Counterfactual Explanation for Anomaly Detection using Graph Neural Network

Xiangyu Shi [a,b], **Abhishek Srinivasan** [a,b,*] **and Sepideh Pashami**[c, d]

[a]Scania CV AB, Vagnmakarvägen 1, Södertälje, Sweden
[b]Department of Computer Science, KTH University, Stockholm, Sweden.
[c]Halmstad University, Halmstad, Sweden
[d]RISE AB, Isafjordsgatan 28 A, Kista, Sweden
ORCID (Xiangyu Shi): https://orcid.org/0000-0002-0356-1941, ORCID (Abhishek Srinivasan): https://orcid.org/0000-0003-4178-5257, ORCID (Sepideh Pashami): https://orcid.org/0000-0003-3272-4145

Abstract.

In industrial settings, anomalies often indicate critical events such as equipment failures or system faults. These events are rare but highly impactful and require urgent attention and often have financial or safety consequences. Deep learning models, especially Graph Neural Networks (GNNs) have gained prominence due to their ability to capture intricate dependencies between sensor signals as graphs. Understanding the reasons behind the predicted anomalies is essential for effective response, however, the black-box nature of GNNs poses a significant challenge.

To address this limitation, we propose a counterfactual explanation framework that offers human-understandable insights by identifying minimal input changes capable of altering the model's decision. Our method employs a two-stage process: (i) selecting the most relevant nodes contributing to the anomaly using graphs, and (ii) generating counterfactual instances by perturbing only these selected nodes. We evaluate our approach on two real-world CPS datasets: SWaT and WADI. Experimental results show that our method produces significantly sparser explanations compared to existing techniques. Additionally, our ablation study shows using graph information for node selection helps in generating sparse explanations. These counterfactual insights enhance model transparency, support better operational decision-making, and ultimately foster greater trust in anomaly detection systems.

1 Introduction

In the age of cyber-physical systems (CPS), where physical processes are tightly integrated with computation and communication infrastructure, ensuring reliable and safe operation of these systems is of paramount importance. As these systems become increasingly complex, continuous monitoring of their health has emerged as a vital component of operational safety and performance optimization. One of the key techniques employed in this context is anomaly detection (AD), which involves identifying patterns in system behavior that deviate from expected norms. Accurate anomaly detection enables early fault diagnosis, minimizes downtime, and helps prevent catastrophic failures.

* Corresponding Author. Email: srini@kth.se .

Traditional anomaly detection methods encompass a wide range of statistical and machine learning techniques. These include clustering approaches such as k-means, and density estimation methods like One-Class SVM and Isolation Forests [4]. More recently, deep learning-based methods such as autoencoders, recurrent neural networks (RNNs), and variational autoencoders (VAEs) have been employed to model the normal behavior of time-series data and identify deviations [17]. While effective in many cases, these approaches generally operate under the assumption that sensor observations are independent or sequentially dependent, and they often fail to account for the structural inter-dependencies among sensors in a system.

Traditional anomaly detection methods often treat sensor observations independently or assume simplistic temporal dependencies, ignoring the inherent structural relationships between different sensing components. In many CPS applications—such as industrial automation, energy distribution networks, and autonomous vehicles—the behavior of a sensor is often influenced by the states of its neighboring sensors due to underlying physical or logical connections. Capturing these interactions is essential for robust modeling of system behavior. Graph-based representations provide a natural and expressive framework to encode these inter-sensor relationships. Recent advances in Graph Neural Networks (GNNs) have made it possible to effectively leverage graph-structured data for tasks like classification, prediction, and anomaly detection in multivariate time series data [8].

Several studies have demonstrated that modeling sensor dependencies through graph structures can significantly enhance the performance of anomaly detection systems in CPS settings [6, 16, 8]. Despite these promising results, a major limitation persists: the lack of explainability. GNN-based anomaly detection models are often treated as black boxes, offering little insight into why a particular anomaly was detected. This is particularly a problem in safety-critical domains, where human operators must understand and trust the decisions made by automated systems.

To bridge this gap, the machine learning community has increasingly focused on explainability, with methods generally categorized into local explanations—targeting individual predictions and global explanations—describing overall model behavior [13]. While several explanation techniques have been proposed for standard deep learning models, the explainability of GNNs, especially in time-series contexts, remains an underexplored area. Moreover, existing expla-

nation methods often rely on feature attribution or saliency maps, which may lack causal grounding and are limited in the types of counterfactual insights they can provide. Our primary focus is to explore whether graph structure in GNNs be harnessed for better explaining the model's decisions.

In this work, we propose a novel framework for counterfactual explanation tailored to GNN-based anomaly detection models operating on time-series sensor data. Counterfactual explanations aim to answer the question: "What minimal change to the input would alter the model's prediction?"—thus providing actionable and intuitive insights into model decisions. Counterfactual explanations can give a clue as to the root cause of the anomalies.

Our approach comprises a two-stage process. In the first stage, we identify the most influential sensors that contribute to an anomaly, along with their local graph neighborhoods. This localization step leverages node-level deviations and GNN attention mechanisms to pinpoint regions of the graph that are most responsible for the prediction. In the second stage, we generate counterfactual instances by perturbing sensor readings in a minimal and plausible manner, aiming to flip the model's prediction from anomalous to normal (or vice versa). These counterfactuals serve as transparent, case-specific explanations that can assist operators in understanding failure modes and potential corrective actions. By integrating such support-systems reduces cognitive load on the human decision-makers, while allowing them to effectively validate model outputs.

