Received: 24 March 2024 | Revised: 6 September 2024 | Accepted: 11 November 2024

DOI: 10.1111/sjos.12763

ORIGINAL ARTICLE

Scandinavian Journal of Statistics

Graph neural networks for the localization of faults in a partially observed regional transmission system

Mantautas Rimkus[1] | Piotr Kokoszka[1] | Dongliang Duan[2] | Xuao Wang[2] | Haonan Wang[1]

[1]Department of Statistics, Colorado State University, Fort Collins, Colorado, USA

[2]Department of Electrical and Computer Engineering, University of Wyoming, Laramie, Wyoming, USA

Correspondence
Piotr Kokoszka, Department of Statistics, Colorado State University, Fort Collins, CO 80523-1877, USA.
Email: Piotr.Kokoszka@colostate.edu

Funding information
National Science Foundation, Grant/Award Numbers: DMS-1923142, DMS-2123761

Abstract
Localization of faults in a large power system is one of the most important and difficult tasks of power systems monitoring. A fault, typically a shorted line, can be seen almost instantaneously by all measurement devices throughout the system, but determining its location in a geographically vast and topologically complex system is difficult. The task becomes even more difficult if measurements devices are placed only at some network nodes. We show that regression graph neural networks we construct, combined with a suitable statistical methodology, can solve this task very well. A chief advance of our methods is that we construct networks that produce localization without having being trained on data that contain fault localization information. We show that a synergy of statistics and deep learning can produce results that none of these approaches applied separately can achieve.

KEYWORDS
fault localization, graph neural networks, regional power grid

1 | INTRODUCTION

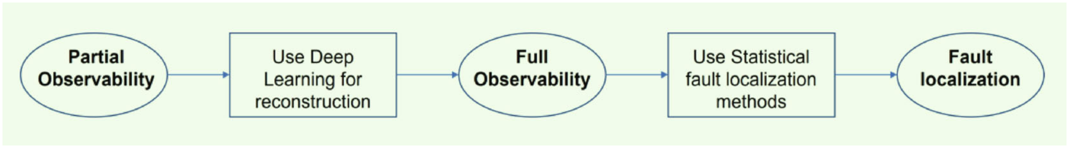
Electric power systems are one of the most crucial infrastructures of any nation or economic area. The functioning of any modern economy and society depends on an efficient power transmission system. Cyberattacks that make information about parts of the power system unavailable or corrupted are a distinct possibility even if these parts are not physically damaged. Natural disasters, like floods or hurricanes, military action or terrorist acts can damage parts of the power grid and cut off information from affected sensors.

Our objective is to show that graph neural networks (GNNs) can be used to recover enough information to ensure that statistical fault detection can be used even if no information is streamed from parts of a regional transmission system. We show how these deep learning networks must be constructed and explain which forms of GNNs work well and which do not. Our conclusions are based on statistical analysis of an exceptionally large database of faults simulated in a realistic model of a regional transmission system. Statistical anomaly detection, including deep learning methods, is a rapidly expanding area, see, for example, Harrou et al. (2020), and GNNs are receiving growing attention, for example, Zhou et al. (2020) and Chen et al. (2021). There have been many recent advances in GNNs. For example, Hamilton et al. (2017), Abu-El-Haija et al. (2019) and Ying et al. (2021). In the brief review below, we focus on most relevant contributions within the framework of power systems.

Faults in power systems cause excessive currents and pose safety threats to people and property, and may cause major fires with substantial economic and social impacts. Therefore, there is an increasing interest in fault detection and localization to reduce the damage to the system and inconvenience to customers. This motivates the development of new methods to complement, or potentially replace, traditional methods. Data-driven methods can now be easily implemented due to the installation of phasor measurement units (PMUs), Zhang et al. (2012); Xu et al. (2014); Kokoszka et al. (2023), among many others. In principle, the entire power system can be measured and monitored by using PMUs on each electric grid bus. However, such placement could be costly and prone to increased device failure rates Shafiullah et al. (2022). Thus, the research on fault detection and localization without the assumption of full network observability has principal importance. This paper concentrates on the issue of locating a faulted line in a transmission network under partial network observability.

Fault detection and localization under partial observability of the power grid using machine learning have been the subject of study in recent years. Li et al. (2019) proposed a methodology to detect a faulted line employing classification based on convolution neural networks (CNN) using bus voltages and discussed the optimal PMUs placement. The study by Afonin and Chertkov (2021) primarily focuses on using deep learning classification methods to locate faults in a network under partial observability. They adopt a similar setting as Li et al. (2019), but investigate a wider range of models: linear regression (one-layer neural network), feed-forward neural networks, AlexNet, and graph convolutional neural networks. All of these models are tuned in to predict the location of the faulted line. The performance of the models depends upon the location of PMU devices. Zhao and Barati (2021) proposed a methodology for fault localization in distribution networks based on a novel feature vector design and CNN classification. The feature vector is built using the differences in voltage and current before and after the fault, the phase angle's difference of each phase before and after the fault, and zero, negative, and positive sequences for phase voltages and currents. Results were tested on a real power distribution feeder with 25 possible locations for the fault. Han et al. (2022) proposed to locate the faults by converting multichannel electrical signals to similarity images. Using CNNs, which are traditionally used for image classification, the authors apply algebraic operations to power signals and convert them to images. The model was tested on a fully observed 24-bus system. The authors argued that the model they proposed was robust to changes in the topology of the power grid. Devi et al. (2018) and Zainab et al. (2019) proposed a similar method as Li et al. (2019), but with additional utilization of electrical area information. As the above review shows, the work done so far considers the limited availability of measuring devices (PMUs) to build classification models for fault localization. Some work considers the approach where one trains the model on limited PMUs and examines its performance Devi et al. (2018); Li et al. (2019); Afonin and Chertkov (2021). There are several limitations to the above classification-based approaches. First, if a particular measuring device fails, the model needs to be re-trained. Second, classification models provide only 0-1 answers without additional information about the unobserved readings. Third, these models do not incorporate the fact that with full PMU data availability, the current fault detection/localization methods work with high accuracy Salehi-Dobakhshari and Ranjbar (2014); Furse et al. (2021). We emphasize that classification-based methods perform very well when trained on rich and appropriately selected data. However, during training, they must know the location of the fault, even if the fault occurs in an area of the grid that is not covered by measuring devices.

Rather than proposing a new fault localization method applicable under partial network observability, this study aims to bridge the gap between limited PMU availability and fault localization methods that are highly accurate under the assumption of the full PMU coverage of the power grid. We propose deep learning networks that can reconstruct data at locations where data are missing. With such an approach, any method that works well under full grid observability can be applied. Our approach is thus akin to data imputation, but with a specific focus on fault localization. It is summarized by the following flow diagram:


Partial Observability
Use Deep Learning for reconstruction
Full Observability
Use Statistical fault localization methods
Fault localization

Our approach is related to self-supervised learning that can be viewed as an extension of model training enabling to obtain competitive results with less available training data, for example, You et al. (2020). Our final recommended method aligns with the principles of self-supervised learning. In the self-supervised learning paradigm, one initially pre-trains the network to accomplish a different task using the same data and subsequently proceeds with using the pre-trained model to train for the primary task. In our case, we utilized network pre-training to achieve imputation (which is not the primary task of fault localization) and subsequently employed the results in a statistical approach to suggest the faulty location. Thus, the difference from self-supervised learning is that we propose for the last step to use a statistical approach instead of a neural network, for reasons noted above.

The topic of missing data imputation in power grids has gained attention in recent years. Zhu and Lin (2021) propose a method that utilizes spatiotemporal correlations for imputing missing PMU data. This method is applied in situations where device malfunctions and communication failures lead to poor PMU measurements and data loss. The approach utilizes not only past information but also a global and local spatial perspective in order to achieve a higher accuracy of data imputation. Foggo and Yu (2022) fill in missing values through the use of two components: a non-dynamic component that is predicted using past data, and a dynamic component that is inferred from all other available PMUs. The proposed model corrects past data and incorporates events data that can be inferred from all available PMUs data for the modulus of voltage. However, these methods only address the missing data scenarios after the failure of the PMUs and assume that the past data are available. Dynamic state estimation is another area where significant progress has been made, Jakir and Rahnamay-Naeini (2021); Park et al. (2023). These authors used deep learning methods and a large amount of diverse data to learn the relationship between power grid nodes under normal operating conditions. Our work has a different focus. We use as little data as possible, namely the modulus of voltage, to reconstruct trajectories *during a fault* with a focus of using them in statistical fault localization methods. Our data are described in Section 2.

We are aware of only one study in the field of power grids that concentrates on employing reconstructed trajectories in certain fault detection techniques, namely Li and Deka (2021) who developed a physics-informed learning approach for detecting high impedance faults. They employed auto-encoders for feature learning and utilized unlabeled data for training, which is similar to our methodology. Our research focuses on utilizing GNNs for reconstructing missing trajectories, with specific applications in fault localization. We offer techniques for enhancing the accuracy of the reconstruction process and propose models that can be utilized in various data availability scenarios. Most research in this area has relied on labeled data (fault locations are known). Our work addresses the challenge of localizing faults in situations where obtaining labeled data is difficult or expensive. We propose employing GNNs to impute data trajectories at buses without PMU devices during a fault in a power grid, and using these data in an existing fault detection technique proposed in Kokoszka et al. (2023). Different techniques could be used, but they are difficult to employ in our experimental test bed of the Western Regional grid due to the lack of deployable code. We explore different feature transformations, loss functions, and strategies for optimal PMU placement. We emphasize that our methodology uses only unlabeled data, that is, the location of a fault is not available during network training. Moreover, our PMU data reconstruction methodology contributes to research on mitigation of cyberattacks on power grids, as we show that corrupted PMU readings can be replaced by imputed readings from our model.

The paper is organized as follows. Section 2 introduces the transmission system and the grid data we work with. In Section 3, we formulate the problem and focus of the paper in greater detail. In Section 4, we describe the methods we propose for data reconstruction using GNNs. We examine various loss functions, optimal PMU placement, and success rates for fault localization. Section 5 reports the performance of various of our method, while Section 6 summarizes our contribution and main conclusions.

