Master's Thesis

Improving the Explainability of Graph Neural Networks for Power Grid Topology Error Identification

---

Cora Hartmann

Examiner: Prof. Dr. Hannah Bast
Advisers: Sebastian Walter,
Dr. Bodo Rückauer

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair for Algorithms and Data Structures

January 23, 2025

Declaration

I hereby declare, that I am the sole author and composer of my Thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Freiburg i. Br., 23.01.2025

Place, Date

Signature

Abstract

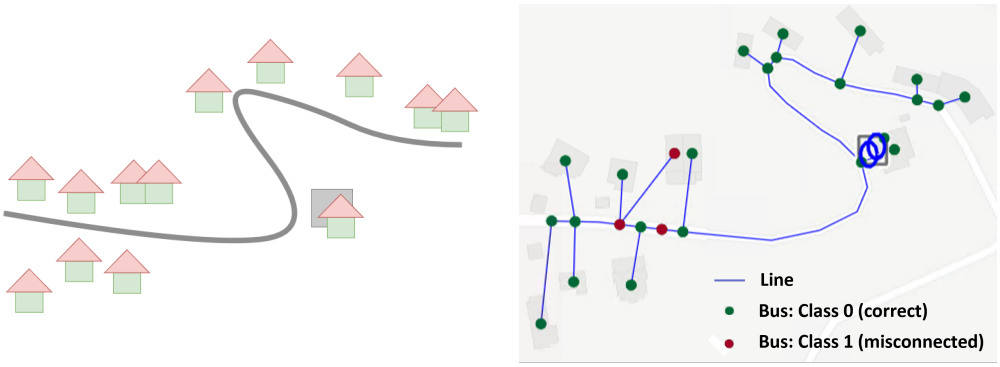
Accurate power grid models are essential for a successful energy transition. These digital twins of the physical grid enable advanced simulation and analysis for optimized grid planning and operation. Currently, maintaining an accurate model of the distribution grid involves time-consuming and error-prone manual work due to frequent changes in the underlying topology. To address this challenge, the Fraunhofer Institute for Solar Energy Systems ISE developed a Graph Neural Network (GNN) to detect errors in distribution grid models. However, the difficulty to retrace how an AI model arrived at a conclusion restricts its application in the critical power industry. In this Thesis, we investigate and improve the explainability and trustworthiness of this GNN using three complementary approaches: confidence calibration, model visualization, and explainable AI. Confidence calibration ensures that the model's prediction scores represent the probability of the prediction's correctness. Model visualization provides insights into the internal representations of the models. Explainable AI methods offer interpretable explanations for the model's predictions. We show that the GNN underestimates the probability of its correct predictions. We improve the calibration with histogram binning and temperature scaling. Our visualizations of the GNN with dimension reduction methods reveal clusters determined by topographical proximity and erroneous nodes. We categorize and analyze explanation subgraphs, identifying patterns in explanations for incorrectly classified nodes. We also quantitatively analyze the explainability of correctly and incorrectly classified nodes. Additionally, we enhance the GNNs loss function with an explainability term which improves the explainability while maintaining model performance. These methods make GNNs for grid topology error identification more accessible to the energy systems industry, addressing the needs of modern smart grids.

Contents

List of Figures

1. Introduction

Electrical power grids are part of the critical infrastructure. They are indispensable for modern society, daily life, and economic activities [1]. Therefore, they must function under various conditions and failures. The energy transition introduces new challenges for grid operators. The trends of national carbon neutrality, and clean and distributed energy are developing rapidly. Decentralized energy sources like wind turbines and solar panels, and new types of electric loads such as electric vehicle charging and heat pumps are on the rise. This electrification of the transport and heat sectors has a massive impact on distribution grids [2]. Low-voltage distribution grids represent the last phase of power delivery [3], transmitting electricity from the transmission system to end-users, such as households. The users' experience is dependent on the reliability, security, and quality of the power supply [2]. For instance, users will immediately experience a power outage if the distribution grid is not maintained on time. Intelligent design and digitization of distribution grids [2] is necessary for ensuring continued reliability and quality of the electricity supply even with the new challenges of the energy transition. As a result, the digitization and smartification of distribution grids has become urgently needed for the development of power systems. So-called digital twins [4] are a virtual representation or replica of the physical system. Digital twins can for instance be used for simulation, planning, prediction, monitoring, and operation tasks. However, frequent changes of the topology of the distribution grid, incomplete knowledge of new components, and potentially outdated maps of the physical topology pose a major challenge [5]. Many grid operators have not fully digitized their grids yet. Grid updates often rely on manual input, which are not always implemented promptly. These factors make it difficult to obtain an accurate digital twin, a correct model of the grid topology. As a result, the grid topology models may be incomplete and contain errors like incorrectly connected lines. In Fig. 1, such an example can be seen. Fig. 1a illustrates a small exemplary residential area. Fig. 1b shows a possible corresponding erroneous topology of the distribution grid.



(a) Small exemplary residential area with a street and 13 buildings.

(b) Erroneous topology of the distribution grid of the exemplary residential area with a misconnected line.

Figure 1: Exemplary residential area and its erroneous grid topology.

Graph Neural Networks (GNNs) [6] can aid in identifying errors such as wrongly connected lines, switch states, and transformer and line properties. At Fraunhofer ISE, a Topology Error Identification Graph Neural Network (TEIGR) for detecting topology errors in distribution grids was developed [7, 8]. A bus, i.e., a node, is classified as either "error" or "no error", allowing grid operators to correct their models. GNNs are widely used and are powerful machine learning models [9], but their non-linear representations often make them difficult to understand. Depending on the level of trust in TEIGR, a grid operator has two options regarding a node classified as "error". The grid operator could directly make changes in the grid topology according to TEIGRs output at the risk of acting on a false positive. On the other hand, if the grid operator has less confidence, they could verify the potential error through an in-field inspection. However, this verification would be costly and time-consuming. It is therefore advantageous if the grid operators can gauge the certainty of the output of TEIGR, and, if desired, retrace how TEIGR arrived at its conclusion. The limited explainability of deep learning models such as GNNs often restricts their adoption in critical industries like the power systems sector [1]. Grid operators could benefit significantly from insights into black-box [9] deep learning models.

In this Thesis, we explore and enhance the explainability and trustworthiness of TEIGR through three complementary approaches: confidence calibration [10, 11, 12], model visualization [13, 14, 15], and explainable AI [16, 17, 18]. In the following, we introduce these approaches and present our research questions.

Confidence Calibration

Usually, the output of a machine learning model for a binary classification task is a prediction score. However, the score is not necessarily a probability which reflects the prediction's correctness. For instance, if the model outputs a score of 0.8, one cannot conclude that 80 out of 100 samples with that score have the predicted class. The concept of confidence calibration refers to the idea that the model's output should represent the probability of the prediction's correctness [19].

Calibrated confidence scores can help improving the trustworthiness [10] of TEIGR, because they provide the user an indication how certain TEIGR is with its prediction. Especially for safety-critical use cases this is of significant importance. Changing the grid topology model can have far-reaching consequences, which is why the grid operator must be sure to do so. A prediction with a low confidence score signals uncertainty. A low confidence score prompts the grid operator to treat the prediction with caution and, for example, verify a potential error through an in-field inspection. Well calibrated confidence scores allow grid operators to prioritize further investigation where TEIGR lacks certainty. Probability is valuable information that goes beyond mere prediction.

Regarding confidence calibration, we are interested in the following research questions:

R1.1 How well is TEIGR calibrated?

R1.2 Do the uncalibrated TEIGR show a bias towards over or under-confidence?

R1.3 Can the confidence calibration be improved with calibration methods?

Model Visualization

A well-calibrated model is the basis, on which we can build the other explainability methods considered in this work. An approach of making TEIGR more interpretable is the visualization and analysis of the internal representations of TEIGR. GNNs process information in high-dimensional representations, making them difficult to analyze [20]. To increase the understanding of GNNs, the high-dimensional representations can be visualized in 2D using dimension reduction methods [13, 14, 15]. We explore and compare the model visualizations of three dimension reduction methods. In this context, we are interested in the following research questions:

R2.1 Do the visualizations reveal clusters influenced by topology errors, feature values, or topographical proximity?

R2.2 How are the visualizations influenced by the kind and number of model features considered?

R2.3 Do the clusters become more differentiated through model training?

Explainable AI

The previous approach referred to TEIGR as a whole and can be particularly helpful for developers. Another approach aims to make individual predictions explainable for users such as grid operators. Explainable AI (XAI) can help reveal the inner workings of complex models like GNNs, providing greater insight into their predictions. An explanation for a GNN is typically given by an explanation subgraph [18], which highlights the nodes and edges that have the greatest impact on the prediction. An explanation of a prediction helps to understand how a GNN came to its decision and which factors influenced it. Therefore, XAI can aid to increase the trustworthiness and interpretability of TEIGR. Regarding XAI, we investigate the following research questions:

R3.1 Can a qualitative analysis help divide explanations into categories for false positives and false negatives?

R3.2 How does the explainability of correctly and wrongly classified nodes differ quantitatively?

R3.3 Can TEIGRs' prediction performance and explainability be improved by adding an explainability metric to the loss function?

Main Contributions

Having motivated and clarified the problem we deal with and the methods to overcome it, we will now summarize our main contributions.

- We find that TEIGR is well calibrated and only slightly under-confident.
- We improve the calibration with histogram binning and temperature scaling and discuss in which cases which method is more beneficial for grid operators.
- We visualize the internal representations of TEIGR with dimension reduction methods. The visualizations reveal clusters based on topographical proximity and the node labels.

- We show that model training improves the representation.
- We implement a visualization for explanations of several explainer algorithms. We qualitatively analyze and categorize the explanations for incorrectly classified nodes.
- We quantitatively analyze the explainability of correctly classified and misclassified nodes. We demonstrate that explainer algorithms with a lower total characterization score often have high characterization scores for misclassified nodes.
- We enhance TEIGRs' loss function with an explainability term which improves the explainability while maintaining model performance.

Overall, we enhance the trustworthiness and explainability of TEIGR through various methods. To the best of our knowledge, we are the first who employ confidence calibration, model visualization and explainable AI for GNNs in the field of power grid topology error identification.

Outline

The rest of this Thesis is organized as follows: In Chapter 2, we summarize related work. In Chapter 3, we describe the methods we use. In Chapter 4, we provide an empirical analysis of the previously presented methods for improving the explainability and trustworthiness of TEIGR. In Chapter 5, we point to future research activities. Finally, in Chapter 6, we conclude our findings.

2. Related Work

Several works have addressed various aspects of our topic. Here, we summarize the most relevant studies about topology error identification, confidence calibration, model visualization and explainable AI for GNNs. We outline how they differ from our approach.

2.1. Topology Error Identification with Graph Neural Networks (GNNs)

Earlier work at Fraunhofer ISE [7] dealt with GNNs to identify and correct misconnected lines in distribution grid topologies through voltage and power measurements. The authors generated topology errors to create labeled data for supervised learning. Their proposed Topology Error Identification Graph Neural Network (TEIGR) identifies and corrects errors in one inference step. They conducted a hyperparameter study to maximize TEIGRs performance. They found that attention-based convolutions are key to capturing patterns and overcoming limitations of existing methods.

In a follow-up project at Fraunhofer ISE [8], TEIGR was evaluated in more realistic scenarios and examined for improvements. The authors assessed TEIGR in a realistic setting, optimizing loss functions, learning rates, and input features. They found that the loss function Binary Cross Entropy (BCE), and a reduced feature set deliver satisfactory results. Further experiments showed that subtle errors and limited measurement data reduce performance.

None of these works dealt with the explainability of TEIGR. Our work is based on TEIGR and improves TEIGR by increasing its interpretability and trustworthiness. By improving TEIGRs explainability, we seek to enhance its accessibility for applications in the critical field of power grids.

2.2. Confidence Calibration

Recently, confidence calibration has gained attention in deep learning [10, 11, 12], highlighting that many neural network models are miscalibrated. The models tend to be over-confident, meaning their accuracy is lower than their confidence scores. Several calibration methods [21, 22, 23, 24, 25, 26] were proposed to correct the undesirable difference.

In the field of GNNs, confidence calibration has only recently been studied. Teixeira et al. [27] conducted an initial evaluation of confidence calibration for GNNs by experimenting with various node classification datasets. They found that GNNs are often miscalibrated and that current calibration methods do not always effectively improve calibration to the desired level.

Wang et al. [28] noted that GNNs are typically under-confident, unlike classical neural network models, which are typically over-confident [10]. Based on these observations, they proposed adding a Graph Convolutional Network (GCN) on top of the backbone GNN to enhance calibration.

Liu et al. [19] studied the impact of factors like model capacity and graph density on confidence calibration. The authors introduced a topology-aware calibration method that considers neighboring nodes, demonstrating improved calibration performance over baseline methods.

None of these works have dealt with GNN models for electric power grids or even topology error identification. But especially for safety-critical applications calibrated confidence scores are important. Calibrated confidence scores can enhance the model's trustworthiness by indicating the certainty of its predictions. In grid topology error identification, grid operators could benefit from confidence calibration. To the best of our knowledge, this is the first time the confidence scores of a GNN model for topology error identification are analyzed and calibrated.

2.3. Visualizing GNNs

Visualizing GNN models can be seen as a problem of visualizing high-dimensional data which appears in many different domains. In recent decades, various techniques have been developed for visualizing high-dimensional data, studied in a survey from Oliveira et al. [29]. Most of these techniques merely offer tools for displaying more than two data dimensions, leaving the interpretation of the data to the human observer. This significantly limits their usefulness for real-world datasets that contain thousands of high-dimensional points. In contrast to the aforementioned approach,

dimension reduction methods transform the high-dimensional dataset into a 2- or 3-dimensional dataset [14] which can be visualized in a scatterplot.

The goal of dimension reduction is to retain as much of the essential structure of the high-dimensional data as possible in the low-dimensional representation. Several techniques have been proposed to address this problem, each focusing on preserving diverse types of structure. Traditional dimension reduction methods, such as Principal Component Analysis (PCA; Hotelling [13]) and classical multidimensional scaling (MDS; Torgerson [30]), are linear techniques that prioritize keeping dissimilar datapoints distant in the low-dimensional space.

Many non-linear dimension reduction methods [31, 32, 33, 34] have been proposed with the goal of preserving the local structure of the data. However, van der Maaten et al. [14] stated that they are strong for artificial data sets, but often not for real data. Specifically, most of these methods fail to preserve both the local and global structure of the data. Therefore, van der Maaten et al. [14] proposed a current state-of-the-art dimension reduction method for visualization, the t-distributed Stochastic Neighbor Embedding technique (t-SNE). The authors showed that t-SNE is effective at capturing much of the local structure of high-dimensional data, while also revealing global structures, such as the presence of clusters at various scales.

McInnes et al. [15] proposed another non-linear dimension reduction method, the Uniform Manifold Approximation and Projection method (UMAP). UMAP performed competitively with t-SNE in terms of visualization quality. The authors stated that UMAP better preserves the global structure, all while delivering superior runtime performance.

Our work differs from the presented work in that we do not develop a new dimension reduction method, but apply the presented methods, PCA, t-SNE and UMAP, to visualize TEIGR. Our goal is to use the dimension reduction methods to visualize and analyze the internal representations of TEIGR. We explore and compare the model visualizations of PCA, t-SNE, and UMAP with respect to our use case.

2.4. Explainable AI

GNNs have found broad application in many scientific and technical fields [18]. Compared to other graph processing approaches [35, 36], GNNs have the disadvantage that their predictive mechanism is opaque [18]. This is the case for most architectures based on deep learning. In recent years, many explainable AI (XAI) techniques for deep architectures were developed [16, 37, 38, 17, 39]. These techniques were

adapted to GNNs too [40, 41, 42]. Ying et al. [43] proposed with the GNNExplainer a new XAI method specific for GNNs. They introduced the GNNExplainer as a model-agnostic method to generate interpretable explanations for GNN predictions on any graph-based task. GNNExplainer identified important subgraph structures and node features that influence predictions. The authors showed that GNNExplainer outperforms existing methods, offering valuable insights into model behavior and enhancing interpretability.

However, the use of XAI in the power system sector is not yet widespread. Machlev et al. [44] explored the potential of XAI for applications in energy and power systems. They discussed the challenges and limitations of implementing XAI techniques in this field, where transparency and accountability are crucial. The paper provided a review of recent studies and ongoing trends in XAI for energy systems. The authors emphasized the lack of consideration for explainability in current machine learning applications. Although the research is advancing, to the best of our knowledge, there is no comparable work for our application of XAI for a grid topology error identification GNN yet.

3. Methods

In this chapter, we present the methodologies used in this thesis. Section 3.1 focuses on the data including the modeling of distribution grids as graphs and the definition of the features. We describe the necessity and realization of synthetic grids as well as the error generator. Section 3.2 introduces the Topology Error Identification Graph Neural Network. We define Graph Neural Networks and describe the training process. Section 3.3 introduces calibration methods, including histogram binning and temperature scaling, to address model calibration. Section 3.4 explores the dimensionality reduction techniques PCA, t-SNE, and UMAP, used for model visualization and analysis. Finally, Section 3.5 delves into explainable AI with a focus on GNN Explainability. We present common evaluation metrics as well as explainer algorithms for GNNs.

3.1. Data

In this section, we will introduce the data we used. This includes the structure, features, and preprocessing of the data.

3.1.1. Power Grids Modeled as Graphs

Graphs are a versatile data structure. At its core, a graph consists of a set of objects, known as nodes, and a set of interactions, referred to as edges, which connect pairs of these nodes. Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ between these nodes [6]. An edge from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$ is denoted as $(u, v) \in \mathcal{E}$.

We modeled a distribution grid as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes represent the physical components, the buses, of the grid. An edge connects the corresponding nodes whenever there is a direct physical connection between two buses, a line. In the following, we will use the term “node”, as this term is more general than “bus”. Only when we are talking explicitly about power grids, we will use the term “bus” as a synonym for “node”, as “bus” is more common in this context.

3.1.2. Node Features

The nodes are characterized by features such as power (P), voltage magnitude (V_{mag}), and voltage angle (V_{ang}). In Table 1 all features with a short description can be found. In the following, they will be explained in more detail.

Table 1: Description of GNN features used for topology error identification.

Feature	Unit	Description
P	W (Watt)	Measured active power
Q	Var (Volt-ampere reactive)	Measured reactive power
V_{mag}	V (Volt)	Measured voltage magnitude
V_{ang}	° (Degrees)	Measured voltage angle
$\hat{V}_{\text{mag}}$	V (Volt)	Expected voltage magnitude
$\hat{V}_{\text{ang}}$	° (Degrees)	Expected voltage angle
$V_{\text{mag\_diff}}$	V (Volt)	$= \hat{V}_{\text{mag}} - V_{\text{mag}}$: Voltage magnitude difference
$V_{\text{ang\_diff}}$	° (Degrees)	$= \hat{V}_{\text{ang}} - V_{\text{ang}}$: Voltage angle difference

Power

Loads consume power. Generators, the counterparts to loads, produce power. Apparent power in alternating current (AC) grids consists of two components: active and reactive power. Active power is the power consumed, while reactive power oscillates between the source and the load [8].

Voltage Magnitude

In AC systems, voltage magnitude refers to the absolute value of the AC voltage. Voltage magnitude is conceptually similar to voltage in direct current (DC) systems. In both cases, it represents the potential difference between two points in a circuit and indicates the strength of the electrical force driving the current [8]. x We assume a low-voltage grid operating at 400 volts. We adopt the per unit (p.u.) convention meaning that all voltages in a grid are normalized to the reference voltage at the transformer node. In an ideal scenario without impedance (the AC equivalent of resistance) in the lines, the voltage magnitude would be consistent across all buses. However, due to the presence of impedance, the voltage tends to drop with increasing distance from the transformer or generator nodes.

Voltage Angle

In AC systems, voltage varies over time and follows a sine wave pattern. The voltage angle, or phase angle, represents the phase shift of this sine wave relative to a reference point, typically the slack bus. The slack bus, a virtual node representing the transmission grid [8], balances the energy in the distribution grid by supplying needed power or absorbing excess energy. In contrast to power, voltage magnitude and voltage angle depend on the position of the node in the grid topology.

3.1.3. Power Flow

Power flow calculation computes the voltage magnitude and angle at each node using loads and line impedances [45]. It solves a non-linear system of equations representing Kirchhoff's laws [45] with a Newton-type gradient descent. This tool is important where voltage data is limited. We use power consumption data from smart meters or simulations to create the datasets.

