A Heterogeneous Graph-Based Multi-Task Learning for Fault Event Diagnosis in Smart Grid

Dibaloke Chanda, *Student Member, IEEE*, and Nasim Yahya Soltani, *Member, IEEE*

***Abstract*—Precise and timely fault diagnosis is a prerequisite for a distribution system to ensure minimum downtime and maintain reliable operation. This necessitates access to a comprehensive procedure that can provide the grid operators with insightful information in the case of a fault event. In this paper, we propose a heterogeneous multi-task learning graph neural network (MTL-GNN) capable of detecting, locating and classifying faults in addition to providing an estimate of the fault resistance and current. Using a graph neural network (GNN) allows for learning the topological representation of the distribution system as well as feature learning through a message-passing scheme. We investigate the robustness of our proposed model using the IEEE-123 test feeder system. This work also proposes a novel GNN-based explainability method to identify key nodes in the distribution system which then facilitates informed sparse measurements. Numerical tests validate the performance of the model across all tasks.**

***Index Terms*—Distribution system, explainability, fault event diagnosis, heterogeneous multi-task learning, smart grid.**

I. Introduction

FAULT diagnosis is a crucial task for the operation and maintenance of power systems, particularly in distribution networks due to the nature of complex interconnectivity and scale of the network. Failure to take proper action during a fault event can result in a cascading outage of the distribution system [1], [2]. For uninterrupted operation in the case of a fault occurrence grid operators need to identify the precise location of the fault in addition to the type of the fault. Furthermore, knowing the fault resistance and fault current before taking action for fault isolation and fault clearance guarantees the implementation of appropriate safety measures. This additional information allows grid operators to make more informed decisions and also to plan for necessary repairs or equipment replacements. Not only that, but it also provides insight into post-fault analysis to identify if all the protection systems performed as intended. Due to the advancement in deep learning in recent years, most fault diagnosis systems are utilizing a data-driven approach. However, there is a lack of unified methods that take into account the challenges associated with real-world deployment and many research provides analysis based on theoretical assumptions only.

In this work, we propose a unified heterogeneous MTL-GNN architecture that is capable of performing fault detection, fault localization, fault type classification, fault resistance estimation and fault current estimation. All the tasks are performed in a simultaneous manner as opposed to a sequential manner which ensures the decoupling of tasks.

We call it a heterogeneous MTL in contrast to a homogeneous MTL due to the fact that the proposed model performs both classification and regression tasks. We take into account all 5 types of short circuit faults that can occur in a distribution system [3]. This includes asymmetrical faults consisting of line-to-ground faults (LG), line-to-line faults (LL), line-to-line-to-ground faults (LLG) and symmetrical faults consisting of line-to-line-to-line-to-ground faults (LLLG), line-to-line-to-line faults (LLL). To address the challenges associated with real-world deployment, our analysis takes into account measurement error, variable fault resistance, small dataset, topology changes and sparse measurements. To make our contribution clear in the following section, the related literature in this domain and the drawbacks and scope for development have been reviewed.

II. Literature Review

The literature on fault event diagnosis is very extensive [4], [5], [6]. This includes different kinds of faults such as overvoltage, insulator, voltage sag, arc, and short-circuit faults.

They can be broadly structured into two categories. One category are data-driven approaches which utilize deep learning models [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] and another category includes traditional methods [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37] using statistical measures, signal processing and the physics of fault events. The latter category can be further broken down into separate categories like traveling wave-based [22], [23], [24], [25], impedance-based [26], [27], [28], [29], morphology-based [30], [31], [32], [33], [34], and voltage sag [35], [36], [37] methods.

As regards the data-driven approaches, there is a multitude of methods and architectures but we broadly divide them into two categories which are GNN-based approaches [15], [16], [17], [18], [19], [20], [21], multi-layer perceptron (MLP) and convolutional neural network (CNN) based approaches [7], [8], [9], [10], [11], [12], [13], [14].

Received 23 October 2023; revised 7 March 2024 and 12 June 2024; accepted 17 August 2024. Date of publication 21 August 2024; date of current version 21 February 2025. An earlier version of this work was presented at the Intl. Workshop on Machine Learning for Signal Processing, Rome, Italy, September 2023. Paper no. TPWRS-01670-2023. *(Corresponding author: Nasim Yahya Soltani.)*

The authors are with the Department of Computer Science, Marquette University, Milwaukee, WI 53233 USA (e-mail: dibaloke.chanda@marquette.edu; nasim.yahyasoltani@marquette.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPWRS.2024.3447533.

Digital Object Identifier 10.1109/TPWRS.2024.3447533

A. Traditional Methods

The traveling wave-based methods [22], [23], [24], [25] analyze the characteristics of traveling waves generated when a fault event occurs. During a fault event, the sudden change in voltage and current around the fault location initiates a transient disturbance that propagates along the distribution line. This transient wave and its reflected counterpart are picked up by fault recorders installed at a substation or two substations depending on whether it is a single-end method or a double-ended method. Based on the time of arrival of the transient wave and the reflected wave it is possible to deduce the location of the fault.

In impedance-based methods [26], [27], [28], [29], current and voltage signals are measured along different places on the distribution line. In the case of a fault event, these measured signals are used to isolate the fundamental frequency to estimate the apparent impedance. This apparent impedance is then used to locate the fault event.

Morphology-based methods [30], [31], [32], [33], [34] make use of mathematical morphological operations like dilation, erosion, closing and opening on the waveform generated during a fault event to extract features that correspond to a fault event. After feature extraction is complete, the extracted features are passed to different classifier algorithms like decision trees [31], recursive least square stage [32], and random forest [34].

Voltage sag methods [35], [36], [37] use the characteristic of the reduction of voltage magnitude in the case of a fault event to isolate the fault's exact location. When a fault event occurs there is a sudden dip in the voltage magnitude. This sudden dip occurs only at the location of the fault event which can isolated based on the characteristics of the voltage sag.

B. Deep Learning Based Methods

1) MLP and CNN Based Methods: These methods use historical or software-simulated data relating to fault events in distribution systems and use them to train MLP and CNN architectures to do prediction tasks like fault detection, fault localization and fault classification.

The work in [7] is one of the early papers that use a MLP. The input to their proposed model is the current measures of distributed generation units (DGs) and substation and they perform fault localization as output. Another similar work that uses MLP is [8] where the authors use the IEEE-13 bus system to perform both fault classification and localization. In [12] the authors use MLP but with an additional fuzzy layer and their analysis is on the IEEE-37 bus system.

In [9] the CNN architecture is adopted for fault localization. First, the authors use a continuous wavelet transform (CWT) algorithm to convert current phasors to images which are fed to a CNN model to localize the fault. The work in [10] also uses a CNN to localize faults but considers partial observability of the grid on IEEE-39 and IEEE-68 bus systems. In [11] the authors take a slightly different approach by using 1-D convolutions with double-stage architecture. The first stage extracts the features and the second performs fault identification. Similar to [9], the work in [13] also uses CWT to convert time-domain current signals to image domain and use the transformed data for fault classification and localization on the IEEE-34 bus system. A different approach by using a capsule-based CNN is proposed in [14] to do fault detection, localization and classification.

2) GNN Based Methods: The first prominent work to use GNN for fault localization is in [15]. In this work, IEEE-123 and IEEE-37 are used as the test feeder system and the authors consider a range of factors like metering error, changes in topology, etc. for their analysis. The architecture employed is CayleyNets [38] which is a graph convolutional neural network (GCN) based on spectral theory. The feature vector considered as the input to their model consists of both voltage and current phasors measured from the buses in the distribution system.

The subsequent notable work is [16] which utilizes not only node features but also link features that include branch impedance, admittance and regulation parameters of the distribution lines. The authors validate their approach on a self-designed 6.6 KV system with 12 buses and 8 loads.

Two other related research works are in [17] and [18]. For the first one, the authors consider a gated GNN [39] architecture. However, they only consider single-line-to-ground fault as opposed to [18] which considers all three types of asymmetrical short circuit fault. Not only that, but the authors also consider limited observability and limited labels in their implementation. Their proposed method has a two-stage architecture with only voltage phasors as input.

In [19] there is a more recent work that uses a different variant of GNN, a graph attention neural network (GAT) [40] to do both fault localization and classification in IEEE-37 feeder system. In their analysis, the authors consider a constant fault resistance and the fault localization prediction is dependent on the fault classification task.

These above-mentioned works consider instantaneous current and/or voltage phasors meaning that they require measurement at the fault time, no pre-fault or post-fault measurement is required. But the method proposed in [20] requires a fault waveform sampled at 1 KHz. Similar to [19] their analysis considers constant fault resistance. One important contribution of this research work is that they use MTL to do both fault classification and fault localization at once in contrast to [19].

Another similar recent work employs spatial-temporal recurrent GNN to do three tasks simultaneously which are fault detection, classification and localization [21]. The authors report numerical results tested on a microgrid and IEEE-123 bus system. Similar to the previous approach their analysis considers constant resistance and due to the temporal nature of their proposed method, it requires high time resolution of fault waveshape.

The summary of major technical differences between our proposed model with the existing GNN-based models is outlined in Table I. We make the argument that the models that are only trained for fault localization and/or classification [15], [16], [17], [18], [19], [20] will require a separate method to first distinguish between fault event and other events (non-fault) in distribution system like load change.