2 | DATA DESCRIPTION

We work with data generated using the miniWECC system, which is a reduced-order dynamic model of the Western Electricity Coordinating Council system. The minniWECC has enough complexity to reflect the relevant properties of the full Western Interconnection's bulk power system, which serves over 80 million customers in 14 US states and two Canadian provinces. The simulations conducted with the minniWECC model have been used to work on various power system mode estimation and event detection algorithms, which are now adopted for control room applications, see, for example, Follum et al. (2017); Trudnowski et al. (2013); Byrne et al. (2016). A more detailed description of the miniWECC can be found Trudnowski et al. (2013). A simplified one-line diagram of the minniWECC is given in Figure 3 in Section 5. Only high-voltage lines are shown to make the graph readable. We use a version of the minniWECC that consists of 158 lines between 122 buses (nodes). Some buses can be physically connected by more than one line, but we treat all these lines as a single connecting line because measurements at end buses cannot distinguish between them. Thus, each connection $l \in \{1, \dots, 158\}$ has two buses $(i,j)\,:\, i,j \in \{1, \dots, 122\},\ \ i \neq j$. We consider only pairs (i,j) for which there is a line between buses i and j.

We simulated a large number of faults in the minniWECC using the Power System Toolbox (PST), Cheung et al. (2009), which is based on MATLAB. The description of the fault-generating mechanism in the next paragraph is technical and requires some background in power systems. The essence is that we generated a very large number of faults that can realistically occur in a regional transmission system. Our statistical approach does not require the knowledge of the fault type or its characteristics. *If the fault occurs closer to bus i, we call i the near-end bus and j is the far-end bus.*

The faults were generated using the switching condition matrix. The values of the zero sequence impedance, $z0$, and the negative sequence impedance, zn, were generated randomly for each simulation. System characteristics imply that they can be generated as random variables uniformly distributed on the interval [0.0004, 0.189] ohms. Using the random values of $z0$ and zn, four types of line faults can be generated: line-to-line (LL), line-to-ground (LG), line-to-line-to-ground (LLG), and three-phase (TP) faults. Furthermore, the location of the fault on a line between buses i and j can also be randomly simulated. Ignoring the randomness of $z0$ and zn, 1,264 unique faults (158 lines $\times$ 4 types $\times$ 2 ends). For each random pair $(z0, zn)$, each of the 1,264 options is chosen randomly. We added ambient noise to each simulation using MinniWECC parameters described in Trudnowski et al. (2013), which also increases the randomness, but not crucially, as will become apparent in the following. The total number of distinct faults that can be simulated is practically unlimited.

The simulated data consist of 120 measurements per second for 10 s. Define the time resolution as $\delta = 1/120$. For this paper, we only consider $\frac{32}{120}$ s before the fault and $\frac{32}{120}$ after (in total, 64 time stamps). We assume that each simulation starts at $t = 0$, the fault is applied at $t = \frac{32}{120}$s, and data are recorded up to $t = \frac{64}{120}s$. We set the time of the fault as $t_f = \frac{32}{120}$. The considered length of simulation is approximately equal to 0.5 s. We use only the modulus of voltage, which is sufficient to localize faults.

Figure 1 shows 500 randomly chosen readings at bus 105. Figure 2 shows responses at all buses to three faults applied between bus 97 and bus 66, with bus 97 being a near-end bus. It shows how all buses see the faults. The responses at all buses exhibit similar patterns, but there is a possibility to identify the faulted line, at least the near-end bus. More detailed visual analysis reveals that faults of certain types are more difficult to localize. The TP faults show much less variability between the lines that would allow to identify the faulted line. Our methods are applied without assuming any knowledge of the fault type.

3 | PROBLEM FORMULATION

In this section, we introduce suitable notation, define the problem, describe the tasks, and explain the challenges that arise. We model the system as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{1, 2, \dots, 122\}$ is the set of buses, including both those with PMU devices and those without them, and $\mathbf{E}$ is the set of lines connecting the buses. Notice that if $(i,j) \in \mathbf{E}$, then $(j, i) \in \mathbf{E}$. The neighborhood of bus i is defined as the set $\mathbf{N}_i = \{j\,:\,(i,j) \in \mathbf{E}\}$ of buses that are connected to i by a line. We assume that the power grid $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is fixed.

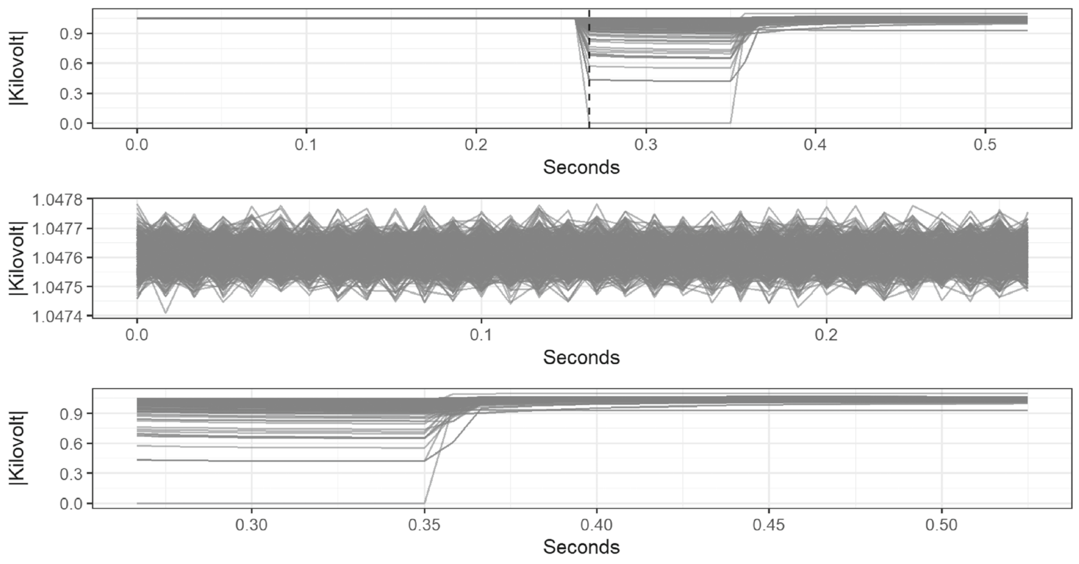


FIGURE 1 Randomly selected modulus of voltage readings at bus 105. The three graphs represent different time stages of simulations. The upper panel represents the whole trajectories, while the lower two represent trajectories before the fault, and after the fault. Notice the different voltage scales at the different stages.

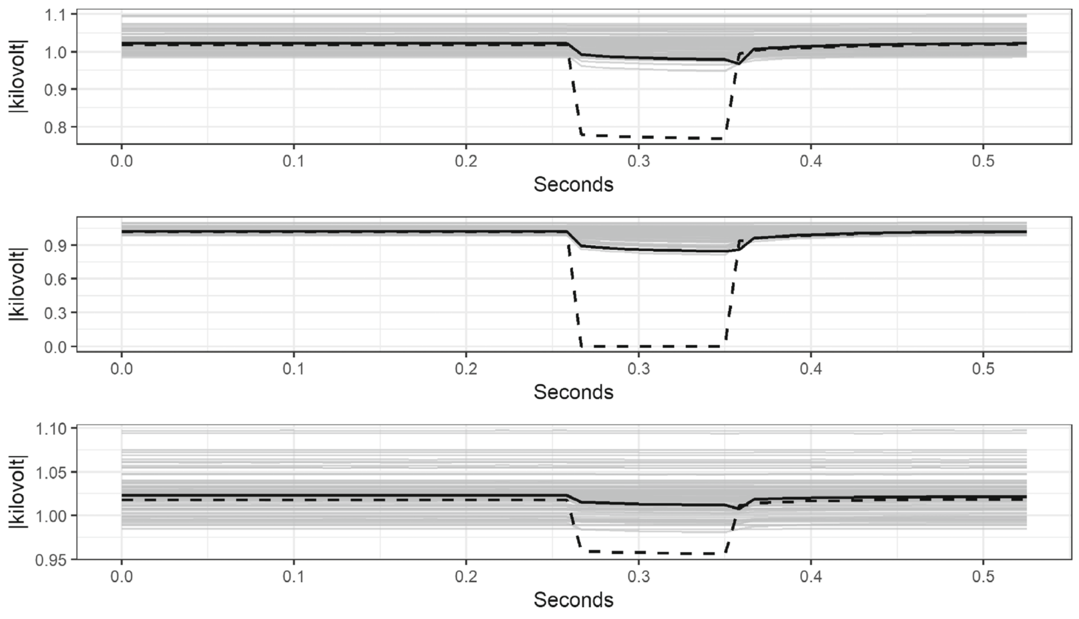


FIGURE 2 Responses at all buses to three faults between bus 97 (near end) and bus 66. Responses at bus 66 are highlighted in solid black and those at bus 97 in dashed black. The three panels represent different types of faults with different parameters.

We assume that a fault occurs at an unknown line $l_0 \in \mathbf{E}$. We consider a prediction $\hat{l}_0$ of the faulted line to be correct if $\hat{l}_0 = l_0$. Additionally, we consider a localization to be partly correct if $\hat{\imath}_0 = i_0$, where $\hat{\imath}_0$ is the predicted near-end bus of the faulted line $\hat{l}_0$. The localization failure ratio is defined by the ratio of incorrect predictions and the total number of simulations.

Our framework assumes that PMUs are available only at a certain subset of buses, which we denote by $\mathbf{K} \subset \mathbf{V}$. The count of buses in $\mathbf{K}$ is denoted by $K := |\mathbf{K}|$. At buses in $\mathbf{K}$, we have records of modulus of voltage during a fault, with m data points before the known fault time t_f and m points after. This leads to a feature matrix $\mathbf{X}_K \in \mathbb{R}^{2m\times K}$. The set of unobserved buses is denoted as the $\mathbf{U} = \mathbf{V} \setminus \mathbf{K}$ with $|\mathbf{U}| = U$. The trajectories at these buses are denoted as $\mathbf{X}_U \in \mathbb{R}^{2m\times U}$. Denote by $\mathbf{X} \in \mathbb{R}^{2m\times(U+K)}$ the dataset under full availability of measuring devices.