By combining the structural strengths of GNNs with the intuitive clarity of counterfactual reasoning, our method advances the state of the art in explainable anomaly detection for cyber-physical systems. On the SWaT and WADI benchmarks, our two-stage approach alters fewer than 6% of sensors, yet still delivers an outstanding sparsity-versus-proximity balance that makes the counterfactuals concise and actionable. This not only enhances trust and accountability but also opens new avenues for troubleshooting and diagnostics.

2 Related Work

2.1 Counterfactual Explanation for Time Series

Several recent studies have explored counterfactual explanation techniques for time series data, with the aim of explaining model decisions by identifying minimal changes in input features that would alter the model output.

For instance, Karlsson et al. 2020, propose a technique for generating counterfactuals using models like k-nearest neighbors and random shapelet forests. In another approach, Wang et al. 2021, focus on univariate time series by mapping data to a latent space, identifying counterfactuals there, and decoding them back to the input space. Native-Guide [5] identifies the nearest contrasting instance, extracts its most influential subsequence, and substitutes it into the original time series. CoMTE [2] selects alternative series from the training set to replace parts of the input in order to induce prediction changes. More recently, CFWoT [14] introduces a model-agnostic framework for both static and multivariate time series, capable of handling continuous and categorical features without needing access to training data or similar samples.

These approaches often do not focus on relational structures present in multivariate time series data, which is the focus of this work.

2.2 Counterfactual Explanation of Graph Neural Networks

Counterfactual explanation methods of graph neural networks aim to identify the smallest possible modifications to the input that would lead to a different model output. By pinpointing which features must be altered to change a prediction, these methods offer valuable insights into the model's decision boundaries and causal reasoning.

A representative method in this category is CF-GNNExplainer [10], which introduces a learnable binary mask over the model's computational graph to indicate edge presence or removal. The mask is optimized to (1) alter predictions (prediction loss) and (2) minimize structural changes (distance loss). The final explanation highlights edges with the highest importance scores from the learned mask.

Another thread of counterfactual explanation methods is to generate counterfactual instances that are close to the original instance but lead to a different prediction. CLEAR [11] employs a graph variational autoencoder (GVAE) to learn a latent representation of the input graph and generate counterfactual graphs by making minimal changes to the original structure or features. The GVAE is trained to reconstruct the original graph while ensuring that the generated counterfactual samples result in a different model prediction, maintaining both proximity (closeness to the original instance) and validity (changing the prediction). RCExplainer [3] uses a neural network that predicts the existence of an edge between two nodes based on their embeddings. To generate counterfactual explanations, RCExplainer modifies these pairwise node embeddings, effectively simulating the addition or removal of edges that lead to a change in the model's prediction. This approach allows for a structured and interpretable way of understanding which edges influence the decision of the GNN model.

However, these methods primarily focus on structural changes to the graph, such as edge addition or removal, rather than utilizing graph structures for time series data, which is the focus of our work. Our approach leverages the inherent relationships between sensors in a time series context, enabling us to generate counterfactual explanations that are both interpretable and relevant to the specific anomalies detected by GNN-based models.

3 Method

3.1 Problem Statement

This paper addresses the task of explaining anomalies in multivariate time series data through counterfactual explanation generation. To support this, we incorporate an initial anomaly detection component as a foundation.

We begin by employing an unsupervised time series anomaly detection model that learns the normal behavior of a system from historical data and detects deviations in unseen data. The input consists of multivariate sensor data V, where $|V| = N$ and N is the number of sensors. The training data is denoted as $\mathbf{s}\text{train} = [\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(T\text{train})}]$, where each $\mathbf{s}^{(t)} \in \mathbb{R}^N$ represents sensor readings at time t. The model assumes training data to be free of anomalies and captures normal system patterns to flag abnormal points in the test data.

The core focus of this work lies in generating counterfactual explanations for the data points identified as anomalous. The counterfactual explanation provides human-interpretable insights into the model's decision-making process by answering the question:

What minimal change would make an anomalous instance be considered normal? Formally, given a test data sequence $\mathbf{s}_{\text{test}} = [\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(T_{\text{test}})}]$ and a set of anomaly predictions, the goal is to generate, for each detected anomaly $\mathbf{s}_{\text{test}}^{(i)}$, a modified version $\mathbf{s}_{\text{test}}^{(i)'}$ such that the model classifies $\mathbf{s}_{\text{test}}^{(i)'}$ as normal, and $\text{stest}^{(i)'}$ remains as close as possible to $\mathbf{s}_{\text{test}}^{(i)}$ under a suitable distance metric.

3.2 Overview

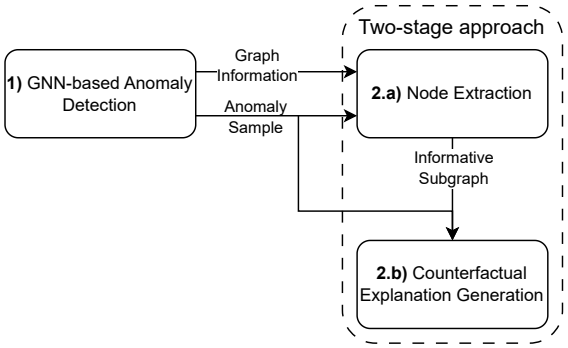

Figure 1. The overall framework of our approach. The framework consists of two primary modules: (1) a GNN-based model for time series anomaly detection, and (2) a two-stage approach for generating counterfactual explanations: (a) the node selector and (b) the counterfactual generator.

