CIEXPLAINER++: CAUSAL AND INTERPRETABLE EXPLANATIONS FOR GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Explainable Artificial Intelligence aims to make black-box models more trustworthy by presenting, in a human-understandable manner, the elements that lead to the model's output. This involves both (i) identifying components and connections with model-level interventional effect on outputs and (ii) translating such structures into an interpretable representation. For the former, we introduce CIExplainer, a novel perturbation-based method grounded in causal inference for explaining Graph Neural Networks (GNNs). CIExplainer identifies the subgraph with the highest model-level interventional effects on GNN predictions using the Potential Outcome Framework. We evaluate and compare CIExplainer on various GNN architectures (GCN, GraphSAGE, GAT, GIN) and datasets. To bridge subgraph explanations with human interpretability, we further propose G2TeXplainer, a method that transforms node-induced interventional explanation subgraphs into natural language explanations that capture both feature-level and relational information.

1 INTRODUCTION

Graph Neural Networks (GNNs) have emerged as a powerful class of deep learning models for learning from graph-structured data (Zhang et al., 2024a). They have achieved strong performance across diverse domains, including recommendation systems (Wu et al., 2021a; Fan et al., 2019), healthcare (Duarte et al., 2021; Valdeira et al., 2023; Sungmin et al., 2018; Kumar et al., 2024), and crime prediction (Jin et al., 2020; Wang et al., 2022).

Despite their success, GNNs typically operate as black-box models (e.g., models that natively do not provide a clear understanding of the mechanisms that lead to a given prediction). This lack of transparency poses significant challenges in high-stakes environments, where understanding model predictions is essential for informed decision-making and increased trustworthiness and accountability. Consequently, developing principled methods to explain the GNN predictions has become a critical research direction (Amara et al., 2022).

Explainable Artificial Intelligence (XAI) (Molnar, 2022; Minh et al., 2022; Lipton, 2018) has emerged to address questions such as

> How can we **trust** a model prediction? How can we know **why** the model made the wrong prediction? How can we **explain** the model behavior?

XAI provides explanation methods to clarify the model's inner mechanisms. However, many explanation methods rely on identifying associations and correlations in the data to justify predictions. While these explanations are often transparent and easy to understand, they do not always offer a full picture of how a model works because "correlation does not imply causation". This means that associations and correlations may overlook deeper, cause-and-effect relationships that drive the models' predictions.

To address this limitation, we propose **CIExplainer**, a local perturbation-based (Yuan et al., 2022; Zhang et al., 2024a) causal explanation framework grounded in the Potential Outcome Framework. CIExplainer estimates the model-level interventional effect of feature-level interventions on model predictions, enabling the identification of graph components that have a causal impact on GNN outputs. GNNs take graphs as input and produce embeddings for graph elements, which are then

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

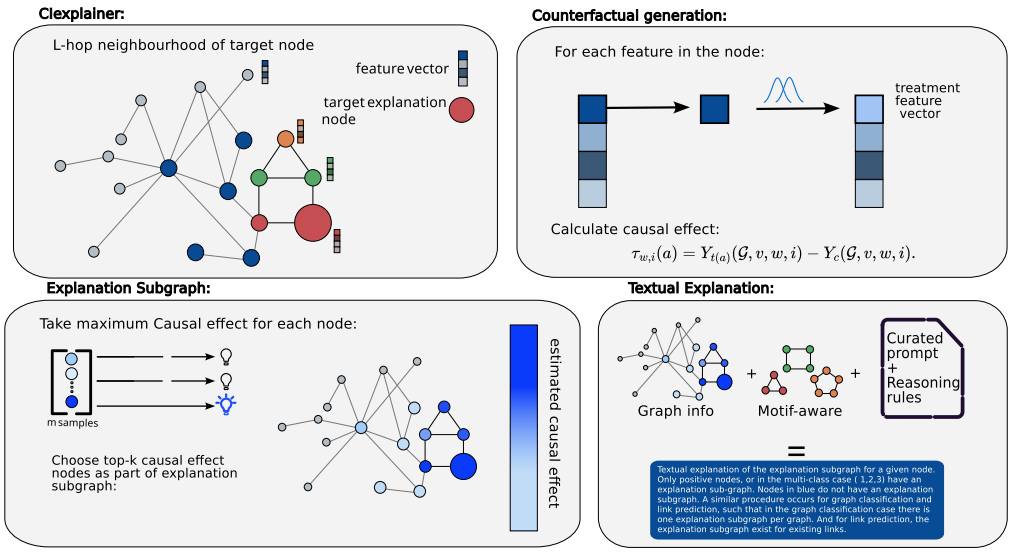


Figure 1: Diagram of the CIExplainer++ proposed pipeline. Using as example the explanation of a node classification task, it outputs the nodes with highest CE, alongside a textual description of the explanation subgraph.

used for tasks such as node classification, graph classification, or link prediction. As is common in GNN explanation methods, the goal of CIExplainer is to produce an explanation subgraph. The explanation subgraph is nonetheless limited in interpretability, as it requires an expert to understand its structure and features (Cedro & Martens, 2025).

To bridge this gap, we further propose **G2TeXplainer**, a graph-to-text model that translates node-induced interventional explanation subgraphs into natural language explanations. Given the explanation subgraph produced by CIExplainer and the key structural motifs within the graph we use a Large Language Model (LLM) to verbalize them in a way that is accessible to a broader audience without sacrificing technical rigor, directly addressing the questions posed above: now we can **understand** why a prediction was made, **trust** the reasoning behind it, and **explain** it. To evaluate the quality of the generated explanations, we design an automated LLM-as-judge (Gu et al., 2026) evaluation framework that assesses the best prompting strategy for generating faithful and clear explanations.

All code and datasets are available at https://anonymous.4open.science/r/CIExplainer-5966.

2 Related Work

Graph Neural Networks. The primary goal of GNNs is to perform representation learning on graphs, which means that they create a condensed representation of a graph, node, or edge in a lower-dimensional space while preserving the original relational information of the input graph. This condensed representation, or embedding, can then be applied in subsequent tasks, such as graph classification and link prediction. Graph Convolution Network (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), Graph Attention Network (GAT) (Veličković et al., 2018), and Graph Isomorphism Network (GIN) (Xu et al., 2019) are some of the most prominent GNN models to date. **GCN** (Kipf & Welling, 2017) is a graph convolutional network that applies a localized first-order approximation of spectral graph convolutions (Hammond et al., 2011), combining each node's features with those of its immediate neighbors to capture local graph structure. **GraphSAGE** (Hamilton et al., 2017) is designed to generate node embeddings from large graphs. It accomplishes this by sampling a K-hop neighborhood around the target node and aggregating the features of the target node and its neighbors. Through this process, GraphSAGE learns a series of aggregator functions that, once trained, can generate embeddings for previously unseen nodes. **GAT** (Veličković et al., 2018) leverages masked self-attentional layers that allow nodes to focus on their neighbors' features with different weights. The **GIN** model (Xu et al., 2019) is based on the

Weisfeiler-Lehman test (Leman & Weisfeiler, 1968), which is well known for its ability to differentiate many types of graph structures by iteratively gathering information from neighboring nodes. The main idea of GIN is to use a very expressive aggregation function that can model injective functions. This allows the model to fully capture the set of node features in a graph neighborhood.

The field of GNN is extensive and a comprehensive survey is available in (Wu et al., 2021b). Other relevant and recent models include GCN-II (Chen et al., 2020), Anti-Symmetric Deep Graph Network (ADGN) (Gravina et al., 2023), Graph Neural Diffusion (GRAND) (Chamberlain et al., 2021), Substructure Assembling Network - Soft Sequence with Context Attention (SAN-SSCA) (Yang et al., 2023) and Interpretable and Efficient Heterogeneous GCN (ie-HGCN) (Yang et al., 2021).

GNN Explanation. Several perturbation-based methods have been proposed to provide instance-level explanations for GNN predictions. **GNNExplainer** (Ying et al., 2019) is a local explanation method applicable to graph, node, and edge level tasks. Given a trained GNN and its prediction, it returns a compact explanation subgraph and a subset of node features that predominantly influence the output by maximizing the mutual information between the prediction and candidate subgraphs. However, it must be retrained for each instance. **PGExplainer** (Luo et al., 2020; 2024) similarly learns discrete edge masks by training a mask predictor to maximize mutual information between original and masked predictions, but is trained once and generalizes across examples. GRAPH-MASK (Schlichtkrull et al., 2021) is a post-hoc method that learns to identify removable edges while preserving predictive performance using stochastic gates optimized with L_0 regularization to encourage sparsity, yielding a graph that can be seen as an explanation graph. **SubgraphX** (Yuan et al., 2021) combines Monte Carlo Tree Search and Shapley values to search for important connected subgraphs through iterative node pruning, where Shapley values quantify the contribution of each subgraph to the prediction. Unlike other approaches, SubgraphX guarantees connected explanation subgraphs. GEM (Lin et al., 2021) proposes a generative explanation framework that learns to produce compact explanation subgraphs guided by an objective inspired by Granger causality.

Other methods for local post-hoc, agnostic, GNN explanation mostly include surrogate models, such as PGM-Explainer (Vu & Thai, 2020), GraphLIME (Huang et al., 2023), DnX (Pereira et al., 2023), RC-Explainer (Wang et al., 2023), GOAt (Lu et al., 2024) and GstarX (Zhang et al., 2022), that learn a different, white-box model, to explain the predictions of the original model. Also, and while not specifically designed for GNNs, gradient based methods such as IntegratedGradient (Shrikumar et al., 2017) and saliency maps (Simonyan et al., 2013) can determine the importance of the input by evaluating the gradient of the model output with respect to the input features.

Furthermore, there exist methods that explicitly learn counterfactual explanation graphs (Lucic et al., 2022), meaning producing the minimal subgraph with modified edges such that a prediction is meaningfully changed, or that focus on regression based tasks (Zhang et al., 2024b).

Graph-to-Text Explanation. While most GNN explanation methods focus on identifying important subgraphs or feature attributions, translating these structural explanations into natural language remains relatively underexplored. GraphXAIN (Cedro & Martens, 2025) generates narrative explanations by prompting an LLM with explanatory subgraphs and feature importance scores produced by GNNExplainer, improving perceived understandability and user satisfaction. For text-attributed graphs, GraphNarrator (Pan et al., 2025) converts saliency-based node and edge importance scores into structured textual descriptions and uses them to train a dedicated explanation generator. In contrast, LOGIC (Baghershahi et al., 2026) aligns internal GNN embeddings with the LLM token space via a learned projector, constructing hybrid prompts that combine graph structure, node text, and latent representations. However, both GraphNarrator and LOGIC are tailored to text-attributed graphs, limiting their applicability to settings where nodes lack textual features.