Also, the models that consider constant resistance [19], [20], [21] will only perform well as long as the actual fault resistance is similar to the fault resistance considered during the generation of the training dataset. Even though these research work report their model performance with different resistance values, these reported values are only applicable to that specific resistance value. In contrast, [15], [16], [17], [18] takes a more practical approach and train their model considering a range of possible fault resistance values. As long as the fault resistance is within that range, the model is expected to hold its performance.

TABLE I
SUMMARY OF TECHNICAL DIFFERENCES WITH PREVIOUS GNN-BASED LITERATURE

Tasks	Chen et al. [15]	Sun et al. [16]	Freitas et al. [17]	Li et al. [18]	Mo et al. [19]	Hu et al. [20]	Nguyen et al. [21]	Ours
Fault Resistance	**Variable**	**Variable**	**Variable**	**Variable**	**Constant**	**Constant**	**Constant**	**Variable**
Fault Detection	✗	✗	✗	✗	✗	✗	✓	✓
Fault Localization	✓	✓	✓	✓	✓	✓	✓	✓
Fault Classification	✗	✗	✗	✗	✓	✓	✓	✓
Fault Resistance Estimation	✗	✗	✗	✗	✗	✗	✗	✓
Fault Current Estimation	✗	✗	✗	✗	✗	✗	✗	✓
Types of Fault Considered	*LG, LL, LLG*	*LG, LL, LLG LLL, LLLG*	*LG*	*LG, LL, LLG*	*LG, LL, LLG*	*LG, LL, LLG LLL, LLLG*	*LG, LL, LLG LLL, LLLG*	*LG, LL, LLG LLL, LLLG*

The drawback of [20], [21] is that their proposed model is dependent on temporal characteristics. To perform well the dataset needs to have high temporal resolution. As the scale of the distribution system grows data acquisition, storage and training overhead for this approach becomes progressively more demanding. In addition, there is a requirement for perfect time synchronization which further complicates the system design.

Considering these drawbacks the key contributions of our work are outlined as:

1) We propose a unified heterogeneous MTL-GNN architecture to perform 5 different tasks simultaneously for a fault event which are fault detection, fault localization, fault type classification, fault resistance estimation and fault current estimation.
2) The proposed model performs well in the presence of measurement error, variable resistance and topology changes as common factors considered in real-world deployments.
3) We utilize an explainability algorithm specific to GNN to identify key nodes in the grid which provides the opportunity for informed sparse measurement.

The remaining parts of the paper are organized as follows. In Section III, a brief overview of the test feeder system and the process involving the dataset generation is provided. The mathematical framework for the overall methodology and architecture used is given in Section IV. In the following Section V, we outline the details of the architecture, training procedure and hyperparameters assumed during training. Numerical results and discussions on them are presented in Section VI. Finally, Section VIII concludes the paper.

III. DATASET GENERATION

This section briefly covers the details of the IEEE-123 node feeder system and the dataset generation process as well as the underlying assumptions considered in the generation process.

A. IEEE-123 Node Feeder System

The IEEE-123 node feeder system [41] shown in Fig. 1 operates at a nominal voltage of 4.16 KV and consists of both overhead and underground lines. It has three-phase, two-phase and single-phase lines and a couple of open and closed switches, voltage regulators and a transformer.

There are a total of 85 nodes that have loads connected to them and most of them are connected to single-phase buses and the rest are connected to three-phase buses. For simulating all 5 types of short circuit faults including asymmetrical and symmetrical

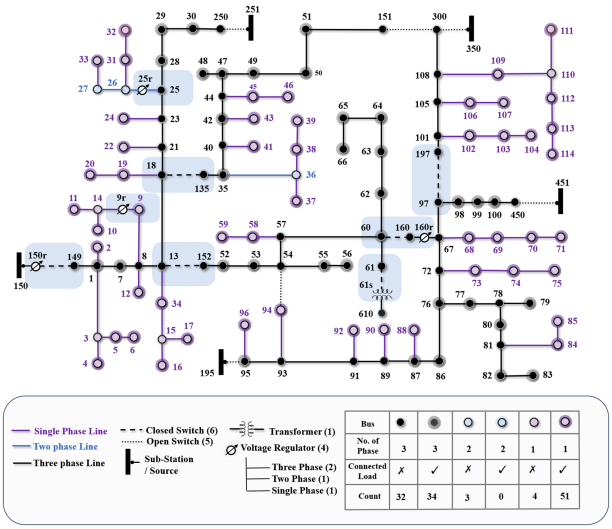

Fig. 1. Diagram of IEEE-123 node feeder system. The highlighted blue blocks represent the node pairs that are considered connected. The number of voltage regulators, transformers, and switches is mentioned in (·) and the number of buses, their phases and load connectivity are mentioned in the table. The active substation (source bus) is connected to the node $150r$.

faults all three phases need to be considered. Therefore, in our analysis, similar to [15], [18] we only consider three-phase nodes which tally up to 68 nodes including the 2 three-phase regulators 150r and 160r. Also, similar to [15], [18] we also make the assumption that some specific pairs of nodes are connected which are (149, 150r), (18, 135), (13, 152), (60, 160, 160r,), (61, 61s), (97, 197), (9, 9r), (25, 25r). The reason behind this assumption is the pairs (18, 135), (13, 152), (61, 61s), (97, 197), (60, 160) are connected via closed switches and the pairs (149, 150r), (9, 9r), (25, 25r) consists of buses and their corresponding regulators. This is shown in Fig. 1 via highlighted blue sections. One important thing to note here, in actuality, considering all the 4 regulators as separate nodes the total number of nodes in the feeder system results in 128 nodes.

B. Dataset Description and Generation Procedure

For dataset generation, we use OpenDSS [42], open-sourced by the electric power research institute (ERPI) as a power flow equation solver engine and use *py_dss_interface* module in Python to interface with it.

We opt to utilize only voltage phasors as measurements based on [15], where the authors showed that the performance of

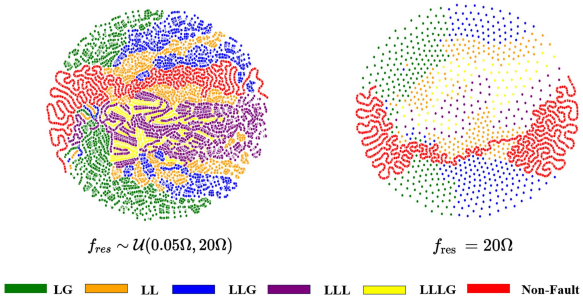

Fig. 2. t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of all the data points. (**Left**) shows the dataset generated with a variable range of fault resistance sampled from a uniform distribution $\mathcal{U}(0.05\,\Omega, 20\,\Omega)$. (**Right**) shows the dataset generated with a constant fault resistance $20\,\Omega$.

their model is almost identical with or without current phasors. Therefore, there is no incentive to use current phasors as features because that would just double the amount of computation needed at the expense of no performance increase. For the three-phase buses shown in Fig. 1 for all three phases (Phase A, Phase B, Phase C) voltage amplitude and angle (in radian) can be measured. For single-phase and two-phase buses values for the missing buses are padded with zero.

As our proposed model does 5 tasks simultaneously, for each data point we generate 5 labels as fault detection, fault location, fault classification, fault resistance, and fault current labels. The first three labels are discrete values whereas the last two labels are continuous values. As mentioned before for practical consideration we assume a range of fault resistance instead of a single fault resistance value. Fault resistance values are sampled from a uniform distribution (rounded up to 2 decimal point) which is $f_{res} \sim \mathcal{U}(\min_{res}, \max_{res})$, where f_{res} is the sampled fault resistance value and $\min_{res}$ and $\max_{res}$ are the lower and upper bound of the uniform distribution. For our initial analysis, we assume a lower bound of $0.05\,\Omega$ and an upper bound of $20\,\Omega$. The practical consideration claim we make is justified by Fig. 2 which clearly shows that for constant resistance analysis [19], [20], [21] the fault diagnosis becomes trivial. Fig. 2 shows t-SNE visualization of the high-dimensional features by projecting them into a 2-dimensional domain. For constant resistance of $20\,\Omega$, the features have minimal overlap and there is considerable distance between two neighboring points. Hence, the underlying data distribution is easily separable with non-linear decision boundaries which can be easily learned using any ML method. In our approach with variable fault resistance, the overlap between features is much more pronounced which makes it a much harder prediction task. However, in case of real-world deployment, our proposed approach with the variable fault resistance is much more practical.

For each fault type, we generate 20400 data samples. This results in a total of $20400 \times 5 = 102000$ data points for the fault events where there are 300 samples per bus. For non-fault event data generation, we vary all 91 loads connected to the 85 buses according to another uniform distribution given by $f_{load} \sim \mathcal{U}(\min_{load}, \max_{load})$. We assume $\min_{load} = 20$ KW and $\max_{load} = 80$ KW based on the typical load range associated with the IEEE-123 node feeder system. For non-fault events, the number of data points is also 20400. The sequence of steps for the data generation process is visualized in Fig. 3. The double-circled digits in the diagram specify the iteration hierarchy.