Figure 1 illustrates the overall framework of our proposed methodology. Our framework consists of two primary modules: (1) a graph neural network (GNN) architecture designed for time-series anomaly detection, and (2) a two-stage mechanism for producing counterfactual explanations. The GNN component processes the input time series sequences and produces binary classifications (normal versus anomalous) for individual temporal observations. Subsequently, the two-stage explanation module utilizes both the original time series input and the GNN's classification outcomes to construct counterfactual explanations specifically for data points identified as anomalous.

3.3 GNN-based Model for Time-Series Anomaly Detection

This section presents GNN-based model for time-series anomaly detection, which utilizes the methodology proposed by Deng and Hooi 2021. The model produces an anomaly score for time series data, labeling it as anomalous if its score exceed a specified threshold. Following the GDN architecture [6], the implementation integrates structural learning techniques with graph neural networks, comprising four interconnected modules: sensor embedding, graph structure learning, graph attention-based forecasting, and graph deviation scoring.

For each sensor i is represented by a trainable embedding vector $\mathbf{e}_i \in \mathbb{R}^d$, learned jointly with the forecasting objective. These embeddings capture the behavior patterns of the sensors and can be used to identify which sensors are similar to each other. Sensors that are highly correlated will have similar embedding vectors.

To explicitly represent inter-sensor relationships, we build a data-driven directed graph. For every pair of sensors embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$, we compute cosine similarity as:

$$\mathbf{A}'_{ij} = \frac{\mathbf{e}_i^T \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\|}, \tag{1}$$

and retain the top-k neighbors of each node to obtain the adjacency matrix $\mathbf{A}$. This resulting directed graph explicitly encodes dominant inter-sensor relationships, informing subsequent forecasting and anomaly scoring steps.

With the learned adjacency matrix $\mathbf{A}$, graph-attention layers process each time window $\mathbf{x}^{(t)} \in \mathbb{R}^{N \times w}$. For node i at time t, the hidden state is

$$\mathbf{h}_i^{(t)} = \text{ReLU}\left(\alpha_{i,i}\mathbf{W}\mathbf{x}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{W}\mathbf{x}_j^{(t)}\right), \tag{2}$$

where attention weights α_{ij} are softmax-normalized cosine similarities of concatenated node features.

A fully connected layer then maps sensor representations into predicted sensor values:

$$\hat{\mathbf{s}}^{(t)} = f_\theta\left(\left[\mathbf{e}_1 \cdot \mathbf{h}_1^{(t)}, \mathbf{e}_2 \cdot \mathbf{h}_2^{(t)}, \ldots, \mathbf{e}_N \cdot \mathbf{h}_N^{(t)}\right]\right), \tag{3}$$

where f_θ is a fully connected layer. The output $\hat{\mathbf{s}}^{(t)}$ is the predicted values of the sensors at time t. The model is trained using a mean squared error (MSE) loss function:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{T_{\text{train}} - w} \sum_{t=w+1}^{T_{\text{train}}} \|\hat{\mathbf{s}}^{(t)} - \mathbf{s}^{(t)}\|_2^2, \tag{4}$$

where T_{train} is the total number of training samples.

Deviations between predicted and actual values are calculated as the deviation score for each sensor $\text{Err}_i(t) = |\hat{\mathbf{s}}_i^{(t)} - \mathbf{s}_i^{(t)}|$, where $\hat{\mathbf{s}}_i^{(t)}$ is the predicted value and $\mathbf{s}_i^{(t)}$ is the actual value of sensor i at time t.

To ensure that all deviation scores are on the same scale, we normalize the deviation score as follows:

$$\text{AS}_i(t) = \frac{\text{Err}_i(t) - \tilde{\mu}_i}{\tilde{\sigma}_i}, \tag{5}$$

where $\tilde{\mu}_i$ and $\tilde{\sigma}_i$ are the median and inter-quartile range (IQR) of the deviation scores of sensor i over the training set, **as followed by [6]**.

The final anomaly score at time t is given by taking the maximum across all sensors:

$$\text{AS}(t) = \max_{i=1}^{N} AS_i(t), \tag{6}$$

where N is the number of sensors. The system is flagged as anomalous if the score exceeds a predefined threshold.

3.4 Two-stage Approach

Anomaly samples detected by the GNN-based anomaly detection model are fed into the two-stage approach for generating an explanation. The first stage involves node extraction, which identifies the most relevant sensors to guide the counterfactual explanation method. The second stage uses a counterfactual explanation method that generates counterfactual instances by altering only the sensors identified in the extracted node set from the first stage.

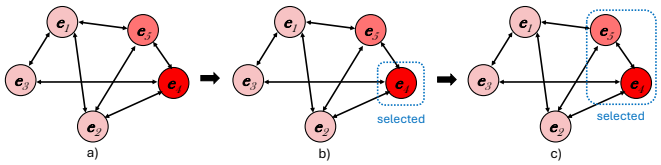

Figure 2. Node extraction module. This figure illustrates the process of selecting the most important sensors based on their anomaly scores. Sensors with darker red coloring indicate higher anomaly scores. With parameters $k_1 = 1$ and $k_2 = 1$, the 4th sensor is first selected (subfigure b), and then the 5th sensor is selected as it is connected to the 4th sensor (subfigure c). The final selected node set contains the 4th and 5th sensors.