Difference between the Measured and Expected Voltage Magnitude and Angle

Let T represent the real grid topology, which accurately reflects the actual physical layout of the power grid. On the other hand, $T*$ represents the given grid topology [8] which could contain errors due to e.g., incomplete knowledge of new components and faulty manual updates. The goal of the GNN is to identify errors in $T*$. We assume we are provided with power and voltage measurements of the real grid topology T, as these are measured on the real grid and are not influenced by a possibly incorrect topology $T*$. Because the power is not influenced by the grid topology but determined by the loads at each node, it is the same for both $T*$ and T. In contrast, voltage magnitude and voltage angle are different for $T*$ and T. Therefore, we can compute the power flow based on the given power values and $T*$. This provides us the expected voltage values for each node in $T*$. The difference between the measured and expected voltage has proven to be a valuable feature in verifying a grid's topology.

3.1.4. Synthetic Grids

Real grid data is not always readily available, e.g., due to the fact that some of the information contained is confidential. Therefore, synthetic grids are widely used in energy system research [46, 47].

Our data includes real grid topologies from grid operators as well as synthetic data. The synthetic data was generated by using real-world data as a base. OpenStreetMap

data was used to create power grid topologies for various cities, with distribution lines following city streets [7]. While the synthetic grids may not perfectly match the actual grids in those cities, they reflect key features and characteristics of real-world grids. We ensure similar graph properties, such as the number of nodes and average node degree, between real-world and synthetic grid topologies. This statistical similarity ensures that the synthetic grids imitate the complexity of real power grids. The GNN models were trained on four grids (three synthetic and one real) and validated and tested on two grids each (one synthetic and one real) [48]. The properties of these eight grid topologies can be found in Appendix A.1.

A tool at Fraunhofer ISE, called synPro [49], allows for time series forecasting based on realistic assumptions about consumption and generation. They developed three grid scenarios for the years 2022, 2030, and 2040. Each scenario includes snapshots of voltage and power values taken throughout the year, with a 15-minute resolution [48].

3.1.5. Error Generator

The original dataset comprises realistic grid topologies free of topology errors. Grid topology errors were simulated using an error generator, which was developed as part of previous work [7, 8]. This approach allowed for training with labeled data, using the original dataset as the ground truth. The error generator reroutes a single line in the grid to create a slight error while maintaining a fully connected topology. Multiple faulty variations can be generated from the same source grid. To ensure realistic, subtle errors, the generator avoids placing lines between buildings and prefers shorter lines over longer ones when adding new connections [8].

3.2. Topology Error Identification Graph Neural Network (TEIGR)

This section introduces the Topology Error Identification Graph Neural Network (TEIGR) used for the confidence calibration, the visualizations and the explainability experiments in our empirical analysis. TEIGR was developed in previous work [7, 8] at Fraunhofer ISE. We present Graph Neural Networks in general and describe the training of TEIGR. Since the focus of this work is not on TEIGR itself, the discussion of its architecture and training is brief. For more information, refer to previous work [7, 8].

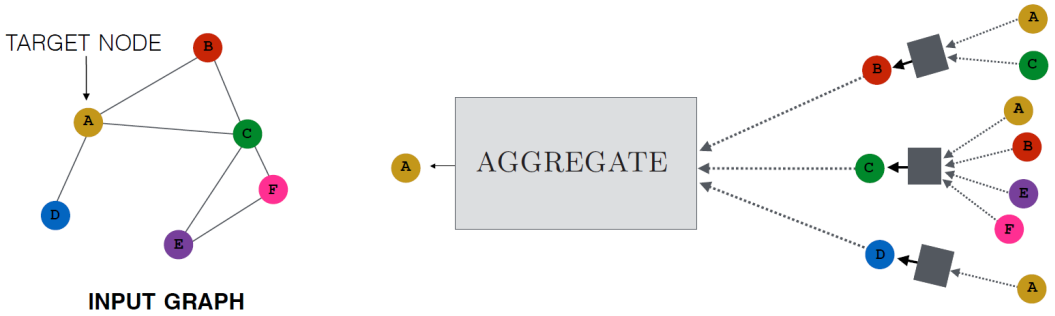


Figure 2: Message passing: Illustration of how a single node aggregates messages from its local neighborhood. Node A collects messages from its immediate neighbors (B, C, and D). These messages, in turn, are derived from information aggregated by each of their respective neighbors, continuing in a recursive manner. The visualization represents a two-layer message-passing model. Figure from Hamilton [6].

3.2.1. Graph Neural Networks

Graph Neural Networks (GNNs) are neural networks that operate on graph data. In this work, we adopt the widely used message-passing framework to formalize GNNs, following the notation from Hamilton [6]. The idea of this message-passing framework is illustrated in Fig. 2 with a single node that aggregates messages from its local neighborhood.

Given a graph $\mathcal{G}$ and a set of d node features $\{x_u \mid u \in \mathcal{V}\}, x_u \in \mathbb{R}^d$, the goal is to compute node embeddings $z_u, \forall u \in \mathcal{V}$. In each message-passing iteration k of a GNN, a hidden embedding $h_u^{(k)}$ for each node $u \in \mathcal{V}$ is updated based on the information aggregated from the graph neighborhood $\mathcal{N}(u)$ of node u. A message-passing iteration can be viewed as analogous to a layer in other neural network architectures. The message-passing update is formally defined as follows:

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right)\right) \tag{1}$$

$$= \text{UPDATE}^{(k)}\left(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}\right). \tag{2}$$

UPDATE and AGGREGATE are arbitrary differentiable functions (e.g., neural networks). $m_{\mathcal{N}(u)}$ represents the "message" aggregated from node u's neighborhood $\mathcal{N}(u)$. Superscripts are used to distinguish the embeddings and functions at different iterations of the message-passing process.

At each message passing iteration k, the AGGREGATE function takes the set of

embeddings of the nodes in u's graph neighborhood $\mathcal{N}(u)$ as input. It produces a message $m_{\mathcal{N}(u)}^{(k)}$ by aggregating this neighborhood information. The portion of the graph $\mathcal{G}$ used in aggregation is called the computation graph $\mathcal{G}_C$ with respect to a specific node u. The computation graph includes a subset of the node set $\mathcal{V}$. The UPDATE function then combines this message with the previous embedding $h_u^{(k-1)}$ of node u to compute the updated embedding $h_u^{(k)}$. Usually the initial embeddings at $k = 0$ are set to the input features x_u:

$$h_u^{(0)} = x_u, \forall u \in \mathcal{V}. \tag{3}$$

After performing K iterations of message passing, the output of the final layer can be used to define the embeddings for each node:

$$z_u = h_u^{(K)}, \quad \forall u \in \mathcal{V}. \tag{4}$$

GNNs have strong inductive and transfer learning capabilities. Inductive learning refers to the ability to generalize knowledge from seen data to unseen data, even if the latter has a different structure. This allows power grid operators to train GNN models on various grid topologies and effectively apply them to new, unseen ones [50].

3.2.2. Model Training

The Topology Error Identification Graph Neural Network (TEIGR) is implemented with the deep learning library PyTorch [51]. We trained TEIGR to classify nodes into two categories: nodes with errors and nodes without errors. This is a supervised learning task, as TEIGR was trained on labeled data. The training process comprises 1.000 epochs on an NVIDIA Quadro RTX 8000 GPU with CUDA 11.0. The model that achieved the best performance on the validation set was selected as the final model. The model and training hyperparameters were optimized through a hyperparameter study, as detailed in previous work [7]. The best-performing hyperparameter values, which we used for TEIGR, including our used optimizer, activation function, and learning rate, are provided in the Appendix A.3.

Loss Function

The loss function is of particular importance for our work, as we are improving the loss function in the experimental study. That is why we briefly introduce the loss

function here.

A loss function is used to guide the optimization process of a ML model, helping to adjust the model's parameters to improve model performance. TEIGR is optimized with the Binary Cross-Entropy (BCE) loss. BCE is a widely used loss function for binary classification tasks. BCE quantifies the difference between predicted probabilities and actual class labels, penalizing incorrect predictions. The BCE loss is calculated as the negative log-likelihood of the true labels given the predicted probabilities. The gradient of the loss is then backpropagated through the parameters of the GNN [6].

F1-Score

To evaluate the performance of TEIGR, we use the F1-score, which is computed as the harmonic mean of precision and recall. Precision P is defined as the ratio of true positive (TP) predictions to the total number of positive predictions consisting of true positives and false positives (FP):

$$P = \frac{TP}{TP + FP} \tag{5}$$

Recall R is defined as the ratio of true positive predictions to the total number of actual positive nodes consisting of the sum of true positives and false negatives (FN):

$$P = \frac{TP}{TP + FN} \tag{6}$$

The F1-score is then defined as:

$$F1 = 2\frac{P \cdot R}{P + R} \tag{7}$$

In previous work [8], TEIGR achieved a F1-score of 0.66 on the validation dataset.

3.3. Confidence Calibration

In the following section, we introduce the concept of confidence scores and calibration. We present two common calibration methods, histogram binning and temperature scaling.

3.3.1. Definition Calibration

Let for each node $u \in \mathcal{V}$ be $y_u \in \kappa := \{1, 2, ..., K\}$ the ground-truth label, where κ indicates the possible classes and K stands for the total number of classes. Let $h_u \in \mathbb{R}^d$ be the node embedding. The function $g : \mathbb{R}^d \to [0, 1]^K$ takes as input h_u and outputs the probability vector $g(h_u)$. Thereby, $g(h_u)_l$ represents the l-th element. Then for node u the predicted class is given by $\hat{y}_u = \arg\max_{l \in \kappa} g(h_u)_l$. The corresponding confidence score is $\hat{p}_u = \max_{l \in \kappa} g(h_u)_l$. We define perfect calibration as [19]:

$$\mathbb{P}(\hat{y}_u = y_u | \hat{p}_u = p) = p, \forall p \in [0, 1]. \tag{8}$$

According to this definition, a GNN is considered perfectly calibrated if the confidence $\hat{p}_u$ matches the true probability of obtaining a correct prediction for each node.

Reliability Curve

For visualizing the calibration of the model, model accuracy can be plotted against confidence which is called reliability curve [52] (see Fig. 5 (left)). Therefore, the nodes are divided according to their predicted confidences $\hat{p}_u$ into $M \in \mathbb{N}$ equally-spaced interval bins. An accuracy and average confidence are computed for each bin B_m, where $m \in \{1, 2, ..., M\}$ and $|B_m|$ denotes the number of samples in bin B_m:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{u \in B_m} \mathbb{1}[\hat{y}_u = y_u] \tag{9}$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{u \in B_m} \hat{p}_u \tag{10}$$

If $\text{acc}(B_m) = \text{conf}(B_m), \forall m$ the model is perfectly calibrated.

The reliability curve provides a method to determine whether a model is over-confident or under-confident. If the curve appears above the diagonal line, it indicates that accuracy exceeds the confidence, signifying the model is under-confident. Conversely, if the curve is below the diagonal, the model is considered over-confident. The diagonal represents the perfect calibration [19].

Expected Calibration Error (ECE)

The calibration error (CE) measures the gaps in the reliability diagram between the reliability curve and the diagonal and is given by

$$\mathrm{CE} = |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{11}$$

For quantifying the overall miscalibration, these gaps in the reliability diagram between the reliability curve and the diagonal can be averaged. This metric is called the expected calibration error (ECE) and is given by

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{12}$$

for a total number of N nodes. A lower value indicates better calibration, with 0 being perfectly calibrated [19].

3.3.2. Calibration Methods

Two widely used calibration methods are histogram binning and temperature scaling. They improve the reliability of models by correcting their predicting probabilities. Histogram binning and temperature scaling will be introduced in the following.

Histogram Binning

Histogram binning [53] is a non-parametric calibration method. It calibrates probabilities by dividing the predicted probabilities into bins (usually uniformly spaced) and adjusting the predicted probabilities within each bin to match the actual observed frequencies [10]. It is piecewise constant: Adjustments in probabilities are based on averages within each bin, resulting in a stepwise calibration function [19].

Temperature Scaling

Temperature scaling [10] is a parametric calibration method that adjusts the confidence of predictions by scaling logits with a single parameter (temperature). It scales the logits uniformly across all predictions, adjusting the overall confidence but preserving the relative ordering of the predicted probabilities. Given the output logit vector z, a rescaling of z by a temperature factor $T > 0$ is performed, resulting in z/T. The parameter T is learned on the validation set, minimizing the negative log likelihood [19].

3.4. Dimension Reduction Methods

Dimension reduction methods in the context of machine learning are applicable in several ways. They can either be used for visualization, compression or interpolation directly or for other algorithms as a preprocessing step, for example to reduce the classifiers input size [34]. We concentrate on the application of dimensionality reduction for model visualization.

In this section, we will introduce the linear dimension reduction method PCA as well as the non-linear methods t-SNE and UMAP.

3.4.1. Principal Component Analysis (PCA)

The standard Principal Component Analysis (PCA) [13] is a statistical technique that reduces the data to its essential features, the so-called principal components (PCs). The goal is to map high-dimensional data to a lower dimensional space for example to help visualize the data [54]. PCs are selected linear combinations of the original variables that capture the maximum variance across all variables retaining as much information as possible. The first PC captures the most information about the data. The second PC contains more information than the third, and so forth. By excluding less important PCs a dimensionality reduction is performed and the data is projected onto a space of lower dimensionality. The PCs possess geometric properties that facilitate an intuitive interpretation of the key features [55].

The PCs are identified via an eigenvalue decomposition of the covariance matrix $S = \frac{1}{N-1}X^T X$. In this process, we arrange the normalized high-dimensional observations $x_i \in \mathbb{R}^d$ into an $N \times d$ matrix X where each column has a mean of zero and a standard deviation of one. The confidence matrix S contains d orthonormal $u_1, ..., u_d$ eigenvectors associated with a set of sorted eigenvalues $\lambda_1 \geq ... \geq \lambda_d$. For the PCA analysis, we select the first k eigenvectors to create the projection matrix P. The reduced low-dimensional $N \times k$ data matrix Y is then computed using this projection matrix P by $Y = XP$ [20].

3.4.2. T-distributed Stochastic Neighbor Embedding Technique (t-SNE)

The t-distributed Stochastic Neighbor Embedding technique (t-SNE) [14] is a non-linear dimension reduction method for visualizing high-dimensional data. The core idea is to identify representations in a lower-dimensional space where points with a high probability of being close in the original high-dimensional space remain likely

to be close in the reduced space. The goal is to best preserve neighborhood identity. T-SNE builds upon the SNE approach [34]. Both methods apply a Gaussian kernel in the high-dimensional space. However, t-SNE uses a Student's t-distribution of first order for the low-dimensional kernel [20].

For each high-dimensional data point x_i, and each potential neighbor x_j, we compute the conditional probability $p_{j|i}$ of x_j being close to x_i:

$$p_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}\right)}. \tag{13}$$

σ_i represents the variance of the Gaussian centered at x_i. By assuming symmetry and for a total number of N data points, we obtain the probability $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$.

Mapping the high-dimensional data points into a lower d-dimensional space, we get the low-dimensional images $\mathbf{y}_i \in \mathbb{R}^d$. The induced low-dimensional probabilities q_{ij} that low-dimensional data point $\mathbf{y}_i$ picks point $\mathbf{y}_j$ as its neighbor are then given by:

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}}. \tag{14}$$

Typically, the dimensionality d of the $\mathbf{y}$-space is 2 or 3 [34, 20] to enable visualization.

The goal is to align these two probability distributions p_{ij} and q_{ij} as closely as possible. This is accomplished by minimizing the cost function, which is the sum of Kullback-Leibler divergences between the original p_{ij} and induced q_{ij} distribution over neighbors for each data point:

$$C = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right). \tag{15}$$

An important parameter is the user-defined perplexity which can be understood as a smooth measure of the effective number of neighbors. T-SNE determines the variance σ_i through a binary search, yielding the result P_i with a fixed perplexity value. The perplexity is defined as [14]:

$$\mathrm{Perp}(P_i) = 2^{H(P_i)}, \tag{16}$$

where the Shannon entropy $H(P_i)$ of P_i, measured in bits, is defined by:

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}. \tag{17}$$

3.4.3. Uniform Manifold Approximation and Projection (UMAP)

UMAP (Uniform Manifold Approximation and Projection) [15] is an advanced manifold learning method for dimension reduction. Compared to t-SNE it better retains global structure of the data while achieving faster performance. One key difference between t-SNE and UMAP lies in the initialization of the low-dimensional embedding. UMAP uses a spectral embedding of the graph Laplacian, while t-SNE typically starts with a random initialization. Additionally, UMAP employs stochastic gradient descent for optimizing the low-dimensional representation, whereas t-SNE relies on standard gradient descent methods [20].

Like t-SNE, UMAP constructs a high-dimensional distribution from the data and then seeks a matching low-dimensional distribution. UMAP uses Riemannian geometry and fuzzy sets to link local metrics with the data through a nearest-neighbor graph. The number of neighbors k is a hyperparameter similar to t-SNE's perplexity [20].

High-dimensional, unnormalized probabilities are calculated as edge weights in a fuzzy graph by

$$p_{j|i} = \exp\left(-\frac{\max\left(0, d\left(x_i, x_j\right) - \rho_i\right)}{\sigma_i}\right), \tag{18}$$

where $d\left(x_i, x_j\right)$ represents the distance between points x_i and x_j, and $1 \leq j \leq k$, $1 \leq i \leq N$. ρ_i is the distance from x_i to its nearest neighbor:

$$\rho_i = \min\{d(x_i, x_{ij}) \mid 1 \leq j \leq k,\ d(x_i, x_{ij}) > 0\}. \tag{19}$$

The parameter σ_i corresponds to a normalization factor and is found by binary search to satisfy the relation $\sum_{j=1}^{k} p_{j|i} = \log_2(k)$. The symmetric unnormalized probabilities p_{ij} are defined by $p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} * p_{j|i}$. The low-dimensional distance probabilities are given by

$$q_{ij} = \left(1 + a\left(y_i - y_j\right)^{2b}\right)^{-1}. \tag{20}$$

The parameters a and b are determined through least-squares fitting:

$$q_{ij} \approx \begin{cases} 1 & \text{if } y_i - y_j \leq \text{min\_dist}, \\ e^{-(y_i - y_j) - \text{min\_dist}} & \text{if } y_i - y_j > \text{min\_dist}. \end{cases} \tag{21}$$

The hyperparameter min_dist controls the minimum distance between data points in the low-dimensional embedding. Lastly, the loss function is defined using binary

cross-entropy [20]:

$$C = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) + (1 - p_{ij}) \log \left(\frac{1 - p_{ij}}{1 - q_{ij}} \right). \tag{22}$$

3.5. Explainable AI

In this section, we introduce explainable AI with a focus on GNN Explainability. We present common evaluation metrics for explanations. Lastly, we discuss popular explainer algorithms for GNNs.

Complex machine learning models are typically developed without interpretability, making them "black boxes" [9] with opaque decision-making processes. The term "black box" refers to the difficulty in explaining these models, which are often incomprehensible even to domain experts. This can reduce the trust in the models. Explainable Artificial Intelligence (XAI) is a set of methods that enable users to comprehend the outputs of models, aiming to make models more explainable. XAI seeks to improve the interpretability of machine learning models for developers, researchers, domain experts, and users, without sacrificing performance or accuracy. For this purpose, many explainer algorithms exist, which will be discussed below.

3.5.1. GNN Explainability

Explainability of GNNs is challenging due to the unique nature of graph data, which lacks a fixed, tabular structure. Nodes have varying numbers of neighbors. GNN predictions rely on the interactions between nodes, edges, and their features. Unlike text or image data, where individual words or pixels can be analyzed, GNNs require capturing structural information as well. The complexity of graphs, with diverse features, makes generating human-interpretable explanations difficult [9].

An explanation for a GNN model is usually an explanation subgraph of the computation graph [19]. This explanation subgraph highlights the nodes and edges that contribute most to the prediction. Each edge in the subgraph has an importance score [56]. The concept of an explanation subgraph is illustrated in Fig. 3, which presents a schematic example of a GNN used for classifying sports activities [43].