3 METHODOLOGY

One of the goals of the work presented in this paper is to construct an explanation method tailored to produce causal explanations for GNN-based tasks. As such, we formulate CIExplainer, a local explanation method dedicated to generating causal inference explanations. With CIExplainer, we provide a node-induced interventional explanation subgraph, that can, paired with its textual interpretation, improve the quality and transparency of GNN-based models, enabling their trustworthy

deployment in critical decision-making environments like clinical risk modeling and financial fraud detection. In the following, we will develop the methodology in the context of node prediction.

3.1 Problem Statement

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. Each node $v \in \mathcal{V}$ is associated with a feature vector $\mathbf{x} = \{x_1, x_2, ...x_n\}, x_i \in \mathbb{R}$. Let Φ denote a trained GNN model for a particular dataset $\mathcal{D}$.

Inspired by Ying et al. (2019), we emphasize that the computation graph of a node v used for node prediction is defined by the neighborhood-based aggregation of the GNN. This graph fully determines all the information the GNN uses to compute a node-level prediction $\hat{y}$ for v. Specifically, the computation graph of v guides the GNN in creating the embedding for v. Let $Ne_K(v) = (\mathcal{V}_{Ne}, \mathcal{E}_{Ne}), \mathcal{V}_{Ne} \subseteq \mathcal{V},\ \mathcal{E}_{Ne} \subseteq \mathcal{E}$ represent the computation graph of v, which is a K-hop sampled neighborhood around v.

The prediction of a GNN is given by $\hat{y} = \Phi(Ne_K(v))$, meaning it depends entirely on the model Φ, the structural information of the graph $Ne_K(v)$, and the related node features. This observation shows that to explain $\hat{y}$, we only need to focus on the sampled neighborhood $Ne_K(v)$ and the corresponding node features. Thus, CIExplainer returns as an explanation a subgraph $\mathcal{G}_{EXP} \subset Ne_K(v)$ with the nodes that had the highest model-level interventional effect on $\hat{y}$. The model-level interventional effect of each node on $\hat{y}$ is calculated using the Potential Outcome Framework (Holland, 1986).

3.2 Potential Outcome Framework

The Neyman-Rubin Causal Model (Holland, 1986), also known as the Potential Outcome Framework, formalizes causal inference in terms of unit-level counterfactual outcomes. Let U denote a population of units, where each unit $u \in U$ is the basic object of study. Variables are real-valued functions defined over U, assigning measurements to each unit.

The central concept in the model revolves around the potential to either expose or refrain from exposing each unit to a causal influence. Let $S(u) \in t, c$ denote a binary treatment assignment, where t represents treatment and c control. For each unit u, define the potential outcomes $Y_t(u)$ and $Y_c(u)$ as the responses that would be observed under treatment t and control c, respectively. The unit-level causal effect is defined as:

$$CE = Y_t(u) - Y_c(u). \tag{1}$$

According to the framework, equation 1 expresses that treatment t causes the effect $Y_t(u) - Y_c(u)$ on unit u (relative to treatment c).

3.3 CIExplainer

Causal Estimand. For a fixed trained GNN Φ, a target prediction instance $(\mathcal{G}, v)$, a candidate node w in the graph and a feature i define the unit as the explanation instance associated with $(\mathcal{G}, v, w, i)$. The control condition c keeps the original feature value $x_{w,i}$. The treatment condition $t(a)$ replaces $x_{w,i}$ by an admissible intervention value a, while keeping the trained model fixed. The outcome is the prediction score, logit, or probability assigned by Φ to the target class. Then,

$$\tau_{w,i}(a) = Y_{t(a)}(\mathcal{G}, v, w, i) - Y_c(\mathcal{G}, v, w, i). \tag{2}$$

For continuous features, several counterfactual values are sampled from $q_i(a)$, the chosen intervention distribution for feature i. Define the intervention values as an i.i.d. batch of size B $a_{w,i}^{(1)}, \ldots, a_{w,i}^{(b)}, \ldots, a_{w,i}^{(B)} \overset{\text{iid}}{\sim} q_i$. Define the sample-level aggregation operator

$$\underset{b=1,\ldots,B}{\text{Agg}} : \mathbb{R}^B \to \mathbb{R},$$

that can be the average, the sum, or the point maximum. Then, the finite-budget estimand is

$$\tau_{w,i}^{\text{Agg},B} = \mathbb{E}_{\left\{a_{w,i}^{(b)}\right\}_{b=1}^{B} \overset{\text{iid}}{\sim} q_i} \left[\underset{b=1,\ldots,B}{\text{Agg}} \tau_{w,i}\left(a_{w,i}^{(b)}\right) \right], \tag{3}$$

Algorithm 1 CIExplainer for node prediction

Input: target node v, GNN model Φ, predicted label $\hat{y}_v$
Output: explanation subgraph $\mathcal{G}_{EXP}$ containing the nodes that caused $\hat{y}_v$
Sample a K-hop neighborhood $Ne_K(v)$ around node v
Compute the node prediction $\hat{y}_v$ using Φ on $Ne_K(v)$
for each node $w \in Ne_K(v)$ **do**
  for each feature x_i of w **do**
    Generate a counterfactual by perturbing x_i
    Compute the model-level interventional effect CE_i of the perturbation on $\hat{y}_v$
  end for
  Node model-level interventional effect $CE_u = \max_i |CE_i|$
end for
Return subgraph of $Ne_K(v)$ containing top l nodes by CE_u and edges between those nodes

where $\tau_{w,i}(a)$ was defined in equation 2.

The parameter B is a parameter of the algorithm and depends on the computational budget available. The current implementation uses $B = 10$ and Agg $= \max$, yielding a finite-budget maximum-effect score. Other choices, such as mean, maximum absolute effect, or median, correspond to different intervention-effect summaries and are assessed in ablation studies (Appendix A.5). In practice, with one sampled batch, the empirical score for max-aggregation is

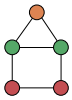
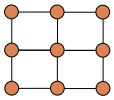
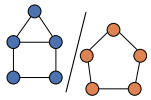
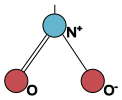
$$\hat{\tau}^{B}_{w,i} = \max_{b=1,\cdots,B} \left\{ Y_{t\left(a^{(b)}_{w,i}\right)}(\mathcal{G}, v, w, i) - Y_c(\mathcal{G}, v, w, i) \right\}.$$

Explanation Subgraph Given a node v extracted from a graph $\mathcal{G}$, a trained GNN model Φ, and a node prediction probability $\hat{y}_p$ generated by Φ for that particular node by sampling a K-hop neighborhood $Ne_K = (\mathcal{V}_N, \mathcal{E}_N)$ of the node, CIExplainer yields a subgraph $\mathcal{G}_{EXP} = (\mathcal{V}_{EXP}, \mathcal{E}_{EXP})$ as an explanation for $\hat{y}$, denoting the inferred node label by analyzing $\hat{y}_p$. This subgraph contains the top l nodes and the edges between those nodes that caused Φ to output $\hat{y}_p$. Specifically, $\mathcal{V}_{EXP} \subseteq \mathcal{V}_N$, $\mathcal{E}_{EXP} \subseteq \mathcal{E}_N$, and $|\mathcal{V}_{EXP}| = l$, where $l \in \{1, 2, ..., |\mathcal{V}_N|\}$ is a hyperparameter enforcing the returned explanation to be concise and informative. Our proposed method, CIExplainer, selects the top l nodes by evaluating the model-level interventional effect that each node in Ne_K exerts on the prediction $\hat{y}$, then prioritizing the l nodes with the highest model-level interventional effect values.

Let $\hat{y}$ be the node classification of the GNN for node v and S be a binary variable representing the cause, t or c, to which $\hat{y}$ is exposed. We note that to generate a node classification probability $\hat{y}_p$, a GNN model Φ only needs the sampled K-hop neighborhood Ne_K from node v. As such, manipulating $\hat{y}$ consists of manipulating Ne_K. Hence, let c denote the absence of manipulation of Ne_K and let t denote the manipulation of Ne_K, specifically, manipulating the feature values of a node in Ne_K. Maintaining the original Ne_K used to generate $\hat{y}$ is denoted by $S(u) = c$ and it represents the *actual* outcome, while, manipulating the features of nodes in Ne_K is denoted by $S(u) = t$ and it represents the *counterfactual* outcome. Let Y denote the node prediction probability generated by Φ for some sampled K-hop neighborhood. Then, $Y_c(u)$ denotes the node prediction probability when u is exposed to c, that is, it denotes the original or *actual* node prediction probability $\hat{y}_p$. Whereas $Y_t(u)$ denotes the node prediction probability when u is exposed to t, that is, the *counterfactual* node prediction probability obtained by constructing a *counterfactual* K-hop neighborhood.

In this context, the **Temporal Stability** assumption holds because the output produced by the model Φ remains consistent regardless of when the input is provided to Φ. Essentially, the value $Y_c(u)$ remains unchanged irrespective of the timing of exposing c to u and measuring $Y_c(u)$. Similarly, the **Causal Transience** assumption is valid because exposing c to u does not alter the overall network structure of Ne_K, and computing the output of Φ for Ne_K does not change the weights of the Φ. Consequently, when computing the output for the original network configuration Ne_K, denoted as $Y_t(u)$, prior exposure of u to c does not influence the result. Therefore, the causal effect CE of altering the feature value of a node in Ne_K on the predicted outcome $\hat{y}$, as assessed by Y, can be determined using equation 1.

Table 1: Dataset statistics. The characteristics are defined per graph. BA-Shapes and Tree-Grid contain a single graph, with multiple motifs attached, where each motif is a ground-truth explanation.

	BA-Shapes	**Tree-Grids**	**BA-2motifs**	**MUTAG**
Motif				
Graphs	1	1	1000	4337
Nodes	700	1231	25	418
Edges	2055	1565	≈ 26	≈ 225
Features	2	2	2	14
Classes	4	2	2	2
Avg. Degree	5.87	2.54	2.04	1.07
Density	0.0084	0.0021	0.0849	0.0026

Given the aforementioned input conditions, and considering each node possesses n features, our method CIExplainer generates an explanation in accordance with Alg. 1.