The objective is to find a function $f(\mathbf{X}_K) = \widehat{\mathbf{X}}_U$ that satisfies two conditions:

1. minimizes a suitable loss function $\ell(\mathbf{X}_U, \widehat{\mathbf{X}}_U)$;
2. minimizes the difference in fault localization accuracy based on $\widehat{\mathbf{X}} = \mathbf{X}_K \cup \widehat{\mathbf{X}}_U$ (the available and reconstructed values) relative to the same method based on $\mathbf{X}$ (the full dataset).

Objective 2 can be seen as applying a discrete, problem-focused loss function. The optimal reconstruction should produce trajectories that lead to fault localization that is as close as possible to the localization based on the completely observed system. To our knowledge, no previous work has addressed the task of training the model for imputing trajectories during a fault without using information of the fault location with the intention to employ the trained model for fault localization. We use the method of Kokoszka et al. (2023), but any other method for which code applicable to minniWECC is available could be used. Our statistical evaluation approach could also be used in different testbeds.

We consider two scenarios to obtain f: (1) The training process of f uses $\mathbf{X}_U$, (2) The training process of f does not use $\mathbf{X}_U$. These two scenarios differ in the amount of information used during the training process, and can be applied to different real-world situations. As we emphasized before, in either scenario, during the training process, the information about which line is faulted is not included. We refer to it as working with unlabeled data. In either case, the function f is a network that is suitably trained, as described in the following.

We also explore the following options to potentially improve trajectory reconstruction and fault detection:

1. investigate whether different choices of the loss function ℓ increase the accuracy of fault localization;
2. examine feature transformation techniques that could help increase the accuracy;
3. investigate selecting an optimal set of $\mathbf{K}$ (optimal PMU placement).

In the next section, we only describe the methods that are used to obtain the final, tabulated or displayed, results. However, in Section 5, we also discuss some other techniques and options that we experimented with.

4 | METHODS

In this section, we describe the proposed methodology. We begin with a brief account in Section 4.1 of GNNs focusing on aspects relevant to the methods we propose. In Section 4.2, we explain the details of the proposed development of GNNs to make them the applicable to PMU data reconstruction with the ultimate objective of fault localization. The remaining subsections discuss the loss functions, selection of an optimal set $\mathbf{K}$, a fault localization scheme used after trajectories in **U** have been reconstructed, and a benchmark model against which the performance of our method is judged.

The development of the methodology in this section is analogous to the development of traditional statistical methodology. We work with data objects that have the form of structured $n \times (2m)$ matrices, and we basically want to solve a missing data problem with a specific objective in mind. For this, we need to propose a model for the data and show how it can be estimated and used for purpose-focused data imputation. The model is formulated in terms of the layers of a deep learning network. There are no explicit formulas for estimating the parameters of the layers: they must be estimated through a training process that must be specified. Details of such approaches are presented, for example, in Goodfellow et al. (2016) and Zhang et al. (2023), where terminology used in the remainder of this paper is also explained.

4.1 | Graph neural networks

GNNs utilize graph data structures as inputs to learn and make predictions. They have seen a rise in popularity due to their ability to generalize the convolution operator to graph structures, Kipf and Welling (2017). These networks have proven to be effective in many applications, such as drug discovery, recommendation systems, social networks, molecular structures, and electrical grids, see Li et al. (2018); Yu et al. (2018), among many others. Reviews of the of GNN are given by Xu et al. (2018) and Wu et al. (2021). GNNs have been extensively studied in recent years, with significant progress in the development of various architectures and techniques for improving their performance, for example, Zhou et al. (2020); Hamilton (2020). Under suitable assumptions, a GNN model can approximate in probability all functions on graphs with any required precision, Scarselli et al. (2009). Liao et al. (2022) discuss their applications in power systems. Recently, models that combine kriging convolution networks and GNNs have been proposed Hamilton et al. (2017); Wu et al. (2020); Appleby et al. (2020).

One of the key components of GNNs is the use of graph convolutional layers designed to capture the local and global structure of the input graph. The computation is driven by the input graph topology, which is described by its adjacency matrix. The adjacency matrix **A** of a graph **G** with n nodes is an $n \times n$ matrix such that $\mathbf{A}(i,j) = 1$ if there is an edge between nodes i and j, and $\mathbf{A}(i,j) = 0$ otherwise. In our setting, nodes are buses, and edges are the lines connecting them. The first layer of a GNN typically encodes the graph structure and the initial node features, while subsequent layers aggregate information from the node's neighbors and update the node's representation. The final layer of a GNN produces the output, which can be used for tasks such as node classification, link prediction, or recommendation. These layers use the adjacency matrix of the graph and trainable weight matrices to compute the outputs according to the equation

$$\mathbf{H}_{l+1} = \sigma(\mathbf{H}_l; \mathbf{A}, \mathbf{\Theta}_{l+1}),$$

where $\mathbf{H}_l$ is the output of the previous convolutional layer, **A** is the adjacency matrix, $\mathbf{\Theta}_{l+1}$ is the weight matrix for the layer $l+1$, and σ is an activation transformation.

Most inductive GNNs are designed to generalize to new, unseen nodes in a graph. In this paper, we assume that the graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is fixed, which does not fully align with the definition of inductive learning. However, our model needs to generalize to buses that are in **G**, but their trajectories are unseen. To incorporate different data availability scenarios listed in Section 3, we consider two approaches:

1. the training process of f can use both known and unknown trajectories $\mathbf{X}_K$ and $\mathbf{X}_U$, respectively. In this approach, the whole information about $\mathbf{E}$ is used, and the training set patterns and their targets are used for training. This approach is similar to traditional regression models. We call it *regression graph neural network* (RGNN);
2. the training process of f cannot use unknown trajectories $\mathbf{X}_U$ and thus needs to learn from the known trajectories $\mathbf{X}_K$ with a knowledge of fixed edges $\mathbf{E}$. We call this approach *semi-inductive graph neural network* (SIGNN).

We need to predict the trajectories at unseen nodes, but the training can use the information about these buses' existence. Such information is crucial for letting the network properly learn the message-passing mechanism. The work of Hamilton et al. (2017) showed that our considered approaches are a reasonable attempt to tackle the challenges listed in Section 3. Although there are more methods to tackle the problem under Scenario (1) than under Scenario (2), we chose to use the same model concept for both to better understand the challenges that need to be addressed when moving from (1) to (2). Moreover, efficient results under Scenario (1) can demonstrate the proof of concept and provide a foundation for addressing the challenges of Scenario (2). Using the same model for both scenarios, we can also investigate how the model's performance changes with respect to the availability of data.

4.2 | GNNs for PMU data reconstruction

In this section, we describe how we propose to use the RGNN and the SIGNN for the purpose of PMU data reconstruction. We use the GNN architecture proposed by Wu et al. (2020) as the basis of our work and develop it to fit our needs. The initial architecture uses an L-layer Diffusion Graph Convolution Network (DGCN), Li et al. (2018). The DGCN performs the following operations:

$$\mathbf{H}_{l+1} = \sum_{k=0}^{\mathcal{K}} T_k(\overline{\mathbf{A}})\mathbf{H}_l\boldsymbol{\Theta}_l^{(k)},$$

where $\mathbf{H}_l$ is the output feature matrix of layer l, $\overline{\mathbf{A}}$ is the transition matrix calculated from the adjacency matrix $\mathbf{A}$, $\mathcal{K}$ is the maximal order of diffusion convolution, $T_k(x) = x^k$ (pointwise). We denote by $\boldsymbol{\Theta}$ the set of all weight matrices $\boldsymbol{\Theta}_l^{(k)}$. In the following, we use the term *learning*, to describe finding the set $\boldsymbol{\Theta}$ such that some loss function ℓ is minimized. The dimension of matrices $\boldsymbol{\Theta}_l^{(k)}$ can be determined by the user by specifying the *width of the network*, z. Specifically,

$$\boldsymbol{\Theta}_1^{(k)} \in \mathbb{R}^{2m\times z}, \quad \boldsymbol{\Theta}_l^{(k)} \in \mathbb{R}^{z\times z}, \; 2 \le l \le L-1, \quad \boldsymbol{\Theta}_L^{(k)} \in \mathbb{R}^{z\times 2m}. \tag{1}$$

Based on many numerical experiments, we determined that, in our context, better results are obtained using the network

$$\mathbf{H}_l = \sigma\left(\sum_{k=0}^{\mathcal{K}} T_k(\overline{\mathbf{A}})\mathbf{H}_{l-1}\boldsymbol{\Theta}_l^{(k)}\right) + \mathbf{H}_{l-1}, \quad 1 \le l \le L, \tag{2}$$

where σ the scaled exponential linear unit (SELU) activation function, Klambauer et al. (2017), with standard parameters. The initial input is the matrix $\mathbf{H}_0 \in \mathbb{R}^{n\times 2m}$, where $n = 122$ is the number of buses and $m = 32$ is the count of time points before and after the fault used in our algorithms. We thus use only 64 measurements corresponding to the interval of about 0.53 s centered at the time of the fault. The input $\mathbf{H}_0$ has the unobserved trajectories masked, as described in Algorithms 1 and 2 below. The masked buses only pass 0 to their neighbors in the first layer, while the remaining $L - 1$ layers are responsible for more generalized representations. The Lth layer is used to output the reconstructed trajectories for each bus. Thus, $\mathbf{H}_L \in \mathbb{R}^{n\times 2m}$ is the reconstruction results that we use to calculate the loss and update weights $\boldsymbol{\Theta}$. In our experiments, we explored different structures and building pieces (GAT, Chebynet, different activation functions) that would bring the gain in fault localization accuracy without loss in efficiency. Through the series of experiments, we found that the proposed structure is the most efficient for the specific problem we aim to solve. In many applications $L = 3$ is sufficient; however, for power grid data, deeper networks may be more effective, as demonstrated in a different context by Ringsquandl et al. (2021). The parameters $\mathcal{K}$, the order of convolution, and z, the number of hidden channels of the weight matrix $\boldsymbol{\Theta}_l^{(k)}$, may also play an important role. Using small $\mathcal{K}$ values allows us to keep the diffusion process more localized in a power grid, while the parameter z controls the number of parameters for each filter $T_k(\overline{\mathbf{A}})\mathbf{H}_{l-1}$. It is not practical to find the optimal values of L, $\mathcal{K}$, and z for each model and setup. We performed a limited grid search to find the optimal settings for L, $\mathcal{K}$ and z. We discuss the results in Section 5. Other issues that must be addressed are: implementing a validation scheme in order to choose the best model and getting reconstruction results from $\mathbf{H}_L$. We proceed with detailed descriptions of all steps.