Formally, given a graph $\mathcal{G}$ with node set $\mathcal{V}$, edge set $\mathcal{E}$, and a trained GNN model f, the explanation task aims to identify the most influential compact subgraph $\mathcal{G}_S$ for the model's prediction. This is achieved by generating an edge mask $M \in \mathbb{R}^{|\mathcal{E}|}$ that represents edge importance scores. To make the explanations more interpretable

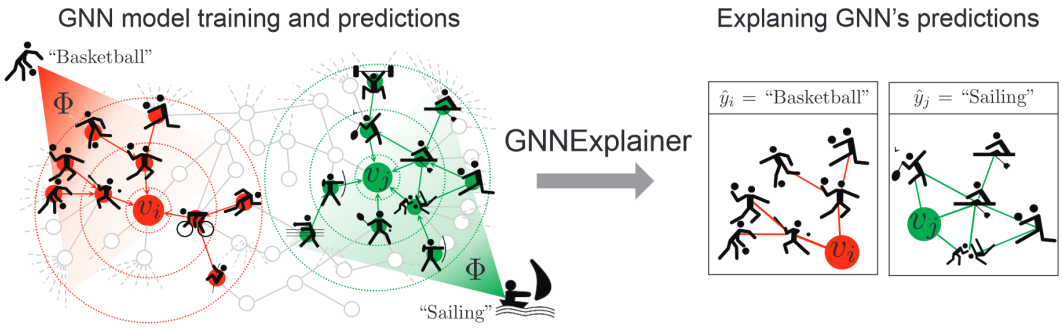


Figure 3: Schematic illustration of an explanation for an example GNN: The example shown is a hypothetical node classification task where a GNN is trained on a social interaction graph to predict future sports activities. A social interaction graph represents people as nodes and their social connections as edges. For a prediction $\hat{y}_i$ = "Basketball" for person v_i, an explanation subgraph is identified. The subgraph (right side of the figure) highlights the nodes that contribute most to the prediction. In this case, the explanation reveals that many friends in one part of v_i's social circle enjoy ball games, leading the GNN to predict basketball for v_i. Similarly, for a prediction $\hat{y}_j$ = "Sailing" for v_j, the explanation shows that v_j's friends and their friends enjoy water and beach sports, leading to the sailing prediction. Figure from Ying et al. [43].

for humans, the edge mask M can be transformed into a sparse matrix by retaining only the highest values and setting the rest to zero. An explainer algorithm can then be defined as a function $h : \mathcal{G} \to \mathcal{G}$, which produces an explanation subgraph $h(\mathcal{G})$ for a given input graph $\mathcal{G}$ [56].

3.5.2. Evaluation Metrics

Evaluating explanations is inherently complex. In contrast to model prediction accuracy or computational complexity, explanation quality is not easily measurable. The challenge lies in the subjective nature of explanations. Unlike e.g., model accuracy, which has well-defined mathematical formulations, interpretability lacks a universally accepted quantitative measure. Moreover, it is not trivial to distinguish model vs. explainer errors. A poor explanation might arise because the model itself makes mistakes, which the explainer fails to justify meaningfully. Alternatively, the model might behave correctly, but the explainer provides misleading or nonsensical explanations [48].

Two common approaches for evaluating explanations are [48]:

1. Measuring accuracy by comparing explanations to ground-truth explanations, often referred to as plausibility.

2. Assessing fidelity or faithfulness of explanations using objective metrics to evaluate how well they reflect the model's reasoning process.

The first approach has two key limitations: Human evaluation is impractical as ground-truth explanations are often unavailable and it is subject to human judgment, which may not always reflect the model's reasoning. Measuring accuracy using ground-truth explanations assesses plausibility rather than the actual correctness of the explanations. Petsiuk et al. [57] argue that excluding human perspectives from the evaluation process leads to a more fair and accurate reflection of the model's reasoning. In contrast, model-focused evaluation assesses the consistency of model predictions by either removing or retaining the explanatory graph entities [1]. As a result, fidelity and faithfulness metrics have become the most widely used evaluation criteria in recent XAI literature [58, 56, 59].

In our case, ground-truth explanations are not available. Therefore, we choose the second approach by relying on objective metrics to evaluate the fidelity of explanations. These metrics are defined and discussed in detail below.

Fidelity-

Amara et al. [60] identify three key user requirements for explanations: explanations must be sufficient, necessary, and efficient. To evaluate whether an explanation is sufficient, the model's prediction based solely on the explanation subgraph G_S is compared to its original prediction on the entire graph. This explanation metric, referred to as fidelity-, is calculated as:

$$\text{fid}_- = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i^{G_S} = \hat{y}_i). \tag{23}$$

Here, N represents the number of nodes in the node set $\mathcal{V}$, and $\hat{y}_i$ is the original prediction for node i. For each node i, the indicator function $\mathbb{1}$ checks whether the prediction $\hat{y}_i^{G_S}$, made using only the explanation subgraph G_S, matches the original prediction $\hat{y}_i$. The fidelity- metric is normalized, with values closer to 0 indicating a more sufficient explanation. An explanation is considered sufficient if it independently leads to the model's initial prediction. Since other configurations within the graph could also produce the same prediction, multiple sufficient explanations may exist for a given prediction. [48, 56, 60]

Fidelity+

Measuring fidelity solely in terms of sufficiency using the metric fidelity- introduces a bias that favors larger explanations. This is not desired since the explanation subgraph should be the minimal subgraph containing all the necessary information for making a prediction [61]. An explanation is considered necessary if removing it from the original graph results in a change in the model's prediction. To address this, a second fidelity score, fidelity+, is introduced. The necessity or comprehensiveness of an explanation can be evaluated by removing the explanation subgraph G_S from the entire computation graph G_C. In other words, by providing only the complement of the explanation $G_{C\setminus S}$ to the model. Fidelity+ is defined as the difference in prediction accuracy between the original model's predictions and the modified model's predictions when relevant inputs are masked out. It is calculated as follows:

$$\text{fid}_+ = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i^{G_{C\setminus S}} = \hat{y}_i). \tag{24}$$

The fidelity+ metric is normalized, and values closer to 1 indicate that the explanation is more necessary [60].

Characterization Score

The two fidelity scores, fidelity- and fidelity+, can be combined into a single metric referred to as the characterization score. Amara et al. [60] propose the characterization score as a global evaluation metric because it effectively balances the sufficiency and necessity requirements. This approach is like combining the model precision and recall in F1-score. The characterization score is defined as the weighted harmonic mean of fidelity+ and $1 -$ fidelity$-$:

$$\text{charact} = \frac{w_+ + w_-}{\frac{w_+}{\text{fid}_+} + \frac{w_-}{1-\text{fid}_-}} = \frac{(w_+ + w_-)\cdot \text{fid}_+ \cdot (1-\text{fid}_-)}{w_+ \cdot (1-\text{fid}_-) + w_- \cdot \text{fid}_+}, \tag{25}$$

where $w_+, w_- \in [0,1]$ are the weights for fid_+ and $1 - \text{fid}_-$, respectively, and satisfy the condition $w_+ + w_- = 1$. These weights allow for adjusting the relative importance of fid_+ and fid_-. A typical choice for the weights is $w_+ = w_- = 0.5$ [48].

Efficiency

The third users' need, efficiency, reflects the balance between performance, as measured by e.g., the characterization score, and the time required to compute an explanation.

A method is considered highly efficient if it can quickly produce explanations that effectively capture the reasoning behind model predictions. This criterion is crucial, as users might want rapid explanations.

3.5.3. Explainer Algorithms

In this section we will give an overview about types and categorizations of explainer algorithms as well as summarize the explainer algorithms we used. Fig. 4 shows an overview of the explainer algorithms used. In the following, the categorization and the individual explanation algorithms are described in more detail.

Perturbation-based and Gradient-based Explainer Algorithms

There are two main types of explainer algorithms: perturbation-based and gradient-based. Perturbation-based algorithms generate explanations by changing parts of the input data to create explanation subgraphs. They do this by masking input features and analyzing how the model performance is affected. On the other hand, gradient-based algorithms use the gradients of the prediction with respect to the input features. During the backward pass, the relevance of the input to the output is computed by calculating the gradients through backpropagation. However, the gradients are often noisy, which can cause attributions to include irrelevant features. To address this, several variants of gradient-based methods have been developed, differing in how the gradients are calculated. Additionally, there are other types of explainer algorithms, such as surrogate- and decomposition-based methods [62, 63]. These were not explored in our project and will not be discussed further.

In the following, we present the explainer algorithms used in this project. The focus is on state-of-the-art algorithms for GNNs. These include algorithms designed for GNNs and others adapted from different model types. The algorithms outlined in red in Fig. 4, Saliency and GNNExplainer, are the most important for this work. They are described in more detail.

Saliency [64] is a gradient-based explainer algorithm. Saliency assigns importance to each node by computing the gradient of the output with respect to the node features. The gradient of a function represents the direction of maximum increase, highlighting the input features that cause the steepest local slope in the output. Saliency can be interpreted as applying a first-order Taylor expansion of the network at the input [60, 65].

InputXGradient [66] extends Saliency by multiplying the gradients of the output

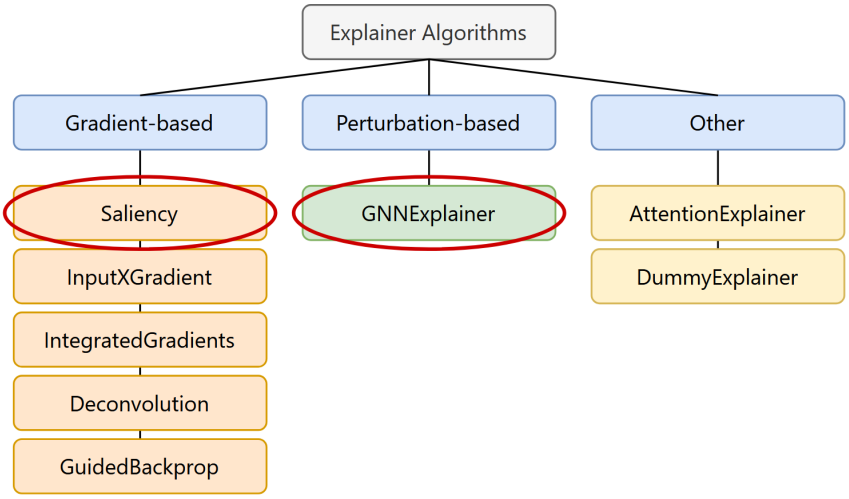


Figure 4: Overview of the used explainer algorithms, categorized into gradient-based, perturbation-based and other. The two algorithms outlined in red are the most important for this work.

with respect to the input by the input feature values.

IntegratedGradients [67] addresses the issue of small gradients in flat regions of the network by accumulating gradients along a path from a baseline input, typically a zero vector, to the actual input.

Deconvolution [39] computes gradients of the output with respect to the input but only backpropagates non-negative values for ReLU functions.

GuidedBackprop [42] modifies ReLU gradients similar to Deconvolution, but instead of applying the ReLU function to the output gradients, it is applied to the input gradients.

GNNExplainer [43] is a perturbation-based explainer algorithm. GNNExplainer is the first algorithm designed specifically for GNNs. It learns a mask for each node's input graph. The masks are initialized randomly and treated as trainable variables. Formally, GNNExplainer is formulated as an optimization task to maximize the Mutual Information (MI) between the GNN's predictions and potential subgraphs. MI measures the shared information between two variables, linked to entropy in information theory. The key idea is that GNN predictions are determined by the graph structure and node features, meaning that only these two aspects need to be considered for explanations. MI computes how the prediction changes when restricting the graph and features to subsets. The objective is to maximize the MI between the predicted class labels and the subgraph respectively feature subset. However, due

to the large number of possible subgraphs, the optimization problem is not directly solvable, so a relaxed version is used as a variational approximation [18].

AttentionExplainer [68] is an algorithm designed for attention-based GNNs. AttentionExplainer uses attention scores on edges to neighboring nodes to determine the importance of information propagation for the predictive task.

DummyExplainer serves as a control algorithm. It generates random explanations by sampling nodes and edge importances from a uniform distribution.

4. Empirical Analysis

This chapter provides an empirical analysis of the previously presented methods for improving model interpretability and trustworthiness. Section 4.1 presents an analysis of the confidence calibration of TEIGR for grid topology error identification. We present the results of the confidence calibration methods, histogram binning and temperature scaling. Section 4.2 explores the visualization of TEIGR, providing insights into the internal structure of TEIGR. We analyze and compare the visualizations of the dimension reduction methods PCA, t-SNE and UMAP. Section 4.3 discusses the visualization of explanations, including a qualitative analysis of explanation categories. Section 4.4 compares the explainability of correctly classified versus misclassified nodes. Finally, Section 4.5 examines the integration of an explainability term into the loss function of TEIGR and evaluates the effect on performance and interpretability.

4.1. Confidence Calibration

In this section, we analyze the confidence scores and calibration of TEIGR. We show the reliability curve and the confidence histogram for TEIGR. The results of applying the confidence calibration methods histogram binning and temperature scaling will be shown. We discuss in which cases which method is preferred.

4.1.1. Setup

We inspect the confidence calibration of TEIGR, where we take the hyperparameters and training settings from former studies [7, 8]. More information about the hyperparameters can be found in Appendix A.3. We apply the sigmoid function to the outputs of TEIGR to convert the logits to probabilities between 0 and 1. The bin number for calculating the ECE is set to 10. We rerun the experiments with 100 different random seeds and plot the error bars representing the standard deviation for the reliability curve.

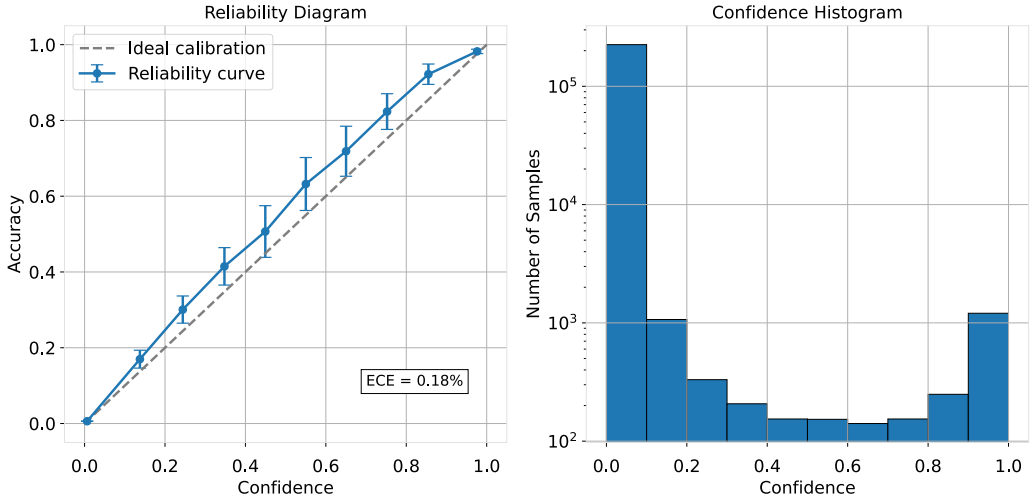


Figure 5: Reliability diagram (left) and corresponding confidence histogram (right) for uncalibrated GNN on validation dataset.

4.1.2. Results

Fig. 5 (left) shows the reliability diagram. The reliability curve of TEIGR is close but not perfectly on the diagonal. The ECE is exceptionally low with 0.18%. TEIGR returns well calibrated predictions. A reason could be the small model size of three layers. According to Guo et al. [10] the model capacity negatively affects model calibration. The shape of the reliability curve of TEIGR can be interpreted as slightly under-confident because it is above the diagonal. This is in line with the findings of studies about confidence calibration for GNNs [19, 28] that GNNs tend to be under-confident.

Additionally, next to the reliability diagram, we plot the corresponding confidence histogram showing the number of samples in each predicted probability bin (Fig. 5 (right)) on a logarithmic scale. We find that the majority of the samples fall into the first bin, which represents confidence values in the range [0, 0.1). Since we have a binary classification task this can be interpreted as TEIGR being highly sure that these samples are not class 1 (error), but class 0 (no error). Most nodes can be easily classified as no error. Only a small proportion of the samples fall into the middle bins (0.4-0.6), where TEIGR is uncertain. The high number of samples where TEIGR is highly confident that it is class 0 explains the low ECE: The ECE quantifies miscalibration by averaging the absolute differences between accuracy and confidence

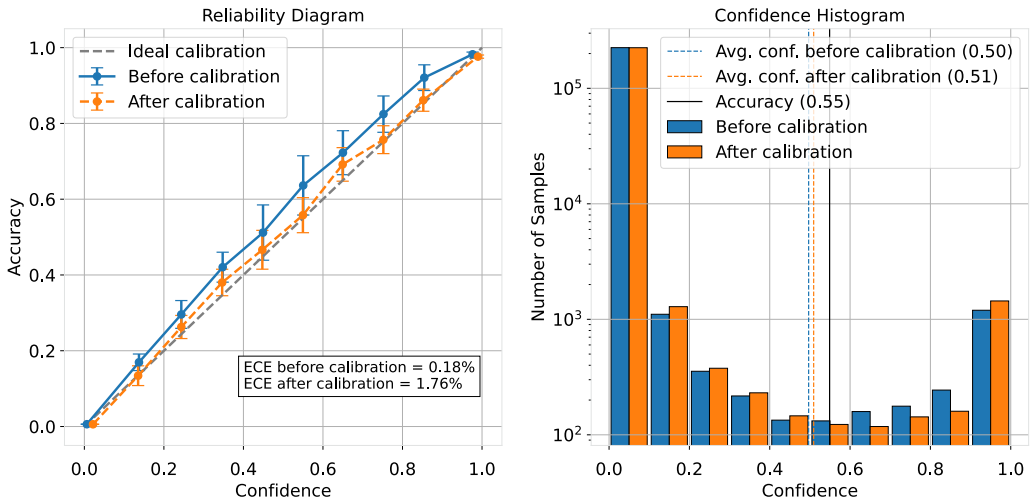


Figure 6: Confidence calibration: Histogram binning. Reliability diagram (left) and corresponding confidence histogram (right). In the confidence histogram, the vertical lines indicate the accuracy and average confidence, before and after calibration. The average confidence before and after calibration is close together.

across all confidence bins, weighted by the proportion of samples in each bin. Thus, the ECE calculation is dominated by the majority of the samples, for which TEIGR is highly confident in this case.

Overall, TEIGR is already well calibrated. To further improve the calibration, we applied the calibration methods histogram binning and temperature scaling.

4.1.3. Histogram Binning

In Fig. 6 we present the reliability curve and the corresponding confidence histogram after applying the calibration method histogram binning (orange curve). For comparison, we show the results for the original GNN model before calibration too (blue curve). Fig. 6 (left) shows that the reliability curve after the calibration is closer to the diagonal which suggests that histogram binning worked and improved the calibration. However, the ECE increased from 0.18% before the calibration to 1.76% after the calibration. This can happen for several reasons, and it highlights that the metrics ECE and reliability curve focus on different aspects. There is a trade-off in calibration between global improvement and local errors. The reliability curve suggests a global improvement because the reliability curve is closer to the diagonal.

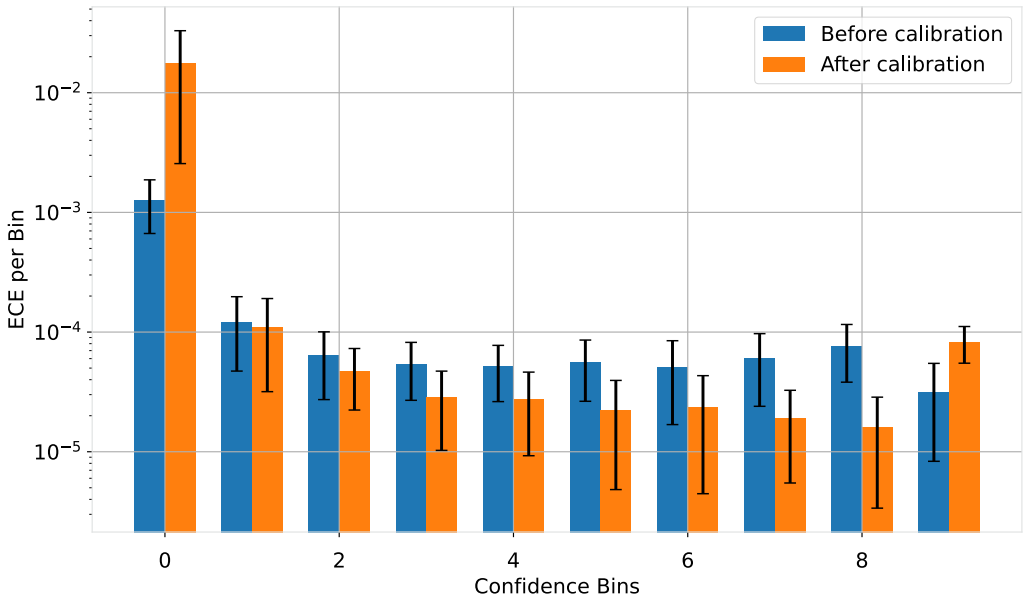


Figure 7: Confidence calibration: Calibration error per confidence bin before and after histogram binning calibration.