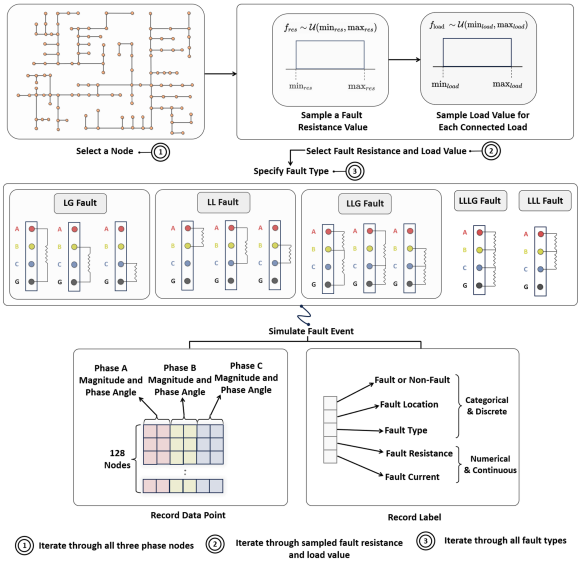

Fig. 3. Visualization of the sequence of procedures for a single data point and label generation. The double-circled digits represent the loop iteration points in the algorithm. By performing all the iterations in a hierarchical manner (the digits specify the order of the hierarchy) the entire dataset is generated.

1) At the first iteration point, we iterate through all 68 three phase nodes and for a specific node execute the following two steps.
2) At the second iteration point, for that specific node a fault resistance value f_{res} and load values f_{load} for the connected loads are sampled. In actual implementation, this is just iterating through a list consisting of 300 tuples of f_{res} and f_{load} values that were sampled in advance.
3) At the third iteration point, we iterate through all 5 fault types and fault is simulated for each fault type.

Going through the above-mentioned process results in $68 \times 300 \times 5 = 102000$ data points for fault events with associated labels. It should be pointed out that the fault simulation strategy for the asymmetric fault types and symmetric fault types differs slightly. For symmetric fault types which are LG, LL and LLG, the samples consist of 100 samples for 3 different states (based on connection difference between phases) which results in $3 \times 100 = 300$ samples for each of them.

All the values in the feature vector are standardized by subtracting the mean value and dividing by the standard deviation.

IV. Mathematical Framework

GNN uses message passing between nodes in the graph for the propagation of features which allows feature representation learning in addition to topological representation learning. GCN is a specific variant of GNN, first introduced by [43]. The proposed model consists of GCN layers as a common backbone followed by dense layers as prediction heads. GCN perform message passing between a given target node u in a particular layer l and the neighbors of the target node $v \in \mathcal{N}(u)$ according

to the following equation:-

$$\mathbf{h}_u^{(l)} = \sigma\left(W^{(l)} \sum_{v \in \mathcal{N}(u)\cup\{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right) \tag{1}$$

where, $\frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}$ is the normalization factor, $\mathbf{h}_v$ is the hidden representation of the neighboring nodes, $W^{(l)}$ is the weight matrix consisting trainable parameters, σ is the non-linear activation function and $\mathbf{h}_u^{(l)}$ is the hidden representation of the target node. This equation gives an intuitive understanding of how the message passing algorithm works, but for implementation (2) is more common in literature.

$$\mathcal{H}^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{\mathcal{A}}\tilde{D}^{-\frac{1}{2}}\mathcal{H}^{(l)}W^{(l)}\right) \tag{2}$$

where, $\tilde{\mathcal{A}} = \mathcal{A} + I$ is the adjacency matrix with self-loops and $\mathcal{A}$ is the original adjacency matrix without self-loops, $\tilde{D}$ is the degree matrix considering the modified adjacency matrix $\tilde{\mathcal{A}}$ and $\mathcal{H}^{(l)}$ is the lth GCN layer containing all the feature representation of all the nodes for that particular layer.

The node feeder system is represented by a graph $\mathcal{G} := (\mathcal{V}, \mathcal{E}, \mathcal{A})$ where $\mathcal{V}$ represents the set of nodes in the feeder system which can be represented as the union of three disjoint sets as shown in the following equation:

$$\mathcal{V} := \mathcal{V}_{1p} \cup \mathcal{V}_{2p} \cup \mathcal{V}_{3p} \tag{3}$$

where $\mathcal{V}_{1p}$, $\mathcal{V}_{2p}$ and $\mathcal{V}_{3p}$ respectively represent the nodes associated with three-phase, two-phase and single-phase buses and regulators. With the inclusion of the voltage regulators as nodes, $|\mathcal{V}_{3p}| = 68$, $|\mathcal{V}_{2p}| = 4$, $|\mathcal{V}_{3p}| = 56$ which equate to $|\mathcal{V}| = 128$. The set of edges is denoted by $\mathcal{E}$ where $|\mathcal{E}| = 127$ and $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the symmetric adjacency matrix. As $\mathcal{G}$ is a sparse graph, the adjacency matrix is replaced by a coordinate list format (COO) representation denoted by $E_c \in \mathbb{R}^{2\times 2|\mathcal{E}|}$. E_c holds the node pair that is connected by an edge. Owing to the fact that the graph under consideration is undirected in nature, if there is an edge between node pair (p, q) then both (p, q) and (q, p) are included in E_c.

From this point, we use superscript k to denote the index associated with a data point, subscript i to denote the index of a specific node and subscript t to denote the task index.

The generated dataset contains a feature vector associated with each node in the graph. That is represented by $\mathbf{z}_i^k \in \mathbb{R}^6$ which is the feature vector for ith node of the kth data point.

Each feature vector holds the value of voltage phasor meaning voltage amplitude (V_i) and angle (ϕ_i) for three phases. This can be mathematically represented by (4).

$$\mathbf{z}_i^k := \left[V_i^A, \phi_i^A, V_i^B, \phi_i^B, V_i^C, \phi_i^C\right]^k \tag{4}$$

where, the superscript A, B, C represent respectively the value associated with Phase-A, Phase-B and Phase-C. It is worth noting that since for the nodes in $\mathcal{V}_{2p}$ and $\mathcal{V}_{1p}$, values for some specific phases do not exist, they are replaced with zeros.

Stacking all the feature vectors for all $|\mathcal{V}|$ nodes in a graph $\mathcal{G}^k$ results in a feature matrix $\mathcal{X}^k \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ associated with that graph. A particular row i in the feature matrix $\mathcal{X}^k$ corresponds to the feature vector for the ith node. It should be noted the COO representation of the adjacency matrix is the same for all the data points *i.e.* $E_c^k = E_c, \forall k$. Therefore, the kth input data point of our dataset can be represented by $X^k := (\mathcal{X}^k, E_c)$.

For each input data point, there are 5 labels which are represented respectively by y_{detect}, y_{loc}, y_{type}, y_{res}, y_{current}. These signify the fault detection label, fault classification label, fault type label, fault resistance label and fault current label. For ease of representation, the subscripts are replaced by the corresponding task index. The first three labels are for classification type prediction and the last two labels are for regression type prediction. Now we define Y as an ordered list of all the labels which can be represented as follows:-

$$Y^k := (y_1^k, y_2^k, y_3^k, y_4^k, y_5^k) \tag{5}$$

Therefore, our dataset can be succinctly written as

$$\mathcal{D}^k := \{(X, Y)^k\} \tag{6}$$

where a single data point and the corresponding labels are indexed by k. The size of the dataset is denoted by $|\mathcal{D}^k| = N$ and the training and testing part of the dataset is represented by $\mathcal{D}_{tr}^k$ and $\mathcal{D}_{test}^k$ respectively, each of which has the size $|\mathcal{D}_{tr}^k| = N_{tr}$ and $|\mathcal{D}_{test}^k| = N_{test}$. Similarly, the inputs associated with the training and testing dataset are expressed as X_{tr} and X_{test} and labels associated are expressed by Y_{tr} and Y_{test}.

Now in implementation, we modify the labels of fault current because they can have a varying range of magnitude up to order of 10^3 or more. This can make the entire optimization process unstable and result in exploding gradients during the training phase. Hence we first normalize the labels to get $\tilde{y}_{t=5}$ followed by taking the negative log which can be expressed as

$$y_5^k := -\ln\left(\frac{y_5^k}{\sum_{k=1}^N y_5^k}\right) \tag{7}$$

This makes the fault current labels in the same range as fault resistance and hence results in much more effective training.

