sparse infrastructure of power grids, to construct matrix A (see Eq. 3). The hyper-parameters $\lambda_{\mathrm{I}} = 5\times 10^{-3}$, $T_1 = 10, T_2 = 10$, and the learning rate is 0.001. $\mathcal{G}_{\mathrm{II}}$ has two layers with $n_l = 3n$ for $l = 1, 2$. We take hyper-parameter $k_{\mathrm{II}} = 120$ for B (see Eq. 8). The learning rate for $\mathcal{G}_{\mathrm{II}}$ is 0.001 and $\lambda_{\mathrm{II}} = 5\times 10^{-5}$. We train the proposed model with Adam and implement the the structure through Pytorch [23]. Based on a MacBook Pro with CPU of 2.4 GHz 8-core Intel i9 series, memory of 32 GB, the per-iteration running time of the proposed graph neural network $\mathcal{G}_I$ is 0.03 seconds, which becomes 0.02 seconds if using 1 NVIDIA Tesla GPU.

Performance Metrics We adapt three performance metrics: F1-score, location accuracy rate (LAR), and LAR$^{1-\mathrm{hop}}$ [24]. The definitions of these three metrics are based on four basic concepts: True Positive (TP$_i$) is the number of correctly predicted samples of location i; False Positive (FP$_i$) is the number of wrongly predicted samples of location i; True Negative (TN$_i$) is the number of correctly predicted samples of locations rather than i; False Negative (FN$_i$) is the number of wrongly predicted samples of locations rather than i. Let TP$'_i$ denote the number of data samples where the predicted node is in the immediate neighborhood of the true fault node i. Then *Precision* is $P_i = \mathrm{TP}_i/(\mathrm{TP}_i + \mathrm{FP}_i)$, *Recall* is $R_i = \mathrm{TP}_i/(\mathrm{TP}_i + \mathrm{FN}_i)$, *F1-score* is $F_i = 2R_iP_i/(R_i + P_i)$, $\mathrm{LAR}_i = \mathrm{TP}_i/(\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{TN}_i + \mathrm{FN}_i)$, and $\mathrm{LAR}_i^{1-\mathrm{hop}} = \mathrm{TP}'_i/(\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{TN}_i + \mathrm{FN}_i)$. The notations LAR and LAR$^{1-\mathrm{hop}}$ are the average of LAR$_i$ and $\mathrm{LAR}_i^{1-\mathrm{hop}}, i \in [1, c]$ of all the locations.

5.2. Performance Comparison

We compare our model with three baseline classifiers (CNN, NN, GCN) in Figure 4, which includes the location performance for different types of faults

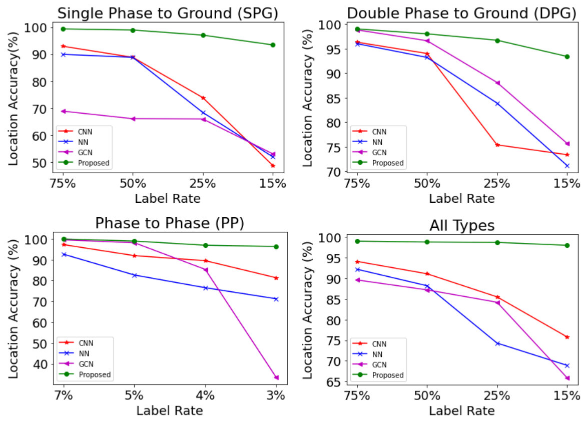


Figure 4. LAR at Different Label Rates

respectively. Because PP faults are less impacted by the changes of grounded impedance, we find that locating the PP faults is much easier than others. We show the location performance for PP faults when label rates are from 7% to 3% and 75% to 15% for others. Note that the power grid operator can first determine the type of each fault by other approaches, such as [18, 19].

We find that when the label rates are high, GCN and CNN can achieve comparable performance with the proposed model, but the high performance depends on the sufficient training data, and their LARs decline dramatically when label rates become low. On the contrary, the proposed algorithm shows more stable and accurate performance, outperforming the baselines by significant margins at realistic low label rates.