On the other hand, ECE captures local errors within individual bins: The total ECE increases if small but significant miscalibration exists in bins with many samples, even if the overall curve looks smoother.

For verifying this, we visualize in Fig. 7 the individual calibration errors (CE), i.e., the ECE per bin. After applying histogram binning, the CE has improved in all bins with the exception of bin 0, and 9. As bin 0 has the highest number of samples $|B_m|$, the contribution of the corresponding term to the sum in Eq. 12 is higher, thus increasing the ECE. Here, histogram binning overfits to sparse bins with few samples, making the predictions highly accurate for a small group of samples while leading to poorer overall calibration. Therefore, we tried another calibration method, temperature scaling, for comparison.

4.1.4. Temperature Scaling

In Fig. 8 we present the reliability curve and the corresponding confidence histogram after applying the calibration method temperature scaling (orange curve). For comparison, we also show the results for the original GNN model before calibration (blue curve). As can be seen in Fig. 8 (left), the results for temperature scaling are quite

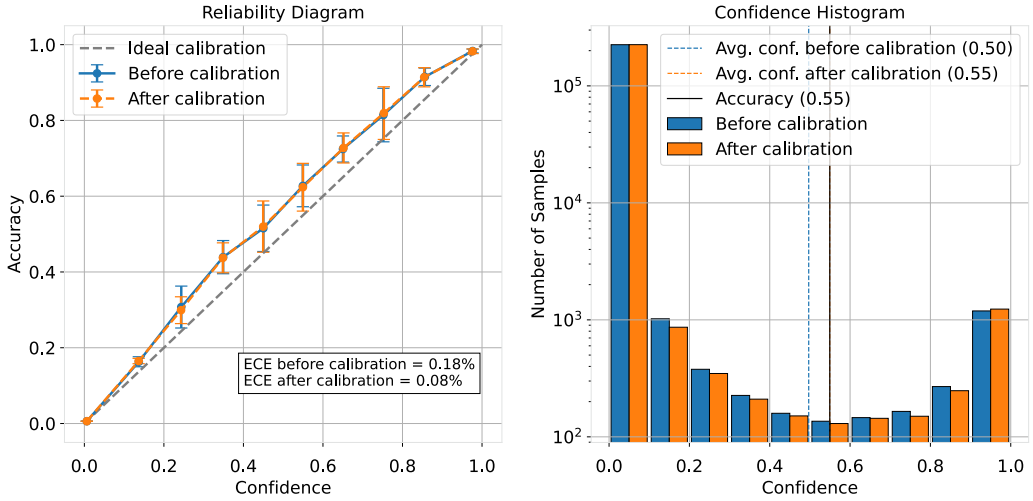


Figure 8: Temperature scaling: Reliability diagram (left) and corresponding confidence histogram (right). In the confidence histogram, the vertical lines indicate the accuracy and average confidence, before and after calibration. The average confidence after calibration is close to the accuracy.

different compared to histogram binning: while the reliability curve remains largely the same, the ECE improves. Before calibration, the ECE was 0.18%, afterwards it is 0.08%.

A reason for these results could be the global adjustment and the lack of granularity of temperature scaling. Temperature scaling applies a single scaling parameter to adjust the sharpness of the model's predicted probabilities. Therefore, temperature scaling cannot make fine-grained adjustments to specific regions of the probability space. Thus, temperature scaling may leave certain areas miscalibrated, especially in unbalanced datasets, even though the overall ECE improves. For unbalanced datasets, temperature scaling may disproportionately benefit the majority class, which dominates the ECE calculation, while leaving the minority classes with the same level of miscalibration. This can be seen in Fig. 9: The calibration is strongly improved for the biggest bin, bin 0, but not for the bins with less samples. This can result in the reliability curve appearing unchanged, even though the aggregate ECE improves.

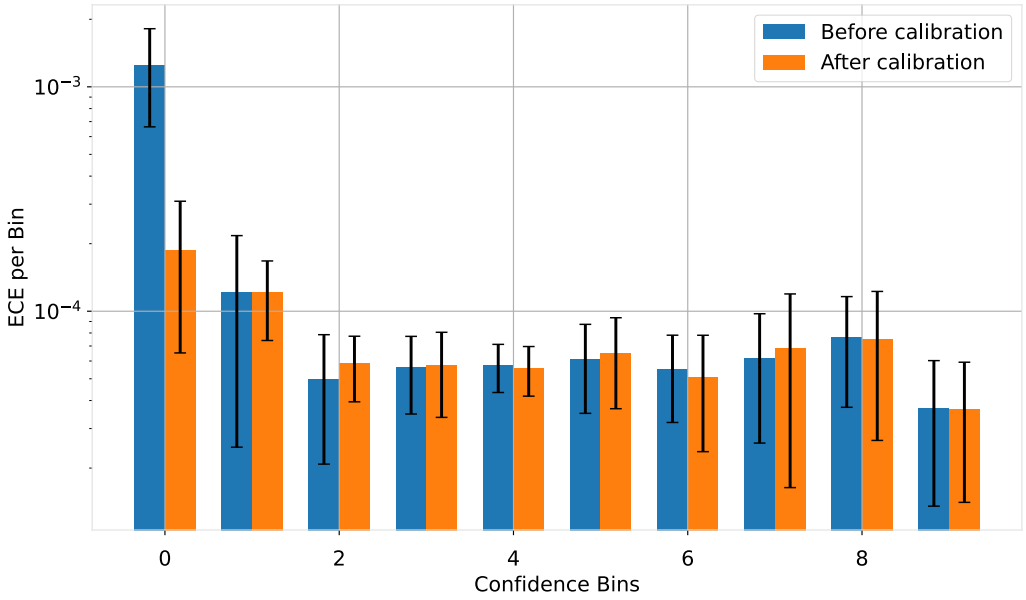


Figure 9: Temperature scaling: Calibration error per confidence bin before and after calibration.

4.1.5. Discussion

Both calibration methods, histogram binning and confidence calibration, could be beneficial for grid operators. Which method is preferred depends on the objectives of the calibration. Histogram Binning improves the calibration of the middle bins with confidences around 0.5. These bins contain samples for which the model is not sure whether they are class 0 or 1. For grid operators who want to use TEIGR for unclear decisions, histogram binning can offer a decisive advantage. Histogram binning improves the confidence of unclear samples in the middle bins. Verifying the grid topology in practice can be expensive, which explains why grid operators may fully rely on TEIGR.

On the other hand, for grid operators who only want to make decisions based on TEIGR if TEIGR is very reliable, temperature scaling can be beneficial. Changing the grid topology can have far-reaching consequences on grid operation, which makes it understandable if grid operators act with caution. For these grid operators, it is more important to increase the confidence of the outer bins to be able to make even safer decisions there. Both cases are conceivable. It is important to discuss these options with users and highlight effects and possibilities of the calibration methods.

4.2. Model Visualization

In the following section, we visualize TEIGR with three dimension reduction methods, PCA, t-SNE and UMAP. We expect nodes in the grid that are similar functionally and topologically to have similar representations in TEIGR. We use various color schemes to encode node similarity in terms of features or topological proximity. We analyze the effect of model training on the representations. Moreover, we compare different feature sets regarding their influence on the representations. The goal of these visualizations and analyses of the internal high-dimensional representations of TEIGR is to make TEIGR easier to interpret and understand. The visualizations also serve as a sanity check, ensuring that the hidden representations exhibit a reasonable structure, which reduces the perception of TEIGR as a "black box". Additionally, the clustering of these representations might help to identify potential false positives and false negatives.

4.2.1. Setup

Just like Solheim et al. [20] we use for t-SNE a perplexity value of 200 and for UMAP the number of nearest neighbors $k = 200$ and min_dist $= 0.5$. The only exception is in Fig. 16 where we use lower hyperparameter values to investigate the robustness of the perplexity parameter.

We use the small synthetic grid topology, which we have shown as an example in the introduction (see Fig. 1b), with just 26 nodes to obtain a visualization that is as clear as possible. In Appendix A.1 the properties of this grid, called "Minimal (syn)", can be found (first line in Tab. 2). In Fig. 10 the corresponding grid topology is visualized. It contains an error in form of an incorrectly connected line. The original line between node 22 and 24 is wrongly connected between node 17 and 24. Therefore, nodes 17, 22, and 24 are erroneous and have the ground truth label 1, while all other nodes have the label 0. We show the visualizations here using this specific error as an example. The visualizations are consistent for different errors. An example with a different error can be found in Appendix B.1. We use 30 snapshots with different feature values, but the same wrong connection. These snapshots are randomly selected from voltage and power data for the year 2022. Each snapshot corresponds to one marker for each node in the scatter plot (see Fig. 11). Similar to the study from Solheim et al. [20], this approach allows us to obtain multiple data points for the same node and error condition, but with slightly different load conditions.

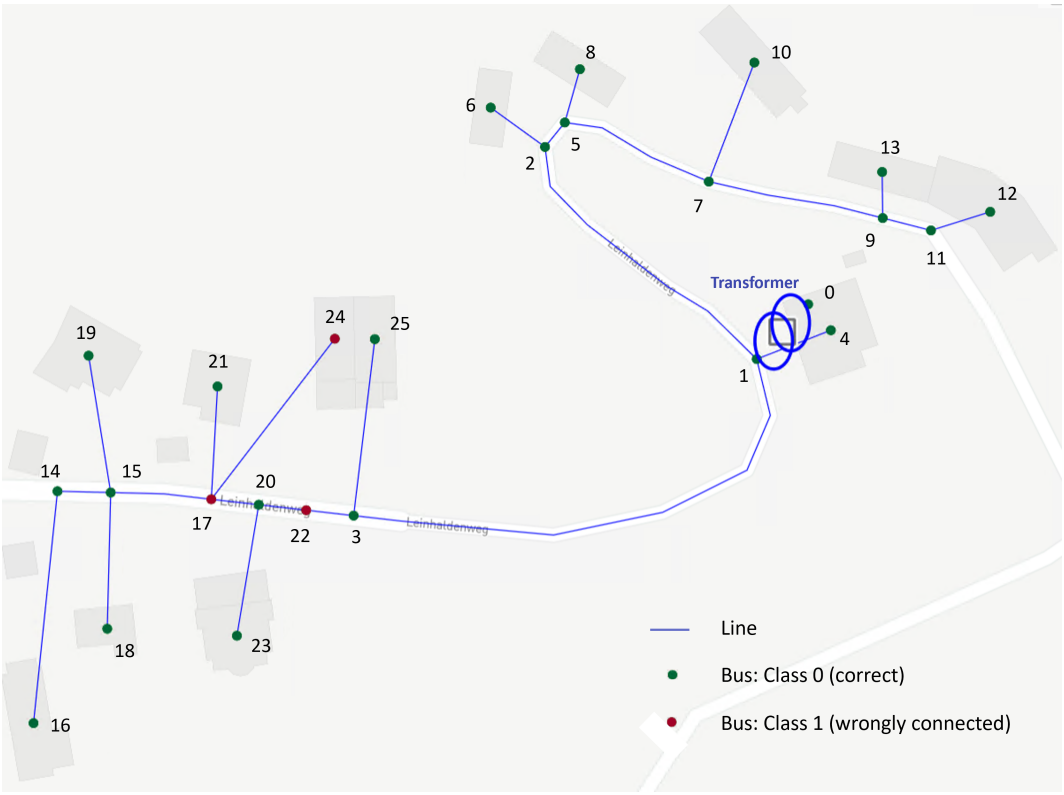


Figure 10: Grid topology of small synthetic grid. The bus colors indicate the ground truth class. Red buses are wrongly connected.

4.2.2. Results

In Fig. 11, we present visualizations of the final message-passing layers of TEIGR using the three different dimensionality reduction techniques: PCA, t-SNE, and UMAP. We assigned each node (0-25) a unique color, uniform for all snapshots. The PCA plot in Fig. 11 (left) does not exhibit any clear clustering of nodes. This pattern closely resembles typical 2D PCA visualizations of datasets such as the handwritten digits database MNIST [69], as demonstrated in [70]. Only node 0 (red), the so-called slack bus, is separated from the other nodes. The reason for this could be that the slack bus differs significantly from the other buses in terms of features and function. The slack bus is usually not a physical entity but represents the transmission grid. The slack bus is modeled as supplying the energy required by the distribution grid. If the distribution grid generates a surplus of energy, the slack bus is considered to "consume" that excess energy [8]. The slack bus supplies a voltage of 400 volts, which is derived from the transmission grid's voltage magnitude and stepped down through

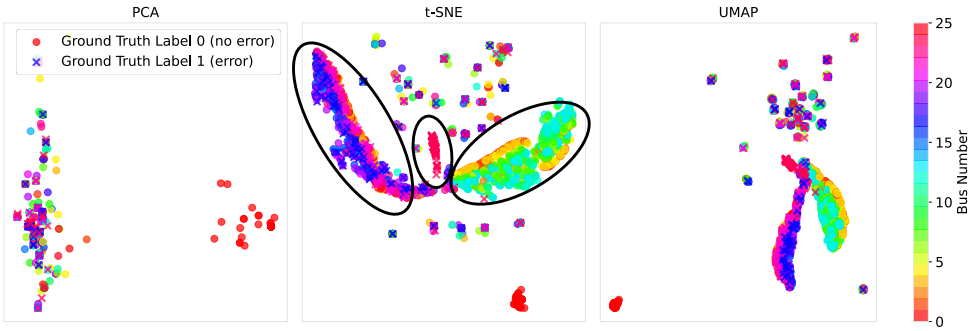


Figure 11: PCA, t-SNE and UMAP visualizations of the trained TEIGR. The color represents the node number. The markers indicate the ground truth labels: circles are correctly connected nodes, crosses are wrongly connected nodes. For better recognizability, we have marked the clusters for t-SNE with black ovals, as the clusters can be easily clarified here. Similar clusters could be drawn for UMAP, but this has been omitted for reasons of clarity. No clear clusters are visible for PCA.

a transformer. Because of impedance in the lines, other buses have lower voltages. A typical voltage magnitude distribution can be seen in Fig. 12. Such feature value differences could explain the separation of the slack bus in the PCA visualization.

The t-SNE and UMAP visualizations in Fig. 11 reveal clearer clusters compared to PCA. While the t-SNE and UMAP representations are quite similar, UMAP appears to produce slightly less fragmented clusters for the chosen hyperparameter settings. For t-SNE, three clusters can be identified. They are marked in Fig. 11 (middle) with black ovals. The left cluster mainly consists of dark blue and pink points corresponding to nodes with high numbers (14-25). These nodes are part of the lower left area in the grid topology in Fig. 10. Similarly, the right cluster in the t-SNE visualization with yellow and light blue points corresponds to the nodes 5 to 12 in the upper right area of the grid in Fig. 10. This shows that topographical proximity of the nodes and the associated correlating features are relevant. The smaller cluster in the middle of the t-SNE visualization stands out. This cluster mainly consists of cross-markers, i.e., wrongly connected nodes with ground truth label 1. The visualization shows that TEIGR can separate wrongly connected nodes and learns in a meaningful way. However, not all faulty nodes are in this “error”-cluster. One reason could be the imperfection of TEIGR, as not all errors are correctly recognized as such.

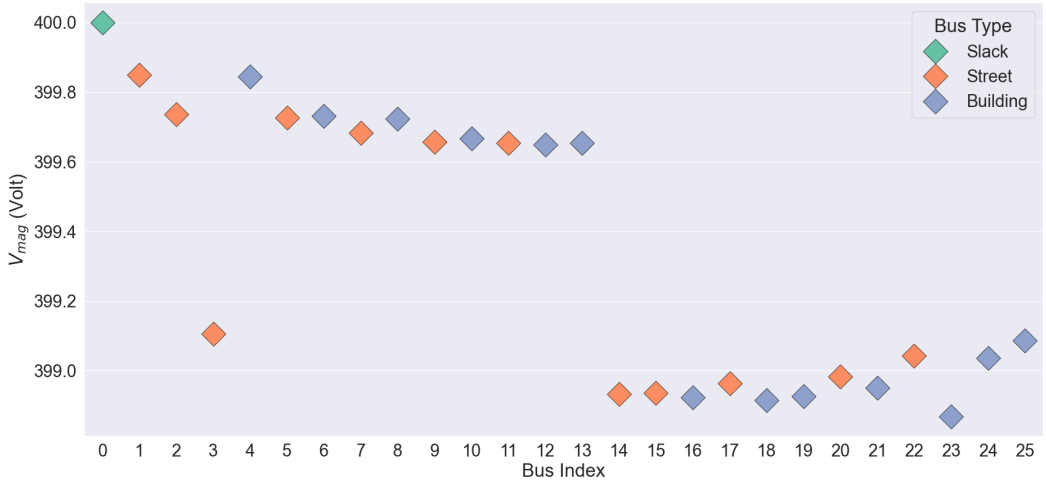


Figure 12: Typical voltage magnitude values computed during power flow analysis using synthetic power profiles in the 26-node grid shown in Fig. 10 [8]. The slack bus provides a reference voltage of 400 V, which is derived from the transmission grid and subsequently transformed down to this value. Based on similar voltage magnitudes, buses can be grouped into three clusters: (0, 1, 4), (2, 5, 6, ..., 12, 13), (3, 14, 15, ..., 24, 25). The clustering corresponds to the spatial proximity of the buses which can be seen in Fig. 10. In particular, the long line between buses 1 and 3 explains the significant voltage drop observed between their respective clusters.

Effect of Model Training on Representation

In case of an untrained version of TEIGR, visualized in Fig. 13, no such cluster of wrongly connected nodes can be identified. Compared to Fig. 11, the cluster of crosses, which is marked by the small black oval in the middle in Fig. 11, is missing. Instead, the crosses are distributed across the visualization. They are more common in the left cluster, which is because the faulty connection is in this cluster. The node classes cannot be distinguished in an untrained model. The representation improves through training. An even greater difference is expected for more complex models [20]. However, the topographical clusters are still visible, again marked with black ovals. The reason for this invariance could be that the topographical information is inherent to the data and linked to the feature values of voltage magnitude and angle.

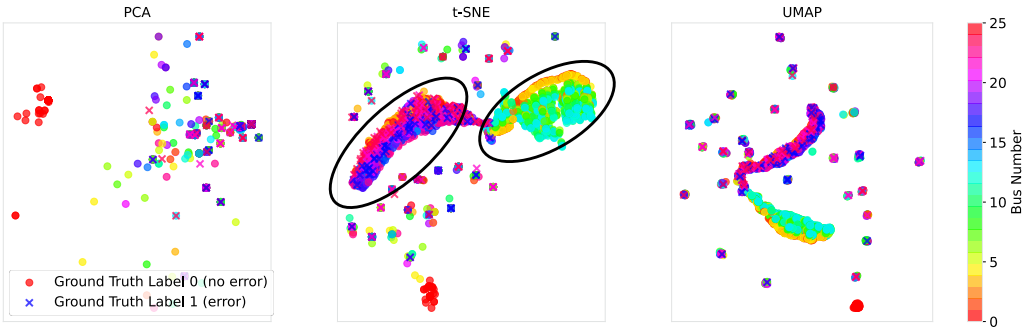


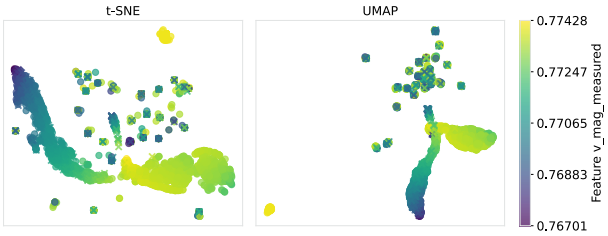
Figure 13: PCA, t-SNE and UMAP visualizations of an untrained version of TEIGR. The color represents the node number. The markers indicate the ground truth labels: circles are correctly connected nodes, X's are wrongly connected. In contrast to the visualization of the trained TEIGR, no cluster of wrongly connected nodes can be seen.