For discrete (binary or categorical) features, counterfactual nodes are generated via direct value intervention, replacing the observed feature with an alternative admissible category (for binary variables, its complement). For continuous features, let $\mathbf{x}_w = (x_{w,1}, \ldots, x_{w,d})$ denote the feature vector of node w. To generate a counterfactual with respect to feature $x_{w,i}$, we hold all remaining features fixed and replace $x_{w,i}$ with a value sampled from its empirical marginal distribution, $x'_{w,i} \sim p(x_i)$, yielding the intervened feature vector $\mathbf{x}'_w = (x_{u,1}, \ldots x'_{u,i}, \ldots, x_{u,d})$. The distribution $p(x_i)$ is estimated from the training data. In our experiments, we impose a fixed per-feature sampling budget of 10 counterfactual perturbations per feature, with ablations available in Appendix A.5.

Link Prediction. For the link prediction task of an edge between a pair of nodes v, w, we adjust CIExplainer by sampling the K-hop neighborhood $Ne_K(v)$ around v, w while keeping the rest of the algorithm unchanged.

Graph Classification. For the graph classification task of a graph $\mathcal{G}$, we modify CIExplainer to use the entire graph $\mathcal{G}$ instead of a sampled neighborhood Ne_K.

3.4 G2TEXPLAINER

Given an explanation subgraph $\mathcal{G}_{EXP}$ and the neighborhood Ne_K the goal is to provide a textual explanation for a node classification task, and more generally for any task. Following the work in GraphXAIN (Cedro & Martens, 2025) we also include a textual explanation of the GNN task. We propose four prompt variants that progressively increase structural richness and reasoning guidance. **P1 (Graph)** provides the raw graph edges without explicit reasoning instructions. **P2 (1-hop)** retains only the 1-hop neighborhood of the explanation subgraph, including edges among selected nodes and their immediate neighbors. **P3 (1-hop + R.)** retains P2's input structure while adding explicit reasoning rules to focus on selected nodes and the predicted motif. Finally, **P4 (Motif-Aware + R.)** replaces raw edge lists with detected motifs (e.g., triangles, 4- and 5-cycles) while retaining P3's structured reasoning, aligning the representation with motif-level reasoning. Overall, whereas prior work relies on structured human user studies assessing subjective dimensions (e.g., understandability, trust, and satisfaction), we also use an LLM-based judge Gu et al. (2026) to assess explanation correctness and faithfulness to the input and extracted subgraph, enabling a more task-oriented and reproducible evaluation.

4 EXPERIMENTAL STUDIES

Datasets. We follow and extend PGExplainer (Luo et al., 2020) experimental settings and use two synthetic explanation evaluation datasets for node classification: BA-Shapes and Tree-Grids; alongside one synthetic explanation evaluation dataset and one real-world dataset for graph classification: BA-2motifs and MUTAG. Table 1 shows the details of synthetic and real-world datasets.

Table 2: Explanation results for node classification. Standard deviation estimated from 10 independent runs. Best result in **bold**, second best is underlined.

	Models	GCN		GAT		GIN		GraphSAGE	
		IoU (↑)	**Pr** (↑)	**IoU** (↑)	**Pr** (↑)	**IoU** (↑)	**Pr** (↑)	**IoU** (↑)	**Pr** (↑)
BA-shapes	Random Explainer	0.0029±.01	0.0052±.01	0.0037±.00	0.0062±.01	0.0032±.00	0.0057±.01	0.0040±.00	0.0071±.00
	IntegratedGradient	0.6590±.00	0.7714±.00	0.7902±.00	0.8619±.00	0.5961±.00	0.7238±.00	0.8424±.00	0.9000±.00
	InputXGradient	0.7021±.00	0.8000±.00	0.8299±.00	0.8857±.00	0.5848±.00	0.7143±.00	0.8424±.00	0.9000±.00
	Shapley Values	0.6107±.03	0.7343±.02	0.7249±.03	0.8195±.02	0.5228±.03	0.6633±.03	0.7370±.02	0.8314±.01
	PGExplainer	0.8163±.05	0.8710±.10	0.5731±.05	0.6395±.06	**0.8566±.03**	**0.8929±.02**	0.6378±.14	0.7210±.12
	GNNExplainer	0.5683±.02	0.7019±.01	0.2922±.01	0.3967±.01	0.5372±.03	0.6614±.02	0.4487±.03	0.5976±.02
	SubgraphX	0.5363±.01	0.6729±.01	0.5935±.03	0.7010±.02	0.7369±.02	0.8181±.02	0.6036±.02	0.7338±.01
	CIExplainer (Ours)	**0.8909±.02**	**0.9314±.01**	**0.8448±.01**	**0.8738±.01**	0.7915±.02	0.8667 ±.01	**0.9294±.01**	**0.9576±.04**
Tree-grid	Random Explainer	0.0036±.00	0.0067±.00	0.0053±.00	0.0099±.01	0.0040±.00	0.0075±.00	0.0024±.00	0.0046±.00
	IntegratedGradient	0.8868±.00	0.9361±.00	0.8395±.00	0.8976±.00	**0.8741±.00**	**0.9285±.00**	0.8868±.00	0.9361 ±.00
	InputXGradient	0.8887±.00	0.9376±.00	0.8381±.00	0.8954±.00	0.0182±.00	0.0228±.00	0.8890±.00	0.9376 ±.00
	Shapley Values	0.7375±.01	0.8356±.01	0.7453±.02	0.8444±.01	0.1072±.00	0.1813±.01	0.7487±.02	0.8440±.01
	PGExplainer	0.0971±.17	0.1335±.20	0.0333±.00	0.0580±.02	0.8701±.02	0.9253±.01	0.1960±.06	0.1960±.07
	GNNExplainer	0.7754±.01	0.8658±.01	0.7413±.01	0.8382±.00	0.0182±.00	0.0228±.00	0.7449±.01	0.8397±.00
	SubgraphX	0.2985±.01	0.4464±.01	0.3042±.01	0.4534±.02	0.3118±.01	0.4632±.02	0.3124±.02	0.4647±.02
	CIExplainer (Ours)	**0.9008±.01**	**0.9381±.01**	**0.8439±.00**	**0.9075±.01**	0.8407±.00	0.8988±.00	**0.9325±.02**	**0.9595±.01**

Considering how CIExplainer works, we extend the synthetic datasets that originally lack node features by assigning features to the nodes. To make GNN models use these features for predictions, we introduce node features artificially by using different Gaussian distributions with varying means for each node class while keeping the variance fixed at 2. For a dataset with C classes, each node's feature vector is sampled as $x \sim \mathcal{N}(\mu_c, 2)$, where $c \in \{0, 1, ..., C\}$.

Baselines. For comparison we choose more widely known GNN-specific explainers, and explanation methods that can be generally applied to any prediction model. Thus, we compared our method with GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020) (which is a surrogate model that generates perturbations), and SubgraphX (Yuan et al., 2021). And also, with gradient based methods IntegratedGradient, InputXGradient (Shrikumar et al., 2017) and Shapley Values (Shapley, 1951), we also compare with a random explainer (Fey & Lenssen, 2019), which is a simple explainer that makes random explanations, thus it can be used as a basic reference for all other methods.

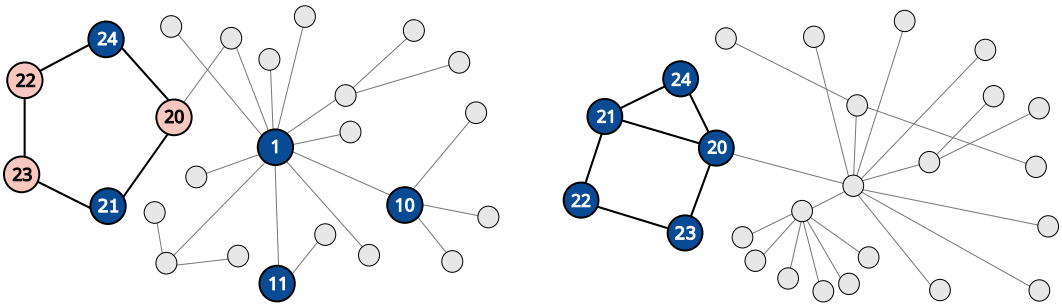
Metrics. The main metrics used in this work are the Jaccard Index, also known as IoU, to measure the overlap between the ground truth and the estimated explainability subgraph, and precision. We also evaluate the inference time per node, measured on a 10GB partition of a NVIDIA A100 GPU. CIExplainer is the fastest of the perturbation based method, averaged across tasks and datasets (see Table 5 in Appendix).

Experiment Details. We run each explanation method with each GNN model on each dataset 10 times and report the average results with the standard deviation. We experiment with four types of GNN models: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019).

G2TeXplainer. We generate explanations using LLaMA 3 8B with stochastic decoding and evaluate them using complementary LLM-as-judge and human evaluation. Full generation and evaluation details, including decoding configurations and evaluation protocols, are provided in Appendix A.7.

4.1 Quantitative Results

The results for node classification are reported in Table 2. Across both datasets, CIExplainer consistently identifies the correct explanation subgraph across all backbone models, with stability across multiple runs. In contrast, PGExplainer exhibits high variability depending on the underlying GNN, with the largest standard deviations, indicating unstable precision across runs. Although never outperforming other methods, SubgraphX remains stable across datasets and models, never collapsing for any (dataset, model) pair. While the considered tasks are relatively simple, GIN performs poorly on the Tree-Grid dataset (see Table 7), which significantly degrades the explanation quality



(a) (b)

Figure 2: Two BA-2Motifs examples with the explained subgraph and textual description.

of GNNExplainer. This behavior aligns with the intuition that explanation quality is bounded by the predictive performance of the underlying model.

Overall, CIExplainer produces more accurate explanation subgraphs with consistent behavior across architectures. Although its performance reflects the predictive quality of the backbone GNN, it does not collapse when the predictor is imperfect. On the BA-shapes dataset, approximately 40% of CIExplainer explanations achieve a perfect match (IoU = 1) with the ground-truth motif. In terms of inference time, CIExplainer is significantly faster than comparable methods that do not rely on offline training, as can be seen in Table 5. We exclude the inference cost of G2TeXplainer from this comparison, as other explainers do not include a graph-to-text interpretation component.

The graph classification results are reported in Table 4 in Appendix A.2. On BA-2motif, CIExplainer achieves the best or second-best performance across most backbone architectures, remaining competitive for all models with the exception of GAT. On MUTAG, results are more heterogeneous. GNNExplainer achieves the strongest performance for GCN, GAT, and GraphSAGE, while CIExplainer performs best under the GIN backbone. We note that GIN is the model with the best performance on the MUTAG dataset, as seen in Table 8, and that we found a positive correlation between the explanation and the model quality, for the CIExplainer.