3.4.1 Node Extraction

To generate counterfactual explanations focused on the most relevant sensors, we need to extract a node set containing only the most important sensors from the original graph. The node extraction module uses the anomaly score for each sensor to identify the most important sensors in the graph. The extraction process consists of three steps:

1. Select the top k_1 sensors with the highest anomaly scores, where k_1 is a hyperparameter that controls the size of the initial node set.
2. Select k_2 additional sensors that are connected to the selected k_1 sensors in the graph, where k_2 is a hyperparameter that controls the size of the extended node set.
3. Combine both sets of sensors to form the final set of selected sensors $S^{(t)}$.

Figure 2 illustrates this node extraction process. For a given time step t, we extract a node set based on anomaly scores. In the first step, we select the top k_1 sensors $S_1^{(t)}$ according to their anomaly scores $\text{AS}_i(t)$ from the sensor set V:

$$\mathbf{S}_1^{(t)} = \{i \in V \mid \text{rank}(\text{AS}_i(t)) \leq k_1\}, \tag{7}$$

where $\text{rank}(\cdot)$ ranks sensors by anomaly scores in descending order. This step selects sensors with the highest anomaly scores, as they are most likely to contribute to the detected anomaly and are therefore most relevant for generating counterfactual explanations.

In the second step, we select k_2 sensors $S_2^{(t)}$ that are connected to the selected k_1 sensors:

$$\begin{aligned} \mathcal{N}(S_1^{(t)}) &= \{j \in V \setminus S_1^{(t)} \mid \exists i \in S_1^{(t)} : A_{ij} = 1\}, \\ \mathbf{S}_2^{(t)} &\subseteq \mathcal{N}(S_1^{(t)}), \quad |\mathbf{S}_2^{(t)}| = k_2, \end{aligned} \tag{8}$$

where $\mathcal{N}(S_1^{(t)})$ represents the neighboring sensors of the selected k_1 sensors. Several strategies exist for selecting $\mathbf{S}_2^{(t)}$ from $\mathcal{N}(S_1^{(t)})$, including choosing sensors with the highest anomaly scores, those most connected to $S_1^{(t)}$, or random selection. We choose the top k_2 sensors with the highest anomaly scores as this provides a simple and effective way to select the most relevant sensors.

Finally, we combine both sets to form the final selected sensor set:

$$S^{(t)} = S_1^{(t)} \cup S_2^{(t)}. \tag{9}$$

The selected node set $S^{(t)}$ is then used as input to the counterfactual explanation method, which generates counterfactual instances by altering only the sensors in the extracted set.

3.4.2 Counterfactual Explanation Generation

The counterfactual explanation generation module creates counterfactual instances by altering the signals of the sensors in the extracted node set. We use a perturbation-based approach that generates counterfactual instances by adding small changes to the original signal.

We employ gradient optimization, a technique commonly used in adversarial attacks, to compute these perturbations effectively. The perturbation is found by minimizing the objective function $\mathcal{L}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta})$, where $\mathbf{x}$ is the original signal, and $\boldsymbol{\delta}$ is the perturbation. The objective function is defined as:

$$\mathcal{L}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}) = \mathcal{L}_{\text{CE}}(f(\mathbf{x} + \boldsymbol{\delta}), y_{\text{target}}) + \lambda \cdot \|\boldsymbol{\delta}\|, \tag{10}$$

where λ controls the trade-off between the two terms, $f(\cdot)$ is the model, y_{target} is the target class, and $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss. The first term pushes the model to produce a specific output (the target class), while the second term keeps the perturbation small.

The perturbation is computed using gradient descent:

$$\boldsymbol{\delta}^{(t+1)} = \boldsymbol{\delta}^t - \eta \nabla_{\boldsymbol{\delta}} \mathcal{L}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}), \tag{11}$$

where η is the learning rate, and t is the iteration number. We initialize the perturbation to zero: $\boldsymbol{\delta}^0 = 0$.

To focus only on the extracted sensors, we apply a mask to the gradient. The mask $\mathbf{m}$ is defined as:

$$\mathbf{m}_i = \begin{cases} 1, & \text{if } i \in S^{(t)}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

This mask zeros out the gradients for sensors not in the extracted node set. The masked gradient is computed as:

$$\nabla'_{\boldsymbol{\delta}} \mathcal{L}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}) = \nabla_{\boldsymbol{\delta}} \mathcal{L}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}) \odot \mathbf{m}, \tag{13}$$

where $\odot$ denotes element-wise multiplication. The perturbation is then updated using the masked gradient:

$$\boldsymbol{\delta}^{(t+1)} = \boldsymbol{\delta}^t - \eta \nabla'_{\boldsymbol{\delta}} \mathcal{L}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}). \tag{14}$$

This process continues until we reach the maximum number of iterations or obtain a valid counterfactual instance. The final step adds the perturbation to the original signal:

$$\mathbf{x}_{\text{cf}} = \mathbf{x} + \boldsymbol{\delta}. \tag{15}$$

The generated counterfactual instance $\mathbf{x}_{\text{cf}}$ is a modified version of the original signal that produces a different model output. This counterfactual instance explains the model's decision by showing how the prediction changes when influential sensors are altered. By only perturbing sensors in the extracted node set, we focus on the most relevant sensors, which helps minimize the perturbation size and improve the quality of explanations.