4.2.1 | Detailed description of the RGNN implementation

The learning process utilizes information from $\mathbf{K}$ to learn the trajectories of the voltage modulus in $\mathbf{U}$, and records the results of the loss function. The loss function is then used to update the parameters $\boldsymbol{\Theta}$. To achieve this, we utilize a mask matrix denoted as M_{sample}, which keeps buses in $\mathbf{K}$ observed and buses in $\mathbf{U}$ unobserved. We calculate the Kronecker product of this matrix and training data, and run the result through layers of GNN and compute the results using the true readings in $\mathbf{U}$. The Kronecker product is denoted by $\otimes$ in line 7 of Algorithm 1. This operation sets to zero all rows of the matrices in $\{\mathbf{X}_{1:S}\}$ that correspond to buses in $\mathbf{U}$. Therefore, the information about the trajectories in $\mathbf{U}$ is only used during the loss step and not within the network. Please refer to Algorithm 1 for all steps. Recall that we denote by $\mathbf{X}$ the $n \times (2m)$ matrix obtained from a single fault simulation. In Algorithm 1, we denote by $\mathbf{X}^{\text{Tr}}$ the set of simulations used in training (we use 10,000 of them) and by $\mathbf{X}^{\text{Val}}$ the simulations used for validation (we use 2000). Due to the structure of the model, it is recommended to use a high number of iterations.

4.2.2 | Detailed description of the SIGNN implementation

For each training iteration, we randomly divide buses from $\mathbf{K}$ into $\mathbf{K}_0$ (prediction buses) and $\mathbf{K}_1$ (target buses). The size of these sets is K_0 and K_1, respectively. The training process utilizes information from $\mathbf{K}_0$ to learn the trajectories of modulus of voltage in $\mathbf{K}_1$ and records the results of the loss function. We utilize a mask matrix, denoted M_{sample}, to keep buses in $\mathbf{K}_0$ as observed and buses in $\mathbf{K}_1$ and $\mathbf{U}$ as unobserved, see Algorithm 2. The size K_1 is randomly selected as a

Algorithm 1. Random mask generation and model learning for RGNN

```
Input: Training data $\mathbf{X}^{\mathrm{Tr}}$, validation data $\mathbf{X}^{\mathrm{Val}}$, graph structure $\mathbf{G}$, length of each simulation $2m$,
    observed buses $\mathbf{K}$ and unobserved buses $\mathbf{U}$; Parameters: size of each iteration $S$, maximum
    iteration $I_{\max}$, maximum epoch $E_{\max}$, loss function $\ell$.
 1: for epoch $= 1 : E_{\max}$ do
 2:     for iteration=1 : $I_{\max}$ do
 3:         for sample $= 1 : S$ do
 4:             Randomly choose a simulation $\mathbf{X}_{\mathrm{sample}}$ from $\mathbf{X}^{\mathrm{Tr}}$.
 5:             Generate a mask matrix $M_{\mathrm{sample}}$ of the same size as $\mathbf{X}_{\mathrm{sample}}$,
                    $M_{\mathrm{sample}}[i,:] = \begin{cases} 1 & \text{if } i \in \mathbf{U} \\ 0 & \text{otherwise} \end{cases}$
 6:         end for
 7:         Use sets $\{\mathbf{X}_{1:S}\} \otimes M_{1:S}$ and $\ell$ to train GNNs (update weights matrix $\boldsymbol{\Theta}$)
 8:     end for
 9:     Calculate validation error $Err_{\mathrm{epoch}}$ using $\ell$ and $\mathbf{X}^{\mathrm{Val}}$ with a predictor/response split $\mathbf{K}$
    and $\mathbf{U}$.
10: end for
11: Final learned model is GNN with weight matrix $\boldsymbol{\Theta} = \boldsymbol{\Theta}_{\arg\min_{\mathrm{epoch}\in\{1,\dots,E_{\max}\}} Err_{\mathrm{epoch}}}$
```

number between 2 and 7 for each iteration. During this process, the adjacency matrix $\mathbf{A}$ remains fixed. The information about the existence of $\mathbf{U}$ is contained only in $\mathbf{A}$, no measurements in $\mathbf{U}$ are used. The resampling of $\mathbf{K}_0$ and $\mathbf{K}_1$, in principle, allows the model to learn the relationships between different trajectories, and generalize results over the nodes in $\mathbf{U}$.

To ensure that the validation error is measured uniformly for each epoch, it is necessary to determine sets $\mathbf{K}_0^{\mathrm{Val}}$ and $\mathbf{K}_1^{\mathrm{Val}}$ before the training procedure. To better understand the validation error over nodes that are not seen during training, we randomly select 3 nodes from $\mathbf{K}$ to create $\mathbf{K}_1^{\mathrm{Val}}$. During training, we effectively have $\mathbf{U}$ to include $\mathbf{K}_1^{\mathrm{Val}}$ and subtract it from $\mathbf{K}$ (i.e., $\mathbf{U} := \mathbf{U} \cup \mathbf{K}_1^{\mathrm{Val}}$ and $\mathbf{K} := \mathbf{K} \setminus \mathbf{K}_1^{\mathrm{Val}}$). As a result, the model cannot access trajectories at $\mathbf{K}_1^{\mathrm{Val}}$ during training, but uses them to compute the validation error. This helps us to understand the generalization power of the model, and choose settings that give optimal results for the test error.

The learning procedure is summarized in Algorithm 2. We emphasize that Step 8 in Algorithm 2 guarantees that we preserve fixed graph structure $\mathbf{G}$, but ensure that the training process has no access to trajectories of buses without PMU devices.

4.2.3 | Reconstruction

We explain the process of reconstructing the trajectories in the unseen nodes $\mathbf{U}$ for each simulation (sample) using both SIGNN and RGNN. Assume we have trained the model and obtained the weight set $\boldsymbol{\Theta}$. First, we form masked signals $\mathbf{X}^{M}_{\mathrm{sample}} = [\mathbf{X}_K, \mathbf{X}_U]$ with $\mathbf{X}_U = \mathbf{0}$ and set $\mathbf{H}_0 := \mathbf{X}^{M}_{\mathrm{sample}}$. We input obtained $\boldsymbol{\Theta}$ and $\mathbf{H}_0$ to the GNN described architecture and obtain $\mathbf{H}_M$. Set $\widehat{\mathbf{X}}^{M}_{\mathrm{sample}} := \mathbf{H}_L$. The $\widehat{\mathbf{X}}^{M}_{\mathrm{sample}}$ can be further split into $\widehat{\mathbf{X}}^{M}_{\mathrm{sample}} = [\widehat{\mathbf{X}}_K, \widehat{\mathbf{X}}_U]$. The GNN architecture outputs the fit of

Algorithm 2. Random mask generation and model learning for SIGNN

```
Input: Training data $\mathbf{X}^{\mathrm{Tr}}$, validation data $\mathbf{X}^{\mathrm{Val}}$, graph structure $\mathbf{G}$, length of each simulation
    $2m$, observed buses $\mathbf{K}$ and unobserved buses $\mathbf{U}$; Parameters: size of each batch $S$, maximum
    iteration $I_{\max}$, maximum epoch $E_{\max}$, loss function $\ell$, fixed $\mathbf{K}_1^{\mathrm{Val}}$ set for validation.
 1: for epoch = 1 : $E_{\max}$ do
 2:     for iteration=1 : $I_{\max}$ do
 3:         Generate random integer $k_1 \in \{2, \dots, 7\}$.
 4:         Randomly sample $k_1$ buses from $\mathbf{K} \setminus \mathbf{K}_1^{\mathrm{Val}}$ to form $\mathbf{K}_1$.
 5:         Set $\mathbf{K}_0 = \mathbf{K} \setminus (\mathbf{K}_1 \cup \mathbf{K}_1^{\mathrm{Val}})$.
 6:         for sample = 1 : $S$ do
 7:             Randomly choose a simulation $\mathbf{X}_{\mathrm{sample}}$ from $\mathbf{X}^{\mathrm{Tr}}$.
 8:             Set rows of $\mathbf{X}_{\mathrm{sample}}$ corresponding to buses in $\mathbf{U}$ and $\mathbf{K}_1^{\mathrm{Val}}$ to 0.
 9:         end for
10:         Generate a mask matrix $M_{\mathrm{sample}}$ of the same size as $\mathbf{X}_{\mathrm{sample}}$,

                $$M_{\mathrm{sample}}[i,:] = \begin{cases} 1 & \text{if } i \in \mathbf{K}_0 \\ 0 & \text{otherwise} \end{cases}$$

11:         Use sets $\{\mathbf{X}_{1:S}\} \otimes M_{1:S}$ and $\ell$ to train GNNs (update weights matrix $\mathbf{\Theta}$)
12:     end for
13:     Calculate validation error $Err_{\mathrm{epoch}}$ using $\ell$, $\mathbf{X}^{\mathrm{Val}}$ with a predictor/response split $\mathbf{K}_0^{\mathrm{Val}}$ and
    $\mathbf{K}_1^{\mathrm{Val}}$, and obtained weight matrix $\mathbf{\Theta}$. Corresponding buses in $\mathbf{U}$ and $\mathbf{K}_1^{\mathrm{Val}}$ of $\mathbf{X}^{\mathrm{Val}}$ are equal 0.
14: end for
15: Final learned model is GNN with weight matrix $\mathbf{\Theta} = \mathbf{\Theta}_{\arg\min_{\mathrm{epoch}\in\{1,\dots,E_{\max}\}} Err_{\mathrm{epoch}}}$
```

the trajectories at all buses, but the final product of reconstruction is only $\widehat{\mathbf{X}}_U$. During the training process of RGNN, the intermediate results have the same structure as the final reconstruction, but that is not the case for SIGNN. In SIGNN, a similar structure is used to obtain reconstruction results within each iteration, but $\mathbf{X}^M_{\mathrm{sample}}$ is replaced with $\mathbf{X}^M_{sample,K_0,K_1} = [\mathbf{X}_{K_0}, \mathbf{X}_{K_1}, \mathbf{X}_U]$, where $\mathbf{X}_{K_1} = 0$ and $\mathbf{X}_U = \mathbf{0}$.