We define our heterogeneous MTL model as f_θ, parameterized by θ. The model can be sectioned into two parts. The first part is the common backbone GNN represented by $g_{\theta^{sh}}$ where θ^{sh} are the parameters of the common backbone g. Now for each task t, there is a sequence of separate dense layers which can be represented by h_{θ^t} where θ^t represent parameters associated with a specific task t. As there are a total of $T = 5$ tasks, we can represent t as $t \in \{1, 2, \ldots, T\}$ which means the network parameters can be expressed as $\theta := \{\theta^{sh}, \theta^1, \theta^2, \ldots, \theta^T\}$. Now the predicted output $\hat{y}_t^k$ of a task t for kth data point can be expressed as (8)

$$\hat{y}_t^k := h_{\theta^t} \circ g_{\theta^{sh}}(X_{tr}^k) \tag{8}$$

For each task, we define a loss function $\mathcal{L}_t$ which takes in X_{tr}^k and the parameters of the proposed model and computes the loss for that particular task. Each loss is weighted by a weighting factor w_t. The overall objective function including L2 regularization can be expressed as (9) where $\lambda \in \mathbb{R}$ is the regularization hyperparameter.

$$\min_\theta \sum_{k=1}^{N_{tr}} \sum_{t=1}^{T} w_t \mathcal{L}_t(\theta, X_{tr}^k) + \lambda\|\theta\|_2 \tag{9}$$

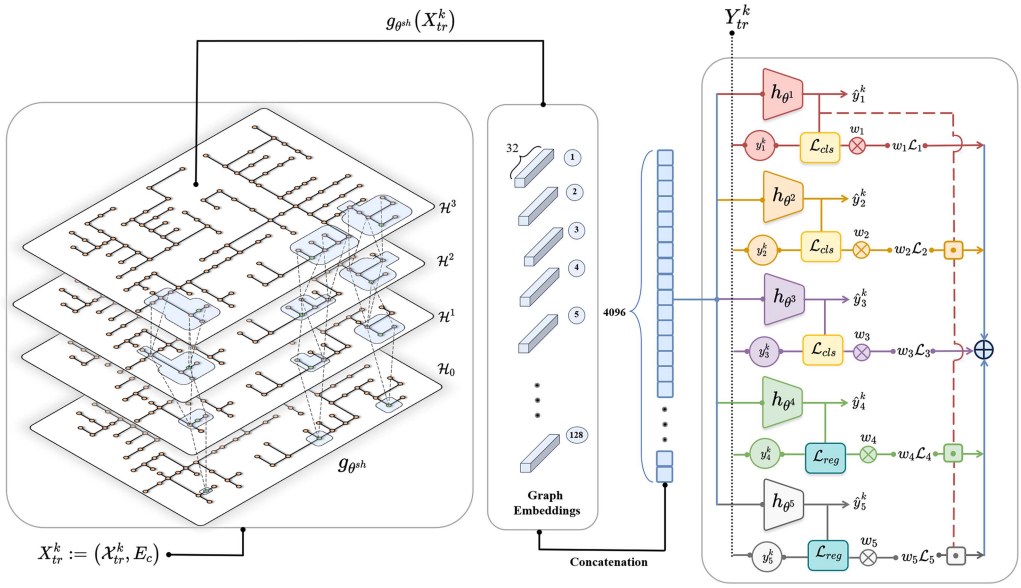

Fig. 4. Architecture of the proposed heterogenous MTL-GNN. The input features go through the common backbone GNN to generate graph embeddings. For visual clarity message passing across the layers for a couple of nodes (highlighted in green) is shown, where the blue highlighted sections signify nodes included in the message passing process (**Left**). The embeddings generated by 128 nodes are flattened and concatenated together to convert to a one-dimensional vector (**Middle**). The concatenated feature vector is passed to 5 heads, three classification heads and two regression heads. The corresponding loss is computed based on the predicted output ($\hat{y}_t^k$) and ground truth label (y_t^k). The computed loss for each task is weighted and summed together (**Right**).

For the classification layers jointly expressed as $h_{\theta^{cls}} = \{h_{\theta^1}, h_{\theta^2}, h_{\theta^3}\}$ the negative log-likelihood (NLL) loss function is used and for the regression layers jointly expressed as $h_{\theta^{reg}} = \{h_{\theta^4}, h_{\theta^5}\}$ the mean squared error (MSE) loss is used. For generality, we express the classification losses as $\mathcal{L}_{cls}$ and regression losses as $\mathcal{L}_{reg}$ and they are defined as follows over entire training data:-

$$\mathcal{L}_{cls} = -\sum_{k=1}^{N_{tr}} y_t^k \log \frac{\exp(h_{\theta^t} \circ g_{\theta^{sh}}(X_{tr}^k))}{\sum_{j=1}^{m} \exp(h_{\theta^t} \circ g_{\theta^{sh}}(X_{tr}^k))} \tag{10}$$

where $t \in \{1, 2, 3\}$ which are the task index for classification task and m is the number of classes for task t.

$$\mathcal{L}_{reg} = \frac{1}{N_{tr}} \sum_{k=1}^{N_{tr}} (\hat{y}_t^k - y_t^k)^2 \tag{11}$$

in this case $t \in \{4, 5\}$ which are the indices for regression tasks.

V. Architecture Detail and Model Training

The GNN backbone of the proposed model consists of 3 GCN layers which are $\mathcal{H}^1, \mathcal{H}^2, \mathcal{H}^3$ and $\mathcal{H}^0$ is the input layer.

The number of layers is restricted to 3 to avoid over smoothing issue [44] which is a common problem for deep GNNs. For normalization of the features flowing through the layers, layer normalization is used [45].

In Fig. 4 the entire forward propagation through the network is shown. The forward propagation through GNN allows feature representation learning through message passing. In addition, topological information is captured in the learned representations.

After that, learned embedding is extracted from all 128 nodes which are then flattened and concatenated together to generate a feature vector of dimension $128 \times 32 = 4096$. This feature vector is passed through the different heads for different prediction tasks. It is crucial to note that for non-fault events we

TABLE II
Number of Parameters for a Single Batch of Size 32

Layers	# Parameters	Output Shape
Input Layer ($\mathcal{H}^0$)	—	$(32, 128, 6)$
GNN Backbone($g_{\theta^{sh}}$):		
GCNConv ($\mathcal{H}^1$)	224	$(32, 128, 32)$
Layer Normalization	64	$(32, 128, 32)$
GCNConv ($\mathcal{H}^2$)	1056	$(32, 128, 32)$
Layer Normalization	64	$(32, 128, 32)$
GCNConv ($\mathcal{H}^3$)	1056	$(32, 128, 32)$
Concatenation Layer	—	$(32, 4096)$
Classification Heads($h_{\theta^{cls}}$):		
Fault Detection Head (h_{θ^1})	172,974	$(32, 2)$
Fault Localization Head (h_{θ^2})	135,328	$(32, 128)$
Fault Type Classification Head (h_{θ^3})	173,222	$(32, 6)$
Regression Heads($h_{\theta^{reg}}$):		
Fault Resistance Estimation Head (h_{θ^4})	172,969	$(32, 1)$
Fault Current Estimation Head (h_{θ^5})	172,969	$(32, 1)$
Total	829,926	

TABLE III
Hyperparameters and Their Values

Hyperparameters	Value
Batch Size	32
Epochs	500
Hidden Layer Activation ($g_{\theta^{sh}}$)	ReLU
Output Layer Activation ($h_{\theta^{cls}}$)	Log Softmax
Output Layer Activation ($h_{\theta^{reg}}$)	–
Optimizer	AdamW
Initial Learning Rate (α)	0.001
Gradient Clipping Threshold ($\mathcal{C}$)	5
Weight Decay (λ)	10^{-3}
Dropout Rate (D_r)	0.2
Train-Test Split	$80 - 20$
$w_t \rightarrow (w_1, w_2, w_3, w_4, w_5)$	$(0.01, 0.8, 0.9, 0.1, 0.04, 0.05)$
Random Seed (for reproducibility)	66

Algorithm 1: Training Algorithm of MTL-GNN.

Input: $\mathcal{D}_{tr}, \alpha, \lambda, \mathcal{C}, w_t$ ▷ *Training set and hyperparameters*
Output: θ ▷ *Learned parameters of the model*

$X_{tr}, Y_{tr} \leftarrow \mathcal{D}_{tr}$
while $s <$ number of epochs **do**
  for $k = 1$ to N_{tr} **do**
    for $t = 1$ to T **do**
      $\hat{y}_t^k \leftarrow h_{\theta t} \circ g_{\theta sh}(X_{tr}^k)$
      if $t = 1$ and $\hat{y}_1^k == 1$ **then** ▷ *Non-fault event*
        $\mathcal{L}_s \leftarrow \sum_k \sum_{t\in\{1,3\}} w_t \mathcal{L}_{cls}$
      else
        $\mathcal{L}_s \leftarrow \sum_k \sum_t w_t \mathcal{L}_t$
  $\nabla_\theta \mathcal{L}_s \leftarrow \begin{cases} \nabla_\theta \mathcal{L}_s, & \text{if } \|\nabla_\theta \mathcal{L}_s\| \le \mathcal{C} \\ \frac{\mathcal{C}}{\|\nabla_\theta \mathcal{L}_s\|} \cdot \nabla_\theta \mathcal{L}_s, & \text{otherwise} \end{cases}$
  $\theta_s \leftarrow \text{AdamW}(\theta_s, \alpha, \lambda, \nabla_\theta \mathcal{L}_s)$
$\theta \leftarrow \theta_s$

restrict all the losses except for $\mathcal{L}_1$ and $\mathcal{L}_3$, as all the other losses correspond to a fault event. This allows gradient flow (during backpropagation) through the network only for $\mathcal{L}_1$ and $\mathcal{L}_3$ loss for non-fault data samples.

Without this mechanism, the parameters associated with fault events $(\theta_2, \theta_4, \theta_5)$ would get updated also.