Overall, CIExplainer demonstrates strong performance on in node classification tasks, and comparable performance with other explainers in graph classification, at a reduced computational cost.

Table 3: Evaluation of G2TeXplainer and GraphXAIN on 20 test examples with 30 explanations each. **Sim.**: similarity. LLM-as-judge metrics: **N. Fid.**: node-importance fidelity; **Str.**: motif and node-membership accuracy; **Clar.**: explanation clarity. Scores range from 1 (poor) to 5 (perfect). Human metrics: **FacC.**: factual consistency; **Clar.**: clarity; **Ovr.**: overall quality, all rated 1–5.

	Prompt		LLM-as-Judge				Human Eval		
		Sim.	**Str.**	**Clar.**	**N. Fid.**	**PW.%**	**FacC.**	**Clar.**	**Ovr.**
GraphSAGE	GraphXAIN Adap.	–	3.754	3.862	3.881	–	4.095	3.143	3.190
	P1 Graph	0.712	3.576	3.284	3.687	40.2	4.000	3.381	2.810
	P2 1-hop	0.727	3.448	3.254	3.697	38.8	**4.333**	4.048	3.476
	P3 1-hop + R.	0.689	3.770	2.994	3.459	32.0	3.095	3.190	2.286
	P4 Motif-Aware + R.	**0.751**	**4.143**	**4.340**	**3.763**	**89.0**	**4.333**	**4.238**	**3.857**

4.2 Qualitative Results

Per-Metric Analysis. Table 3 shows a clear progression from P1 to P4 for GraphSAGE, with substantial gains in **Node Fidelity** and **Structure**, indicating that motif-aware prompting with scope constraints improves structural correctness and reduces overclaiming. **Clarity** and **Similarity** remain stable, suggesting lower sensitivity to prompt design. Across GNN backbones (Table 10), small P4 differences indicate that prompt design is the main driver of explanation quality.

Table 3 reports LLM-as-judge **Pairwise Win-rates**, measuring how often each prompt was preferred over alternatives. P4 shows a decisive advantage, preferred over the other prompts in 89.0% of comparisons, while P1–P3 remain below 50%. This gain likely stems from P4 explicit reasoning constraints, which enforce strict scope limitations (e.g., restricting discussion to selected nodes). This design likely explains P4 consistent preference across 6300 pairwise comparisons. Human evaluation further corroborates these findings: P4 achieves the highest **Clarity** and **Overall Quality**, and the joint-best **Factual Consistency** (4.333). This results suggest that the improvements identified by the LLM-as-judge translate to clearer, more accurate, and more useful explanations, also perceived by human annotators.

Figure 2 shows two curated BA-2Motif examples from our pipeline. In (a), CIExplainer overvalues highly connected nodes (e.g., node 1), which, although outside the motif, strongly affect the GNN prediction. The text correctly identifies the 5-cycle and summarizes the explanation subgraph. In (b), the model correctly identifies the complete "house" motif. Overall, house motifs yield clearer and more faithful descriptions (Figure 10), possibly because their defining triangle is easier to describe than the absence of one in "circle" motifs.

5 Discussion and Limitations

The algorithm used to calculate the model-level interventional effect is bound to be limited by its sampling range. While for binary features it is possible to evaluate all possible counterfactual outcomes, the same is not true in the large finite set and continuous features. We evaluate this limitation on Appendix A.5. Similarly, use of the marginal distribution instead of the joint feature distribution is ablated, showing very similar results, with the choice to use the marginal distribution as the default approach due its simplicity, and to guarantee scalability for high-dimensional data, where it is infeasible to estimate and to sample from such a joint distribution. We also like to acknowledge that our work is based on the Neyman-Rubin Causal Model and not the Pearl Causal Model (Pearl, 2009). Regarding textual explanations, a key limitation is their dependence on problem-aware prompting, as task-specific information generally improves explanation quality. Evaluation also partly relies on an LLM-as-judge, which may introduce subjectivity. Finally, results on BA-2Motif may not generalize to more diverse real-world graphs.

6 Conclusions

We presented a framework for explanations of GNN models that transforms black-box predictions into explanation subgraphs with textual descriptions. We conducted a robust evaluation of CIExplainer across multiple GNN architectures, datasets, and tasks, using diverse explanation metrics and comparisons against state-of-the-art GNN explainers. Specifically, for node classification, CIExplainer achieved the strongest overall performance while also exhibiting the lowest inference time among perturbation-based methods. The key takeaway is that the Potential Outcome Framework can be used for causal inference and explanation, for both graph and node classification. Considering the gap between real and synthetic datasets for GNN explanation, one research direction includes the use of more complex, realistic datasets, that still have ground truth explanations, but with rich and meaningful feature spaces, that more realistically represented most use-cases for GNNs. This would enable us to better understand the strengths and weaknesses of different explanation methods and support the design of improved alternatives. As future work, we also aim to improve the realism of the generated causal counterfactuals to better reflect plausible interventions. And, as GNN explanations gain adoption in real-world scenarios, calibration analysis becomes an important research direction. Furthermore, accurate textual descriptions, alongside feature-based explanations, will become more necessary to bridge the gap between ML experts and end users.

Reproducibility statement

The work is fully available at https://anonymous.4open.science/r/CIExplainer-5966. The repository contains the CI-Explainer and G2TeXplainer, alongside the trained models (Appendix A.6) used to evaluate the explanation methods. The main algorithm is presented in Algorithm 1, with further details such as the counterfactual generation methods presented in detail in Appendix A.4. Finally, we also include for completeness the prompt used to guide the G2TeXplainer in Appendix A.9.

AI use statement

We have not used LLMs for any of the sub-tasks that are required to be disclosed. LLMs are directly used in this work, explicitly to turn node-induced interventional explanation subgraphs into a textual description. During the development of this manuscript, we used generative AI tools mainly to edit code and correct bugs. We have also used LLMs to structure LaTeX tables, and improve the clarity and wording of the manuscript text. We have reviewed all AI-assisted work.

References

Kenza Amara, Zhitao Ying, Zitao Zhang, Zhichao Han, Yang Zhao, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. In Bastian Rieck and Razvan Pascanu (eds.), *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pp. 44:1–44:23. PMLR, 09–12 Dec 2022.

Peyman Baghershahi, Gregoire Fournier, Pranav Nyati, and Sourav Medya. From nodes to narratives: Explaining graph neural networks with LLMs and graph context. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41975–41994, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1944.

Mateusz Cedro and David Martens. GraphXAIN: narratives to explain graph neural networks. In *World Conference on Explainable Artificial Intelligence*, pp. 91–114. Springer, 2025.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International conference on machine learning*, pp. 1407–1418. PMLR, 2021.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Regina Duarte, Qiwei Han, and Claudia Soares. Referral prediction in healthcare using graph neural networks. In *GReS – Workshop on Graph Neural Networks for Recommendation and Search*, pp. 12, 2021. Co-located with the ACM RecSys '21 Conference.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pp. 417—426. Association for Computing Machinery, 2019.

Yassir Fathullah and Mark Gales. Generalised Probabilistic Modelling and Improved Uncertainty Estimation in Comparative LLM-as-a-judge. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanovič, Ramona Bendias, Barghi Alexandria, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan E. Lenssen, and Jure Leskovec. PyG 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Yang, et al. The Llama 3 Herd of Models. July 2024. doi: 10.48550/arXiv.2407.21783.

Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Zhouchi Lin, Bowen Zhang, Lionel Ni, Wen Gao, Yuanzhuo Wang, and Jian Guo. A survey on LLM-as-a-judge. *The Innovation*, pp. 101253, 2026. doi: doi.org/10.1016/j.xinn.2025.101253.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Paul W Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81 (396):945–960, 1986.

Zhengyu Hu, Linxin Song, Jieyu Zhang, Zheyuan Xiao, Tianfu Wang, Zhengyu Chen, Nicholas Jing Yuan, Jianxun Lian, Kaize Ding, and Hui Xiong. Explaining Length Bias in LLM-Based Preference Evaluations. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, Suzhou, China, November 2025. Association for Computational Linguistics.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Trans. on Knowl. and Data Eng.*, 35(7):6968–6972, July 2023. doi: 10.1109/TKDE.2022.3187455.

Guangyin Jin, Qi Wang, Cunchao Zhu, Yanghe Feng, Jincai Huang, and Jiangping Zhou. Addressing crime situation forecasting task with temporal graph convolutional neural network approach. In *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pp. 474–478, 2020.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Sahil Kumar, Francisco Caldas, and Cláudia Soares. Causal inference explanations for graph neural networks. In *9th Causal Inference Workshop at Conference in Uncertainty in Artificial Intelligence (UAI)*, 2024.

Andrei Leman and Boris Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

Wanyu Lin, Hao Lan, and Baochun Li. Generative Causal Explanations for Graph Neural Networks. In *International Conference on Machine Learning*, 2021.

Zachary C Lipton. The mythos of model interpretability. *Communications of the ACM*, 61(10): 36–43, 2018.

Shengyao Lu, Keith G. Mills, Jiao He, Bang Liu, and Di Niu. GOAt: Explaining graph neural networks via graph output attribution. In *The Twelfth International Conference on Learning Representations*, 2024.

Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 4499–4511. PMLR, 2022.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

Dongsheng Luo, Tianxiang Zhao, Wei Cheng, Dongkuan Xu, Feng Han, Wenchao Yu, Xiao Liu, Haifeng Chen, and Xiang Zhang. Towards inductive and efficient explanations for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5245–5259, 2024.

Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55(5):3503–3568, 2022.

Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. URL `https://christophm.github.io/interpretable-ml-book`.

Bo Pan, Zhen Xiong, Guanchen Wu, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. Graphnarrator: Generating textual explanations for graph neural networks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23–42, 2025.

Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3, 2009.

Tamara Pereira, Erik Nascimento, Lucas E. Resck, Diego Mesquita, and Amauri Souza. Distill n' explain: explaining graph neural networks using simple surrogates. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6199–6214. PMLR, Apr 2023.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. In *International Conference on Learning Representations*, 2021.

Lloyd S. Shapley. Notes on the n-person game - ii: The value of an n-person game. Technical Report RM-670 / ATI 210720, RAND Corporation, 1951.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3145–3153. PMLR, 06–11 Aug 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Rhee Sungmin, Seo Seokjun, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3527–3534. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

Gemma Team. Gemma 3. 2025. URL `https://goo.gle/Gemma3Report`.