4 Experiments

4.1 Experiment Setup

4.1.1 Datasets

We evaluate our approach on two multivariate time series datasets from industrial control systems, comprising both public benchmarks. Dataset statistics are summarized in Table 1.

Table 1. Dataset statistics and characteristics.

Dataset	#Feature	#Train	#Test	Anomaly Ratio
SWaT [12]	51	47,520	44,991	12.20%
WADI [1]	128	118,800	17,280	5.77%

We use two widely-adopted water treatment testbed datasets: SWaT [12] and WADI [1]. The Secure Water Treatment (SWaT) dataset contains data from a scaled water treatment plant with 51 sensors monitoring various physical processes. The Water Distribution (WADI) dataset extends SWaT with a more comprehensive 128-sensor water distribution system. Both datasets include two weeks of normal operations followed by controlled attack scenarios that simulate real-world anomalies through physical system manipulations.

We apply consistent preprocessing across all datasets following [6]: (1) median downsampling to 0.1 Hz (one sample per 10 seconds) to reduce noise and computational overhead, (2) sensor-wise min-max normalization to [0,1] range, and (3) sliding window segmentation into 50-second chunks (5 downsampled measurements) for model input, following the previous works [6].

4.1.2 Baseline Methods

We compare the GNN anomaly detection approach against several baseline models, including six traditional machine learning models, and one GNN-based model. The compared models are listed as follows:

- KNN: K Nearest Neighbors utilizes the distance of each point to its k nearest neighbors as the anomaly score and classifies the point as anomalous if the score is greater than a specified threshold.
- IForest: Isolation Forest is an ensemble-based anomaly detection model that isolates anomalies by randomly partitioning the data into smaller subsets. It builds an ensemble of isolation trees and uses the average path length of the trees to compute the anomaly score.
- OCSVM: One-Class SVM is a support vector machine-based anomaly detection model that learns a decision boundary around the normal data points and classifies points outside the boundary as anomalous.
- AutoEncoder: AutoEncoder consists of an encoder and a decoder which reconstruct data samples from the input data. The reconstruction error is used as the anomaly score.
- VAE: Variational AutoEncoder is a improved version of AutoEncoder, which learns a probabilistic model of the data.
- PCA: Principal Component Analysis looks for a low-dimensional projection of the data that captures most of the variance of the data. The reconstruction error is used as the anomaly score.
- FuSAGNet [7]: FuSAGNet introduces Fused Sparse Autoencoder and Graph Net, which jointly optimizes reconstruction and forecasting while explicitly modeling the relationships within multivariate time series.

For counterfactual explanation generation, we compare against two additional baselines: (1) Reconstruction, which directly uses autoencoder reconstructions as counterfactual explanations under the assumption that reconstructions project onto the normal space, and (2) Without Node Extraction, which represents our method without the node extraction component.

4.1.3 Evaluation Metrics

We evaluate our approach using two sets of metrics: anomaly detection performance and counterfactual explanation quality.

Anomaly Detection Performance. We assess the anomaly detection model using standard classification metrics: precision, recall, F1-score, AUC-ROC, and PRC-AUC. AUC-ROC and PRC-AUC provide a comprehensive assessment of the model's performance across different threshold values and are widely used metrics for evaluating classification models.

Counterfactual Explanation Quality. We evaluate generated counterfactuals using three quantitative metrics alongside qualitative visual inspection. *Validity* measures the fraction of counterfactuals that successfully flip the model's prediction:

$$\text{Validity} = \frac{1}{N_{\text{cf}}} \sum_{i=1}^{N_{\text{cf}}} \mathbb{I}(f(\mathbf{x}_{\text{cf}}^i) < \theta) \tag{16}$$

where N_{cf} is the number of counterfactuals, $f(\cdot)$ is the model, θ is the classification threshold, and $\mathbb{I}(\cdot)$ is the indicator function.

Sparsity quantifies the average fraction of sensors modified per counterfactual:

$$\text{Sparsity} = \frac{1}{N_{\text{cf}}} \sum_{i=1}^{N_{\text{cf}}} \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(|\delta_j^i| > \epsilon) \tag{17}$$

where N is the number of sensors, δ_j^i is the perturbation for sensor j in counterfactual i, and ϵ is a minimal change threshold.

Proximity measures the average magnitude of perturbations:

$$\text{Proximity} = \frac{1}{N_{\text{cf}}} \sum_{i=1}^{N_{\text{cf}}} \|\boldsymbol{\delta}^i\| \tag{18}$$

where $\boldsymbol{\delta}^i$ represents the perturbation vector for counterfactual i.

Higher validity indicates more effective counterfactuals, while lower sparsity and proximity reflect better explainability through minimal, localized changes.