The total number of parameters per layer and output shapes are mentioned in Table II. The model was trained for 500 epochs with a batch size of 32 with AdamW [46] optimizer on an NVIDIA A100 80GB GPU. For stability of training gradient clipping and L2 regularizer are used as well as dropout is used ($D_r = 0.2$) to prevent overfitting. The hyperparameters are outlined in Table III.

The training procedure is shown in Algorithm 1. For training the model the training dataset $\mathcal{D}_{tr}$ and the hyperparameters : initial learning rate (α), regularizer hyperparameter/weight decay (λ), gradient clipping threshold ($\mathcal{C}$), loss weighting factors (w_t) need to be specified.

To determine the w_t hyperparameters a trial run with the same weight was conducted to get a gauge about which task is easier to learn for the model and after that, these values are set taking into account the importance of the task. We specify the epoch index as s and the corresponding overall weighted loss associated with that epoch is $\mathcal{L}_s$. After a forward pass through the network, $\mathcal{L}_s$ is calculated followed by the gradient of the loss with respect to model parameters which is $\nabla_\theta \mathcal{L}_s$. Gradient clipping is applied if necessary and finally, the AdamW optimizer updates the model parameters based on the defined hyperparameters.

VI. Numerical Tests

In this section, we first introduce the evaluation metrics used to assess the model performance, followed by the regression and classification performance of the model. Then we evaluate the model performance on sparse measurements where the sparse node-set is strategically chosen based on an explainability algorithm.

A. Metrics Used for Evaluation of the Model Performance

For fault detection which is a binary classification task, we report balanced accuracy and f1-score. The reason for reporting balanced accuracy as opposed to accuracy is the class imbalance for fault detection. For fault localization, we report the location accuracy rate (LAR) and f1-score. The LAR^h (h indicate the number of hops) is a common metric to evaluate the performance for fault localization. It quantifies the percentage of correctly identified fault locations within a certain h-hop distance from the actual fault location. LAR^0 capture the accuracy for exact fault location. In contrast, LAR^1 measures the performance of the identified fault locations within 1-hop distance from the actual fault location. LAR^2 computes the same thing but for 2-hop distance. The reason for computing this metric is to evaluate the model's ability to provide an estimation of the fault's approximate position, even if the model cannot precisely identify the exact fault location.

For fault type classification we report accuracy and f1-score. In addition, we also use the confusion matrix which provides insight into the model's performance specific to a fault type.

For the regression tasks, fault resistance and fault classification estimation we report MSE and mean absolute percentage error (MAPE) for the test data set. These two metrics are given by the following equation.

$$\text{MSE}_{test} = \frac{1}{N_{test}} \sum_{k=1}^{N_{test}} (\hat{y}_t^k - y_t^k)^2 \tag{12}$$

$$\text{MAPE}_{test} = \frac{1}{N_{test}} \sum_{k=1}^{N_{test}} \left| \frac{\hat{y}_t^k - y_t^k}{y_t^k} \right| \times 100 \tag{13}$$

where $t = 4$ indicates the metrics for fault resistance and $t = 5$ indicates the metrics for fault current. The reason for reporting MAPE in addition to MSE is the sensitivity to scale for MSE. As we report performance for different ranges of fault current and resistance, this varying range needs to be taken into account. All results are summarized in Table IV. For simulating possible measurement errors, we introduce noise n sampled from a zero mean gaussian distribution $\mathcal{N}(0, \sigma_{noise})$ with a variance value set to σ_{noise}. The variance σ_{noise} is respectively set to 0.0001, 0.001 and 0.01 for different levels of noise. We also consider the model performance under varying ranges of fault resistance values which are $0.05\,\Omega - 20\,\Omega$, $20\,\Omega - 100\,\Omega$ and $100\,\Omega - 500\,\Omega$. Furthermore, we consider the fact there might be a lack of historical data for fault events. To imitate this, we decrease the size of the dataset and evaluate the model performance on the decreased dataset. The smallest size we consider is 15% of the original which results in only 45 samples per node. We also report another set of metrics considering two possible topology changes in the feeder system.

B. Performance of the Model for Classification Tasks

From Table IV it is apparent that the model can easily distinguish between a load change event and a fault event which is intuitive given that there are only two classes. In addition, the voltage fluctuation in the case of a fault event differs significantly from that of a load change event. In the case of fault localization, despite the different variations, the model is robust enough to maintain relatively high LAR^1 and LAR^2. When we consider measurement error, first we evaluate the performance with an out-of-distribution (OOD) test set meaning the training dataset didn't contain noisy samples. For low and moderate noise, even with OOD samples the fault localization performance holds. For

TABLE IV
PERFORMANCE OF PROPOSED MODEL ON ALL 5 TASKS CONSIDERING MEASUREMENT ERROR, RESISTANCE CHANGE AND DATASET SIZE

Criteria	Fault Detection		Fault Localization				Fault Classification		Fault Resistance Estimation		Fault Current Estimation	
	Balanced Accuracy	F1-Score	LAR^0	LAR^1	LAR^2	F1-Score	Accuracy	F1-Score	MSE	MAPE	MSE	MAPE
Measurement Error												
†Low Noise: $n \sim \mathcal{N}(0, 0.0001)$	1.0	1.0	0.982	0.999	0.999	0.983	0.991	0.991	0.108	0.094	0.022	0.008
†Moderate Noise: $n \sim \mathcal{N}(0, 0.001)$	1.0	1.0	0.980	0.999	0.999	0.980	0.992	0.992	0.111	0.102	0.022	0.008
†High Noise : $n \sim \mathcal{N}(0, 0.01)$	1.0	1.0	0.715	0.910	0.963	0.714	0.989	0.824	0.208	0.111	0.031	0.010
High Noise : $n \sim \mathcal{N}(0, 0.01)$	1.0	1.0	0.954	0.999	0.999	0.952	0.992	0.992	0.136	0.116	0.040	0.011
Resistance Range Change												
$0.05\Omega - 20\Omega$	1.0	1.0	0.984	0.999	0.999	0.984	0.993	0.993	0.133	0.112	0.026	0.008
$20\Omega - 100\Omega$	1.0	1.0	0.983	0.999	0.999	0.982	0.991	0.991	1.468	0.020	0.008	0.006
$100\Omega - 500\Omega$	1.0	1.0	0.917	0.990	0.999	0.914	0.949	0.948	68.92	0.021	0.028	0.009
% Samples /Fault Type												
15%	1.0	1.0	0.934	0.997	0.999	0.931	0.991	0.991	0.206	0.302	0.040	0.014
25%	1.0	1.0	0.957	0.995	0.999	0.957	0.992	0.992	0.146	0.174	0.030	0.011
50%	1.0	1.0	0.975	0.999	0.999	0.975	0.991	0.991	0.211	0.181	0.040	0.011
Topology Change												
†Open 97-197, Close 151-300	1.0	1.0	0.988	0.999	0.999	0.987	0.993	0.993	0.091	0.113	0.024	0.008
†Open 18-135, Close 151-300	1.0	1.0	0.987	0.999	0.999	0.987	0.993	0.993	0.093	0.118	0.024	0.009

†Out-of-Distribution (OOD) data

TABLE V
PERFORMANCE OF TRADITIONAL ML MODELS UNDER DIFFERENT NOISE INTENSITY (FOR RESISTANCE RANGE $0.05\,\Omega$–$20\,\Omega$)

Model	Noise Level	Detection Accuracy	Classification Accuracy	Localization Accuracy	Resistance Estimation MAPE	Current Estimation MAPE
XGBoost	Low	1.0	0.98	0.88	5.27	0.28
	Moderate	1.0	0.97	0.84	6.03	0.43
	High	1.0	0.91	0.68	9.07	1.46
Random Forest	Low	1.0	0.98	0.90	1.15	0.07
	Moderate	1.0	0.97	0.88	1.82	0.11
	High	1.0	0.96	0.75	4.38	1.50

high levels of noise, the performance goes down but if the noisy samples are included in the training set the location accuracy improves significantly. However, this robustness to noise can be due to various reasons. Hence, further analysis is performed to evaluate if the robustness holds in case the proposed model is swapped out with other ML models. As regards fault detection, Table V demonstrates that the measurement noise has no impact on the obtained accuracy. However, for the other 4 tasks, the robustness to noise does not hold and the performance of other ML models (see Table V) is significantly worse than the proposed multi-task model as detailed in Table IV. Also, the model manages to sustain relatively high accuracy despite a broad range of resistance values. Similar conclusions can be made for varying dataset sizes. One important thing to note here, even when the model is not able to localize the exact fault point, it can approximate the location with high accuracy.

The performance retains for topology changes in the feeder system. Note that, these metrics are computed with OOD test samples meaning the test samples with the modified topology were never included in the training data. When generating these OOD samples the connectivity information, E_c, was changed to reflect the topology change. For fault-type classification, it is important to outline per-class performance as the probability of all fault types is not the same and varies according to fault type; LG (70%–80%), LLG (17%–10%), LL (10%–8%), LLL, LLLG (3%–%) [3]. This means it is more important the model is able to classify the asymmetrical faults compared to the symmetrical faults.