Influence of Model Features

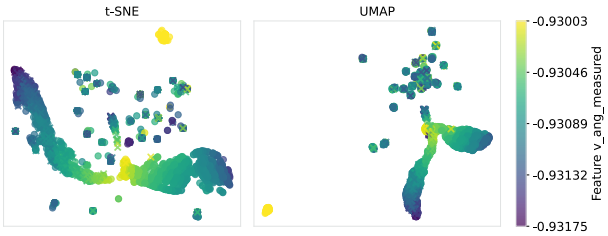
To analyze the reasoning of TEIGR more deeply, we did the same visualization for the trained TEIGR with different color encoding: Instead of the node number coloring in Fig. 11, we colored the data points in Fig. 14 according to their feature values. We do not show the PCA visualization here, as it does not provide any relevant insights. In Fig. 14a we see that the topographical cluster distinction correlates with the voltage magnitude measurements. The left cluster in the t-SNE visualization has consistently lower voltage magnitudes than the right cluster. This observation is in line with a typical voltage magnitude distribution as seen in Fig. 12. The significant voltage magnitude drop in the first cluster is probably primarily influenced by the long line between node 1 and 3 (see Fig. 10).

Looking at the voltage angle coloring in Fig. 14b we do not see a clear difference between the clusters, but a smooth gradient within the clusters. Nodes located further out have a higher voltage angle. Like voltage magnitude, the voltage angle is also influenced by the position of a node within the grid topology. The direction of increasing voltage angle is varying across the two clusters. This lack of a consistent global ordering within clusters is expected, as t-SNE and UMAP axes are not inherently interpretable. Global features are not guaranteed to be preserved in a t-SNE and UMAP projection [20].

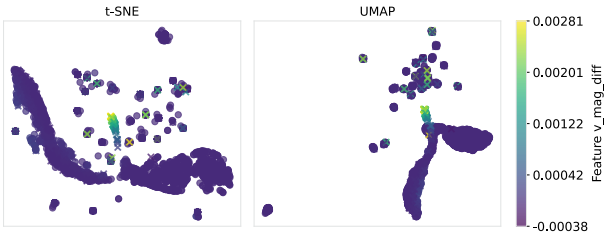
For feature 3 and 4, a different, revealing picture emerges. The visualization does not differ significantly between these two features, so we will discuss them together.



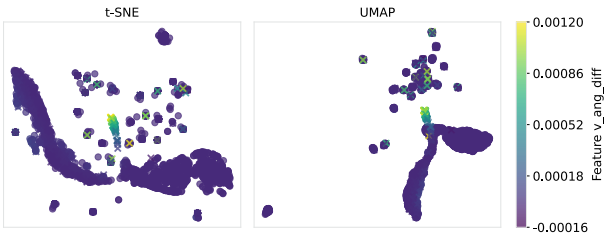
(a) Color by feature 1: Measured voltage magnitude.



(b) Color by feature 2: Measured voltage angle.



(c) Color by feature 3: Voltage magnitude differences.



(d) Color by feature 4: Voltage angle differences.

Figure 14: t-SNE and UMAP visualizations colored by feature values.

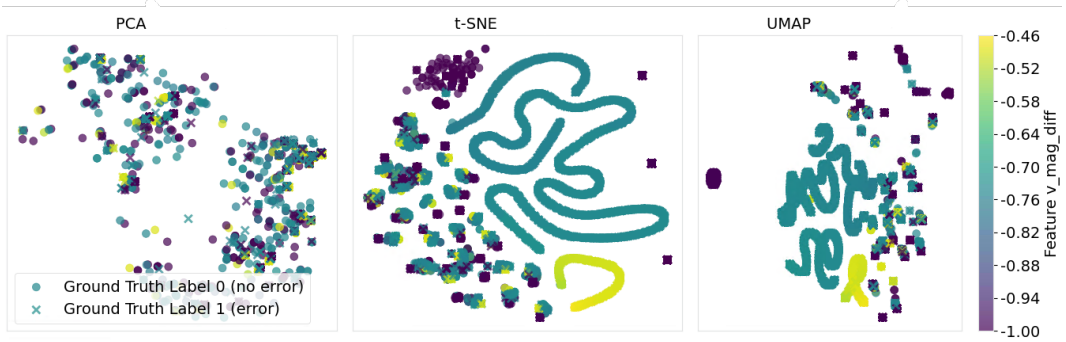


Figure 15: PCA, t-SNE and UMAP visualizations of a version of TEIGR with just one feature. The color represents the feature values, the voltage magnitude differences between the measured and expected values.

The two features describe the difference between the measured and expected voltage magnitude respectively angle. It is noticeable that the difference for wrongly connected nodes is significantly greater than for correct nodes (crosses appear brighter overall than circles). This is in line with former research which showed that the difference between the measured and the expected voltage is a highly effective feature for verifying a grid's topology [8]. This difference is so relevant that even a model trained with only one of these two features instead of the original four together (measured voltage magnitude, measured voltage angle, voltage magnitude difference, and voltage angle difference) achieves comparable results [8]. Therefore, we were interested in how the visualization for such a model with only one feature looks like.

Visualization of Model with One Feature

In Fig. 15, the PCA, t-SNE and UMAP visualizations of the final message-passing layer of the model trained on a modified dataset with just one feature, voltage magnitude difference, can be seen. The visualizations are colored by the feature values, i.e., the voltage magnitude differences of each node. The long serpentine lines in the t-SNE visualization are striking. We hypothesize that this pattern arises because the data is inherently one-dimensional, and the t-SNE and UMAP representations effectively capture this characteristic.

Influence of the Visualization Hyperparameter Perplexity

The t-SNE and UMAP model visualizations are sensitive to the choice of the hyperparameters. The influence of a lower perplexity of 10 and a number of nearest

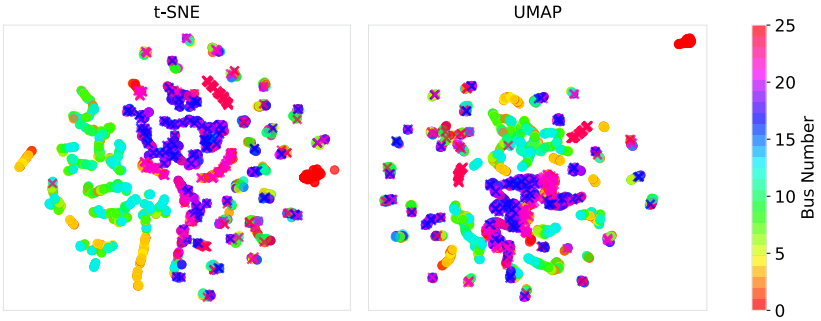


Figure 16: Influence of a lower perplexity of 10 and a number of nearest neighbors $k = 10$.

neighbors $k = 10$ can be seen in Fig. 16. The clusters are much more fragmented, and connections are more difficult to recognize.

4.2.3. Discussion

In this section we visualized the internal representations of TEIGR. By using dimension reduction methods, we have gained insights into the hidden representations underlying the decision process. The visualizations consist of clusters related to the topographical proximity and the labels of the nodes. By analyzing the influence of the features with various color schemes, we were able to confirm the importance of the features which describe the difference between measured and expected voltage magnitude respectively angle. Our visualizations also highlight how the representation improves through model training. For an untrained model, the visualization does not distinguish between correctly and wrongly connected nodes.

For comparison, we have done the visualizations for other wrong connections too and varied the position of the wrong line. An example with a different error can be found in Appendix B.1. The visualization results are transferable to larger grids. Such an example can be found in Appendix B.2.

The visualizations help in understanding TEIGR and building confidence in its development. In future work, dimension reduction visualizations could potentially also aid to improve the model architecture of TEIGR. In an iterative process, such visualizations could play a significant role in identifying models with desired properties. Such desired properties could be the consistency of clustering or robustness to noise, i.e., the ability to maintain meaningful internal representations even when the input data is noisy. A clear differentiation of classes by separate clusters would be desirable.

Another desired property could be generalization, i.e., the ability of the model to generalize well to new power grid topologies. This generalization ability can be verified by visualizations that show consistency of their structure across different grid topology variations.

4.3. Explanation Visualization

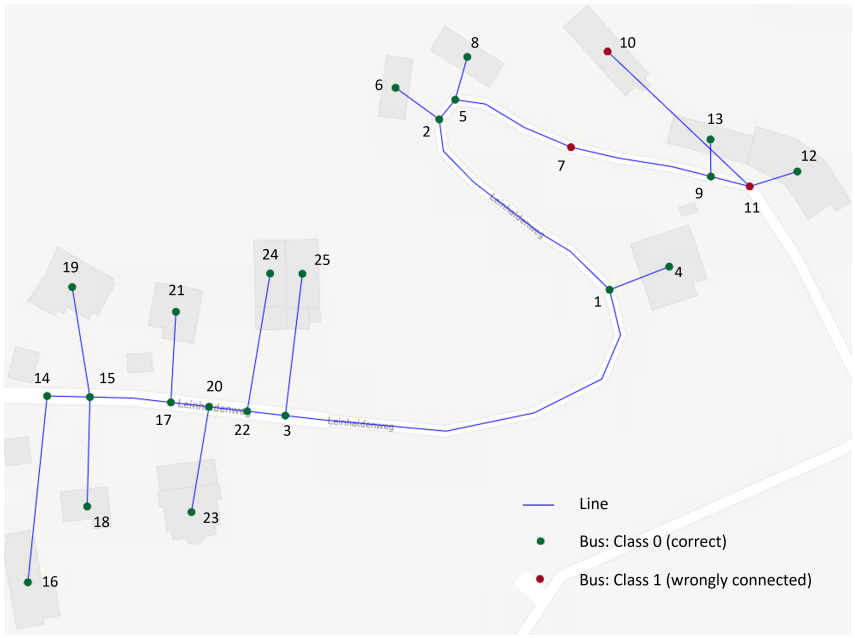
The previous approach visualized TEIGR as a whole, which is particularly useful for the developers of TEIGR. In contrast, in this section, we focus on making individual predictions explainable for users, such as grid operators. Explainable AI can uncover the inner workings of complex models like GNNs, offering deeper insights into their predictions. For GNNs, an explanation often takes the form of a subgraph that highlights the most influential nodes and edges for a given prediction. We implemented a visualization that embeds the explanation subgraph into the original grid topology. This makes the explanation naturally understandable for grid operators.

In this section, we present an exemplary explanation and show how grid operators can benefit from it. Moreover, we qualitatively analyze and categorize explanations.

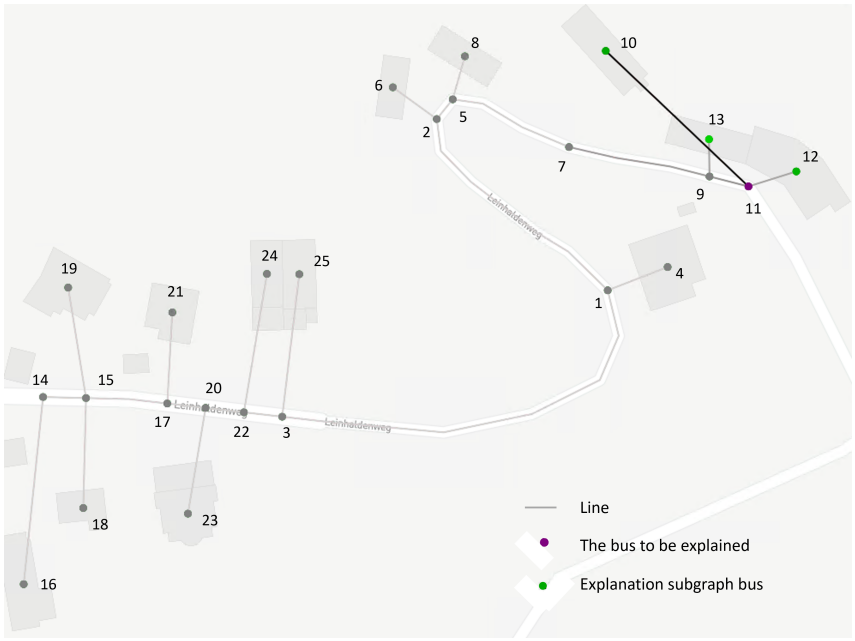
4.3.1. Example

In this subsection, we provide an example explanation to illustrate the output of explainer algorithms. For this example and the following qualitative analysis, we use the explainer algorithm Saliency because it is the best performing explainer algorithm regarding characterization score and efficiency in [48]. We use the same grid as in the introduction (see Fig. 1b) and in the previous section (see Fig. 10). However, we have varied the error, as the benefit for the grid operators can be better explained here. The grid topology and the model prediction can be seen in Fig. 17a. This topology contains an error in form of an incorrectly connected line. The original line between node 7 and 10 is wrongly connected between node 10 and 11. Therefore, nodes 7, 10, and 11 are erroneous and have the ground truth class label 1, while all other nodes have the class label 0.

In Fig. 17b, we visualized the corresponding explanation for the model prediction for the wrongly connected node 11. The explanation indicates why TEIGR predicted for node 11 class 1. The node to be explained, node 11, is colored purple. The nodes which are part of the explanation subgraph are colored green. The node to be explained is also part of the subgraph, but has been colored differently to make it



(a) Model prediction: Grid topology of small synthetic grid. The bus colors indicate the ground truth class. Red buses are wrongly connected.



(b) Explanation for model prediction for erroneous bus 11. Bus 11 has the class label 1: wrongly connected. The explanation shows why the model predicted for bus 11 class 1.

Figure 17: Exemplary model prediction and its explanation for small grid.

better visible. The darker the green of the explanation subgraph nodes, the more important the nodes are for the prediction. The same applies to the lines: The darker the lines, the more significant the corresponding edges in the explanation subgraph. In this case, the line between node 10 and node 11 has the greatest influence on the prediction of node 11. That makes sense since this is the wrongly connected line. The explanation could therefore correctly show that node 11 has the class 1 because of its connection to bus 10 and not, for example, because of the connection to node 9 or 12.

Most topology model errors are likely to be more subtle than the one presented in this example. The explanation is then all the more important so that grid operators can understand the prediction. Moreover, in practice the grids are larger than in this example. Our visualization scales for larger grids. We show, analyze and categorize such examples in the next subsection.

Overall, while the prediction only recognizes the incorrectly connected nodes, the explanation can also identify the incorrectly connected line responsible for this. The explanation can be helpful in other cases too. In which cases the explanation is helpful and which patterns can be found here are examined in more detail in the following subsection.

4.3.2. Qualitative Analysis: Explanation Categories

In the previous section, we presented a true positive (TP) example: The node was correctly classified by the model as positive (class 1: error). However, of particular interest to grid operators are explanations for incorrectly classified nodes, as these need to be identified and analyzed in more detail. Incorrectly classified nodes can be either false positives (FP) or false negatives (FN). False positives are cases where a node is incorrectly classified as belonging to class 1 (error). False negatives are cases where a node that actually belongs to class 1 is not identified as error.

By analyzing individual explanations, it has been shown that they can be categorized. In the following, we present categories for explanations for false positives and false negatives. We divide these into helpful and unhelpful explanations. For each category we show the visualization of a representative explanation example of that category. We give an estimate of which categories occur frequently and which rarely. Due of the manual evaluation involved, it was not feasible to determine incidence numbers based on a large sample size. However, we provide an empirical estimate after inspecting 30 random cases.

False Positive (FP) Categories

For the false positive categories, we chose examples from the grid "Spaltenstein" with 80 nodes. The properties of this grid can be found in Appendix A.1 (second line in Tab. 2). The errors in the grid and the nodes to be explained vary between the examples. To clarify the categories, we have focused on obvious errors. These are unrealistic in practice, but in the examples, they make it possible to recognize the error at first glance.

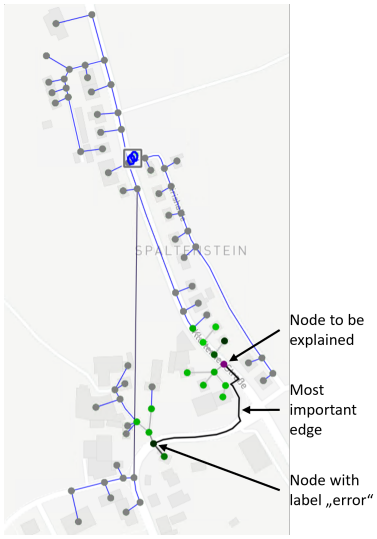
1. **A neighbor node is the real error and the explanation indicates it.**

   An example for this category is visualized in Fig. 18a. The purple node is the one that is explained. This node is a false positive: TEIGR has predicted the label "error", although the node in the ground truth has the label "no error". The most important edge in the edge mask of the explanation subgraph is the one to the neighbor node with the label "error". The neighbor node is erroneous, because it has a missing connection. The neighbor node has also a high importance value in the node mask, represented by a dark green color. The explanation is helpful as it indicates why the purple node was predicted incorrectly: TEIGR was misled by the faulty neighbor node. The category occurs frequently.

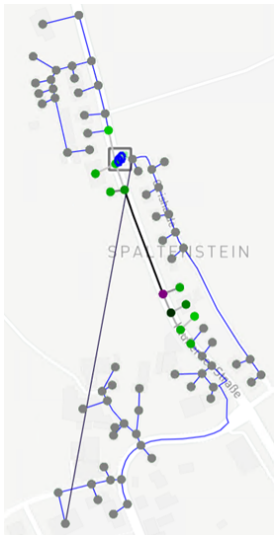
2. **The real error is close by and the explanation indicates it.**

   An example for this category is visualized in Fig. 18b. This category is similar to the previous category. The difference is that the real error is not the direct neighboring node, but only close by. The real error can be two or three nodes away. Nevertheless, the real error affects the model prediction and causes TEIGR to incorrectly predict the label error. The explanation reflects that by giving the highest importance value to the edge pointing in the direction of the real error. This can help the grid operator to understand the model prediction and even find the correct error.

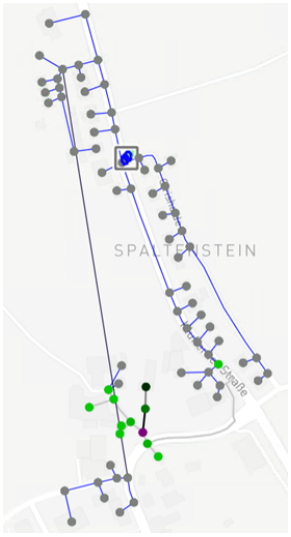
   However, sometimes edge and node mask are contradictory. This can also be seen in Fig. 18b: the node with the highest importance, recognizable by the dark green shade, is not close to the real error. Overall, we have found that the edge mask of the explanation is more helpful than the node mask, especially for false positives. This can be an important indication for grid operators. The category occurs frequently.



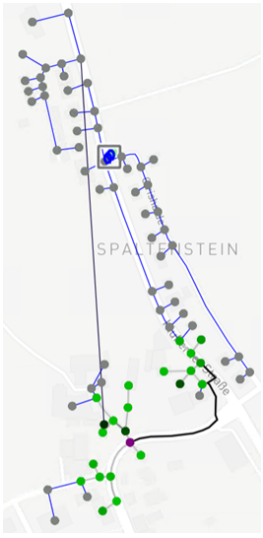
(a) FP-Category 1: Neighbor node is real error and explanation reflects this.



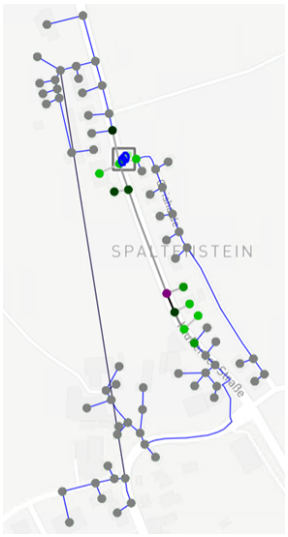
(b) FP-Category 2: Real error is close by and the explanation reflects this.



(c) FP-Category 3: Neighbor node is real error, but explanation does not reflect this.



(d) FP-Category 4: Real error is close by, but explanation does not reflect this.



(e) FP-Category 5: No real error close by.

Figure 18: Example explanations for the five false positive categories.

3. **A neighbor node is the real error, but the explanation does not indicate it.**

   As mentioned in category 1, it is not always the case that the explanation points in the direction of the real error. Such an example can be seen in Fig. 18c. Both edge importances and node importances of the explanation point in a direction where no erroneous node is located. Because this category also occurs frequently, it is not possible to derive an automated recommendation for action for grid operators based on the explanations.

4. **The real error is close by, but the explanation does not indicate it.**

   An example for this category is visualized in Fig. 18d. Three nodes away from the node to be explained is a node that has an incorrect additional connection. This is an example where the node mask would be more helpful than the edge mask. The most important edge is misleading, but the most important node is the one with the actual error. This example shows that not every false positive falls into a category with a simple correction mechanism. The category is rare.