4.1.4 Implementation Details

We implement the proposed approach using PyTorch and PyTorch Geometric. The model is trained with Adam optimizer with learning rate 1×10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.99)$ for 50 epochs. We include early stopping with a patience of 10 epochs. The embedding dimension for the sensors is 128 for WADI dataset, and 64 for SWaT dataset. Training is performed on a single Tesla T4 GPU with 16 GB memory. For the node extraction module, we set $k_1 = 2$ and $k_2 = 1$. The perturbation is computed using gradient descent with a learning rate of 0.001 and a maximum of 100 iterations, with Adam optimizer. λ for the objective function is 0.1 for SWaT dataset and 0.001 for WADI dataset.

4.2 Benchmark Comparison

In this section, we conduct two benchmark comparisons. The first benchmark is to compare the anomaly detection performance of the proposed GNN-based model with the other baseline models. This benchmarking acts a sanity check for anomaly detection. The second benchmark is to compare the generated counterfactual explanations with the baseline models.

Table 2. Anomaly detection performance of different models on the WADI and SWaT datasets. The best results are highlighted in bold. Higher values are better.

Dataset	Model	F1	Precision	Recall	ROC-AUC	PRC-AUC
WADI	KNN	0.5295	0.7824	0.4002	0.7685	0.4829
	IForest	0.2984	0.3010	0.2959	0.7375	0.2104
	OCSVM	0.5109	0.6772	0.4102	0.7872	0.4897
	AutoEncoder	0.5434	0.8124	0.4082	0.7775	0.4928
	PCA	0.5159	0.7036	0.4072	0.7449	0.4685
	VAE	0.3652	0.2614	**0.6058**	0.7962	0.4753
	FuSAGNet	0.4697	0.5195	0.4273	**0.8109**	0.4884
	GDN	**0.5646**	**0.8263**	0.4293	0.8051	**0.5089**
SWaT	KNN	0.7423	0.9826	0.5965	0.8058	0.7007
	IForest	0.7075	0.9237	0.5733	0.8213	0.6155
	OCSVM	0.7503	0.9922	0.6032	0.8178	0.7148
	AutoEncoder	0.7411	0.9864	0.5936	0.8101	0.6959
	PCA	0.7404	**0.9939**	0.5899	0.8151	0.7044
	VAE	0.7378	0.9899	0.5881	0.8063	0.6960
	FuSAGNet	0.7799	0.9847	0.6455	0.8676	0.7607
	GDN	**0.8152**	0.9403	**0.7209**	**0.8917**	**0.7988**

Anomaly Detection Performance As a sanity check for GNN model, we compare the performance of anomaly detection for the proposed GNN-based model and the other baseline models on the two datasets. The results are shown in Table 2.

On the WADI dataset, GDN achieves the highest F1, precision and PRC-AUC, while FuSAGNet leads in ROC-AUC. VAE achieves the best recall. These results suggest that GNN-based models offer more balanced performance.

On the SWaT dataset, GDN consistently outperforms others across nearly all metrics. PCA achieves the highest precision but with lower recall, indicating a stricter anomaly boundary that may misclassify normal instances.

Explanation Performance We compare the performance of counterfactual explanations across different models. In addition to our proposed method, we apply the two-stage approach using FuSAGNet. For baseline models without graph structures, we skip the node extraction step and apply the counterfactual method directly. We also evaluate a reconstruction-based counterfactual approach on both GDN and FuSAGNet. Results are shown in Table 3. Note that KNN and IForest are excluded, as their non-differentiable nature prevents gradient-based counterfactual generation.

Table 3. Performance of counterfactual explanations for anomalous instances on the WADI and SWaT datasets. *W/o Node Extr.* refers to methods without the node extraction step, while *W/ Node Extr.* includes it. *Reconstruction* indicates reconstruction-based counterfactual generation. Metrics include validity, sparsity, and proximity, where higher validity and lower sparsity/proximity indicate better performance. Best results in each category are shown in bold.

Dataset	Anomaly Detection Model	Counterfactual Generation	Validity	Sparsity	Proximity
WADI	OCSVM	w/o Node Extr.	0.4412	1.0000	0.2490
	AutoEncoder	w/o Node Extr.	0.4557	1.0000	0.1336
	PCA	w/o Node Extr.	0.4581	1.0000	0.0561
	VAE	w/o Node Extr.	0.6131	1.0000	0.0808
	FuSAGNet	w/o Node Extr.	0.6714	0.0140	0.0075
		Reconstruction	0.0892	1.0000	0.5617
		w/ Node Extr.	**0.6714**	**0.0138**	**0.0075**
	GDN	w/o Node Extr.	0.5148	0.1551	0.2837
		Reconstruction	0.0023	1.0000	0.4868
		w/ Node Extr.	**0.5718**	**0.0091**	**0.0124**
SWaT	OCSVM	w/o Node Extr.	1.0000	1.0000	0.1132
	AutoEncoder	w/o Node Extr.	1.0000	1.0000	0.1002
	PCA	w/o Node Extr.	1.0000	1.0000	0.1680
	VAE	w/o Node Extr.	0.9997	1.0000	0.2386
	FuSAGNet	w/o Node Extr.	**0.9980**	0.1517	0.0571
		Reconstruction	0.1330	1.0000	0.3188
		w/ Node Extr.	0.1380	**0.0552**	**0.0157**
	GDN	w/o Node Extr.	0.9010	0.7714	0.0324
		Reconstruction	**1.0000**	1.0000	0.3218
		w/ Node Extr.	0.9740	**0.0547**	**0.0141**