4.3 | Loss functions

In addressing the problem formulated in Section 3, a typical approach would be to use the mean squared error (MSE) loss function. However, this may not always be the best approach. Most deep learning methods rely on the MSE or its variants (such as MAE and quantile loss). In our case, the primary goal is to accurately predict the fault location, which may not be best achieved through the MSE alone. To address this, we have explored alternative loss functions, including bias-weighted MSE, normalized soft time warping, and others Cuturi and Blondel (2017); Guen and Thome (2022). We only discuss the MSE and the Bias-weighted MSE, as they are used to report results. The methods are additionally evaluated by their ability to minimize fault detection failure rate, see Section 4.5, which can also be seen as a loss function, even though it is not used in training, but is used to arrive at final recommendations.

To simplify the explanation, let us consider the case of evaluating the loss function for a single simulation. This can be easily extended to multiple simulations by taking the average. In this section, we define the matrix of reconstructed trajectories as $\widehat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_r)$, and the true matrix is defined as $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_r)$, where r is the number of reconstructed trajectories. Further, define each reconstructed trajectory $\hat{\mathbf{y}}_i = (\hat{y}_{i,1}, \dots, \hat{y}_{i,2m})$ in terms of $2m$ data points. Similarly, set $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,2m})$.

The MSE loss function is defined as:

$$\ell_{\mathrm{MSE}}(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{r}\sum_{i=1}^{r} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2 = \frac{1}{r}\sum_{i=1}^{r}\sum_{j=1}^{2m} (\hat{y}_{i,j} - y_{i,j})^2.$$

For the bias-weighted MSE, we propose to decompose the MSE into two components: the difference in means (bias) and the difference in variance. This is motivated by the statistical fault localization technique employed in this paper, where a significant bias in the reconstruction has a lesser impact on the fault localization accuracy compared with a large difference in shape, approximated by the variance. The bias of prediction for each $i \in \{1, \dots, r\}$ is defined as $b_i = \bar{y}_i - \bar{\hat{y}}_i$, where the averages are taken of the $2m$ data points. We then define the bias-adjusted reconstruction matrix as $\widehat{\mathbf{Y}}_{ba} = (\hat{y}_1 - b_1, \dots, \hat{y}_r - b_r)$. With a few simple steps, it can be shown that

$$\ell_{\mathrm{MSE}}(\widehat{\mathbf{Y}}, \mathbf{Y}) = \ell_{\mathrm{MSE}}(\widehat{\mathbf{Y}}_{ba}, \mathbf{Y}) + \frac{2m}{r}\sum_{i=1}^{r} b_i^2.$$

We propose weighting these two parts of $\ell_{\mathrm{MSE}}(\widehat{\mathbf{Y}}, \mathbf{Y})$, putting more emphasis on the first part. Using this notation and including a tuning parameter $\alpha \in (0, 1)$, we define the bias-weighted MSE as

$$\ell_{\mathrm{BMSE}}(\widehat{\mathbf{Y}}, \mathbf{Y}) = \alpha \ell_{\mathrm{MSE}}(\widehat{\mathbf{Y}}_{ba}, \mathbf{Y}) + (1 - \alpha)\frac{2m}{r}\sum_{i=1}^{r} b_i^2.$$

The hyperparameter α is determined during the training process. Notice that if $\alpha = 0.5$, $\ell_{\mathrm{BMSE}} = \frac{1}{2}\ell_{\mathrm{MSE}}$.

4.4 | Optimal PMU placement

Our framework assumes that PMUs are available only at a subset of buses $\mathbf{K} \subset \mathbf{V}$. A question of interest is what the locations $\mathbf{K}$ should be to minimize aggregated loss functions and the fault localization failure rate, given that $K = |\mathbf{K}|$ is fixed. A brute-force search for the optimal $\mathbf{K}$ by testing every possible configuration given K is infeasible due to the astronomical number of possibilities. With $K = 70$ and 122 possible locations for PMUs, there are approximately 1.02E+35 possible combinations for different placement of PMUs.

PMU placement algorithms discussed in the literature, for example, Aminifar et al. (2010); Enshaee et al. (2012); Abiri et al. (2014, 2015); Shafiullah et al. (2022) do not necessarily ensure optimal results for reconstruction and fault localization accuracy, which are the criteria we use in our work. To address this issue, we propose optimizing the PMU placement set $\mathbf{K}$ using one of the proposed loss functions ℓ. We propose a version of a regression technique called the

Algorithm 3. Optimal PMU placement strategy

```
Input: Training data $\mathbf{X}^{\mathrm{Tr}}$, validation data $\mathbf{X}^{\mathrm{Val}}$, graph structure $\mathbf{G}$, length of each simulation $2m$;
    Parameters: sample size each iteration $S$, maximum iteration $I_{\max}$, maximum epoch $E_{\max}$,
    desired $\mathbf{K}$ size $K$.
Output: Optimal set $\mathbf{K}_{opp}$ of size $K$
1: Set $\mathbf{K}_{opp} = \{1, \dots, 122\}$ and $\mathbf{U}_{opp} = \{\}$
2: while $|\mathbf{K}_{opp}| > K$ do
3:     for $b$ in $\mathbf{K}_{opp}$ do
4:         Run Algorithm 2 with $\mathbf{U} := \{\mathbf{U}_{opp}, b\}$ and $\mathbf{K} := \mathbf{K}_{opp} \setminus \{b\}$
5:         Record validation error $Err_b$
6:     end for
7:     Set $\hat{b} = \arg\min_{b \in \mathbf{K}_{opp}} Err_b$
8:     Set $\mathbf{K}_{opp} := \mathbf{K}_{opp} \setminus \{\hat{b}\}$ and $\mathbf{U}_{opp} := \{\mathbf{U}_{opp}, \hat{b}\}$
9: end while
```

Backward Stepwise procedure, specified in Algorithm 3. Starting with $\mathbf{K}$ equal to the set of all buses ($K = 122$, the algorithm progressively removes buses from the set $\mathbf{K}_{opp}$ of optional buses and adds them to the set $\mathbf{U}_{opp}$ of used buses until $|\mathbf{K}_{opp}| = K$. The decision of which bus to move is based on the loss function ℓ. We consider moving each possible bus from the set $\mathbf{K}_{opp}$ and evaluate the cost of the decision using ℓ. For simplifications, we employed this algorithm with a simpler model (RGNN) using the MSE loss function and without features transformation (Algorithm 2). As Algorithm 3 requires running Algorithm 2 many times, we reduced computational burden by lowering settings (reducing dimension for graph convolution, using fewer iterations, and fewer epochs). We provide a discussion in Section 5.2.

4.5 | Fault localization scheme

The overall structure of the proposed fault localization method under partial observability involves two modules: 1) reconstructing the modulus of voltages at unobserved buses, and 2) using existing methods, which require full grid observability, to predict the buses (i_0, j_0) connected by the faulted line l_0. Module 1) has already been described. In this subsection, we focus on Module 2), assuming possession of the full data set $\mathbf{X}$. In practice, this requires replacing $\mathbf{X}$ by $\widehat{\mathbf{X}}$.

In principle, any fault localization method can be used, but we use a slightly revised method of Kokoszka et al. (2023) because we have code for it that works in the minniWECC setting, and we have been unable to gain access to code for other methods. We assume that the time of the fault, t_f, is estimated using one of the available methods, for example, Shafiullah and Abido (2018); Li and Deka (2021); Kokoszka et al. (2023); Rimkus et al. (2023). In the following, t_f is treated as known.

We have $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_{122}$, where $\mathbf{x}_k = x_k(0), x_k(\delta), \dots, x_k(2m \cdot \delta)$ is a trajectory of the modulus of voltage at bus k, and δ is the time separation of the consecutive data points ($\delta = \frac{1}{120}$ in this paper). We denote by S_0 and S_1 the time windows (in seconds) used to extract information from trajectories around fault time t_f. For each bus k, we set

$$\bar{x}_k(t, S_0) = \frac{1}{S_0/\delta} \sum_{l=1}^{S_0/\delta} x_k(t - l\delta), \tag{3}$$

$$m_k(t, S_1) = \frac{1}{1 + S_1/\delta} \sum_{l=0}^{S_1/\delta} x_k(t + l\delta). \tag{4}$$

We obtain the prediction of the near-end bus by setting:

$$\hat{\imath}_0 = \underset{k \in \{1,..,122\}}{\arg\max} |m_k(t_f, S_1) - \overline{x}_k(t_f, S_0)|. \tag{5}$$

Once we have predicted the near-end bus $\hat{\imath}_0$ of the faulted line l_0, we predict the far-end bus by employing the properties of fault clearance. Notice that the prediction $\hat{\jmath}_0$ must be one of the buses to which $\hat{\imath}_0$ is connected by a direct line, so there are only a few (or no) choices. For each bus k, we set

$$D_i^*(t, t_f, S_0) = m_k(t, 0) - \overline{x}_k(t_f, S_0), \quad t > t_f. \tag{6}$$

The time of the recovery, t_R, is defined as the smallest $t > t_f$ such that

$$\left| \frac{D_{\hat{\imath}_0}^*(t, t_f, S_0) - D_{\hat{\imath}_0}^*(t - \delta, t_f, S_0)}{D_{\hat{\imath}_0}^*(t - \delta, t_f, S_0)} \right| > \tau_1. \tag{7}$$

The parameter τ_1 was determined Kokoszka et al. (2023) who also explained that its choice is not critical, as long as it remains in a reasonable range. We obtain the prediction of the remote end bus by setting

$$\hat{\jmath}_0 = \underset{k}{\arg\max}(|D_k^*(t, t_f, S_0)| - |D_k^*(t - \delta, t_f, S_0)|), \tag{8}$$

where the maximum is taken over busses k that connect directly to $\hat{\imath}_0$.