5. **No real error close by.**

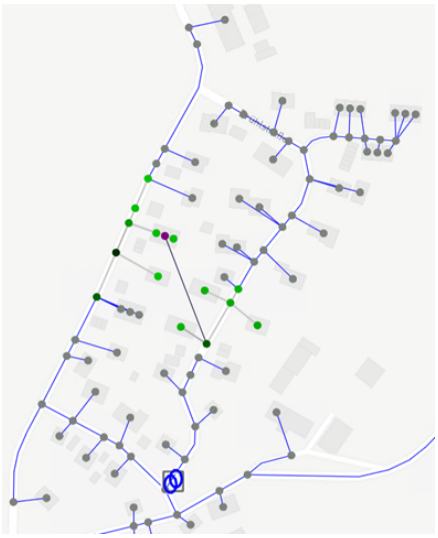
   The category differs fundamentally from the previous four. No node near the examined false positive is a real error. Such an example can be seen in Fig. 18e. The explanation does not provide a helpful indication of how the model arrived at the wrong prediction. This category is rare.

False Negative (FN) Categories

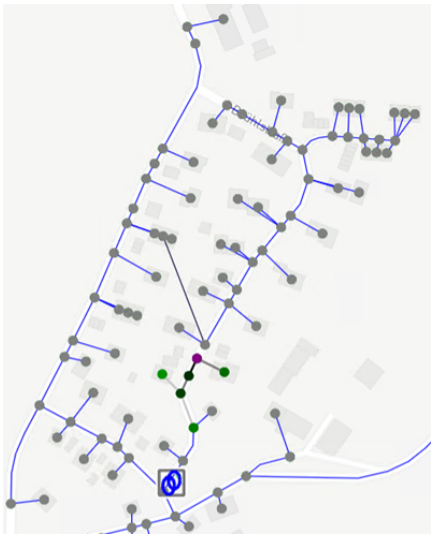
For the false negative categories, we use examples from the grid "Eggenweiler" as the categories become clearer here. The grid is larger than "Spaltenstein", it has 115 instead of 80 nodes.

1. **The explanation can narrow down the search space.**

   An example for this category is visualized in Fig. 19a. The purple node is the one that is explained. The node is a false negative. This means that TEIGR has predicted the label "no error", although the node in the ground truth has the label "error". The explanation node mask gives the neighbor node to which the examined node is wrongly corrected a high importance. However, other nodes that are further away and not faulty are also given high importance. Therefore, the explanation is mainly helpful to limit the error search to a small subset of candidate nodes. This category occurs frequently.



(a) FN-Category 1: Explanation is somewhat helpful.



(b) FN-Category 2: Explanation not helpful, because the error type is "missing connection".

Figure 19: Example explanations for the two false negative categories.

2. **The explanation is not helpful because the error type is a "missing connection".**

   An example for this category is visualized in Fig. 19b. The examined node has the ground truth label "error" because it has a missing connection. Therefore, the explanation cannot point in the direction of any wrongly connected nodes, because there exists no connection to them. The category occurs frequently.

4.3.3. Discussion

The presented qualitative analysis provides insights into the strengths and limitations of explanations for false positives and false negatives in the error identification task. While not every explanation category leads directly to actionable insights, even partial indications offer value. These partial indications, such as narrowing down the search space, can significantly reduce the labor- and time-intensive validation work for grid operators. For example, helpful explanations for false positives, such as those in FP-Category 1 and FP-Category 2, demonstrate that TEIGR captures relevant patterns in the grid topology, even if the predictions are incorrect. Similarly, FN-Category 1 highlights how explanations can guide grid operators toward a subset

of candidate nodes, even if they do not reveal the exact error. Such explanations can help grid operators to better understand TEIGR and increase its trustworthiness, even if the predictions are wrong.

Future work could quantitatively investigate the frequency of the categories and see if these results are transferable to other grid topologies. Future work could also involve working together with grid operators to find out how they would use the explanations and how the explanations could be further improved. In Chapter 5, we present another idea for future work addressing "missing connection" errors.

4.4. Explainability of Correctly Classified versus Misclassified Nodes

In the previous section, we analyzed the explanations qualitatively. Here, we will compare them quantitatively. In a preparatory study [48], we compared different explainer algorithms for TEIGR regarding the explainer metrics characterization score and fidelity. This investigation led to the question how these methods perform if the metrics are calculated separately on correctly classified and misclassified nodes. This question is addressed below.

4.4.1. Characterization Score Analysis

In Fig. 20, the characterization scores for eight explainer algorithms calculated separately for correctly and misclassified nodes are visualized. The explainer algorithms are sorted according to their overall characterization score (grey bars in Fig. 20). The proportion of correctly classified nodes is significantly higher than that of misclassified nodes. The ratio is approximately 60:1. This explains why, for example, for the explainer Saliency the bars for "all nodes" and "correctly classified nodes" are approximately the same, although that of "misclassified nodes" is significantly lower. The lower number of misclassified nodes is also the reason the bars for misclassified nodes show larger error bars. Interestingly, explainer algorithms which have a lower score overall tend to have surprisingly high scores for the misclassified nodes. This observation contrasts with the results of Amara et al. [60], who found a drop of characterization scores for misclassified nodes in their experiments on five real benchmark datasets. Their reason for this is that when explaining correct predictions, the explanations target the true label, thereby explaining both the underlying phenomenon and the GNN model. In contrast, when explaining incorrect predictions, the GNN's

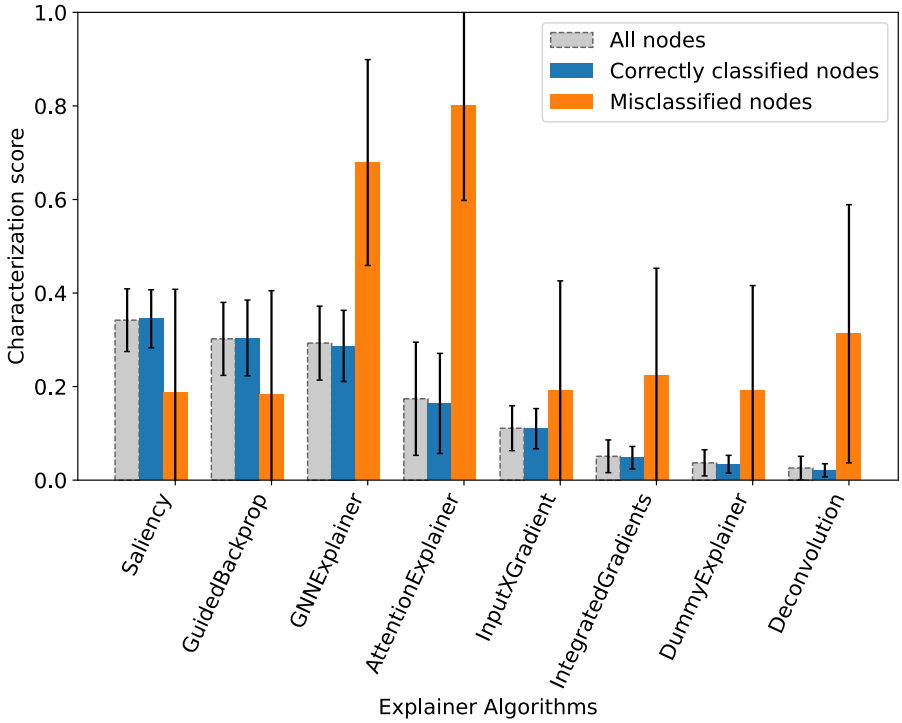


Figure 20: Characterization scores for different explainer algorithms divided into correctly classified and misclassified nodes, represented by blue and orange bars, respectively. The grey bars indicate the total characterization score, which corresponds to the average of the blue and orange bars weighted by the sample size. The standard deviation of the explainer algorithms across 100 snapshots of varying feature values of the grid "Spaltenstein" is visualized as error bars.

outputs diverge from the true label, allowing the explainer algorithms to gain insights only into the GNN's reasoning process [60].

4.4.2. Fidelity Analysis

To investigate this behavior further, we analyze the characterization score results in more detail by breaking them down into the fidelity values. The characterization score is the weighted harmonic mean of fidelity+ and $1 -$ fidelity$-$ (see Eq. 25). In Fig. 21 the fidelity+ and $1 -$ fidelity$-$ scores and their error bars for the eight explainer algorithms are visualized. The markers show the differentiation between correctly

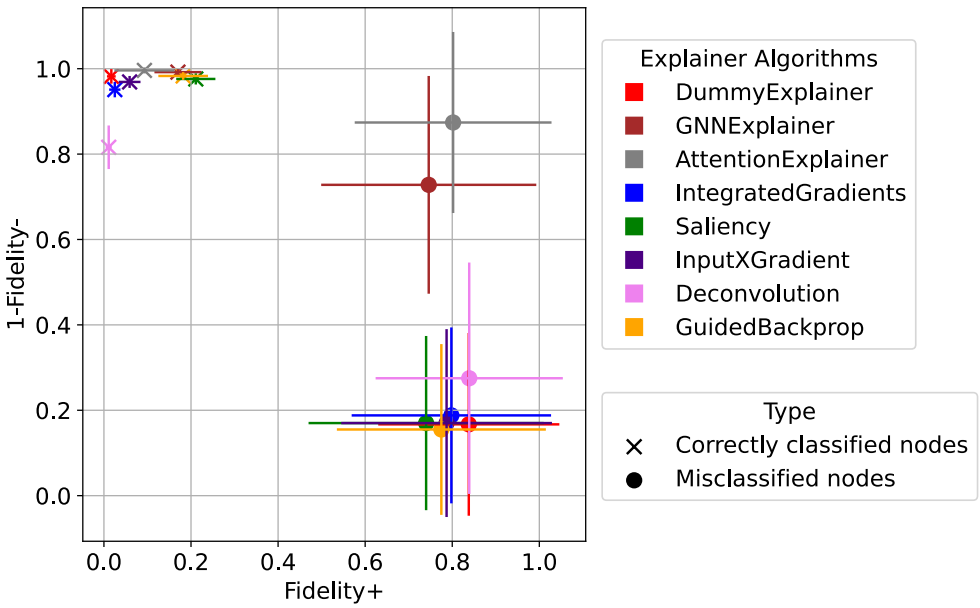


Figure 21: Fidelity scores for different explainer algorithms divided into correctly classified and misclassified nodes. The standard deviation of the explainer algorithms across 100 snapshots of varying feature values of the grid "Spaltenstein" is visualized as error bars.

classified (marker $\times$) and misclassified nodes (marker $\bullet$). To provide a more intuitive visualization and ensure comparability with the similarly structured overview study from Amara et al. [60], we plot $1 - \text{fidelity}-$ on the y-axis instead of fidelity$-$. This adjustment ensures that higher values represent better results.

A significant difference between correctly classified and misclassified nodes can be seen for fidelity+. For misclassified nodes fidelity+ values are significantly higher. While for correctly classified nodes the value fluctuates approximately between 0.01 and 0.25, for misclassified nodes it is between 0.7 and 0.85. Better explanations have higher fidelity+ scores, which is one reason for the high characterization scores of misclassified nodes in Fig. 20. Fidelity+ describes the necessity of an explanation and punishes larger explanations. A reason for the difference between correctly and misclassified nodes regarding fidelity+ could be that it is difficult to find a clear explanation for a correctly classified node. Most correctly classified nodes are inconspicuous and have ground truth class 0 (no error). The features and neighboring

nodes give a consistent picture without anomalies. This makes it hard to find a concise explanation. In the case of a misclassified or erroneous node, it is more likely that a smaller explanation can be found, for example in the form of a specific feature deviation. This could be a reason for the contrasting results of Amara et al. [60], who found a drop of characterization scores for misclassified nodes: They work with multi-class classification datasets such as the Citeseer, Cora and Pubmed citation network datasets. The task is to predict a label for a document based on citation links between documents and document features. In this case, there may also be a meaningful explanation for correctly classified nodes, e.g., a relevant linked document or a document topic indicating a specific label. In contrast, we work with the special use case of identifying errors, which means that most correctly classified nodes are inconspicuous and have the ground truth class 0 (no error). This different classification task could be the reason why we find that misclassified nodes can be explained better for some explainer algorithms, while Amera et al. find that misclassified nodes can be explained worse.

Another hypothesis for our difference between correctly and misclassified nodes regarding fidelity+ is the following: For misclassified nodes, the nodes, edges, and features identified by the explanation as "important" may not actually be important at all, as they caused TEIGR to make incorrect predictions. Fidelity+ is calculated by only providing the complement of the explanation $G_{C \setminus S}$ to the model. So for the calculation of fidelity+ these "important" nodes, which are part of the explanation subgraph, are removed. Removing them gives TEIGR another chance to predict without the "confusing" inputs that propably led to a wrong prediction. Since TEIGR is generally accurate, it is likely to classify the node correctly on this "second attempt". This results in a large difference between the original, wrong prediction based on the full computation graph and the new, probably correct prediction based on the complement of the explanation subgraph. A large difference between these two predictions corresponds to high fidelity+.

For fidelity− the picture is different. As can be seen in Fig. 21, correctly classified nodes achieve satisfactory results for all explainer algorithms. For misclassified nodes, only two explainer algorithms, GNNExplainer and AttentionExplainer, achieve satisfactory results. That explains the high bars for misclassified nodes for GNNExplainer and AttentionExplainer in Fig. 20. However, we cannot yet answer the question as to why these two explainer algorithms show satisfactory results regarding fidelity− and the others do not. This question could be the subject of future work.

Because misclassified nodes are relevant for grid operators to identify, it is desirable

that the explanations for this group are better. This is an argument in favor of using the GNNExplainer, as it has a high total value and performs notably well for misclassified nodes.

4.4.3. Confusion Matrix

The two most promising explainer algorithms regarding the total characterization score as well as the characterization score for misclassified nodes are Saliency (highest total characterization score) and GNNExplainer (best compromise between total and misclassified characterization score). We have examined these two in more detail in Fig. 22 by calculating the characterization scores for true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) separately. Another imbalance should be noted here: Not only is the number of correctly classified nodes significantly higher than the number of incorrectly classified nodes, the number of nodes with ground truth label 0 (no error) is also significantly higher than those with label 1 (error). In Fig. 22, we specified the absolute number of nodes in each category in square brackets below the characterization scores.

The results for the two explainer algorithms, Saliency (Fig. 22 left) and GNN-Explainer (Fig. 22 right), differ significantly. For the category with the most samples, TN, Saliency achieves a higher characterization score. Due to the high number of TNs, this has a major influence on the total characterization score. That explains the higher total characterization score of Saliency compared to GNNExplainer. However, for GNNExplainer the other three categories, TP, FP, and FN, can be explained better compared to Saliency. These categories are more relevant for grid operators than TN. TP and FP are relevant for grid operators because action is potentially needed for "positive", erroneous nodes with ground truth label 1. FP are especially important as incorrectly classified nodes need to be analyzed in more detail. FN are crucial because they represent instances where TEIGR fails to identify actual errors in the grid topology. For grid operators, these undetected errors pose significant risks, as they may lead to unaddressed faults. Unlike FP, which can be analyzed and verified in detail, FN remain invisible to the grid operators, making them impossible to address without external intervention or alternative detection methods. Because GNNExplainer performs significantly better in all three of these categories, GNNExplainer might be best suited for grid operators in practice.

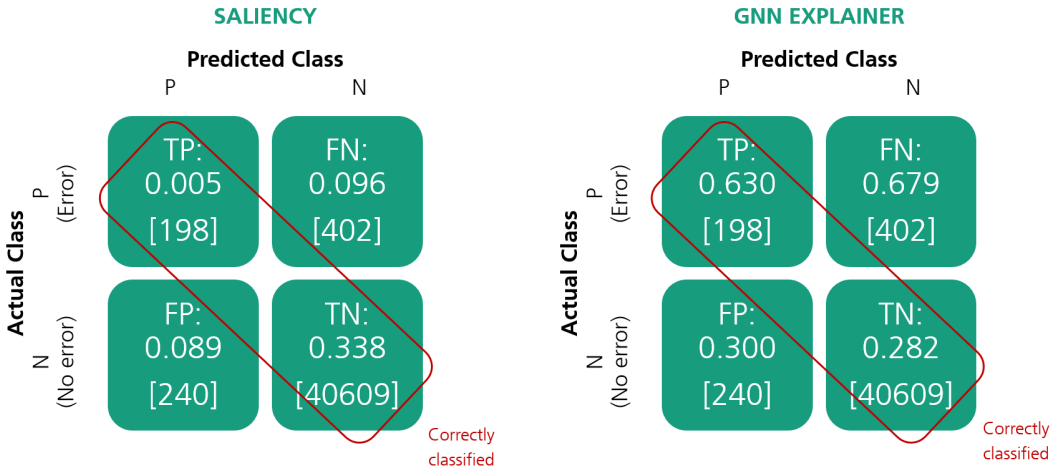


Figure 22: Characterization scores for the explainer algorithms Saliency and GNN-Explainer divided into TP, FN, FP and TN. In square brackets is the absolute number of samples in each category.

4.5. Enhancing the Loss Function of TEIGR with Explainability Term

In the last sections, we have analyzed the explainability of TEIGR. Now we want to improve TEIGR with the aim of increasing its classification performance and explainability. We propose adding an explainability term to the loss function of the model training. In the following, we present the motivation, the implementation and the results thereof.

4.5.1. Motivation

Prediction performance is measured with the F1-score. Explainability is measured with the characterization score. The idea is to augment the loss function by the characterization score to encourage TEIGR to produce more interpretable outputs. This modification might also improve prediction performance, as the explainability term guides TEIGR to find solutions that are both accurate and easy to interpret. The explainability term is an added constraint which could improve the robustness of TEIGR by encouraging TEIGR to align its predictions with clear and interpretable patterns, such as fidelity metrics. This alignment could reduce TEIGRs dependence on spurious correlations or noise in the data, which often lead to overfitting or unstable predictions.

To this end, we introduce a parameter α that balances the trade-off between task performance and explainability. We propose the following adjusted loss function:

$$L_{\text{total}} = L_{\text{task}} + \alpha(1 - \text{charact}) = L_{\text{task}} + \alpha(1 - \frac{w_+ + w_-}{\frac{w_+}{\text{fid}_+} + \frac{w_-}{1-\text{fid}_-}}). \tag{26}$$

The total loss, L_{total}, combines the primary task loss, L_{task}, and the complement of the explainability score, charact, weighted by $\alpha \in [-1, 1]$. The L_{task} is the base loss of TEIGR, in our case the Binary Cross Entropy.

Higher characterization scores indicate better explanations. By using the complement of the characterization score, poorer explanations have a stronger influence on the loss. We allow positive and negative values for α. The interpretation of positive values ($0 < \alpha \leq 1$) is straight forward: we want to maximize explainability and therefore punish poor explainability. The idea behind negative values ($-1 \leq \alpha < 0$) is more complicated to understand. In the previous section 4.4, we found that for many explainer algorithms misclassified nodes have a higher characterization score. The implication here is that penalizing high characterization scores could tend to draw attention to misclassified nodes during training. We limit α to $\alpha \in [-1, 1]$ to ensure the primary task loss L_{task} has sufficient influence.

The characterization score weights $w_+, w_- \in [0, 1]$ satisfy by convention the condition $w_+ + w_- = 1$. In the extreme cases of $w_+ = 0$ or $w_- = 0$, the model's behavior is guided by a single aspect of explainability. When $w_+ = 0$, charact depends entirely on the fidelity score fid_-. This means the model focuses only on sufficiency. Conversely, when $w_- = 0$, charact is determined solely by the fidelity score fid_+. This means the model focuses only on necessity.

In the following, we examine the effects of the adjusted loss function with an explainability term. We analyze the results regarding the model performance and the explainability for varying weight values.

4.5.2. Implementation

Computation of the Characterization Score

In machine learning, the term batch refers to a subset of the training data used in one iteration of model training. The batch size is a hyperparameter that defines the number of samples in a batch. Just like previous work, we used a batch size of 32 [8]. In every training iteration, we compute the characterization score for every training sample in the batch. Then we average across all characterization scores of the batch.

This averaged characterization score is used in the loss function.