On the WADI dataset, the proposed approach achieves a validity score of 0.5718, which is not significantly higher than other models, but still acceptable. Notably, it outperforms others in sparsity and proximity, indicating that the generated counterfactuals are both sparse and close to the original instances. In contrast, baseline models show poor performance, with a sparsity score of 1.0000 and much higher proximity values. While FuSAGNet with node extraction achieves a higher validity score, its sparsity and proximity do not improve significantly. These suggest that generating valid counterfactuals is more easy, but needs more adjustment to the original signal. We also find that the node extraction step is not effective for FuSAGNet, as the validity score is the same as the model without the node extraction step. Reconstruction-based methods perform poorly on WADI, with low validity and sparsity fixed at 1.0000, indicating difficulty in generating meaningful and interpretable counterfactuals.

On the SWaT dataset, a similar trend emerges. The proposed approach achieves a high validity score alongside low sparsity and proximity, indicating effective and interpretable counterfactuals. Although baseline models and reconstruction-based methods reach perfect validity, they suffer from high sparsity and proximity, reducing explinability. When the node extraction step is removed from the proposed approach, validity drops and sparsity increases significantly, which highlights the step's effectiveness. We attribute this to the gradient-based method distributing perturbations across all sensors, leading to less valid and less sparse counterfactuals. Interestingly, FuSAGNet performs worse with node extraction on SWaT, dropping to a validity score of 0.1380. This may stem from its architectural constraints enforcing sparsity in the latent space [7], which limits its adaptability in counterfactual generation.

4.3 Ablation Studies

Table 4. Performance of generated counterfactual explanations with different hyperparameters for GDN on the SWaT and WADI datasets. k_1 and k_2 are the number of selected sensors and the number of neighbors for each selected sensor, respectively. Higher validity and lower sparsity and proximity scores are better.

Dataset	k_1	k_2	Validity	Sparsity	Proximity
SWaT	1	0	0.4160	0.0200	0.0067
	1	1	0.9640	0.0351	0.0113
	2	0	0.9680	0.0361	0.0116
	2	1	0.9740	0.0547	0.0141
	3	0	0.9570	0.0548	0.0161
	5	0	0.9430	0.0798	0.0206
WADI	1	0	0.4123	0.0064	0.0094
	1	1	0.4123	0.0066	0.0096
	2	0	0.5763	0.0089	0.0121
	2	1	0.5718	0.0091	0.0124
	3	0	0.5900	0.0096	0.0128
	5	0	0.5900	0.0109	0.0139

Effect of Node Extraction Hyperparameters for Counterfactual Explanations: We investigate the impact of hyperparameters k_1 and k_2, which control the number of selected sensors and their neighbors, on the quality of counterfactual explanations using GDN on the SWaT and WADI datasets. Results are shown in Table 4.

As k_1 and k_2 increase, the validity score generally improves, indicating that more valid counterfactuals can be generated, with more features to perturb. However, this trend plateaus when the sum $k_1 + k_2$ exceeds 2, suggesting only a small number of informative sensors and their immediate neighborhood are sufficient for effective explanation. Meanwhile, both sparsity and proximity scores in-

crease with k_1 and k_2, reflecting reduced explainability due to more widespread perturbations.

On the SWaT dataset, the best configuration is $k_1 = 2$, $k_2 = 1$, achieving the highest validity of 0.9740 while maintaining relatively low sparsity and proximity. Notably, this configuration outperforms than the one with $k_1 = 3$ and $k_2 = 0$, despite both involving three total nodes. This indicates that leveraging the graph structure to incorporate neighbors provides more targeted and efficient perturbations than selecting more sensors independently, which highlights the benefit of graph-based relational modeling in counterfactual generation. In contrast, too few sensors (e.g., $k_1 = 1$, $k_2 = 0$) result in poor validity (0.4160), while too many ($k_1 = 5$) can dilute the perturbation effect, lowering validity to 0.9430. A similar pattern is observed on WADI, where the best validity (0.5900) occurs at $k_1 = 3$, $k_2 = 0$, though the overall scores are lower, which is likely due to WADI's higher dimensionality and complexity.

Overall, the number of selected sensors should be large enough to ensure generation of valid counterfactuals, but small enough to maintain explainability. Validity gains plateau after a certain point, suggesting a trade-off between completeness and sparsity.

4.4 Visual Analysis Experiments

We show one illustrative example of the generated counterfactual explanations for the detected anomalies. This example is selected from the SWaT dataset, which contains a detected anomaly with a label of 9. The original instance and the generated counterfactual instance are shown in Figure 3.

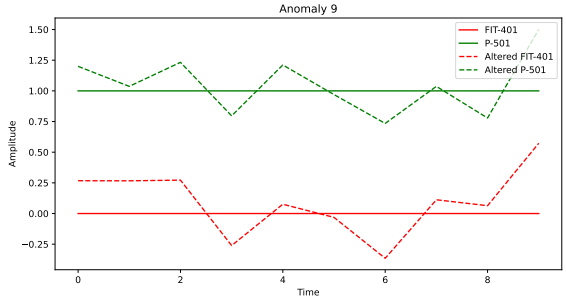

Figure 3. An example of the detected anomaly and the generated counterfactual instance on SWaT dataset. The original signals are shown in solid lines, and the generated counterfactual signals are shown in dashed lines. Only two altered sensors are shown in the figure.