The confusion matrix shown in Fig. 5 summarizes the performance for each fault type. The misclassified fault types belong to the LLL and LLLG fault class which is further corroborated

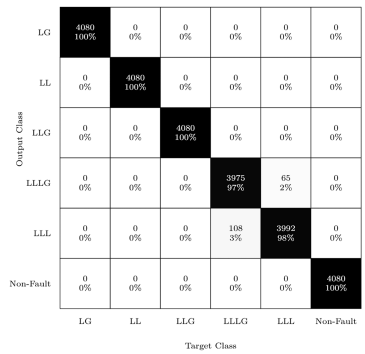

Fig. 5. Confusion matrix for fault type classification task. The first three classes are asymmetric faults, the next two classes are symmetric faults and the final class corresponds to non-fault events.

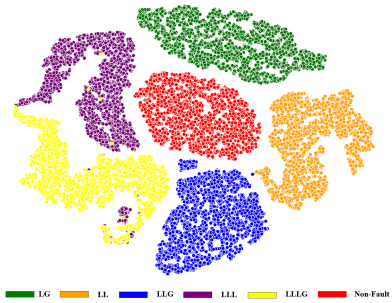

Fig. 6. t-SNE visualization of last layer features of fault type classification head h_θ^3 for the test set $\mathcal{D}_{test}$. Except for a few samples in the LLL fault type and LLLG fault type, most features are clearly separable based on fault type.

by the t-SNE visualization of the last layer features of h_θ^3 shown in Fig. 6. Samples from other fault types are clearly separable even when they are projected into a 2-dimensional feature space.

C. Performance of the Model for Regression Tasks

For both regression tasks, the performance of the model remains consistently high across all the variations considered. Even though for resistance levels $100\,\Omega - 500\,\Omega$, MSE is high due to the scale range, MAPE remains low similar to other resistance levels.

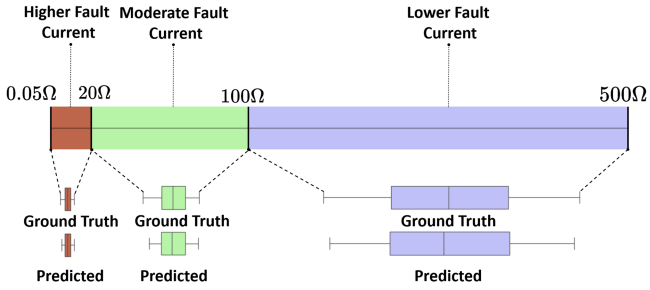

Fig. 7. Boxplot of both the ground truth and predicted distribution of the fault resistance.

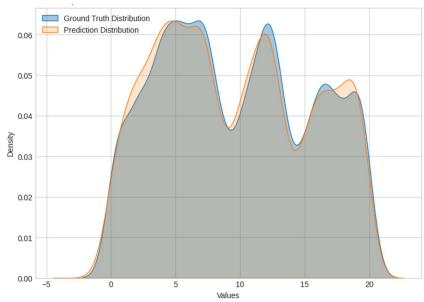

Fig. 8. Ground truth (y_4) and predicted ($\hat{y}_4$) fault resistance distribution on the test set ($\mathcal{D}_{test}$). The grey section is the correctly predicted part of the distribution.

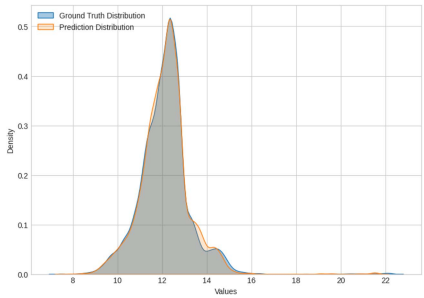

Fig. 9. Ground truth (y_5) and predicted ($\hat{y}_5$) redefined fault current distribution on the test set ($\mathcal{D}_{test}$). The grey section is the correctly predicted part of the distribution.

Fig. 7 visualizes the range and variance difference among the three fault resistance levels considered in this paper. For resistance level $0.05\,\Omega - 20\,\Omega$, the fault resistance variance is the lowest, but the fault current level is higher compared to other resistance levels. For resistance level $100\,\Omega - 500\,\Omega$, it is exactly the opposite. This further highlights the scale sensitivity issue with MSE that results in a high MSE score with the resistance level $100\,\Omega - 500\,\Omega$. As MAPE is computed as a percentage of the actual value, it is scale-independent and is preferred with metrics that involve a broader range and large variance which is the case for $100\,\Omega - 500\,\Omega$.

Fig. 8 shows the distribution plot for the ground truth and predicted value for fault resistance on the test set. For the most part, the predicted distribution closely resembles the ground truth distribution. Similarly, Fig. 9 shows the ground truth distribution and predicted distribution for the fault current. One important thing to mention here, this plot is for the transformed fault current label mentioned in (7) rather than the actual fault current.

Algorithm 2: Sparse Node-Set Generation Algorithm.

Input: $X_{test}, epoch_{GE}, h_{\theta^2} \circ g_{\theta^{sh}}(trained), w_{th}$
Output: $\mathcal{V}_{x\%}$
for $k = 1$ to N_{test} **do**
  $E_{\mathcal{I}}^k \leftarrow \text{GNNExplainer}(X_{test}^k, h_{\theta^2} \circ g_{\theta^{sh}}, epoch_{GE})$
  for $p = 1$ to $|\mathcal{E}|$ **do**
    if $E_{\mathcal{I}}^k(p) > w_{th}$ **then** ▷ *Thresholding with* w_{th}
      $E_{w_{th}}^k(p) \leftarrow 1$
    else
      $E_{w_{th}}^k(p) \leftarrow 0$
  $\{\mathcal{V}_{sparse}\}^k \leftarrow \text{GetConnectedNodePairs}(E_{w_{th}}^k)$
  $\mathcal{V}_{x\%} \leftarrow \bigcup_k \{\mathcal{V}_{sparse}\}^k$

To get the actual fault current prediction and ground truth labels, the inverse operation of (7) can be performed which is given by the following equation:

$$y_5^k := \sum_{k=1}^{N} y_5^k \times \exp(-y_5^k) \tag{14}$$

D. Using Explainability to Identify Key Nodes

Out of all the tasks, fault localization is the most important given that the model performs really well for fault detection. Considering practical implications, for the remaining three tasks, it is acceptable if the model can approximate the prediction. So far in analysis, we assumed complete observability which implies all node voltage phasors can be measured. As the scale of the distribution system grows it becomes more and more harder to maintain complete observability considering cost and system complexity. Therefore, we propose a novel approach to locate key nodes using an explainability algorithm GNNExplainer proposed by [47]. GNNExplainer uses an optimization that maximizes the mutual information between a GNN's prediction and distribution of possible subgraph structures to identify important subgraphs. The most notable characteristic of this algorithm is it does not require ground truth labels. Using this algorithm we generate four sparse node sets $\mathcal{V}_{10\%}$, $\mathcal{V}_{20\%}$, $\mathcal{V}_{50\%}$ and $\mathcal{V}_{75\%}$ where the subscript denotes the percentage of nodes out of 128 which has data available. The feature vector of the rest of the nodes is set to 0.

To generate these sets we pass each test data sample X_{test}^k and the trained model with localization head $h_{\theta^2} \circ g_{\theta^{sh}}$ to the GNNExplainer algorithm which generates an edge importance vector given by $E_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{E}|}$, indexed by p. For each edge in E_c a corresponding weight is generated which signifies the importance of that edge. A weight value closer to 1 means a more important edge and a value closer to 0 means a less important edge. We threshold the values in $E_{\mathcal{I}}$ with a threshold w_{th}. After thresholding, the transformed edge importance vector is expressed by $E_{w_{th}} \in \mathbb{R}^{|\mathcal{E}|}$. Edges that have an importance score of more than w_{th} are kept and the rest are disregarded. The nodes connected to the edges in $E_{w_{th}}$ after thresholding are regarded as the important nodes for the kth data point. In this way, for each data point, we get a sparse node set given by $\{\mathcal{V}_{sparse}\}^k$. Then the union of these sparse important node sets generates the final sparse node set which is given by the

TABLE VI
LAR^h AND F1-SCORE WITH SPARSE NODE-SET

Sparse Sets	LAR^0	LAR^1	LAR^2	F1-Score
$\mathcal{V}_{75\%}$	0.975	0.999	0.999	0.975
$\mathcal{V}_{50\%}$	0.962	0.998	0.999	0.962
$\mathcal{V}_{20\%}$	0.942	0.987	0.992	0.943
$\mathcal{V}_{10\%}$	0.849	0.932	0.955	0.849
$\mathcal{V}_{50\%}^{random_{avg}}$	0.916	0.993	0.999	0.913
$\mathcal{V}_{50\%}^{random_{min}}$	0.886	0.989	0.998	0.881

following equation:-

$$\mathcal{V}_{x\%} := \bigcup_{k=1}^{N} \{\mathcal{V}_{sparse}\}^k \tag{15}$$

where the $x\%$ value depends on the threshold value. We set the value of threshold w_{th} to respectively $0.57, 0.52343, 0.487$ and 0.44 to generate$\mathcal{V}_{10\%}$, $\mathcal{V}_{20\%}$, $\mathcal{V}_{50\%}$ and $\mathcal{V}_{75\%}$.

The entire process is described in the Algorithm 2. In addition to other parameters mentioned above, an epoch number needs to be specified for the internal optimization of the GNNExplainer.