Learnable Parameters

We learned the weights w_+, w_- and α during training based on gradient descent. Gradient descent is an optimization algorithm that iteratively adjusts the model's parameters in the direction of the negative gradient of the loss function to minimize it. We used the initialization values $w_+ = 0.5$, $w_- = 0.5$ and $\alpha = 0$. Just like previous work [7], we used the optimizer RMSProp. RMSProp (Root Mean Squared Propagation) is an optimization algorithm based on gradient descent. RMSProp adjusts the learning rate based on a moving average of the squared gradients to improve training stability. Joint optimization of the loss weights potentially leads to better adaptation and model performance.

4.5.3. Results and Discussion

In the following, we present and discuss the results of adding an explainability term to the loss function of TEIGR.

Learning the parameters resulted in extreme values. The learned value of α was 1. The characterization score weights w_+ and w_- converged at 1 and 0, respectively.

The results of the model performance, measured with the F1-score, can be seen in Fig. 23. To gain further insights, we also examined intermediate parameter values. For the characterization score weights, we analyzed three scenarios. First, the learned values $w_+ = 1$, $w_- = 0$. Second, the opposite values $w_+ = 0$, $w_- = 1$. Third, a balanced characterization score with $w_+ = w_- = 0.5$, which is a typical choice for the weights and was used in a former project [48]. For each of these three characterization score weight characteristics, we trained six models with different values for α. For comparison, we plotted as baseline the F1-score for an original model with an unmodified loss function without explainability term.

The F1-score for the model with the learned parameter configuration ($w_+ = 1$, $w_- = 0$, $\alpha = 1$) is with 0.6447 slightly superior to the F1-score for the baseline model with 0.6424. However, the new model is still within the standard deviation of the baseline. This applies to all parameter configurations. None of them show clear model improvements. Therefore, the enhancement of the models' loss function with an explainability term appears to have limited effectiveness in improving model performance. The hypothesis that penalizing high characterization scores could tend to draw attention to misclassified nodes during training has not shown the expected

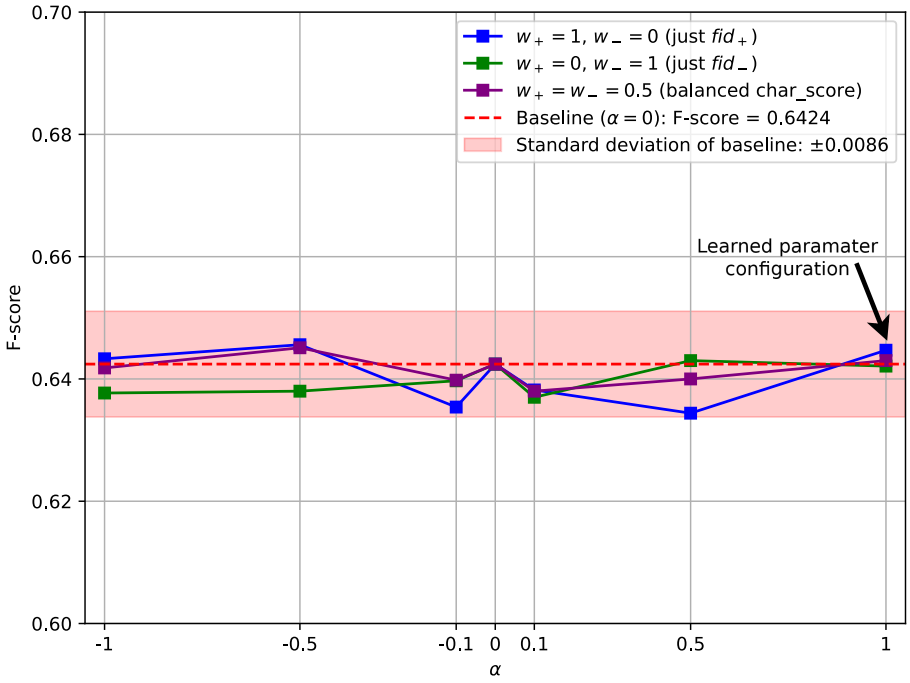


Figure 23: Model performance for modified loss function for different weight parameters α, w_+, and w_- as well as a baseline.

results so far. However, it is possible that alternative configurations could yield different outcomes.

Explainability Results

In the following, we analyze the influence of the loss enhancement of TEIGR on the explainability. Therefore, we plot in Fig. 24 the characterization score for different α. For the sake of clarity, we limit the plot to models with the balanced weight parameters $w_+ = w_- = 0.5$ (purple curve in Fig. 23) and show the error bars. The model with $\alpha = 1$ stands out: its characterization score is with 0.65 significantly higher than the others. This result can be explained by the fact that we penalize poor explainability for high alpha through the modified loss function, thereby maximizing explainability. For $\alpha < 0$ the modified loss function does the opposite: it penalizes high explainability. This explains the low char scores for $\alpha = -0.5$ and $\alpha = -1$. The model with $\alpha = 0.5$ does not fall into this pattern. Here we would have expected

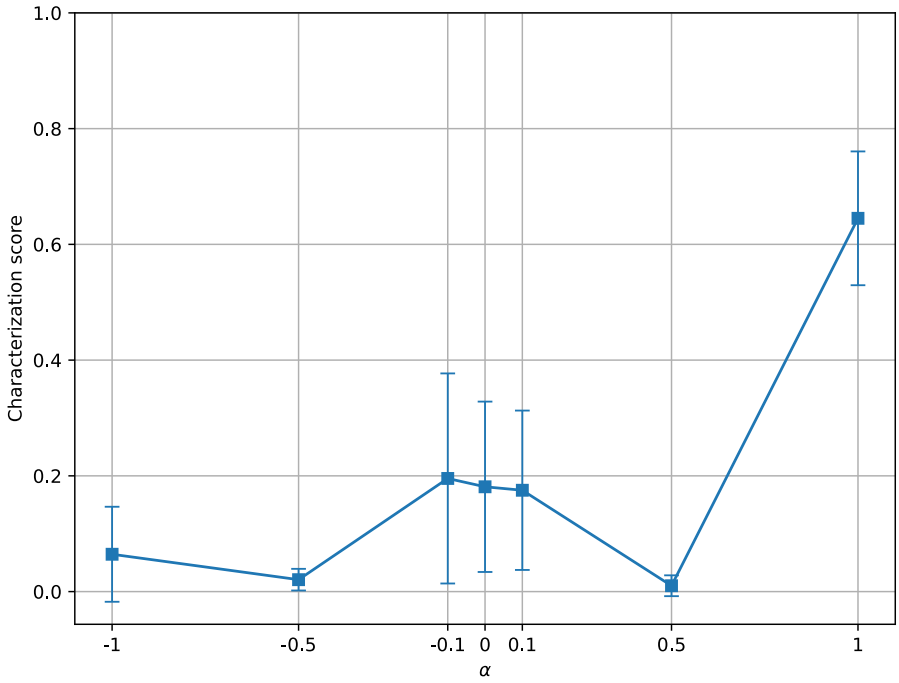


Figure 24: Influence of models' loss enhancement on explainability.

a characterization score between those of the baseline model (0.18) and those of the best model with $\alpha = 1$ (0.65), i.e., a characterization score roughly between 0.2 and 0.6. We cannot explain this outlier of a characterization score of 0.01. This behavior should be verified and analyzed in more detail in future experiments with other models and data sets.

Computational Complexity

Although the explainability for a modified loss function with an explainability term weighted with $\alpha = 1$ improves significantly, there is a disadvantage. The computation time for the modified loss function is significantly higher than for the original one. The reason for this is the computationally intensive calculation of the characterization scores. In the following, we analyze the computational complexity in more detail for the normal loss and the modified loss with the explainability term.

For the normal loss, we must perform a forward pass and the loss calculation. Let

T_{forward} be the complexity for a GNN forward pass on a graph with N nodes. The loss calculation involves a comparison for each node, which has a complexity of $O(N)$ [71], assuming that the small and constant number of layers, node degree and node features is negligible in our case. Thus, the computational complexity of the normal loss is:

$$T_{\text{loss\_normal}} = T_{\text{forward}} = O(N). \tag{27}$$

For the modified loss, we must compute in addition to the normal loss the explainability term consisting of the characterization score. First, an explanation in form of an explainability subgraph must be computed. The computational complexity for computing an explanation depends on the explainer algorithm used. We assume the use of the explainer algorithm Saliency, which computes the gradient of the model's output with respect to the node features. According to Blakely et al. [71], the forward pass and the backward pass of a GNN have the same computational complexity. Thus, we assume a computational complexity of $O(N)$ for computing the explanation. After the explanation, the characterization score for every node must be calculated. For calculating the characterization score, the fidelity$_-$ and fidelity$_+$ are needed. For calculating the fidelity$_-$, for each node i, $\hat{y}_i^{G_S}$ requires computing a prediction based on the explanation subgraph G_S. For calculating the fidelity$_+$, for each node i, $\hat{y}_i^{G_{C\setminus S}}$ requires a prediction based on the complement of the explanation subgraph $G_{C\setminus S}$. This leads to a complexity of $O(N)$ for calculating the characterization score for one node, i.e. $O(N^2)$ for N nodes. Together with the computation of the explanation subgraphs, we have a total complexity of $O(N + N^2) = O(N^2)$ for calculation the explainability term of the modified loss function. That gives us:

$$\begin{aligned} T_{\text{loss\_modified}} &= T_{\text{loss\_normal}} + T_{\text{explainability\_term}} \\ &= O(N) + O(N^2) \\ &= O(N^2). \end{aligned} \tag{28}$$

In our experiments, the computation time of the modified loss is larger by a factor of approximately 10 compared to the normal loss. This factor is highly dependent on the model size because of the quadratic scaling in N. Therefore, due to the large computation time, the modified loss is just feasible for small grids because of the quadratic scaling in N. Grid operators must weigh the advantage of better explainability against the longer training time of the model. Fortunately, TEIGR does not have to be trained in real time, so a longer training time could be accepted.

5. Future Work

In addition to the minor improvement ideas from the previous approaches that we have already mentioned in the respective places, the following ideas could be interesting for future work.

Addressing "Missing Connection" Errors

While categorizing the explanations for false negatives, we realized that the explanation is often not helpful, because the error type is a "missing connection". In this case, the explanation cannot point to any incorrectly connected nodes, as there is simply no connection to them. Revising TEIGR to allow access not only to neighboring nodes, but also to those in spatial proximity, could fix this problem. To avoid potentially impairing TEIGRs ability to detect other error types, this extension could be implemented as a second step after the initial prediction phase. Developing and integrating this functionality would likely require a few months, considering the need for model adaptation and extensive testing.

Sensitivity Analysis to Enhance Explainability

For further improving the explainability of TEIGR, a sensitivity analysis of the effect of model architecture and training on the explainability could be performed. For this purpose, the characterization scores for models with various hyperparameters could be measured. For example, the number of layers, the number of epochs for the training or the non-linearity function of the model could be varied. The implementation time of this approach would be short, but a large number of models would have to be trained. The necessary time to realize that would be approximately two to three weeks.

Enhance Loss Function by Fine-tuning Pre-trained Model

In this work, we enhanced the loss function of TEIGR by learning TEIGR with a modified loss function from scratch. However, learning a model from scratch requires

a large amount of computational resources. An alternative approach would be to start with a TEIGR model that has already been trained with the base loss and fine-tune it by additional training with the modified loss. For example, TEIGR could be pre-trained for 1000 epochs and then fine-tuned for another 1000 epochs. Using an already pre-trained model and fine-tuning it with the modified loss can be a way to reduce training time. Implementing this approach would likely take several weeks, considering the need for monitoring and adjusting the training process through metrics and visualizations to identify and eliminate potential issues.

Alternative Approaches to Enhancing the Loss Function

In this work, we have extended the loss function by an explainability term consisting of the explanation metric characterization score. An alternative approach to integrating the characterization score in the loss function could be to adjust the loss by including other explainability goals, such as sparsity or consistency. Sparsity regularization encourages TEIGR to use a smaller number of nodes. Focusing on only a few important nodes leads to simpler and more interpretable explanations. For the implementation one would have to add a regularization term to the loss function that minimizes the number of utilized nodes. This can be achieved using L1 regularization:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{sparsity}} \cdot \sum_{i=0}^{N} |w_i|, \tag{29}$$

where w_i represents the weight of the node, e.g., explanation importance, and $\lambda_{\text{sparsity}}$ is a hyperparameter controlling the strength of the regularization.

Consistency regularization ensures that similar inputs produce similar explanations. This regularization enhances the robustness and reliability of the explanations. For the implementation one could add a regularization term to the loss that promotes the consistency of explanations between similar inputs:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{consistency}} \cdot \sum_{i,j} \text{sim}(x_i, x_j) \cdot \text{dist}(E_i, E_j), \tag{30}$$

where $\text{sim}(x_i, x_j)$ measures the similarity between two inputs, and $\text{dist}(E_i, E_j)$ measures the difference in the explanations for these inputs. $\lambda_{\text{consistency}}$ is a hyperparameter controlling the consistency regularization. The selection of suitable hyperparameter values for $\lambda_{\text{sparsity}}$ and $\lambda_{\text{consistency}}$ requires careful tuning and validation. A challenge would be the increased computation time. Implementing and testing these two

approaches for enhancing the loss would likely require several weeks.

Adapting TEIGRs Usage Based on High Characterization Scores

Since misclassified nodes often have high characterization scores, we could adapt the use of TEIGR instead of refining the training process. For example, if the characterization score is above a certain threshold, we could generate more predictions with slightly modified graphs to increase the model's robustness. Another idea would simply be to give the grid operators a corresponding notice in such cases. This could probably be implemented in one to two weeks.

Advanced Confidence Calibration Methods

In the field of confidence calibration, it might be interesting to explore advanced calibration methods. For example, hybrid approaches combining the advantages of histogram binning and temperature scaling like class based temperature scaling [72] or the scaling-binning calibrator [73] could be investigated. Also, calibration methods that take the graph structure into account like ratio-binned scaling [19] could deliver interesting results. Depending on the used calibration method, implementation and analysis would likely require a few weeks. Since TEIGR is already well calibrated, this improvement is not as important as the ones previously mentioned.

6. Conclusion

In this Thesis, we investigated different methods to overcome the problem of missing trustworthiness and explainability in the Topology Error Identification Graph Neural Network. The methods included confidence calibration, model visualization, and explainable AI.

First, we examined the calibration of TEIGR, because a well calibrated model is the basis for other methods. In general, TEIGR returns well calibrated predictions. TEIGR tends to be slightly under-confident. We adjusted the calibration by applying the calibration methods histogram binning and temperature scaling. Histogram binning provides a local, discrete correction that improves the visual reliability curve but increases ECE. Temperature scaling applies a global, smooth correction, reducing overall calibration errors but leaving the reliability curve unchanged. Depending on the goals of the calibration, both could be beneficial for grid operators improving the trustworthiness of TEIGR and indicating how reliable the predictions of TEIGR are.

We visualized the internal representations of TEIGR using dimension reduction techniques, providing insights into the reasoning of TEIGR. The visualizations revealed clusters based on topographical proximity and node labels. By analyzing feature influences through various color schemes, we confirmed the significance of features that capture the difference between measured and expected voltage magnitude and angle. Additionally, the visualizations demonstrated how the representation improves through model training. For an untrained version of TEIGR, the visualization does not distinguish between correctly and incorrectly connected nodes. These visualizations enhanced our understanding and confidence in TEIGR.

After focusing on TEIGR as such, we next made individual predictions of TEIGR explainable to users. We developed a visualization that integrates the explanation subgraph into the original grid topology, making the explanation intuitively clear for grid operators. We provided an example explanation and demonstrated its practical benefits for grid operators. Additionally, we presented a categorization of explanations for false positive and false negative model predictions. By analyzing these explanations, we identified both helpful and unhelpful categories. For false

positives, the most frequent and useful explanations were those where the real error was either a direct or close neighbor node. On the other hand, false negatives were often explained by highlighting incorrect or irrelevant nodes, with missing connections being a particularly challenging case.

After we analyzed the explanations qualitatively, we compared quantitatively the explainability of correctly classified and misclassified nodes. We found that misclassified nodes can be explained well. Especially the fidelity+ values are high for misclassified nodes. Explainer algorithms with a lower total characterization score tend to have high characterization scores for misclassified nodes. The explainer algorithm GNNExplainer has a high total characterization score as well as a high characterization score for misclassified nodes. Therefore, GNNExplainer might be best suited for grid operators in practice.

Finally, after analyzing the explainability of TEIGR, we improved the explainability of TEIGR itself. Threrefore, we enhanced the loss function of TEIGR with an explainability term. The performance of TEIGR could not be improved, but the explainability, measured by the characterization score, improved for certain parameter values. However, the computation time increased. Therefore, the loss enhancement is just feasible for small grids. Increasing the interpretability without harming the performance of TEIGR is helpful for ensuring that TEIGRs decisions are both understandable and reliable in practical applications.

Overall, we improved the trustworthiness and explainability in TEIGR with confidence calibration, model visualization, and explainable AI. Our work is an important step towards valid digital twins as basis for advanced grid planning and operation in the energy transition.

Bibliography

[1] A. Varbella, K. Amara, B. Gjorgiev, M. El-Assady, and G. Sansavini, "Powergraph: A power grid benchmark dataset for graph neural networks," in *Proceedings of the 38th Conference on Neural Information Processing Systems*, 2024.

[2] Z. Zhaoyun and L. Linjun, "Application status and prospects of digital twin technology in distribution grid," *Energy Reports*, vol. 8, pp. 14170–14182, 2022.

[3] D. Deka, V. Kekatos, and G. Cavraro, "Learning distribution grid topologies: A tutorial," *IEEE Transactions on Smart Grid*, vol. 15, no. 1, pp. 999–1013, 2023.

[4] N. Susila, A. Sruthi, and S. Usha, "Impact of cloud security in digital twin," in *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, vol. 117, pp. 247–263, Elsevier, 2020.

[5] Y. Weng, Y. Liao, and R. Rajagopal, "Distributed energy resources topology identification via graphical modeling," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 2682–2694, 2017.

[6] W. L. Hamilton, *Graph Representation Learning*, vol. 14 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2020.

[7] J. Nolde, "Topology error identification in electrical distribution grids with graph neural networks," master thesis, Furtwangen University, 2023.

[8] C. Wanninger, "A grid operator's dream: Error detection with GNNs," master project, Albert-Ludwigs-University Freiburg, 2024.

[9] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782–5799, 2022.

[10] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, pp. 1321–1330, PMLR, 2017.

[11] A. Kumar, S. Sarawagi, and U. Jain, "Trainable calibration measures for neural networks from kernel mean embeddings," in *International Conference on Machine Learning*, pp. 2805–2814, PMLR, 2018.

[12] J. Zhang, B. Kailkhura, and T. Y.-J. Han, "Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning," in *International Conference on Machine Learning*, pp. 11117–11128, PMLR, 2020.

[13] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of Educational Psychology*, vol. 24, pp. 498–520, 1933.

[14] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[15] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," 2016. Preprint arXiv. `https://doi.org/10.48550/arXiv.1602.04938`.

[17] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–42, 2018.

[18] A. Longa, S. Azzolin, G. Santin, G. Cencetti, P. Liò, B. Lepri, and A. Passerini, "Explaining the explainers in graph neural networks: a comparative study," *ACM Computing Surveys*, 2024.

[19] T. Liu, Y. Liu, M. Hildebrandt, M. Joblin, H. Li, and V. Tresp, "On calibration of graph neural networks for node classification," in *International Joint Conference on Neural Networks*, 2022.

[20] O. R. Solheim, G. S. Presthus, B. A. Høverstad, and M. Korpås, "Visualizing graph neural networks in order to learn general concepts in power systems," *Electric Power Systems Research*, vol. 237, p. 110717, 2024.

[21] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, 1999.

[22] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[23] C. Tomani and F. Buettner, "Towards trustworthy predictions from deep neural networks with fast adversarial calibration," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[24] R. Muller, S. Kornblith, and G. Hinton, "When does label smoothing help?," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[25] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. S. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.

[26] N. Charoenphakdee, J. Vongkulbhisal, N. Chairatanakul, and M. Sugiyama, "On focal loss for class-posterior probability estimation: A theoretical perspective," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2021.

[27] L. Teixeira, B. Jalaian, and B. Ribeiro, "Are graph neural networks miscalibrated?," 2019. Preprint arXiv. `https://doi.org/10.48550/arXiv.1905.02296`.

[28] X. Wang, H. Liu, C. Shi, and C. Yang, "Be confident! towards trustworthy graph neural networks via confidence calibration," in *Proceedings of the 35th Conference on Neural Information Processing Systems*, 2021.