5.3. Two-stage Location Performance

Table 1. Location Performance of All Types

Label Rate β	75%	50%	25%	15%
F1 Score (%)	98.7	98.5	98.4	97.6
LAR (%)	99.0	98.8	98.7	98.0
LAR$^{1-\mathrm{hop}}$ (%)	100.0	100.0	99.96	99.90

We train $\mathcal{G}_{\mathrm{I}}$ at different label rates β, which is the ratio of the number of training data samples to N. Table 1 reports the location performance in terms of the three metrics when β varies from 75% to 15%. F1 Scores and LARs of our model are stable and remain higher than 97% even only β =15%. Note that the labeled data samples for training are randomly selected for each location to avoid the issue of data imbalance [28]. Crucially, the LAR$^{1-\mathrm{hop}}$ is close to 100%, indicating that the predicted node is in the 1-hop neighborhood of the fault node with a high probability.

5.4. Performance at Different Stages When Label Rates are Low

Table 2. Location Performance of Different Stages

Types of Faults	SPG	DPG	PP
Label Rate	15%	15%	3%
Stage I Only (LAR)	92.9	92.7	95.7
Stage II Only (LAR)	30.7	39.6	78.2
Stage I + II (LAR)	**93.3**	**93.5**	**96.3**
Stage I Only (LAR$^{1-\text{hop}}$)	94.8	95.1	96.6
Stage II Only (LAR$^{1-\text{hop}}$)	37.3	46.3	82.3
Stage I + II (LAR$^{1-\text{hop}}$)	**99.7**	**98.9**	**99.8**

Table 2 shows the performances at different stages when label rates are low, where "Stage I Only" denotes that we only train $\mathcal{G}_{\text{I}}$ to locate faults; "Stage II Only" means that we directly utilize the input data X^p rather than the embedding z^p to construct the matrix B and then locate the unknown faults. Note that "Stage II Only" is the typical manner of label propogation strategy for the semi-supervised learning in the computer vision domain [8]. "Stage I + II" represents our proposed graph learning framework.

Comparing with the "Stage I Only" and "Stage II Only", we observe the superior performance of combining stage I and II. Also, the low LAR of "Stage II Only" indicates the significance of constructing B using z^p rather than X^p. The high accuracy of "Stage I Only" ensures the learned z^p is reliable. Therefore, combination of the stages I and II benefits the location accuracy further.

5.5. Robustness to Out of Distribution Data

We validate the robustness of different classifiers to OOD data due to load variations and topology changes. Note that the performance of each classier here is for the best model in Figure 4, *but without retraining*.

Table 3 demonstrates the robustness to load variations, where Δp denotes the averaged load variation per unit (p.u.) at each node with a load. The Δp for training data is 0.53, and here we increase it up to 0.74 p.u., which immediately causes the measurements to change to different extent. Compared with other classifiers, our proposed method achieves the highest accuracy with less variations and hence is more robust to the load variations.

Also, we change the topology by varying the states of eight switches. Under normal conditions, the first six switches are closed, and the other two are open. Tables 4 reveals the testing accuracy of the datasets with various switch states, where "Close 7&8" denotes that we close the switches 7 and 8 from open states, and "Open 1-6" means that we open the first 6 closed switches. Note that the challenge here is not only the change of topology, but also the data variations caused by topology changes. Still, the proposed one demonstrates better accuracy in all scenarios.

Table 3. LAR of SPG, DPG, PP When All the Loads Vary in Different Ranges

	Δp (p.u.)	0.53	0.58	0.64	0.69	0.74
SPG	CNN	93.9	85.3	84	83.9	82
	NN	92.5	80	77.4	76.7	74
	GCN	64.3	57.7	56.4	55.6	55.1
	Proposed	**98.9**	**96.6**	**96.3**	**95.8**	**95.1**
DPG	CNN	96.5	88.3	87.8	85.3	82.5
	NN	98	89.3	88.2	86.7	85.1
	GCN	98.3	84.0	83.7	82.2	78.8
	Proposed	**98.4**	**94.1**	**93.7**	**92.7**	**92.2**
PP	CNN	97.5	96.2	96.1	95.1	94.6
	NN	95.6	92.2	90.3	87.9	85.9
	GCN	99.5	96.5	96.5	96.6	96.7
	Proposed	**99.9**	**99.6**	**99.4**	**99.2**	**98.4**