To validate this approach we also randomly sample 50% nodes several times and train the model to generate fault localization metrics on the test set. $\mathcal{V}_{50\%}^{random_{avg}}$ represent the average of these metrics and $\mathcal{V}_{50\%}^{random_{min}}$ represent the minimum of these metrics from the samples generated. The results of this analysis are summarized in Table VI. It is apparent that the model is robust enough to maintain the LAR^1 and LAR^2 irrespective of which sparse node-set is used. But for LAR^0 and F1-Score, the sparse set generated with GNNExplainer is comparatively better. $\mathcal{V}_{50\%}$ has a 4.78% increase in LAR^0 compared to $\mathcal{V}_{50\%}^{random_{avg}}$ and a 7.9% increase compared to $\mathcal{V}_{50\%}^{random_{min}}$ which means the sparse node-set generation algorithm was able to identify the important node set for fault localization. For $\mathcal{V}_{10\%}$ which represents only 10% the nodes, even though LAR^0 goes down significantly the LAR^1 and LAR^2 still holds up. Note that, state estimation methods can be used to approximate the measurements from other nodes. To this end, we outline the benefits gained by identifying important nodes through GNNExplainer:

1) *Reduction in cost:* There are several data acquisition systems in a distribution network which include phasor measurement unit (PMU), micro-phasor measurement unit (μ-PMU), distribution-level PMU (D-PMU) and smart meters. As the scale of the distribution system increases, the number of measurements increases. This limits a complete observability of the system while keeping the cost of the system within bounds. Identifying the important nodes and collecting data only from those nodes will significantly decrease the overall cost.
2) *Reduction in system complexity:* With a large number of measurments the system complexity increases substantially as time synchronization and latency in data processing come into the picture. An alternative way to maintain the observability of the whole system with fewer data acquisition units is to use state estimation techniques. However, even with state estimation methods there is the possibility of running into convergence issues and accumulation of errors. These issues can be avoided or their effect can be subsided if the system is designed with a lower number of nodes through the proposed approach.
3) *Reduction in training overhead:* As the scale of the distribution system grows, the training overhead of the proposed model will also increase. Generating sparse features by setting unmeasured values to zero, will greatly reduce the training overhead.
4) *No need for ground truth labels:* GNNExplainer does not require any ground truth labels for the important nodes. This means no domain or expert knowledge about the distribution system is required to annotate important nodes in advance as the algorithm can select those.

VII. CONSIDERATIONS FOR IMPLEMENTATION IN A PRACTICAL SETTING

The proposed method deals with the challenges that come with a practical deployment. However, there are some practical considerations that need to be taken into account including thorough evaluation of the system before deployment, alternative software choice for simulation and day-to-day operation strategy for grid operators. In this section, we discuss how to address these, along with citing relevant research that offers insights into how the proposed method will work in practice.

The National Renewable Energy Laboratory (NREL) has developed a repository with synthetic but realistic data that contains feeder systems with millions of buses and also provides OpenDSS scripts for these distribution systems. This repository named SMART-DS [48] can be used to test the proposed method under different realistic conditions. Another option is to use a framework like OPAL-RT that provides a distribution management system (DMS) that is capable of real-time simulation and provides compatibility with OpenDSS.

A notable work that uses D-PMU data simulated with OPAL-RT in a physical test-bed to detect events is [49]. The framework provided in this work follows IEEE standards throughout and can be used to validate the performance of the proposed method in a realistic setting. Another work that has a working prototype of using μ-PMUs to diagnose distribution level events is [50] which is a distribution network at Lawrence Berkeley National Laboratory. This work is a proof-of-concept for how our proposed method can be used in real time with data aggregated from μ-PMUs.

We have investigated the changes in the observability of nodes. For example, an observability of approximately 20% of the nodes still provides a high accuracy in predictions as shown in Table VI. It is worth noting that state estimation methods can be used to get observability of these 20% nodes which has been explored in previous works [51], [52]. Obviously using state estimation methods will introduce noise into observations but as shown in Table IV our proposed method is robust to noise. Another practical aspect to consider is how the grid operators will manage the system on a day-to-day basis. To ensure the reliability of model predictions, grid operators need access to more information. Fig. 10 illustrates how we envision a user interface. In this figure, a sample data point with fault event at node 77 is used for the visualization. The grid operators will be provided by both the predicted location and the neighboring nodes within 1-hop and 2-hop distances. Based on the results in

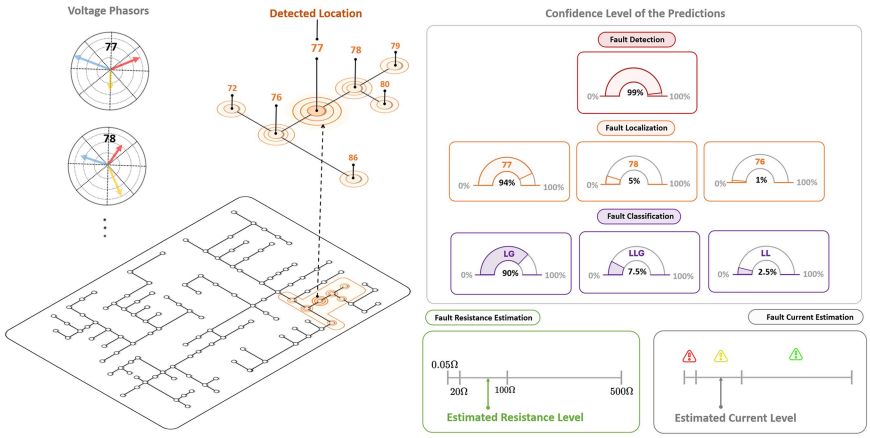

Fig. 10. Envisioned user interface for grid operators. (**Left**) Detected fault location (in this example node 77) in the distribution grid along with 1-hop nodes (in this example 76, 78) and 2-hop nodes (in this example 72, 86, 79, 80) in the neighborhood of the predicted node. (**Right**) The confidence level behind the predictions provides grid operators with additional information to verify the reliability of the prediction and fault resistance and fault current estimation allow grid operators to take appropriate safety measures for fault isolation and clearance.

Table IV almost 100% of the time the predicted fault location is in the 2-hop neighborhood of the actual faulty node. In addition, grid operators will have access to the voltage phasors of these nodes which provides more context behind the predictions. Furthermore, the confidence level of these predictions will allow grid operators to make an informed decision and undertake necessary safety measures before taking any action.

VIII. Conclusion

In this paper, we proposed an MTL-GNN capable of performing 5 different tasks simultaneously even with a sparse node-set. This sparse node-set is generated with a novel algorithm and to the best of our knowledge this is the first work that uses a GNN explainability algorithm for an informed node selection. There are some challenges associated with the proposed method that can be addressed in an extension of this work. The proposed architecture requires more hyperparameter tuning compared to a single-task learning model. A possible future work is automating the process of hyperparameter selections.

Besides that, the number of tasks needed for a distribution network needs to be specified in advance to configure the model architecture. For example, if the power distribution system requires only fault detection and localization, the architecture needs to be modified prior to deployment. A more adaptive framework for optional number of tasks can be of interest.

References

[1] O. P. Veloza and F. Santamaria, "Analysis of major blackouts from 2003 to 2015: Classification of incidents and review of main causes," *Electricity J.*, vol. 29, no. 7, pp. 42–49, 2016.

[2] H. Guo, C. Zheng, H.H.-C. Iu, and T. Fernando, "A critical review of cascading failure analysis and modeling of power system," *Renewable Sustain. Energy Rev.*, vol. 80, pp. 9–22, 2017.

[3] J. L. Blackburn and T. J. Domin, *Protective Relaying: Principles and Applications*. Boca Raton, FL, USA:CRC Press, 2006.

[4] A. Bahmanyar, S. Jamali, A. Estebsari, and E. Bompard, "A comparison framework for distribution system outage and fault location methods," *Electric Power Syst. Res.*, vol. 145, pp. 19–34, 2017.

[5] R. Dashti, M. Daisy, H. Mirshekali, H. R. Shaker, and M. H. Aliabadi, "A survey of fault prediction and location methods in electrical energy distribution networks," *Measurement*, vol. 184, 2021, Art. no. 109947.

[6] J. De La Cruz, E. Gómez-Luna, M. Ali, J. C. Vasquez, and J. M. Guerrero, "Fault location for distribution smart grids: Literature overview, challenges, solutions, and future trends," *Energies*, vol. 16, no. 5, p. 2280, 2023.

[7] S. Javadian, A. Nasrabadi, M.-R. Haghifam, and J. Rezvantalab, "Determining fault's type and accurate location in distribution systems with DG using MLP neural networks," in *Proc. IEEE Int. Conf. Clean Elect. Power*, 2009, pp. 284–289.

[8] H. A. Tokel, R.Al. Halaseh, G. Alirezaei, and R. Mathar, "A new approach for machine learning-based fault detection and classification in power systems," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf.*, 2018, pp. 1–5.

[9] M.-F. Guo, X.-D. Zeng, D.-Y. Chen, and N.-C. Yang, "Deep-learning-based earth fault detection using continuous wavelet transform and convolutional neural network in resonant grounding distribution systems," *IEEE Sensors J.*, vol. 18, no. 3, pp. 1291–1300, Feb. 2018.