Filipa Valdeira, Stevo Racković, Valeria Danalachi, Qiwei Han, and Cláudia Soares. Extreme multilabel classification for specialist doctor recommendation with implicit feedback and limited patient metadata. *arXiv preprint arXiv:2308.11022*, 2023.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Minh N. Vu and My T. Thai. PGM-explainer: probabilistic graphical model explanations for graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Chenyu Wang, Zongyu Lin, Xiaochen Yang, Jiao Sun, Mingxuan Yue, and Cyrus Shahabi. Hagen: Homophily-aware graph convolutional recurrent network for crime forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4193–4200, 2022.

Xiang Wang, Yingxin Wu, An Zhang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced Causal Explainer for Graph Neural Networks . *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(02):2297–2309, February 2023. doi: 10.1109/TPAMI.2022.3170302.

Le Wu, Lei Chen, Pengyang Shao, Richang Hong, Xiting Wang, and Meng Wang. Learning fair representations for recommendation: A graph-based perspective. In *Proceedings of the Web Conference 2021*, pp. 2198–2208, 2021a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021b. doi: 10.1109/TNNLS.2020.2978386.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yaming Yang, Ziyu Guan, Jianxin Li, Wei Zhao, Jiangtao Cui, and Quan Wang. Interpretable and efficient heterogeneous graph convolutional network. *IEEE Transactions on Knowledge and Data Engineering*, 35(2):1637–1650, 2021.

Yaming Yang, Ziyu Guan, Wei Zhao, Weigang Lu, and Bo Zong. Graph substructure assembling network with soft sequence and context attention. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4894–4907, 2023. doi: 10.1109/TKDE.2022.3148299.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Chuting Yu, Hang Li, Guido Zuccon, Joel Mackenzie, and Teerapong Leelanupab. When LLM Judges Inflate Scores: Exploring Overrating in Relevance Assessment. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 4322–4327. Association for Computing Machinery, 2026. doi: 10.1145/3805712.3809905.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pp. 12241–12252. PMLR, 2021.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799, 2022.

He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. Trustworthy graph neural networks: Aspects, methods, and trends. *Proceedings of the IEEE*, 2024a.

Jiaxing Zhang, Zhuomin Chen, Hao Mei, Longchao Da, Dongsheng Luo, and Hua Wei. Regexplainer: Generating explanations for graph neural networks in regression tasks. *Advances in Neural Information Processing Systems*, 37:79282–79306, 2024b.

Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. *Advances in neural information processing systems*, 35: 19810–19823, 2022.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond Yes and No: Improving Zero-Shot Pointwise LLM Rankers via Scoring Fine-Grained Relevance Labels . In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)* , 2024.

A APPENDIX

A.1 DATASETS AND METRICS

Following previous work (Luo et al., 2020), we use established experimental settings and metrics, and two widely known synthetic explanation evaluation datasets for node classification: BA-Shapes and Tree-Grids.

A.1.1 DATASETS

In the graph classification setting, we use one synthetic explanation evaluation dataset and one real-world dataset for graph classification: BA-2motif and MUTAG, where MUTAG as a known ground truth. The **BA-Shapes** dataset, described in(Ying et al., 2019), is a single graph created using a base Barabasi-Albert (BA) graph with 300 nodes. To this base graph, 80 "house"-shaped motifs are attached to nodes selected randomly. After attaching these motifs, additional random edges are added to introduce some perturbation in the graph. In BA-Shapes, nodes do not have features. Nodes from the base BA graph are labeled as 0, while nodes located at the top, middle, and bottom of the "house" motifs are labeled as 1, 2, and 3, respectively. In the **Tree-Grids** dataset, the base graph is an 8-level balanced binary tree. To this base graph, 80 motifs of 3x3 grids are attached randomly. The **BA-2motifs** dataset from (Luo et al., 2020), includes 800 graphs. Each graph starts with a BA base graph. Half of the graphs are attached with "house" motifs, while the other half are attached with five-node cycle motifs. The type of motif attached determines the class label of the graph, making it a binary classification task. In these datasets, the ground truth explanations are artificially designed to correspond to the motifs that nodes belong to.

The **MUTAG** dataset consists of 4,337 molecule graphs, each classified into one of two categories based on its mutagenic effect. Each node has a one-hot encoded feature vector of size 14, indicating the chemical element of the node. It is known that carbon rings with chemical groups like NH_2 or NO_2 have a mutagenic effect. As noted in Luo et al. (2020), carbon rings are present in both mutagenic and non-mutagenic graphs, so they are not discriminative. Therefore, we consider carbon rings as shared base graphs, while NH_2 and NO_2 are treated as motifs for the mutagenic graphs, serving as ground-truth explanations. Non-mutagenic graphs, however, do not have clear motifs. In this setup, we train different GNN models on the whole dataset but evaluate explanations only for predictions on mutagenic graphs since they have ground-truth explanations. We define a positive explanation as one that identifies one mutagenic group completely, regardless of the total number of mutagenic groups in that specific graph.

A.1.2 METRICS

For every prediction $\hat{y} = \Phi(\mathcal{G}_c)$ from the test set of the dataset $\mathcal{D}$ made by a GNN Φ on a computation graph $\mathcal{G}_c$, we calculate an explanation subgraph $\mathcal{G}_{EXP}$. Then, we use all the calculated explanation subgraphs on the test set to evaluate the explanation method using various metrics. We use the **Jaccard Index** to measure the overlap between the ground-truth explanation nodes and the $\mathcal{G}_{EXP}$ nodes. Formally, given the set of nodes $\mathcal{V}_T$ from a ground-truth explanation graph and the set of nodes $\mathcal{V}_{EXP}$ from an explanation subgraph, the Jaccard index, also known as the intersection over union (IoU), is defined as:

$$IoU(\mathcal{V}_T, \mathcal{V}_{EXP}) = \frac{|\mathcal{V}_T \cap \mathcal{V}_{EXP}|}{|\mathcal{V}_T \cup \mathcal{V}_{EXP}|}$$

A good explanation method should have a high Jaccard index, indicating that the explanation subgraph is similar to the ground-truth explanation. For a direct comparison with other works (Ying et al., 2019), we also evaluate the models on **Precision**. Given a ground-truth explanation $\mathcal{G}_T$ and $\mathcal{G}_{EXP}$, Precision (Pr) is defined as:

$$Pr(\mathcal{G}_T, \mathcal{G}_{EXP}) = \frac{TP(\mathcal{G}_T, \mathcal{G}_{EXP})}{TP(\mathcal{G}_T, \mathcal{G}_{EXP}) + FP(\mathcal{G}_T, \mathcal{G}_{EXP})}$$

and

where $TP(\mathcal{G}_T, \mathcal{G}_{EXP})$ is the number of true positives, $FP(\mathcal{G}_T, \mathcal{G}_{EXP})$ is the number of false positives.

A.2 Graph Classification Explaination Results

Table 4 reports detailed explanation performance for graph classification across backbone architectures and datasets.

BA-2motif. On this synthetic motif-based dataset, CIExplainer achieves the highest or second-highest IoU and precision for GCN, GIN, and GraphSAGE backbones. In particular, for GCN and GIN, CIExplainer attains the best overall performance. For GraphSAGE, CIExplainer remains competitive with PGExplainer. Across all GNN models, one key feature that can be seen in the examples shown in Fig. 2 is that CIexplainer tends to identify nodes with high degree as relevant for the explanation.

MUTAG. On the real-world MUTAG dataset, results vary depending on the underlying GNN. GNNExplainer achieves the strongest performance for GCN, GAT, and GraphSAGE, whereas CIExplainer performs best under the GIN backbone. The variability across backbones highlights that explanation quality remains dependent on the inductive biases of the underlying GNN model, particularly in more challenging datasets.

Overall, these results suggest that node-induced interventional explanation subgraph identification is particularly effective when predictions are driven by localized structural motifs, as in BA-2motif, while remaining competitive on real-world molecular graphs. The observed backbone sensitivity further indicates that the interaction between aggregation mechanisms and causal perturbations warrants deeper investigation.

Table 4: Explanation results for graph classification. Standard deviation obtained from 10 independents explanations.

	Models	GCN		GAT		GIN		GraphSAGE	
		IoU (↑)	**Pr** (↑)	**IoU** (↑)	**Pr** (↑)	**IoU** (↑)	**Pr** (↑)	**IoU** (↑)	**Pr** (↑)
BA-2motif	Random Explainer	0.1224±.01	0.2020±.01	0.1234±.01	0.2036±.02	0.1235±.01	0.2048±.02	0.1171±.01	0.1942±.02
	IntegratedGradient	0.0022±.00	0.0040±.00	0.0847±.00	0.1520±.00	0.0000±.00	0.0000±.00	0.1652±.00	0.2700±.00
	InputXGradients	0.0111±.00	0.0200±.00	0.1225±.00	0.2120±.00	0.0078±.00	0.0140±.00	0.2815±.00	0.4280±.00
	Shapley Values	0.0158±.01	0.0278±.01	0.0115±.00	0.0206±.01	0.0161±.00	0.0288±.01	0.0224±.00	0.0392±.01
	PGExplainer	0.6369±.08	0.7216±.05	**0.7696±.08**	**0.8272±.07**	0.3859±.02	0.5210±.02	**0.8823±.08**	**0.9240±.06**
	GNNExplainer	0.1144±.01	0.1896±.02	0.0738±.01	0.1264±.02	0.1056±.01	0.1772±.02	0.1129±.01	0.1880±.01
	SubgraphX	0.3015±.02	0.3554±.03	0.3108±.04	0.3676±.03	0.3301±.04	0.3864±.04	0.3192±.03	0.3726±.03
	CIExplainer (Ours)	**0.6745±.02**	**0.7842±.01**	0.1349±.00	0.2292±.01	**0.3921±.01**	**0.5498±.01**	0.7503±.01	0.8336±.01
MUTAG	Random Explainer	0.1107±.01	0.1747±.02	0.1084±.00	0.1719±.01	0.1105±.01	0.1751±.02	0.1090±.01	0.1723±.02
	IntegratedGradient	0.4459±.00	0.5580±.00	0.1202±.00	0.1985±.00	0.1283±.00	0.2040±.00	**0.3025±.00**	**0.4246±.00**
	InputXGradients	0.0827±.00	0.1100±.00	0.1208±.00	0.1748±.00	0.1281±.00	0.1982 ±.00	0.1097±.00	0.1683±.00
	Shapley Values	0.1884±.01	0.2952±.01	**0.1899±.01**	**0.2922±.01**	**0.2138±.02**	**0.3177±.02**	0.2651±.02	0.3846±.02
	PGExplainer	0.0780±.04	0.1200±.06	0.1132±.03	0.1658±.00	0.1000±.01	0.1549±.05	0.1650±.09	0.2343±.13
	GNNExplainer	**0.6251±.02**	**0.7225±.01**	0.1659±.01	0.2467±.02	0.0611±.01	0.0950±.01	0.1872±.01	0.2871±.02
	SubgraphX	0.0808±.01	0.1159±.01	0.0877±.01	0.1259±.01	0.0861±.01	0.1235±.01	0.0863±.01	0.1240±.02
	CIExplainer (Ours)	0.1160±.01	0.1844±.02	0.1314±.01	0.2141±.02	0.1751±.01	0.2580±.01	0.1213±.01	0.1890±.01

A.3 Inference Time

Across both node and graph classification we present the averaged inference cost per node. The most direct comparison of CIExplainer is with other perturbation-based methods, such as GNNExplainer Ying et al. (2019), and in those cases our proposed algorithm is faster. We note that PGExplainer is a slight different approach, as is it is trained on each dataset separately, to learn a subgraph that maximizes Mutual Information. Since it requires training, it cannot be directly compared with other methods. Finally, gradient based-methods, for the models used in this work, have a very low inference-cost, since their cost scales with the size of the computation graph rather than with the number of candidate subgraphs or coalitions that must be evaluated. And the underlying GNN-architecture used in this work are quite small.