Table 4. LAR of SPG, DPG, PP When Different States of Switches Change Network Topology

	Switch	Close 7&8	Open 1-6	Open 1-3
SPG	CNN	80.0	84.4	88.8
	NN	75.0	82.5	81.7
	GCN	56.9	58.3	59.6
	Proposed	**95.8**	**94.5**	**96.9**
	Switch	Close 7&8	Open 1-6	Open 1-3
DPG	CNN	84.7	88.3	90.3
	NN	82.9	91.0	89.3
	GCN	80.5	66.9	85.6
	Proposed	**93.6**	**94.4**	**96.5**
	Switch	Close 7&8	Open 1-6	Open 1-3
PP	CNN	96.1	95.0	96.9
	NN	93.6	94.1	94.1
	GCN	91.4	95.6	97.3
	Proposed	**97.2**	**99.0**	**99.9**

5.6. Extension to IEEE 37-node Test Feeder

We extend our graph framework to the IEEE 37-node test feeder [11], where 15 nodes are measured. To indicate the adaptation of our graph model, we keep the same graph structures described in Section 5.1. Only the dimensions of inputs become $\bar{X}^p \in R^{36\times 6}, p \in [1, \bar{N}], \bar{y}^p \in \{1, \cdots, \bar{c}\}, \bar{c} = 36$, where $\bar{X}^p$ only has 15 nonzero rows corresponding to those observed nodes.

Table 5. Location Performance of All Types of Faults When Using Various Training Strategies

Supervised	β	75%	50%	25%	15%
	F1 Score	91.0	90.4	88.1	84.7
	LAR	91.1	90.3	88.2	84.8
	LAR$^{\text{1-hop}}$	97.6	96.8	96.5	96.5
Proposed	β	75%	50%	25%	15%
	F1 Score	**95.6**	**94.9**	**93.3**	**91.3**
	LAR	**95.6**	**94.9**	**93.3**	**91.3**
	LAR$^{\text{1-hop}}$	**98.9**	**98.5**	**99.1**	**98.9**

We generate $\bar{N} = 12960$ data samples in the 37-node test feeder, including SPG, DPG, and PP faults at all possible nodes, accompanied with load fluctuating. We train our graph framework with different percentages of datasets and test by the remaining data. The location performance of all types of faults is shown in Table 5 in the line of "Proposed Training", which denotes that we employ the proposed training strategy in Section 3.1. Comparably, "Supervised Training" denotes that the graph model is trained by the conventional supervise learning method, i.e., regard $\mathcal{G}_{\text{I}}$ as a whole and update all the trainable parameters in Θ_{I} in each iteration by optimizing (6).

Table 5 shows that the proposed training strategy can enhance up to 6% of LAR than that using "Supervised Training". Note that the similar effectiveness also appears in the 123-node test feeder. The intuition behind is that the "local aggregation" and "global transformation", functioning as the encoder and decoder of the graphical input data, have better convergence if trained alternatively [7, 9].

6. Conclusions and Future Works

The black-box machine learning fails in power grids mainly due to the practical challenges: low observation, insufficient labeled datasets, and dynamic data distributions. This paper handles these issues by establishing a two-stage robust data-driven algorithm for fault location via embedding power grid physics into a graph learning framework.

We theoretically demonstrate the benefits of the proposed adjacency matrices to address the sparse observability and low label rates challenges. Experimental results illustrate the superior performances of the proposed approach over three baseline classifiers. A large number of OOD datasets validate our approach's robustness to load variations and topology changes. Our future interest is to optimize the placement of PMUs to maximize location accuracy at the minimum cost.

References

[1] Aparicio, M. J., Grijalva, S., and Reno, M. J. (2021). Fast fault location method for a distribution system with high penetration of pv. In *HICSS*, pages 1–9.