The predicted faulted line $\hat{l}_0$ is the line between buses $\hat{\imath}_0$ and $\hat{\jmath}_0$. Under the assumption of PMUs at every possible bus, this methodology produces results with around 4% failure rate. To report fault localization results under partial observability, we also use the failure rate—the percentage of simulations where the faulted line/near-end bus was predicted incorrectly.

4.6 | Benchmark method

To assess the results of the proposed deep learning methodology, we use a relatively simple and intuitive benchmark method (BM). To the best of our knowledge, there is no other work that focuses on fault localization in a regional power grid without using labeled data, so we cannot use an existing benchmark. Recall that **U** is the set of buses where trajectories need to be predicted and **K** is the set of the available buses. The BM is implemented for each trajectory $\mathbf{y}_k$ with $k \in \mathbf{U}$. It takes the trajectories of directly connected buses and averages them to estimate the missing trajectory. If some of the buses connecting directly to bus $k \in \mathbf{U}$ are themselves in **U**, their reconstructed trajectories are used, which are obtained through an iterative process. The iterations continue until each bus in **U** has all neighbors with available trajectories The process can be formalized by setting

$$\hat{\mathbf{y}}_k = \frac{1}{|\mathbf{N}_k|} \sum_{j \in \mathbf{N}_k} \mathbf{y}_j^*,$$

where $\mathbf{N}_k$ is set of buses connecting directly to bus k. Some of the $\mathbf{y}_j^*$ can be real trajectories, if $j \in \mathbf{K}$, some can be reconstructed trajectories, if $j \in \mathbf{U}$.

5 | APPLICATION TO THE WESTERN INTERCONNECTION

In this section, we apply the methods of Section 4 to the data described in Section 2. We implement them with 10,000 training simulations, 2500 validation simulations, and 9068 testing simulations. As in all deep learning approaches, the performance of the methods is evaluated on the testing simulations that were not used during the training and validation process. The selection of the simulations for the three groups was random. The models were implemented using the Torch framework Collobert et al. (2011). We utilized the Adam optimization algorithm Kingma and Ba (2014), which includes the learning rate parameter determining the size of the step for updating the GNN parameters. We conducted experiments with various learning rates (ranging from 0.001 to 0.00005) to update the weight matrix represented as $\boldsymbol{\Theta}$ for different-sized models with various hyperparameters M, z, and $\mathcal{K}$ described in Section 4.2, cf. (1) and (2). We implemented our methods on the CUDA platform with an NVIDIA GeForce GTX 1080 Ti with 11 GB GDDR5X.

We first implemented the optimal PMU placement algorithm, Algorithm 3, using a small RGNN ($M = 3$, $z = 150$, $\mathcal{K} = 2$), and obtained a sequence of the $\mathbf{U}$ sets (the buses without data) used to present the results. Starting with $U = 12$ buses in $\mathbf{U}$, we obtained the sets $\mathbf{U}$ up to $U = 77$. Recall that the total number of buses is 122. By construction, a set $\mathbf{U}$ with a smaller cardinality is always a subset of a set $\mathbf{U}$ with a larger cardinality. Algorithm 3 is a useful tool in its own right. It might allow grid operators to determine where monitoring devices should be placed if only a limited number of them can be monitored. This is illustrated in Figure 3.

Second, we implemented a grid search to find optimal hyperparameters. Choosing the appropriate hyperparameters for neural networks is crucial, as poorly chosen ones can result in suboptimal performance such as slow convergence, overfitting, or underfitting Bengio et al. (2013). There are several hyperparameters that can affect the results, including batch size, learning rate, network depth, the size of inner layers, diffusion localization level, and number of epochs. In our grid search, we focused on network depth M, the size of inner layers z, and diffusion localization level $\mathcal{K}$. Due to the resources required to train each model, it is not feasible to conduct such a search for every $\mathbf{U}$, so we only performed the grid search for RGNN and SIGNN with $U = 17$. The grid search was implemented solely using validation results, leaving the test data untouched to avoid data leakage. We recommend a similar search to determine the settings that work best for any specific application. This issue is elaborated on in Section 5.2.

We evaluated the accuracy of trajectory reconstruction using the loss functions and localization failure rate explained in Section 4.3. We compared these results to the BM of Section 4.6. Our results are reported for two testing data splits:

(1) all testing simulations;
(2) testing simulations where the near-end bus of a faulted line is in the set $\mathbf{U}$.

Finding the bus closest to the fault, the near-end bus, is the most important aspect of the localization algorithm.

Finally, to better understand the mechanics of the models, we used two sets of data: the unmodified data and the rescaled data. The rescaled data was generated by dividing each bus's

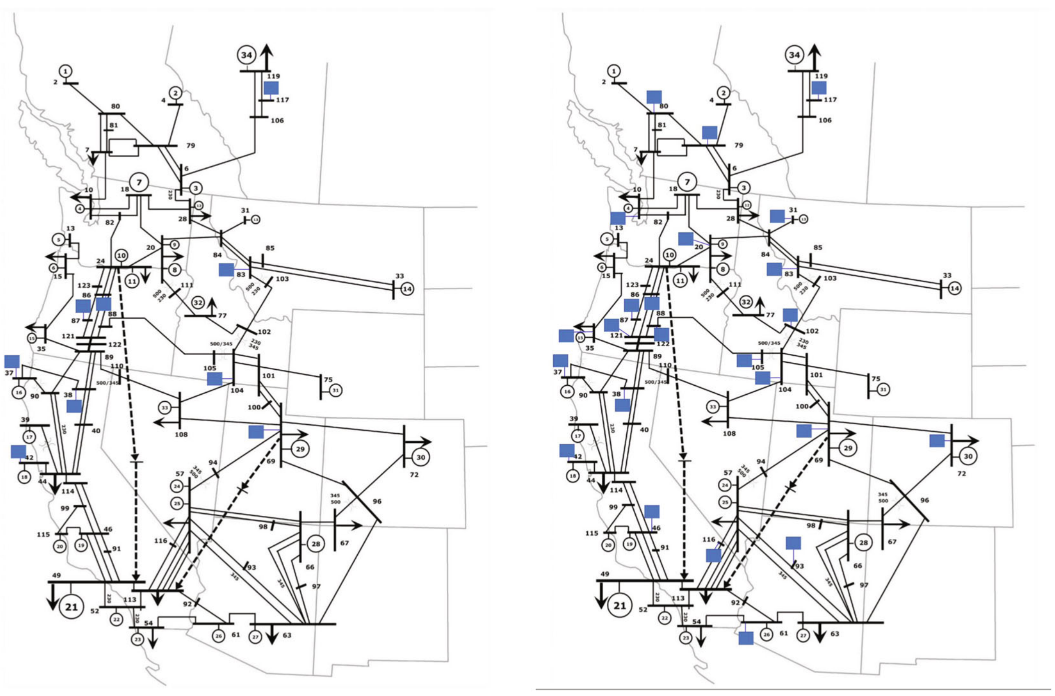


FIGURE 3 The squares indicate recommended locations, according to Algorithm 3, of monitoring devices. Only buses connecting high-voltage lines are shown. Left: results for $K = 27$, Right: results for $K = 57$.

trajectory by its average value prior to the fault, which was determined through the system parameters given in Trudnowski et al. (2013). Such a normalization is feasible in a real system because the average voltages of the lines under normal operations are either known or can be easily measured. The assessment of the models on both the unmodified data and the rescaled data facilitates the identification of their strengths and weaknesses and reveals important distinctions between them. Using the unmodified data, the BM and the SIGNN give results influenced by the differences between the trajectories at different buses before the fault happens, the differences in average voltages of the lines. As the focus of the paper is the reconstruction of the trajectories during a fault, using rescaled data that eliminate this bias is justified. In the following, we use the abbreviations BM or SIGNN to refer to these models trained on the unmodified data, by BM-R and SIGNN-R to these models trained on the rescaled data. More precisely, the application of the BM-R and the SIGNN-R involves the following steps: 1. use rescaled data to train the SIGNN (BM does not need training); 2. use rescaled data to reconstruct missing trajectories; 3. rescale reconstructed trajectories back to the unmodified data space; 4. calculate the MSE and fault localization results as for other methods.

5.1 | The results

During a grid search for optimal hyperparameters, we observed that for RGNN we do not need to use large z, L, and $\mathcal{K}$ values. This suggests that the learning is more local and the trajectories' reconstruction is performed mostly using trajectories that are close in the grid. The results below are presented using the model with $z = 200$, $L = 4$, and $\mathcal{K} = 2$ for $U \leq 42$. For $U > 42$, we used $L = 6$.

Figures 4 and 5 demonstrate that the RGNN exhibits strong performance in terms of trajectory reconstruction and fault localization, generally by a large margin. For a fixed U, the RGNN takes about 30 min, while the BM methods about one minute (on Intel Core i7-6850K @ 3.6 GHz with 32 GB of RAM with NVIDIA GTX 1080Ti). We also examined analogs of Figure 4 using the RMSE, the MAE, and the MAPE. The shapes of the three curves and the gaps between them change, but the curve corresponding to the RGNN is always at the bottom (the best). With only 36% of the power grid observed ($U = 77$), the RGNN correctly detects the faulted line 3 times out of 4. Looking at the bottom two panels of Figure 5, one can see that the RGNN localizes a fault with 60% success rate, even if the fault occurs at a bus not measured by a PMU. That suggests that it can successfully infer where the fault is and reconstruct the trajectories with high success. In Figure 6, we illustrate the accuracy of the fit. We present results for $U = 17$ with a testing simulation in which the line between buses 30 and 31 is faulted, with bus 30 being the near-end bus of the faulted line. Bus 30 is included in $\mathbf{U}$ with $U = 17$. The relative performance of the RGNN remains better than that of BM and BM-R if different loss functions are used, and there is not much change in the localization success rates.