[29] M. Ferreira de Oliveira and H. Levkowitz, "From visual data exploration to visual data mining: a survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 378–394, 2003.

[30] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, p. 401 – 419, 1952.

[31] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. New York: Springer, 2007.

[32] J. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, 1969.

[33] P. Demartines and J. Herault, "Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 148–154, 1997.

[34] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Advances in Neural Information Processing Systems*, vol. 15, MIT Press, 2002.

[35] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016. Preprint arXiv. `https://doi.org/10.48550/arXiv.1607.00653`.

[36] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. 77, pp. 2539–2561, 2011.

[37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, 2019.

[38] W. Wu, Y. Su, X. Chen, S. Zhao, I. King, M. R. Lyu, and Y.-W. Tai, "Towards global explanations of convolutional neural networks with concept attribution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[39] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV: 13th European Conference*, pp. 818–833, Springer, 2014.

[40] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[41] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," in *International Conference on Machine Learning*, PMLR, 2019.

[42] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proceedings of the International Conference on Learning Representations*, 2015.

[43] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[44] R. Machlev, L. Heistrene, M. Perl, K. Levy, J. Belikov, S. Mannor, and Y. Levron, "Explainable artificial intelligence (xai) techniques for energy and power systems: Review, challenges and opportunities," *Energy and AI*, vol. 9, p. 100169, 2022.

[45] T. Brown, J. Horsch, and D. Schlachtberger, "Pypsa: Python for power system analysis," *Journal of Open Research Software*, vol. 6, no. 3, 2018.

[46] H. Li, J. Wert, A. Birchfield, T. Overbye, T. Gomez, C. Mateo, F. Postigo Marcos, P. Martinez, T. Elgindy, and B. Palmintier, "Building highly detailed synthetic electric grid data sets for combined transmission and distribution systems," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 478–488, 2020.

[47] G. Pisano, N. Chowdhury, M. Coppo, N. Natale, G. Petretto, G. G. Soma, R. Turri, and F. Pilo, "Synthetic models of distribution networks based on open data and georeferenced information," *Energies*, vol. 12, p. 4500, 2019.

[48] C. Hartmann, "Cracking the black box of graph neural networks for electrical grid validation," master project, Albert-Ludwigs-University Freiburg, 2024.

[49] D. Fischer, A. Härtl, and B. Wille-Haussmann, "Model for electric load profiles with high time resolution for german households," *Energy and Buildings*, vol. 92, pp. 170–179, 2015.

[50] M. Ringsquandl, H. Sellami, M. Hildebrandt, D. Beyer, S. Henselmeyer, S. Weber, and M. Joblin, "Power to the relational inductive bias: Graph neural networks in electrical power grids," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, p. 1538–1547, ACM, 2021.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[52] A. H. Murphy and R. L. Winkler, "Reliability of subjective probability forecasts of precipitation and temperature," *Applied Statistics*, vol. 26, pp. 41–47, 1977.

[53] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *International Conference on Machine Learning*, vol. 1, 2001.

[54] M. Gallagher and T. Downs, "Visualization of learning in neural networks using principal component analysis," *IEEE Transactions on Systems*, vol. 33, no. 1, pp. 28–34, 2003.

[55] M. Greenacre, P. Groenen, T. Hastie, and et al., "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, p. 100, 2022.

[56] K. Amara, M. El-Assady, and R. Ying, "Ginx-eval: Towards in-distribution evaluation of graph neural network explanations," in *XAI in Action: Past, Present, and Future Applications*, 2023.

[57] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," 2018. Preprint arXiv. `https://doi.org/10.48550/arXiv.1806.07421`.

[58] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van Keulen, and C. Seifert, "From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai," *ACM Computing Surveys*, vol. 55, no. 13s, 2023.

[59] A. Jacovi and Y. Goldberg, "Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness?," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[60] K. Amara, Z. Ying, Z. Zhang, Z. Han, Y. Zhao, Y. Shan, U. Brandes, S. Schemm, and C. Zhang, "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks," in *Learning on Graphs Conference*, pp. 44–1, PMLR, 2022.

[61] P. Li, Y. Yang, M. Pagnucco, and Y. Song, "Explainability in graph neural networks: An experimental survey," 2022. Preprint arXiv. `https://doi.org/10.48550/arXiv.2203.09258`.

[62] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," 2020. Preprint arXiv. `https://doi.org/10.48550/arXiv.2006.11371`.

[63] I. E. Nielsen, D. Dera, G. Rasool, R. P. Ramachandran, and N. C. Bouaynaya, "Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks," *IEEE Signal Processing Magazine*, vol. 39, no. 4, p. 73–84, 2022.

[64] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: visualising image classification models and saliency maps," in *Proceedings of the International Conference on Learning Representations*, 2014.

[65] P. Sturmfels, S. Lundberg, and S.-I. Lee, "Visualizing the impact of feature attribution baselines," *Distill*, 2020. `https://doi.org/10.23915/distill.00022`.

[66] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks," in *Proceedings of the International Conference on Learning Representations*, no. 1711.06104, 2018.

[67] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*, pp. 3319–3328, PMLR, 2017.

[68] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, and A. Wiltschko, "Evaluating attribution for graph neural networks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 5898–5910, Curran Associates, Inc., 2020.

[69] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database." AT & T Labs, 2010. `http://yann.lecun.com/exdb/mnist`.

[70] R. Min, D. A. Stanley, Z. Yuan, A. Bonner, and Z. Zhang, "A deep non-linear feature mapping for large-margin knn classification," in *9th IEEE International Conference on Data Mining*, pp. 357–366, 2009.

[71] D. Blakely, J. Lanchantin, and Y. Qi, "Time and space complexity of graph convolutional networks," 2021. Preprint. `https://qdata.github.io/deep2Read/ /talks-mb2019/Derrick_201906_GCN_complexityAnalysis-writeup.pdf`.

[72] L. Frenkel and J. Goldberger, "Network calibration by class-based temperature scaling," in *29th European Signal Processing Conference*, pp. 1486–1490, 2021.

[73] A. Kumar, P. S. Liang, and T. Ma, "Verified uncertainty calibration," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

A. Data and Model

A.1. Grid Topologies and Their Properties

In Table 2 the grid topologies and their graph properties are presented. $|\mathcal{V}|$ and $|\mathcal{E}|$ represent the number of nodes respectively edges. The average node degree μ_{deg} is the average number of edges connected to a node in the graph. The diameter *dia* measures the number of edges in the shortest path between the two farthest nodes in the graph. The average shortest path length μ_{sp} refers to the average number of edges along the shortest paths between all pairs of nodes in the graph.

Table 2: Grid topology characteristics [7].

Grid	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	μ_{deg}	*dia*	μ_{sp}
Minimal (syn)	26	48	1.92	13	5.23
Spaltenstein (syn)	80	156	1.97	24	10.23
Eggenweiler (syn)	115	226	1.98	37	13.98
Oberraderach (syn)	282	560	1.99	43	17.65
Manzell Nord (syn)	349	694	1.99	45	18.30
E301 (real)	105	210	2.02	23	8.92
E212 (real)	266	538	2.03	36	15.44
E208 (real)	188	385	2.06	28	12.44

A.2. Data Splits for Training, Validation and Test

Table 3 shows the assignment of grid topologies to the training, validation, and test subsets. The eight grid topologies are divided in the three subsets so that their distributions of grid size and the ratio of real-to-synthetic grids are representative of the entire dataset. The training set includes four grid topologies, while the validation and test sets each consist of two topologies. Each topology corresponds to three scenarios for the years 2022, 2030, and 2040.

Table 3: Data splits for training, validation and test [7].

Training	Validation	Test
Spaltenstein (syn)	Oberraderach (syn)	Minimal (syn)
Eggenweiler (syn)	E208 (real)	E301 (real)
Manzell Nord (syn)		
E212 (real)		

A.3. Hyperparameter Configuration

In previous work [7], a hyperparameter study was performed to identify the optimal hyperparameters for TEIGR. The best-performing hyperparameter configuration is provided in the table below. This optimal configuration was used for TEIGR in this project. For a detailed explanation of the hyperparameters and the hyperparameter search, please refer to [7].

Table 4: Hyperparameter configuration.

Hyperparameter	Short Description	Value
K	Number of update layers	3
H_{number}	Number of attention heads	8
H_{width}	Width of the attention heads	16
δ_{dropout}	Dropout rate	0.0118
$\sigma(\cdot)$	Non-linear activation function	ReLU
B	Batch size	200
η	Learning rate	0.0026
$opt(\cdot)$	Optimizer	RMSProp

B. Complex Model Visualizations

B.1. Variation across Clusters

In section 4.2.2 we showed the dimension reduction visualizations for a synthetic grid with an erroneous line within one cluster. Here we examine the visualizations for a faulty line between two clusters. The corresponding grid with such a variation is shown in Fig. 25. The correct line between node 9 and 11 is replaced by the erroenous line between node 11 and 25. Node 9 and 11 belong to the upper right cluster, while node 25 belongs to the lower left cluster of the grid.

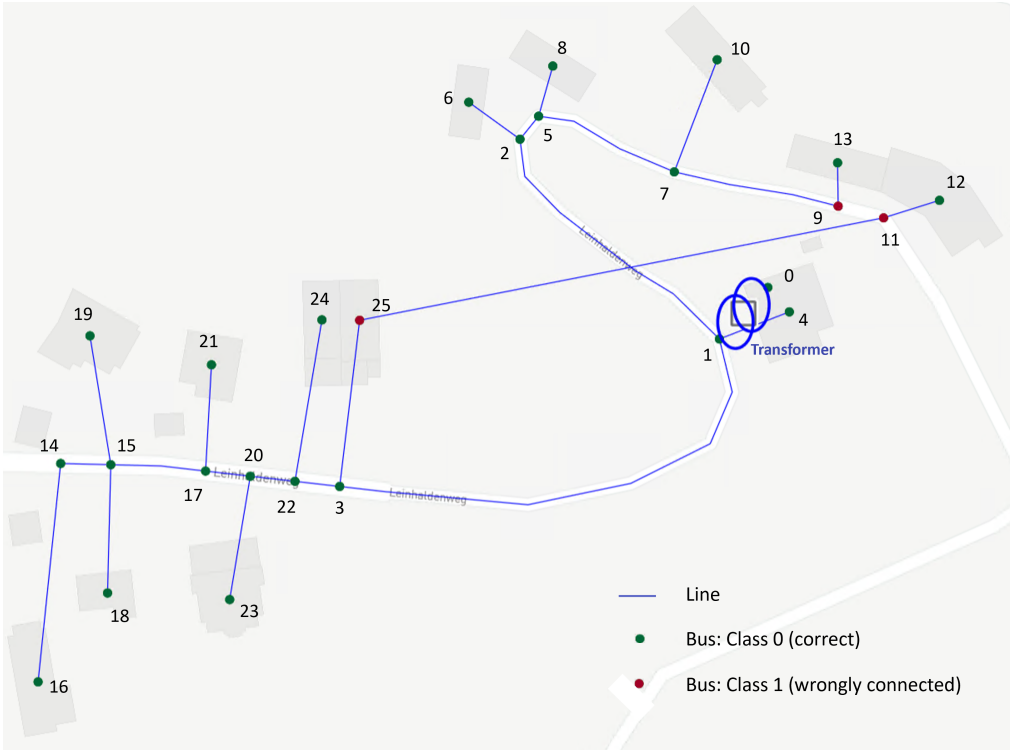
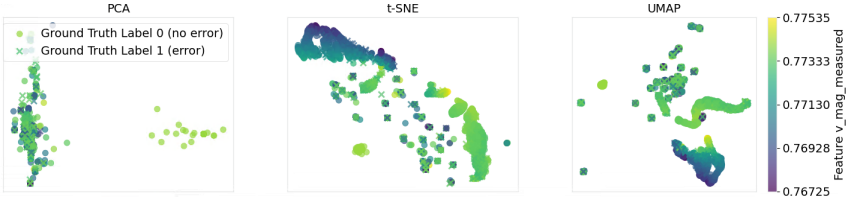
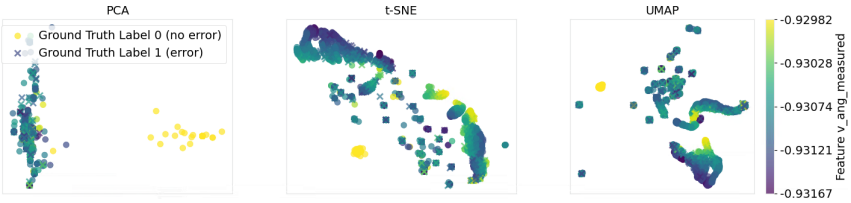


Figure 25: Grid topology of synthetic grid with erroneous line between the two clusters. The bus colors indicate the ground truth class. Red buses are wrongly connected.

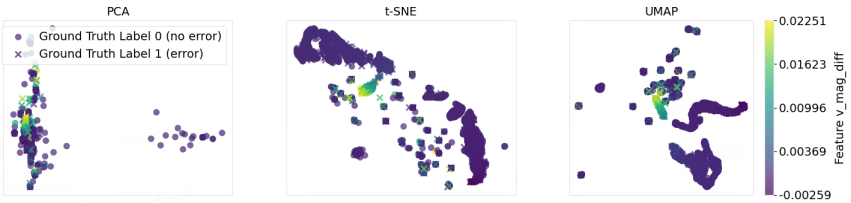
The visualizations of the error between clusters in Fig. 26 look similar to the visualizations of the original error within one cluster in Fig. 14. PCA still does not reveal any clustering besides the slack bus cluster. t-SNE and UMAP, on the other hand, return three clusters: two relating to the topographical proximity of the nodes and one containing erroneous nodes. This shows that the visualizations of TEIGR are stable across grid variations of diverse types.



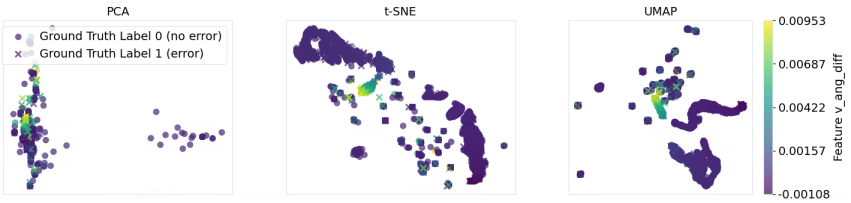
(a) Color by feature 1: Measured voltage magnitude.



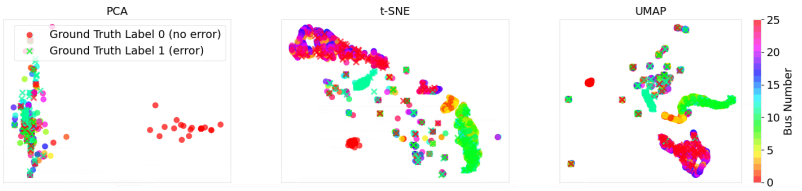
(b) Color by feature 2: Measured voltage angle.



(c) Color by feature 3: Voltage magnitude differences.



(d) Color by feature 4: Voltage angle differences.



(e) Color by node number.

Figure 26: Model visualizations with different colorings for grid with erroneous line between two clusters.

B.2. Visualization for Larger Grid

In section 4.2.2 we showed the dimension reduction visualizations for a small synthetic grid with just 26 nodes. Here we present the visualizations for the grid "Spaltenstein" with 80 nodes. The grid topology with a variation in the upper part of the grid can be found in Fig. 27. This grid can also be divided into two topological clusters at the transformer. These clusters as well as an "error"-cluster can also be found in the model visualization in Fig. 28. The visualizations in Fig. 28 show that the results are transferable to larger grids.

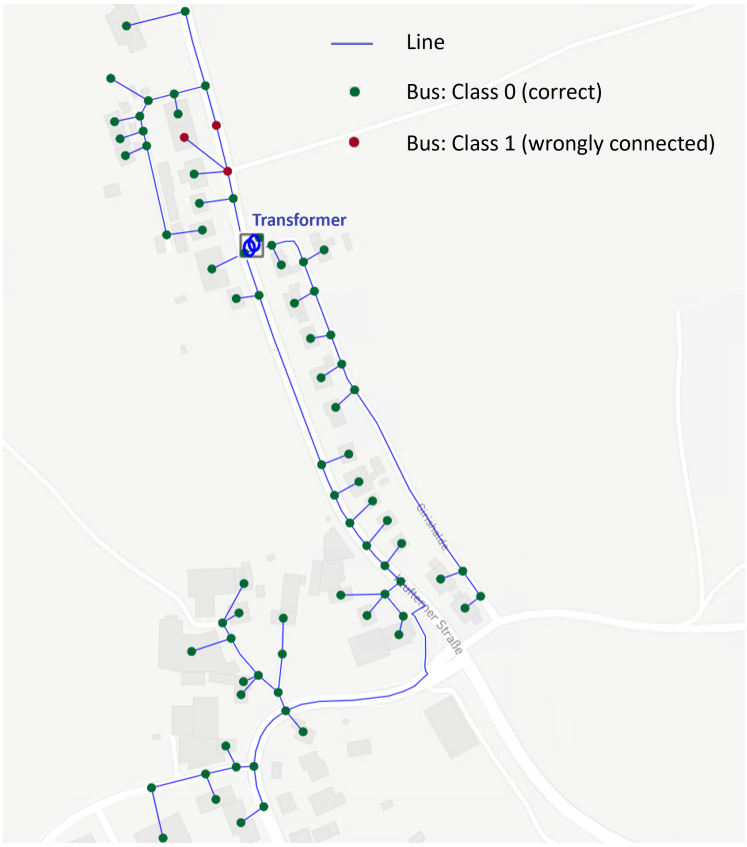
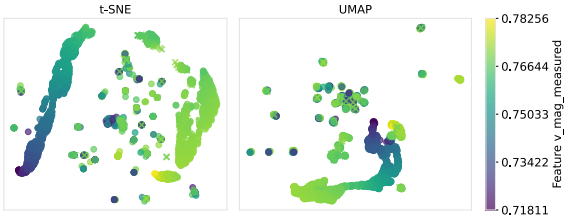
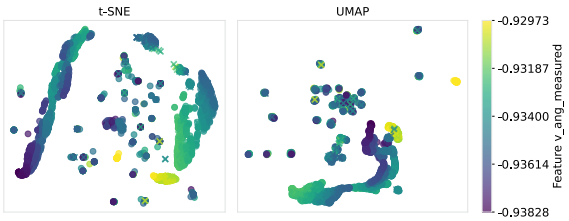


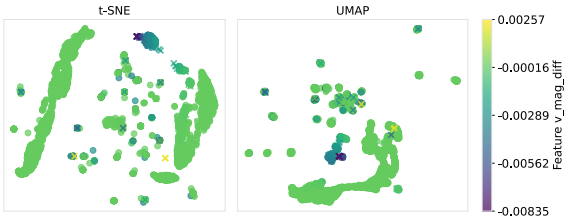
Figure 27: Grid topology of larger grid "Spaltenstein" with 80 nodes. The bus colors indicate the ground truth class. Red buses are wrongly connected.



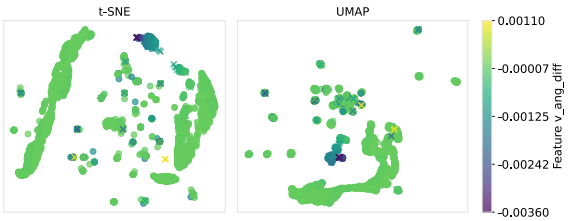
(a) Color by feature 1: Measured voltage magnitude.



(b) Color by feature 2: Measured voltage angle.



(c) Color by feature 3: Voltage magnitude differences.



(d) Color by feature 4: Voltage angle differences.

Figure 28: t-SNE and UMAP visualizations colored by feature values for the larger grid “Spaltenstein”.

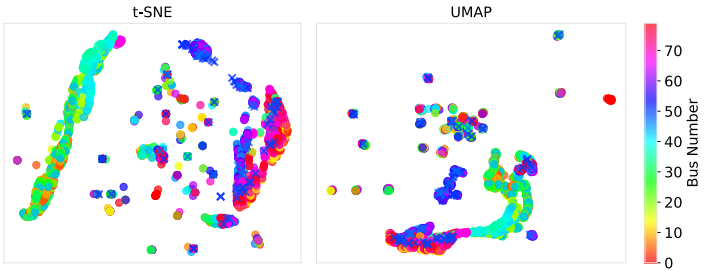


Figure 29: t-SNE and UMAP visualizations colored by bus number for the larger grid "Spaltenstein".