This anomaly example is due to a attack on the sensor FIT-401, which is a flow transmitter sensor. The attack manually set the sensor value to 0, which makes actuator P-501 turns off (from 2 to 1 in value). The original instance is shown in solid lines, and both sensors are in the status of turning off. Our node extraction module selects the most important sensors, i.e., FIT-401 and P-501. The generated counterfactual instance is shown in dashed lines, where the sensor FIT-401 is set to a higher value, and the actuator P-501 is set to a higher value. We can see that there are correlations between the two sensors, which indicates that they are related and can influence each other's behavior. This aligns with the physical setting of the system, as FIT-401 is the upstream sensor of P-501, and the value of FIT-401 has direct influence on the value of P-501. The generated counterfactual instance is valid, as it is close to the original instance and can be interpreted as a valid counterfactual explanation.

5 Discussion

Our experimental results confirm that the proposed two-stage counterfactual framework provides concise, actionable explanations that improve trust and troubleshooting efficiency for system operators. In this section we discuss two main insights.

Effectiveness of graph-aware counterfactuals: Across both datasets, validity increases sharply once the explanation can perturb *at most three* sensors, i.e. the $k_1+k_2 = 3$ setting, where $k_1 > 0$, and $k_2 > 0$ and $k_2 > 0$, denote that neighbors of the selected features are utilised. This shows that usually only a few, closely linked variables drive each anomaly. When we choose some of those sensors using the graph of how they connect (i.e. increasing k_2), the resulting counterfactuals are more valid than if we just picked the sensors with the highest anomaly scores. This backs up our idea that knowing the system's structure is crucial for clear counterfactual reasoning in highly coupled systems.

Trade-off between validity, sparsity and proximity: Letting the algorithm perturb more sensors (higher k_1 or k_2) makes its explanations more often valid, but it also means bigger changes to the data, resulting in the results become harder to read and trust. Looking at Table 4, the sweet spot seems to be $k_1 = 2$ and $k_2 = 1$: we still get over 97% validity on the SWaT dataset while the typical change stays under 0.015 (in normalized units). In practice, engineers can pick these two knobs to suit their goals: smaller values if they want to pinpoint the root cause with minimal edits, larger values if making sure the validity is more important than keeping the edits tiny.

6 Conclusion

In this work, we introduced a novel framework to generate counterfactual explanations tailored for graph neural network-based model. Our approach leverages the representational power of GNNs to model complex inter-sensor relationships in our two-stage explanation mechanism which enables interpretable counterfactual reasoning. Extensive experiments on the SWaT and WADI benchmarks show that our two-stage framework cuts the number of perturbed sensors to less than 6% on average, while generating highly valid counterfactual explanation. This superior sparsity–proximity trade-off means the counterfactuals are both concise and easier for practitioners to act upon.

Our framework contributes to more transparent and trustworthy machine learning solutions for safety-critical domains by bridging the gap between black-box anomaly detection using GNNs and explainable AI. Future work may explore weighted similarity-based relationships in graphs, the integration of domain constraints, real-time explanation generation, and multi-criteria optimization.

Acknowledgements

The work was carried out with support from Vinnova (Sweden's innovation agency) through the Advanced Digitalisation Program as part of the future AI-based maintenance project (project number: 2023-01917).

References

[1] C. M. Ahmed, V. R. Palleti, and A. P. Mathur. Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, pages 25–28, 2017.

[2] E. Ates, B. Aksar, V. J. Leung, and A. K. Coskun. Counterfactual explanations for multivariate time series. In *2021 international conference on applied artificial intelligence (ICAPAI)*, pages 1–8. IEEE, 2021.

[3] M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. C.-H. Lam, and Y. Zhang. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems*, 34:5644–5655, 2021.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[5] E. Delaney, D. Greene, and M. T. Keane. Instance-based counterfactual explanations for time series classification. In *International conference on case-based reasoning*, pages 32–47. Springer, 2021.

[6] A. Deng and B. Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.

[7] S. Han and S. S. Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2977–2986, 2022.

[8] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[9] I. Karlsson, J. Rebane, P. Papapetrou, and A. Gionis. Locally and globally explainable time series tweaking. *Knowledge and Information Systems*, 62(5):1671–1700, 2020.

[10] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4499–4511. PMLR, 2022.

[11] J. Ma, R. Guo, S. Mishra, A. Zhang, and J. Li. Clear: Generative counterfactual explanations on graphs. *Advances in neural information processing systems*, 35:25895–25907, 2022.

[12] A. P. Mathur and N. O. Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016.

[13] C. Molnar. *Interpretable Machine Learning*. 3 edition, 2025. ISBN 978-3-911578-03-5. URL https://christophm.github.io/interpretable-ml-book.

[14] X. Sun, R. Aoki, and K. H. Wilson. Counterfactual explanations for multivariate time-series without training datasets. *arXiv preprint arXiv:2405.18563*, 2024.

[15] Z. Wang, I. Samsten, R. Mochaourab, and P. Papapetrou. Learning time series counterfactuals via latent space representations. In *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*, pages 369–384. Springer, 2021.

[16] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica. Representing long-range context for graph neural networks with global attention. *Advances in neural information processing systems*, 34:13266–13279, 2021.

[17] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1):1–42, 2024.