[10] W. Li, D. Deka, M. Chertkov, and M. Wang, "Real-time faulted line localization and PMU placement in power systems through convolutional neural networks," *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4640–4651, Nov. 2019.

[11] M. Zou, Y. Zhao, D. Yan, X. Tang, P. Duan, and S. Liu, "Double convolutional neural network for fault identification of power distribution network," *Electric Power Syst. Res.*, vol. 210, 2022, Art. no. 108085.

[12] F. G. Y. Souhe et al., "Fault detection, classification and location in power distribution smart grid using smart meters data," *J. Appl. Sci. Eng.*, vol. 26, no. 1, pp. 23–34, 2022.

[13] S. Paul, S. Grijalva, M. J. Aparicio, and M. J. Reno, "Knowledge-based fault diagnosis for a distribution system with high PV penetration," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf.*, 2022, pp. 1–5.

[14] M. R. Shadi, M.-T. Ameli, and S. Azad, "A real-time hierarchical framework for fault detection, classification, and location in power systems using PMUs data and deep learning," *Int. J. Elect. Power Energy Syst.*, vol. 134, 2022, Art. no. 107399.

[15] K. Chen, J. Hu, Y. Zhang, Z. Yu, and J. He, "Fault location in power distribution systems via deep graph convolutional networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 1, pp. 119–131, Jan. 2020.

[16] H. Sun, S. Kawano, D. Nikovski, T. Takano, and K. Mori, "Distribution fault location using graph neural network with both node and link attributes," in *Proc. IEEE PES Innov. Smart Grid Technol. Europe*, 2021, pp. 1–6.

[17] J. T. de Freitas and F. G. F. Coelho, "Fault localization method for power distribution systems based on gated graph neural networks," *Elect. Eng.*, vol. 103, no. 5, pp. 2259–2266, 2021.

[18] W. Li and D. Deka, "PPGN: Physics-preserved graph networks for real-time fault location in distribution systems with limited observation and labels," 2021, *arXiv:2107.02275*.
[19] H. Mo, Y. Peng, W. Wei, W. Xi, and T. Cai, "SR-GNN based fault classification and location in power distribution network," *Energies*, vol. 16, no. 1, p. 433, 2022.
[20] J. Hu et al., "Fault location and classification for distribution systems based on deep graph learning methods," *J. Modern Power Syst. Clean Energy*, vol. 11, no. 1, pp. 35–51, 2022.
[21] B. L. Nguyen, T. V. Vu, T.-T. Nguyen, M. Panwar, and R. Hovsapian, "Spatial-temporal recurrent graph neural networks for fault diagnostics in power distribution systems," *IEEE Access*, vol. 11, pp. 46039–46050, 2023.
[22] R. Liang, G. Fu, X. Zhu, and X. Xue, "Fault location based on single terminal travelling wave analysis in radial distribution network," *Int. J. Elect. Power Energy Syst.*, vol. 66, pp. 160–165, 2015.
[23] S. Shi, A. Lei, X. He, S. Mirsaeidi, and X. Dong, "Travelling waves-based fault location scheme for feeders in power distribution network," *J. Eng.*, vol. 2018, no. 15, pp. 1326–1329, 2018.
[24] Y. Wang, T. Zheng, C. Yang, and L. Yu, "Traveling-wave based fault location for phase-to-ground fault in non-effectively earthed distribution networks," *Energies*, vol. 13, no. 19, p. 5028, 2020.
[25] A. Tashakkori, P. J. Wolfs, S. Islam, and A. Abu-Siada, "Fault location on radial distribution networks via distributed synchronized traveling wave detectors," *IEEE Trans. Power Del.*, vol. 35, no. 3, pp. 1553–1562, Jun. 2020.
[26] R. Krishnathevar and E. E. Ngu, "Generalized impedance-based fault location for distribution systems," *IEEE Trans. Power Del.*, vol. 27, no. 1, pp. 449–451, Jan. 2012.
[27] S. Das, N. Karnik, and S. Santoso, "Distribution fault-locating algorithms using current only," *IEEE Trans. Power Del.*, vol. 27, no. 3, pp. 1144–1153, Jul. 2012.
[28] R. Dashti and J. Sadeh, "Accuracy improvement of impedance-based fault location method for power distribution network using distributed-parameter line model," *Int. Trans. Elect. Energy Syst.*, vol. 24, no. 3, pp. 318–334, 2014.
[29] K. Jia, T. Bi, Z. Ren, D. W. Thomas, and M. Sumner, "High frequency impedance based fault location in distribution system with DGs," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 807–816, Mar. 2018.
[30] S. Gautam and S. M. Brahma, "Detection of high impedance fault in power distribution systems using mathematical morphology," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1226–1234, May 2013.
[31] K. Sekar and N. K. Mohanty, "Combined mathematical morphology and data mining based high impedance fault detection," *Energy Procedia*, vol. 117, pp. 417–423, 2017.
[32] T. Gush et al., "Fault detection and location in a microgrid using mathematical morphology and recursive least square methods," *Int. J. Elect. Power Energy Syst.*, vol. 102, pp. 324–331, 2018.
[33] N. Bayati, H. R. Baghaee, A. Hajizadeh, M. Soltani, and Z. Lin, "Mathematical morphology-based local fault detection in DC microgrid clusters," *Electric Power Syst. Res.*, vol. 192, 2021, Art. no. 106981.
[34] F. Wilches-Bernal, M. Jiménez-Aparicio, and M. J. Reno, "An algorithm for fast fault location and classification based on mathematical morphology and machine learning," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf.*, 2022, pp. 1–5.
[35] S. Lotfifard, M. Kezunovic, and M. J. Mousavi, "Voltage sag data utilization for distribution fault location," *IEEE Trans. Power Del.*, vol. 26, no. 2, pp. 1239–1246, Apr. 2011.
[36] Y. Dong, C. Zheng, and M. Kezunovic, "Enhancing accuracy while reducing computation complexity for voltage-sag-based distribution fault location," *IEEE Trans. Power Del.*, vol. 28, no. 2, pp. 1202–1212, Apr. 2013.
[37] F. C. Trindade, W. Freitas, and J. C. Vieira, "Fault location in distribution systems based on smart feeder meters," *IEEE Trans. Power Del.*, vol. 29, no. 1, pp. 251–260, Feb. 2013.
[38] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," 2017, *arXiv:1705.07664*.
[39] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*.
[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[41] W. H. Kersting, "Radial distribution test feeders," *IEEE Trans. Power Syst.*, vol. 6, no. 3, pp. 975–985, Aug. 1991.
[42] R. C. Dugan and T. E. McDermott, "An open source platform for collaborating on smart grid research," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, 2011, pp. 1–7.
[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[44] C. Cai and Y. Wang, "A note on over-smoothing for graph neural networks," 2020, *arXiv:2006.13318*.
[45] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[47] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[48] B. Palmintier and B.-M. Hodge, "SMART-DS: Synthetic models for advanced, realistic testing: Distribution systems and scenarios," *Nat. Renewable Energy Lab.*, Golden, CO USA, Tech. Rep., 2020.
[49] M. Stifter, J. Cordova, J. Kazmi, and R. Arghandeh, "Real-time simulation and hardware-in-the-loop testbed for distribution synchrophasor applications," *Energies*, vol. 11, no. 4, p. 876, 2018.
[50] A. L. Liao, E. M. Stewart, and E. C. Kara, "Micro-synchrophasor data for diagnosis of transmission and distribution level events," in *2016 IEEE/PES Transmiss. Distrib. Conf. Expo.*, 2016, pp. 1–5.
[51] M. Pignati, L. Zanni, P. Romano, R. Cherkaoui, and M. Paolone, "Fault detection and faulted line identification in active distribution networks using synchrophasors-based real-time state estimation," *IEEE Trans. Power Del.*, vol. 32, no. 1, pp. 381–392, Feb. 2017.
[52] F. Malandra, R. Pourramezan, H. Karimi, and B. Sansò, "Impact of PMU and smart meter applications on the performance of LTE-based smart city communications," in *2018 IEEE 29th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2018, pp. 1–6.

Dibaloke Chanda (Student Member, IEEE) received the B.Sc. degree in electrical, electronic and communication engineering from the Military Institute of Science and Technology, Dhaka, Bangladesh, in 2021. He is currently working toward the Ph.D. degree in computer science with Marquette University, Milwaukee, WI, USA. He was supervised by Dr. Nasim Yahya Soltani for his Ph.D. He is a Member of the machine learning, optimization and data (MOD) laboratory. His research interests include graph learning, interpretability, vision+x models, and representation learning.

Nasim Yahya Soltani (Member, IEEE) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2003, the M.Sc. degree in electrical engineering from the Iran University of Science and Technology, Tehran, in 2006, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2014. From 2014 to 2017, she was a Research Associate with the Digital Technology Center, University of Minnesota,. From 2018 to 2019, she was a Senior Data Scientist with Harley-Davidson Motor Company, Milwaukee, WI, USA. Since 2019, she has been a Northwestern Mutual Assistant Professor with the Department of Computer Science, Marquette University, Milwaukee. Her research interests include statistical signal processing, machine learning, optimization theory and network science with applications to wireless communications and networking, health care, and smart grid.