We note that gradient based methods are necessarily faster in this setting.

Table 5: Computational cost comparison. Train time is reported once per explainer, with inference cost average per model and dataset. Inference time is averaged per explained node. All experiments run on a 10GB partition of a single NVIDIA A100 GPU.

Method	**Train Time (sec.)**	**Inference (sec./node)**
Random Exp.	N/A	0.0003 ± 0.0
PGExplainer	88.2	0.0025 ± 0.00005
IntegratedGradient	N/A	0.258 ± 0.0235
InputXGradient	N/A	0.214 ± 0.0236
Shapley Values	N/A	38.064 ± 0.790
GNNExplainer	N/A	1.314 ± 0.0304
SubgraphX	N/A	7.866 ± 0.512
CIExplainer	N/A	$\mathbf{0.553} \pm 0.0113$

A.4 Counterfactual Generation

As seen in 1, for each feature x_i, a counterfactual is generated, and the model-level interventional effect CE_i is then computed. In this section we expand on the counterfactual generation process, which is non trivial, apart from binary features.

A.4.1 Binary Features

For binary features, the space of possible interventions is inherently discrete and limited to two states. Let $x_i \in \{0, 1\}$ denote a binary a feature. The complement is $x_i' = 1 - x_i$. Consequently, unlike continuous or multi-valued categorical features, binary features do not require sampling over a range of plausible values, as only a single alternative state exists.

However, in realistic settings, dependencies with other features and feasibility constraints may require additional adjustments, that distinguish intervenable feature level nodes from features where the intervention is not possible.

A.4.2 Discrete Features

In this setting, the space of possible interventions is a finite set of discrete values. Let $x_i \in \mathcal{X}_i$, where $\mathcal{X}_i = \{v_1, \ldots, v_K\}$ denotes the finite domain of the feature i. A feature-level counterfactual intervention corresponds to replacing the observed value x_i with an alternative value $x_i' \in \mathcal{X}_i \setminus \{x_i\}$.

While the space can exhaustively evaluated to estimate the maximum model-level interventional effect, doing so might be computationally infeasible for large sets, and higher number of features. Due to this limitation we employ a sampling scheme, that samples from the prior marginal distribution of the feature. Since, for the graph classification case, the prior information might be small and unrealiable, we perform laplace smoothing, mixing the the prior distribution with a uniform distribution. This guarantees that even in small datasets, all values can be sampled from.

Following the same approach as in the binary feature case, not all features can receive an intervention. Moreover, changing just one feature might obtain an out-of-distribution feature vector. For small feature spaces, it is possible to sample from the empirical joint distribution, guaranteeing that the sample is typically found in the distribution, however the joint support size scales exponentially with the number of features, making it computationally costly to sample from this distribution.

Algorithm 2 Discrete Counterfactual with Laplace Smoothing

Input: Feature index j, feature vector $\mathbf{x}$, marginal distribution $\mathcal{M}$, Domain $\mathcal{X}$,smoothing parameter ϵ
Output: feature vector $\mathbf{x}'$ with counterfactual perturbation
obtain current value $v \leftarrow x_j$
valid values $\mathcal{V}' \leftarrow \mathcal{V}_j \setminus \{v\}$
new distribution $\mathbf{p}' \leftarrow \mathbf{p}$ without v
for each k in $\mathbf{p}'$ **do**
    $p'_k \leftarrow p'_k + \epsilon \frac{1}{|\mathcal{V}'|}$
end for
Normalize $\mathbf{p}'$
Sample $v' \sim \text{Categorical}(\mathcal{V}', \mathbf{p}')$
$x'_j \leftarrow v'$
return $\mathbf{x}'$

In Algorithm 2 we showcase how the sampling is perform for one feature in the feature vector, using the marginal distribution. This algorithm is then used for each feature, and repeated m times, from which the model-level interventional effect is extracted.

A.4.3 Continuous Features

In the continuous setting, the space of possible interventions is an uncountable set. Let $x_j \in \mathcal{X}_j \subseteq \mathbb{R}$, where $\mathcal{X}_j$ denotes the domain of the feature. A feature-level counterfactual intervention corresponds to replacing the observed value x_j with an alternative value $x'_j \in \mathcal{X}_j$.

Unlike the discrete case, the intervention space cannot be exhaustively evaluated. As a result, estimating the maximum model-level interventional effect requires sampling-based approximations. In this work, we adopt a sampling scheme, where candidate counterfactuals are drawn from a continuous marginal distribution associated with the feature. For CIExplainer, this marginal distribution can be a bounded uniform distribution, an empirical Gaussian Distribution, a t-Student distribution with heavier tails, or Gaussian Mixture model.

To ensure sufficient and efficient exploration, the algorithm samples from the distribution, and then performs a distance or distribution based filter.

Following the approach in Algorithm 2, we intuitively extend the approach to the continuous case, where the number the number of components in the gaussian mixture model is akin to the discrete set. In Algorithm 3 we also include a filter to guarantee that the obtained sampled is unlikely to be from the original gaussian distribution.

Algorithm 3 Continuous Counterfactual via Bayesian Gaussian Mixture

Input: Feature index j, feature vector $\mathbf{x}$, original value v, BGM model $\mathcal{G}$ with k components
Output: feature vector $\mathbf{x}'$ with counterfactual perturbation
Compute posterior probability given $\gamma_k(v) = P(z = k \mid v)$
Identify component: $k^* \leftarrow \arg\max_k \gamma_k(v)$
Create perturbed model $\mathcal{G}'$ without component k^*
Sample $v' \sim \mathcal{G}'$
Accept with probability:
    $p_{\text{acc}} \leftarrow 1 - \gamma_k(v')$
If accepted, set $x'_j \leftarrow v'$; otherwise repeat step 6
return $\mathbf{x}'$

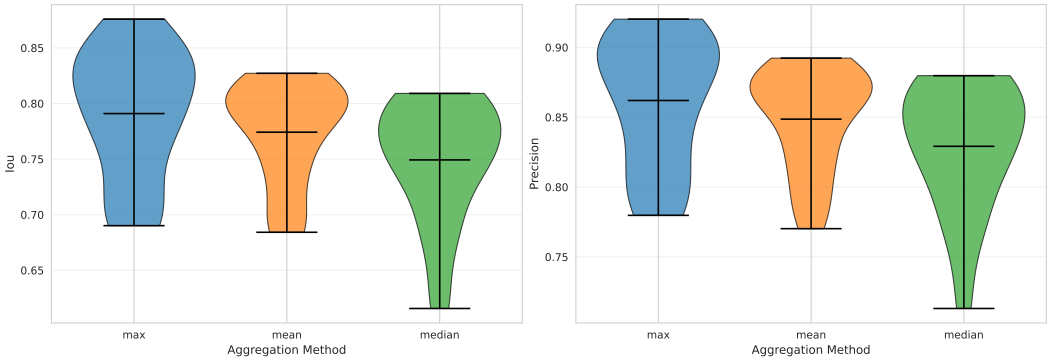
A.5 Ablation Studies

A.5.1 Aggregation

On this ablation study we evaluate the aggregation method over multiple conterfactual samples. For a given sampling budget, then the obtained model-level interventional effect for feature can be

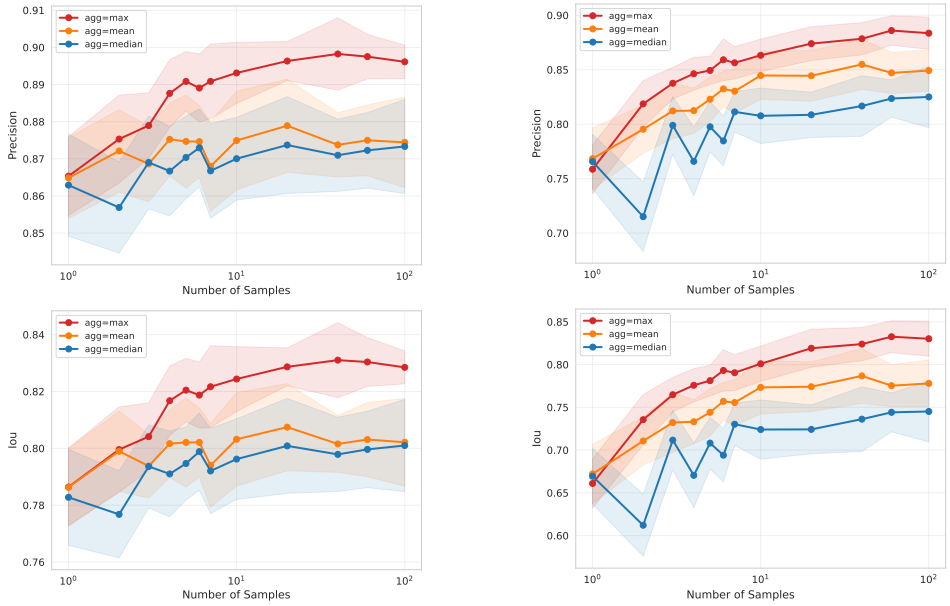
aggregate using maximum, mean or median. Figure 3 seems to indicate that taking the maximum value for each feature yields the best results. While the results are all similar, as expected, using the maximum value has better results across both metrics.