We now turn to the second scenario described in Section 3 (the SIGNN), in which the training process cannot use trajectories in $\mathbf{U}$, and thus needs to learn only from the trajectories in $\mathbf{K}$, with the knowledge of all edges $\mathbf{E}$. The RGNN can use the trajectories in $\mathbf{U}$ during training. The SIGNN must thus solve a much more challenging task. This corresponds to a situation where no PMUs have ever been placed in $\mathbf{U}$. The RGNN is trained on the actual data in $\mathbf{U}$ that may become unavailable due to, say, a natural disaster or a cyberattack. In broadest terms, the default version of SIGNN failed and did not outperform the benchmark models. The SIGNN-R performed slightly better than BM and BM-R and noticeably better than both of them in simulation of the near-end bus in $\mathbf{U}$.

Examination of Figures 1 and 2 suggests that using the Haar (rectangular) wavelet transform might lead to better reconstruction results, as this transform is effective in decomposing signals

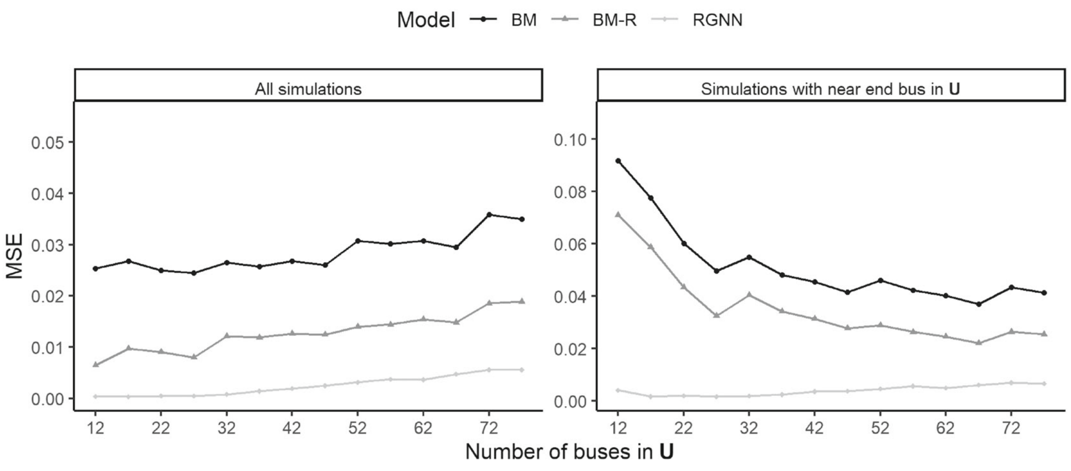


FIGURE 4 Comparison of reconstruction results in terms of the MSE for the RGNN and the benchmark models BM and BM-R, with varying sizes of $\mathbf{U}$. The expected trend of decreasing MSE as U increases is observed in the second panel due to more trajectories being included in $\mathbf{U}$.

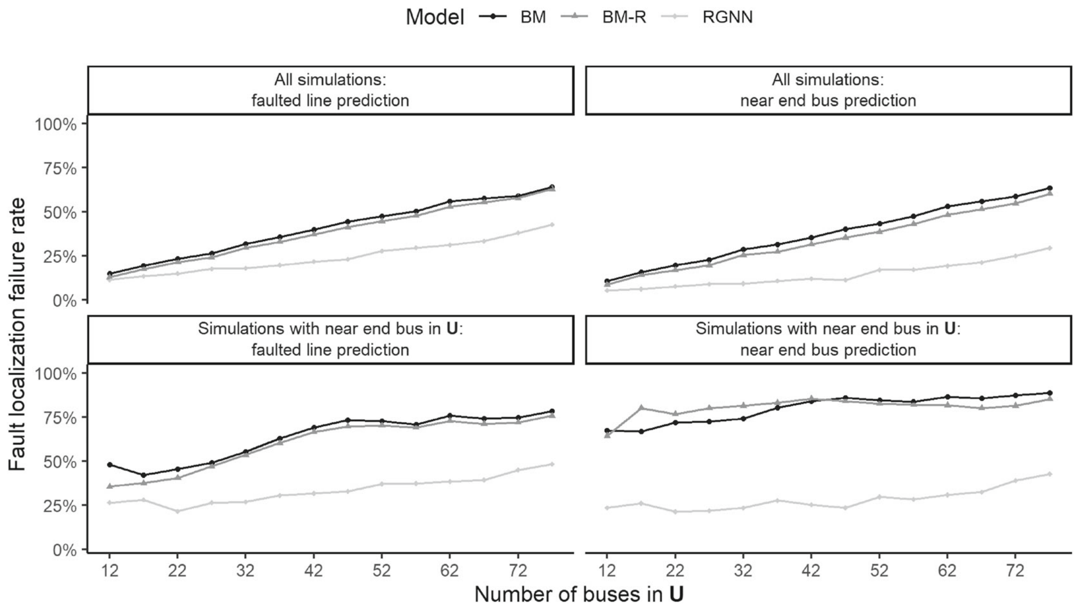


FIGURE 5 Comparison of fault localization results in terms of failure rate for RGNN and the benchmark model (BM and BM-R), with varying sizes of **U**.

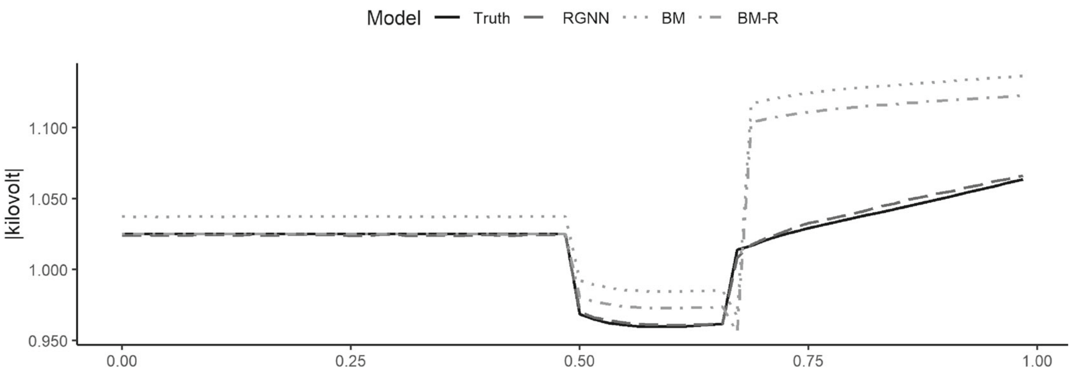


FIGURE 6 Trajectory reconstruction at bus 30 in **U** for $U = 17$. Bus 30 is the near-end bus of the fault. The RGNN reconstruction visually almost coincides with the true.

with "rectangular blocks" into sparse representations. Interested readers are referred to Percival and Walden (2000). We conducted experiments involving the application of the Haar wavelet transform. The inputs and outputs of the GNNs are represented as the wavelet coefficients. From the outputs, the trajectories in time domain can be reconstructed exactly and diagnostics can be applied to the reconstructed trajectories. Although the final results were comparable to those for the raw trajectories, we observed that the application of the Haar wavelet transform facilitated convergence to the solution for RGNN and SIGNN-R, necessitating fewer iterations for model training. In many cases, the time needed to properly train these models decreases by half when Haar wavelets were used.

5.2 | Additional information on network architecture, hyperparameters and training

We conducted a grid search for SIGNN, SIGNN-R, and RGNN with $U = 17$ and $U = 52$ using the validation error. We observed that the parameter $\mathcal{K}$ had minimal impact on the results, and we found that setting $\mathcal{K} = 2$ produced reasonable results. We reported all results with $\mathcal{K} = 2$, except for models trained with $L = 8$. In these cases, due to GPU limitations, we used $\mathcal{K} = 1$ instead. Our findings also showed that there was no significant improvement in increasing z beyond 200. Therefore, we reported all results using $z = 200$. Additionally, we observed that the parameter L (the depth of the network) had an impact on the results, with larger L working better for more complex problems that the network needed to solve. For smaller values of U (e.g., $U = 17$), the SIGNN-R performed best with $L = 6$, while the RGNN with $L = 4$. However, for higher values of U, the SIGNN-R performed the best with $L = 8$, while the RGNN with $L = 6$. This difference can be attributed to the amount of time required for the network to distill the masked trajectories in H_0. In broad terms, the depth L of the network may impact the quality of the reconstruction, whereas increasing its width beyond $z = 200$ has little practical impact. We also experimented with batch sizes and numbers of epochs and iterations, which have a qualitatively different status because these adjustments do not change the network structure. We settled on the batch size of $S = 64$, $E_{\max} = 200$ epochs, and $I_{\max} = 10$ iterations because increasing these values did not bring better reconstruction results and resulted in longer run times.

6 | SUMMARY AND MAIN CONCLUSIONS

We proposed GNNs that can be effectively applied to reconstruct PMU trajectories with the aim of power grid fault localization. Compared with previous methods employed for fault localization, our learning networks do not require training with specified fault locations, *the methods are not given the exact fault locations to learn to find them in the future*. We studied two scenarios for data availability and proposed the RGNN and SIGNN networks (with various variants) that are applicable to them. The RGNN is trained using data in **U** (the unobserved buses). The SIGNN is trained without any access to data in **U**. We emphasize that in none of these scenarios information about fault location is used, so none of the classification-based methods discussed in the Introduction can be used. In particular, our approaches cannot be compared with any of them within our data framework. For this reason, we introduced a BM, which by itself is a potential tool for trajectory reconstruction and fault localization.

The conclusions and contributions of our research can be summarized as follows.

1. The RGNN performs very well and could form a basis for an industrial implementation.
2. The SIGNN and its variants perform broadly on par with the benchmark method. Since the benchmark method is simpler, we cannot recommend any variants of the SIGNN at this point.
3. The benchmark method performs reasonably well in the second scenario where a large part of the grid has never been observed. We are aware of no other published methods that are applicable in such a scenario.
4. We propose an algorithm that identifies optimal buses for placing a limited number of PMUs for the purpose of fault location identification. This algorithm could serve as a starting point for an industrial implementation.

5. Our method is based on reconstructing data at buses where PMUs are not available or do not stream data for some reason. Such reconstruction algorithms, potentially with some modifications, can be useful for other purposes, for example, mitigation of the effects of cyberattacks.
6. We test the performance of the proposed methods on a realistic model of a regional transmission grid that covers about one third of the United States. This is a much larger grid than the testbeds used in previous research.