[2] Calder, J., Cook, B., Thorpe, M., and Slepcev, D. (2020). Poisson learning: Graph based semi-supervised learning at very low label rates. In *International Conference on Machine Learning*, pages 1306–1316. PMLR.

[3] Chen, K., Hu, J., Zhang, Y., Yu, Z., and He, J. (2019). Fault location in power distribution systems via deep graph convolutional networks. *IEEE Journal on Selected Areas in Communications*, 38(1):119–131.

[4] Chen, Y., Wang, Y., Kirschen, D., and Zhang, B. (2018). Model-free renewable scenario generation using generative adversarial networks. *IEEE Transactions on Power Systems*, 33(3):3265–3275.

[5] Dijkstra, E. W. et al. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[6] Dugan, R. C. and McDermott, T. E. (2011). An open source platform for collaborating on smart grid research. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–7.

[7] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Cambridge, MA, USA: MIT Press.

[8] Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

[9] Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

[10] Jia, K., Yang, B., Dong, X., Feng, T., Bi, T., and Thomas, D. W. (2019). Sparse voltage measurement-based fault location using intelligent electronic devices. 11(1):48–60.

[11] Kersting, W. H. (1991). Radial distribution test feeders. 6(3):975–985.

[12] Kingma, D. P. and Ba, J. L. (2014). Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*, pages 1–15.

[13] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph

convolutional networks. *arXiv preprint arXiv:1609.02907*.

[14] Lee, Y.-J., Lin, T.-C., and Liu, C.-W. (2019). Multi-terminal nonhomogeneous transmission line fault location utilizing synchronized data. 34(3):1030–1038.

[15] Li, H. and Weng, Y. (2021). Physical equation discovery using physics-consistent neural network (pcnn) under incomplete observability. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 925–933.

[16] Li, W. and Deka, D. (2021). Physics-informed learning for high impedance faults detection. In *2021 IEEE Madrid PowerTech*, pages 1–6. IEEE.

[17] Li, W., Deka, D., Chertkov, M., and Wang, M. (2019). Real-time faulted line localization and pmu placement in power systems through convolutional neural networks. *IEEE Trans. Power Syst.*, 34(6):4640–4651.

[18] Li, W. and Wang, M. (2019). Identifying overlapping successive events using a shallow convolutional neural network. *IEEE Trans. Power Syst.*, 34(6):4762–4772.

[19] Li, W., Wang, M., and Chow, J. H. (2018). Real-time event identification through low-dimensional subspace characterization of high-dimensional synchrophasor data. 33(5):4937–4947.

[20] Majidi, M., Arabali, A., and Etezadi-Amoli, M. (2015). Fault location in distribution networks by compressive sensing. 30(4):1761–1769.

[21] Majidi, M. and Etezadi-Amoli, M. (2018). A new fault location technique in smart distribution networks using synchronized/nonsynchronized measurements. 33(3):1358–1368.

[22] Novosel, D., Bartok, G., Henneberg, G., Mysore, P., Tziouvaras, D., and Ward, S. (2009). IEEE PSRC report on performance of relaying during wide-area stressed conditions. 25(1):3–16.

[23] Paszke, A. and et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. Curran Associates, Inc.

[24] Pedregosa, F. and et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[25] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

[26] Soltanolkotabi, M. et al. (2014). Robust subspace clustering. *Ann. Stat.*, 42(2):669–699.

[27] Tashakkori, A., Wolfs, P. J., Islam, S., and Abu-Siada, A. (2019). Fault location on radial distribution networks via distributed synchronized traveling wave detectors. 35(3):1553–1562.

[28] Thabtah, F., Hammoud, S., Kamalov, F., and Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513:429–441.

[29] Von Meier, A., Culler, D., McEachern, A., and Arghandeh, R. (2014). Micro-synchrophasors for distribution systems. In *Innovative Smart Grid Technologies Conference (ISGT), 2014 IEEE PES*.

[30] Wang, H. and Leskovec, J. (2020). Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*.

[31] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462.

[32] Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.