We also can observe, in Figure 4, the different aggregation methods across different sampling budgets. To perform an extensive and robust analysis, 5 independent evaluation procedures are performed at each sampling budget, from 1 to 100, at spaced intervals. As expected, increasing the number of samples improves the quality of explainer. We can also conclude that at around 10 samples, the improvements are less significant. Finally, it is clear that using the maximum as the aggregation function is superior for all number of samples.



(a) IoU (b) Precision

Figure 3: Comparison of aggregation method results for CE. The values are calculated across all classification models, for all datasets in the node classification task.



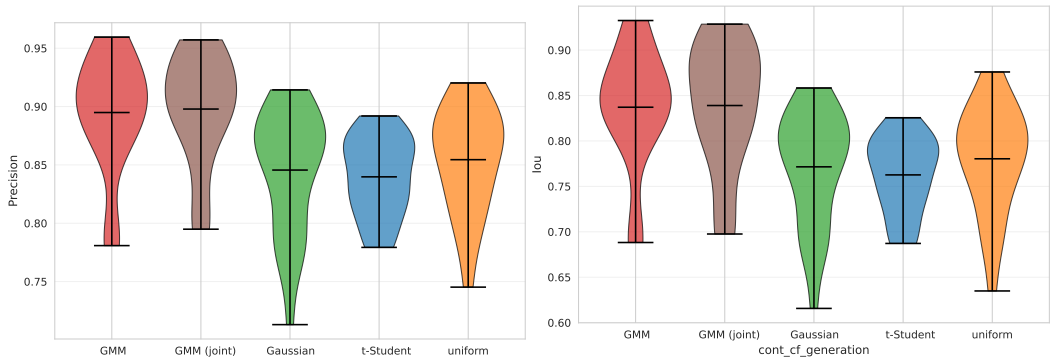
(a) Tree-Grid Dataset (b) BA-Shapes

Figure 4: Comparison of aggregation method results for CE. The values are calculated across all classification models, for all datasets in the node classification task.

A.5.2 CONTERFACTUAL DISTRIBUTION

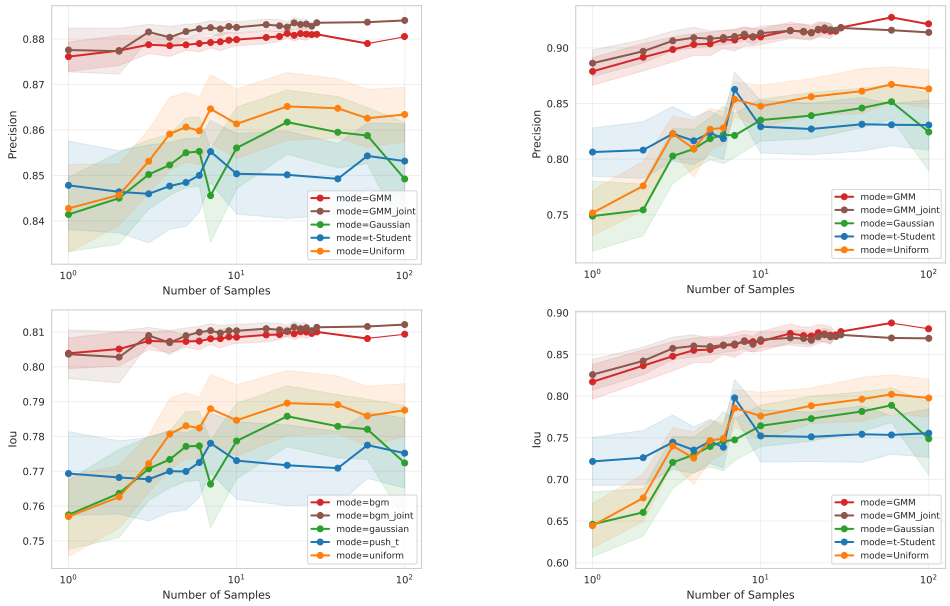
As mentioned in A.4.3, there are different distributions from which to sample counterfactuals. By choosing a specific distribution, it implies assumptions about the underlying data distribution and, consequently, affects the properties of the generated counterfactuals. This section presents an ablation study on the impact of this choice. To do so, we evaluate different counterfactual distributions ($\mathcal{G}$) from which to sample from, for the datasets with continuous features.

In Figure 5 it is clear that the Gaussian Mixture Model (GMM) as a better counterfactual model than the simpler gaussian, uniform or t-student, for a fixed number of samples, using the maximum as the aggregation function. Figure 6 further reinforces this observation, by indicating that the GMM is better across all samplings budgets, including $N = 1$. This shows that the GMM is able, with some success, to correctly represent the feature space. In this ablation study we also evaluate two different approaches, sampling from the marginal distribution and sampling from the joint distribution. For the given datasets, the results are very similar, with some slight advantage to the joint approach, as expected. While we do not extrapolate this finding to more complex feature spaces, it does shows that, for this cases, sampling from the marginal is an acceptable simplification that can be applied with very limited downside.



(a) IoU (b) Precision

Figure 5: Comparison of aggregation method results for CE. The values are calculated across all classification models, for all datasets in the node classification task.



(a) Tree-Grid Dataset

(b) BA-Shapes

Figure 6: Comparison of sampling distributions results for CE. The values are calculated across all classification models, for all datasets in the node classification task.

A.6 Graph Neural Networks Training Results

The main objective of this paper is to propose a causal explanation method for GNNs, that achieves both high explainability but is also interpretable, in the sense that someone can understand the finding of the explanation model. To do this, we trained four different GNNs models for each dataset, using very simple datasets with ground truth labels. While there are more realistic and complex datasets, those datasets lack proper ground-truth label, and the evaluation of the explainers becomes intertwined with the predictive capabilities of the model themselves.

The datasets chosen are commonly used for GNN explainability, and due to their simplicity allow to isolate explainability component. For fairness, we fix the overall model architecture and all hyperparameters within each task setting, with the only difference being the type of layer used in the model. For each dataset, we trained a GCN, a GraphSAGE, a GAT, and a GIN model, using pytorch geometric (Fey et al., 2025). Table 6 shows the configuration of GNNs models for each task. For all datasets, we adopt a random 80/10/10 split for training, validation, and testing, respectively.

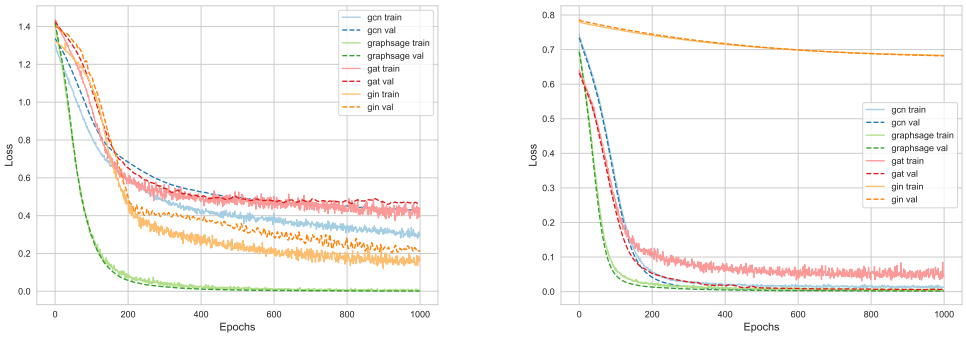
Table 6: GNN model configuration per task.

	Node Classification	Graph Classification
# Layers	3	3
Hidden Dimension	20	20
Pooling Layer	-	Max
Criterion	CE or BCE	BCE
Optimizer	Adam	Adam
Learning Rate	$1e^{-3}$	$1e^{-3}$
# Epochs	1000	1000
Train/Val/Test Split	80/10/10	80/10/10

A.6.1 NODE CLASSIFICATION

We measure model training performance on all datasets using the loss. As shown in Fig. 7, GraphSAGE exhibits the fastest convergence and consistently achieves the lowest final training loss. In contrast, GIN fails to converge on the Tree-Grid dataset. This finding is complemented in Table 7, where the graphSAGE models achieves perfect accuracy in both test sets, and GIN has a subpar accuracy of 0.5887.

Figure 7: Training Loss for the node classification task. Loss is presented for the training and validation sets. Across the models we consistently observe GraphSage as the fastest and more consistent model to achieve close to zero loss, while GIN fails to properly optimize in these tasks.



(a) BA-Shapes (b) Tree-Grid

Table 7: Node classification test metrics.

	BA-Shapes			Tree-Grid		
Model	Loss	Acc	F1	Loss	Acc	F1
GCN	0.3606	0.9000	0.8689	0.0083	**1.0000**	**1.0000**
GraphSAGE	**0.0005**	**1.0000**	**1.0000**	**0.0033**	**1.0000**	**1.0000**
GAT	0.4138	0.8286	0.7423	0.0195	0.9839	0.9832
GIN	0.1733	0.9571	0.9427	0.6827	0.5887	0.3706

A.6.2 GRAPH CLASSIFICATION

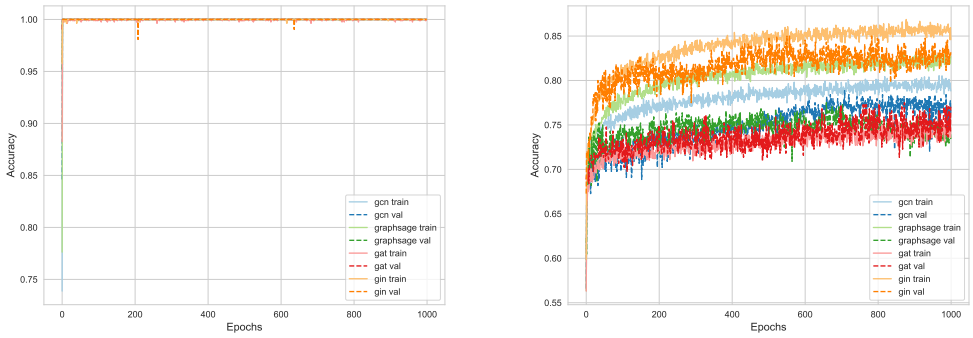
The results for graph classification are similar to node classification in the synthetic graph classification dataset BA-2motifs in Fig. 8 and Table 8. The perfect accuracy results are to be expected, since the features of these datasets are specially created to make it easier for the model to learn and predict. GraphSAGE again has the lowest loss on the test set, and trains the fastest.

Table 8: Graph classification test metrics.