FUNDING INFORMATION

This research was partially supported by NSF grants DMS–1923142 and DMS–2123761.

CONFLICT OF INTEREST STATEMENT

The authors report there are no competing interests to declare.

ORCID

Piotr Kokoszka https://orcid.org/0000-0001-9979-6536

REFERENCES

Abiri, E., Rashidi, F., & Niknam, T. (2015). An optimal PMU placement method for power system observability under various contingencies. *International Transactions on Electrical Energy Systems*, *25*(4), 589–606.

Abiri, E., Rashidi, F., Niknam, T., & Salehi, M. (2014). Optimal PMU placement method for complete topological observability of power system under various contingencies. *International Journal of Electrical Power & Energy Systems*, *61*, 585–593.

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Harutyunyan, H., Alipourfard, N., Lerman, K., Ver Steeg, G., & Galstyan, A. (2019). *Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing*. In *The thirty-sixth international conference on machine learning (ICML'19)*. ACM.

Afonin, A., & Chertkov, M. (2021). Which neural network to choose for post-fault localization, dynamic state estimation, and optimal measurement placement in power systems? *Frontiers in Big Data*, *4*, 692493.

Aminifar, F., Khodaei, A., Fotuhi-Firuzabad, M., & Shahidehpour, M. (2010). Contingency-constrained PMU placement in power networks. *IEEE Transactions on Power Systems*, *25*(1), 516–523.

Appleby, G., Liu, L., & Liu, L. (2020). Kriging convolutional networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*, 3187–3194.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Byrne, R. H., Concepcion, R. J., Neely, J., Wilches-Bernal, F., Elliott, R. T., Lavrova, O., & Quiroz, J. E. (2016). *Small signal stability of the western north american power grid with high penetrations of renewable generation*. In *Proc. of 2016 IEEE 43rd photovoltaic specialists conference (PVSC)* (pp. 1784–1789). IEEE.

Chen, Y., Coskunuzer, B., & Gel, Y. (2021). *Topological relational learning on graphs*. In *Advances in neural information processing systems 34 (NeurIPS 2021)*. NeuroIPS Foundation.

Cheung, K. W., Chow, J., & Rogers, G. (2009). *Power system toolbox* (Vol. *v 3.0*. Technical Report.). Rensselaer Polytechnic Institute and Cherry Tree Scientific Software.

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). *Torch7: A matlab-like environment for machine learning*. In *BigLearn* NIPS Workshop. NeuroIPS Foundation.

Cuturi, M., & Blondel, M. (2017). *Soft-dtw: A differentiable loss function for time-series*. In *Proceedings of the 34 th international conference on machine learning* (Vol. *70*, pp. 894–903). ACM.

Devi, M. M., Geethanjali, M., & Devi, A. R. (2018). Fault localization for transmission lines with optimal phasor measurement units. *Computers and Electrical Engineering*, *70*, 163–178.

Enshaee, A., Hooshmand, R., & Fesharaki, F. (2012). A new method for optimal placement of phasor measurement units to maintain full network observability under various contingencies. *Electric Power Systems Research*, *89*, 1–10.

Foggo, B., & Yu, N. (2022). Online PMU missing value replacement via event-participation decomposition. *IEEE Transactions on Power Systems*, *37*(1), 488–496.

Follum, J., Pierre, J. W., & Martin, R. (2017). Simultaneous estimation of electromechanical modes and forced oscillations. *IEEE Transactions on Power Systems*, *32*(5), 3958–3967.

Furse, M., Kafal, M., Razzaghi, R., & Shin, Y. (2021). Fault diagnosis for electrical systems and power networks: A review. *IEEE Sensors Journal*, *21*(2), 888–906.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org

Guen, V., & Thome, N. (2022). *Deep time series forecasting with shape and temporal criteria*. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (vol. *45*, pp. 342–355). IEEE.

Hamilton, W. (2020). Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *14*(3), 1–159.

Hamilton, W., Ying, R., & Leskovec, J. (2017). *Inductive representation learning on large graphs*. In *Advances in neural information processing systems* (Vol. *30*, pp. 1025–1035). Curran Associates.

Han, J., Miao, S., Li, Y., Yang, W., & Yin, H. (2022). Fault diagnosis of power systems using visualized similarity images and improved convolution neural networks. *IEEE Systems Journal*, *16*(1), 185–196.

Harrou, F., Sun, Y., Hering, A., Madakyaru, M., & Dairi, A. (2020). *Statistical process monitoring using advanced data-driven and deep learning approaches*. Elsevier.

Jakir, M., & Rahnamay-Naeini, M. (2021). *State estimation in smart grids using temporal graph convolution networks*. In *2021 north American power symposium (NAPS)* (pp. 1–5). IEEE.

Kingma, D., & Ba, J. (2014). *Adam: A method for stochastic optimization*. In *International conference on learning representations* (p. 12). ICLR.

Kipf, T., & Welling, M. (2017). *Semi-supervised classification with graph convolutional networks*. In *International conference on learning representations*. ICLR.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). *Self-normalizing neural networks*. In *Advances in neural information processing systems* (Vol. *30*, pp. 971–980). Curran Associates, Inc.

Kokoszka, P., Rimkus, M., Hosur, S., Duan, D., & Wang, H. (2023). Detection and localization of faults in a regional power grid. *Austrian Journal of Statistics*, *52*, 143–162.

Li, W., & Deka, D. (2021). *Physics-informed learning for high impedance faults detection*. In *2021 IEEE Madrid PowerTech* (pp. 1–6). IEEE.

Li, W., Deka, D., Chertkov, M., & Wang, M. (2019). Real-time faulted line localization and PMU placement in power systems through convolutional neural networks. *IEEE Transactions on Power Systems*, *34*(6), 4640–4651.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). *Diffusion graph convolutional recurrent neural network: Data-driven traffic forecasting*. In *International conference on learning representations*. ICLR.

Liao, W., Bak-Jensen, B., Pillai, J., Wang, Y., & Wang, Y. (2022). A review of graph neural networks and their applications in power systems. *Journal of Modern Power Systems and Clean Energy*, *10*(2), 345–360.

Park, S., Gama, F., Lavaei, J., & Sojoudi, S. (2023). *Distributed power system state estimation using graph convolutional neural networks*. In *Proceedings of the Hawaii International Conference on System Sciences*. IEEE.

Percival, D. B., & Walden, A. T. (2000). *Wavelet methods for time series analysis*. Cambridge University Press.

Rimkus, M., Kokoszka, P., Prabakar, K., & Wang, H. (2023). Real-time power grid fault detection. *Communications in Statistics: Case Studies, Data Analysis and Applications* Forthcoming. Taylor & Francis.

Ringsquandl, M., Sellami, H., Hildebrandt, M., Beyer, D., Henselmeyer, S., Weber, S., & Joblin, M. (2021). *Power to the relational inductive bias: Graph neural networks in electrical power grids*. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*. ACM.

Salehi-Dobakhshari, A., & Ranjbar, A. (2014). Application of synchronised phasor measurement to wide-area fault diagnoses and location. *Generation, Transmission and Distribution, IET*, *8*, 716–729.

Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., & Monfardini, G. (2009). Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, *20*(1), 81–102.

Shafiullah, M., & Abido, M. (2018). S-transform based FFNN approach for distribution grids fault detection and classification. *IEEE Access*, *6*, 8080–8088.

Shafiullah, M., Abido, M., & Al-Mohammed, A. (2022). *Power System Fault Diagnosis*. Elsevier Inc.

Trudnowski, D., Kosterev, D., & Undrill, J. (2013). *PDCI damping control analysis for the western north American power system*. In *Proc. of 2013 IEEE power & energy society general meeting* (pp. 1–5). IEEE.

Wu, Y., Zhuang, D., Labbe, A., & Sun, L. (2020). *Inductive graph neural networks for spatiotemporal kriging*. In *AAAI conference on artificial intelligence*. AAAI.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 4–24.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). *How powerful are graph neural networks?* In *International conference on learning representations*. ICLR.

Xu, Z., Xue, Y., & Wong, K. (2014). *Recent advancements on smart grids in China*. In *Electric Power Components and Systems* (vol. *42*, pp. 251–261). Taylor & Francis.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., & Liu, T.-Y. (2021). Do transformers really perform bad for graph representation? *Advances in Neural Information Processing Systems*, *34*, 28877–28888.

You, Y., Chen, T., Wang, Z., & Shen, Y. (2020). *When does self-supervision help graph convolutional networks?* In *ICML'20: Proceedings of the 37th international conference on machine learning* (pp. 10871–10880). ICML.

Yu, B., Yin, H., & Zhu, Z. (2018). *Spatio-temporal graph convolutional neural*. In *Proceedings of the 27th international joint conference on artificial intelligence, IJCAI'18* (pp. 3634–3640). AAAI Press.

Zainab, A., Refaat, S., Syed, D., Ghrayeb, A., & Abu-Rub, H. (2019). *Faulted line identification and localization in power system using machine learning techniques*. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 2975–2981). IEEE.

Zhang, A., Lipton, Z., Li, M., & Smola, A. (2023). *Dive into deep learning*. Cambridge University Press.

Zhang, P., Qian, K., Zhou, C., Stewart, B., & Hepburn, D. (2012). A methodology for optimization of power systems demand due to electric vehicle charging load. *IEEE Transactions on Power Systems*, *27*, 1628–1636.

Zhao, M., & Barati, M. (2021). A real-time fault localization in power distribution grid for wildfire detection through deep convolutional neural networks. *IEEE Transactions on Industry Applications*, *57*(4), 4316–4326.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81.

Zhu, L., & Lin, J. (2021). Learning spatiotemporal correlations for missing noisy PMU data correction in smart grid. *IEEE Internet of Things Journal*, *8*(9), 7589–7599.

How to cite this article: Rimkus, M., Kokoszka, P., Duan, D., Wang, X., & Wang, H. (2025). Graph neural networks for the localization of faults in a partially observed regional transmission system. *Scandinavian Journal of Statistics*, *52*(2), 572–594. https://doi.org/10.1111/sjos.12763