	BA-Shapes			Tree-Grid		
Model	Loss	Acc	F1	Loss	Acc	F1
GCN	0.0000	1.0000	1.0000	0.4972	0.7894	0.7461
GraphSAGE	0.0000	1.0000	1.0000	0.6229	0.7545	0.6865
GAT	0.0002	1.0000	1.0000	0.5705	0.7227	0.6841
GIN	0.0000	1.0000	1.0000	**0.4321**	**0.8182**	**0.7958**

For the MUTAG dataset, we can see in Fig. 8 that the models learn more slowly and do not perform as well on the validation set. The reason is that the features in the MUTAG dataset are not specifically designed to help the model learn and make predictions more easily. Therefore, the quality of the

Figure 8: Training accuracy for the graph classification task. Accuracy is presented for the training and validation sets.



(a) BA-2Motif

(b) MUTAG

predictions is more dependend on the GNN aggregation layer. Even so, for the MUTAG dataset, GIN is the GNN that performs best on the training and validation sets, with the least overfitting. The test performance in Table 8 shows that the models generalize well to unseen data but do not achieve perfect metric scores. This is expected since the MUTAG dataset is more complex and does not have a feature set designed to facilitate model learning and prediction, and it can influence the explanation evaluation, since it is not expected that a model that wrongly classifies the graph will have a correct explanation subgraph.

A.7 G2TeXplainer Generation and Evaluation Details

Generation Configurations. To generate natural language explanations, we use LLaMA 3 8B (Grattafiori et al., 2024) with stochastic decoding via nucleus sampling. We set the temperature to 0.7 to balance coherence and variability, and use a top-p of 0.9 to restrict token selection to high-probability candidates. A mild repetition penalty of 1.1 is applied to improve fluency and reduce redundancy. Random seeds are controlled to ensure reproducibility across samples while allowing some variation between generations. For each example, we produce 30 explanations across 20 **BA-2motifs** instances. In addition, we evaluate on the full test set using a reduced sampling budget of 5 explanations per instance; the corresponding results are reported in Appendix A.8.

Evaluation Criteria. We evaluate each natural language explanation using five complementary criteria: (1) **Node Fidelity**, which evaluates whether the explanation assigns structural importance only to the ground-truth subgraph nodes, and avoids overclaiming importance to other nodes; (2) **Structure**, which evaluates whether the explanation accurately describes the detected subgraph motifs (e.g., cycles, triangles) and correctly identifies which nodes belong to each motif, without inventing structures; (3) **Clarity**, which evaluates how coherently and unambiguously the explanation communicates the reasoning behind why the selected nodes support the GNN's prediction; (4) **Semantic Similarity**, which assesses consistency across independent runs and is computed as the average of TF-IDF lexical similarity and embedding-based semantic similarity; and (5) **Pairwise Preference** which also relies on an LLM-as-judge protocol, where two explanations for the same instance are compared with randomized order to avoid positional bias and the judge selects the preferred explanation (Fathullah & Gales, 2025). The LLM-as-judge evaluations are conducted using Gemma 3 27B (Team, 2025), which scores the relevant criteria on a scale from 1 (very poor) to 5 (perfect) (Zhuang et al., 2024).

In addition to the automatic evaluations, we conduct a human evaluation study to assess the factual consistency and quality of the generated explanations. Explanations are evaluated according to four binary factual criteria: Correct Class, which assesses whether the explanation identifies the predicted class; Correct Nodes, which evaluates whether the relevant nodes are correctly identified; Correct Structure, which assesses whether the described structural relationships are consistent with the target motif; and No Hallucination, which measures whether the explanation avoids introducing unsupported graph elements or structural claims. These four criteria are summed to obtain a Factual Consistency Score then mapped to a 1–5 scale. In addition, evaluators assess, Clarity, and Overall Quality using 1–5 Likert scales. Clarity measures the readability and understandability of the explanation, and Overall Quality captures the evaluator's holistic assessment of the explanation.

A.8 G2TeXplainer Extra Evaluation

Table 9: Evaluation of G2TeXplainer against GraphXAIN on the full test set (5 samples per example). Judge-based metrics are scored from 1 (very poor) to 5 (perfect). **N. Fid.**: correctness of node importance; **Str.**: accuracy of detected motifs and node memberships; **Clar.**: coherence and interpretability of the explanation;

Prompt	N. Fid.	Str.	Clar.
GraphXAIN	3.568	3.296	3.762
P4 Motif-Aware + R.	4.100	3.882	3.436

GraphXAIN achieves slightly higher scores on the full test set see table 9. However, GraphXAIN's responses are, on average, around 40 words longer than those produced by our method. This is important because prior work has identified a known bias in LLM-based evaluation favoring longer responses Hu et al. (2025); Ye et al. (2025), particularly in pairwise "LLM-as-judge" comparisons. More recent studies suggest that this length bias may also persist even when LLM judges score responses independently rather than comparatively Yu et al. (2026).

Figure 9: Distribution of LLM-as-judge Node Fidelity scores for different graph types. Each cell shows the number of explanations (and percentage) receiving a specific Node Fidelity score from 1 (low Fidelity) to 5 (high Fidelity).

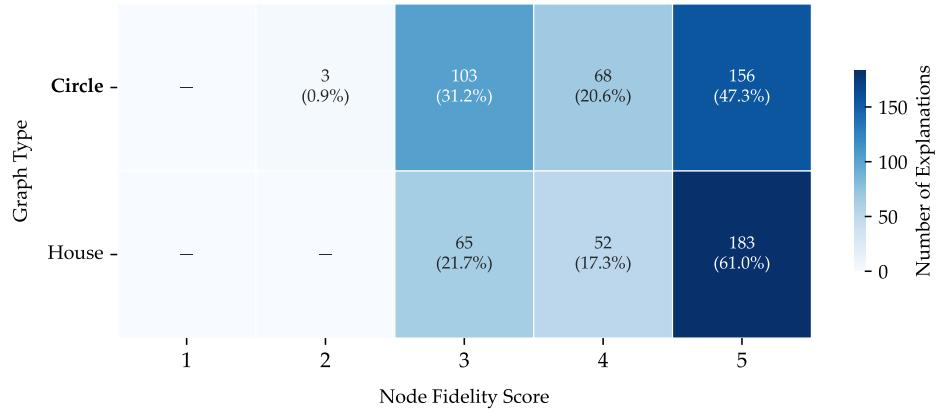


Figure 10: Distribution of LLM-as-judge Structural scores for different graph types. Each cell shows the number of explanations (and percentage) receiving a specific structural score from 1 (poor structure) to 5 (well-structured).

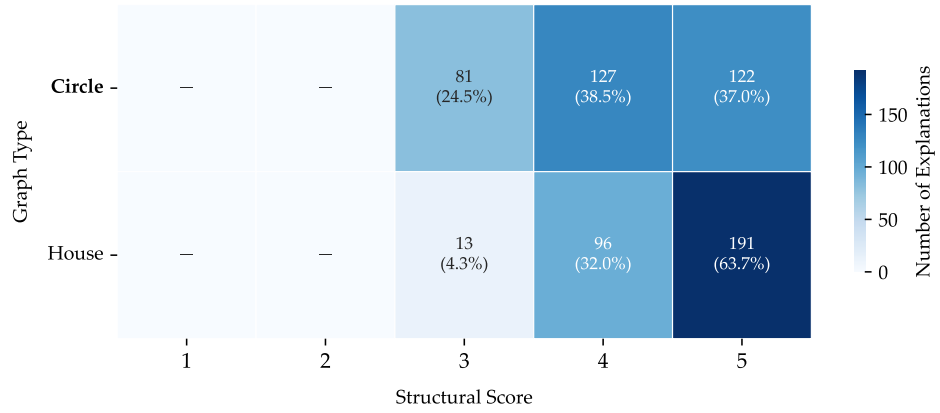


Table 10: Evaluation of the P4 prompt in G2TeXplainer on a subset of the test set (20 examples, with 30 generated explanations per example) across different GNN backbones. **Sim.**: which assesses similarity across independent runs. Judge-based metrics are scored from 1 (very poor) to 5 (perfect). **N. Fid.**: correctness of node importance attribution; **Str.**: accuracy of the identified motifs and node memberships; **Clar.**: coherence and interpretability of the explanation. Bold indicates the best result and underlined values indicate the second-best result.

	Prompt		LLM-as-Judge		
		Sim.	**Str.**	**Clar.**	**N. Fid.**
GraphSAGE	P4 Motif-Aware + R.	0.751	4.143	4.340	<u>3.763</u>
GCN	P4 Motif-Aware + R.	**0.758**	4.262	<u>4.348</u>	**3.789**
GAT	P4 Motif-Aware + R.	0.744	**4.354**	**4.403**	3.538
GIN	P4 Motif-Aware + R.	<u>0.754</u>	<u>4.276</u>	4.271	3.724

Table 10 shows consistent P4 performance across GNN backbones, with only minor variations in **Similarity**, **Structure**, and **Clarity**. **Node Fidelity** varies somewhat more, which is expected given its dependence on the underlying model's node-importance patterns. Overall, the results suggest that P4's effectiveness is largely robust to the choice of GNN architecture.

A.9 G2TEXPLAINER P4 PROMPT

Fine-tuned prompt used for explanation description.

```
You are analyzing why a Graph Neural Network (GNN) identified
specific nodes as important for its prediction.

TASK:

You are NOT asked to explain why the graph is a "{label}".
You are asked to explain whether the EXPLAINABLE NODES support
that prediction.

Rules:
Only discuss nodes in {explain_nodes}.
Do NOT discuss nodes that were not selected.
Do NOT explain the full graph structure.
Focus only on whether the selected nodes align with the
"{label}" motif.

Instructions:

Identify which of the selected nodes belong to the listed
structures.
If selected nodes are part of a listed 4-cycle or 5-cycle,
explain how that supports the "{label}" pattern.
If selected nodes are isolated or not part of the listed
structures, state that they do NOT support the prediction.
Assess whether the explanation nodes meaningfully justify the
"{label}"prediction.
STRUCTURAL DEFINITIONS:
CIRCLE PATTERN:
Definition: A connected 5-node cycle
HOUSE PATTERN:
Definition: 4-cycle with triangle
NONE PATTERN:
The graph does not contain either the Circle or House
structural motifs.
GNN PREDICTION: "{label}"
EXPLAINABLE NODES: {explain_nodes}

STRUCTURES DETECTED IN SUBGRAPH:
{structure_text}

Write 4{6 analytical sentences.
Explanation:
```