Faithful Explanations for Graph Classification using Logic

Alessio Ragno[1], Marc Plantevit[2], and Céline Robardet[1]

[1] INSA Lyon, CNRS, LIRIS UMR 5205, F-69621 Villeurbanne, France
{alessio.ragno,celine.robardet}@insa-lyon.fr
[2] EPITA Research Laboratory (LRE), FR-94276, Le Kremlin-Bicêtre, France
marc.plantevit@epita.fr

Abstract. Most post-hoc explainability methods for graph classification analyze the model's internal representations rather than explicitly capturing its reasoning process. These approaches typically rely on perturbations, gradients, or optimization techniques to infer important features but do not approximate the decision-making function itself. In this paper, we propose a novel approach that directly models the GNN's decision function using a Transparent Explainable Logic Layer (TELL). This logic-based approximation enables both instance-level and global-level explanations, offering insights into how node embeddings contribute to predictions. Unlike conventional methods, our approach derives explanations that are structurally aligned with the model's decision process rather than being externally imposed. Through experiments on synthetic and real-world graph classification tasks, we show that our method produces faithful, sparse, and stable explanations, outperforming existing techniques.

Keywords: Graph Neural Networks · Explainability · Interpretability · Logic

1 Introduction

Graph Neural Networks (GNNs) have become a fundamental tool for learning representations from graph-structured data, enabling breakthroughs in tasks such as node classification, link prediction, and graph classification [4, 12, 13, 22]. However, due to the intricate nature of graphs and the implicit feature aggregation mechanisms employed by GNNs, understanding their predictions remains a significant challenge. This lack of interpretability raises concerns about their deployment in high-stakes applications, such as drug discovery, financial risk assessment, and social network analysis. To address this issue, Explainable AI (XAI) provides methodologies to enhance the transparency of GNNs and offer human-interpretable explanations for their decisions.

These techniques fall into two categories: self-explainable models [7, 18, 19, 30], designed to be inherently interpretable, and post-hoc techniques, which explain predictions of pre-trained models. While self-explainable models require

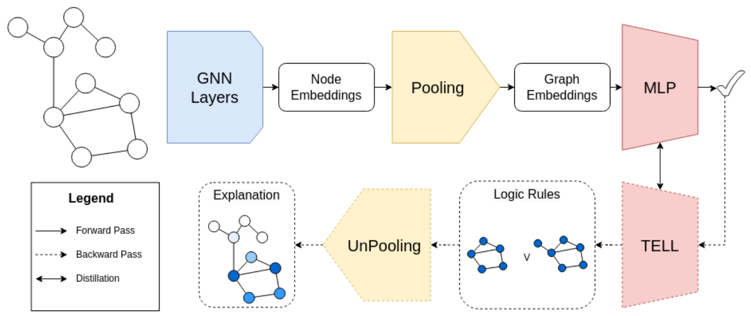


Fig. 1: LogiX Explanation Procedure. A Transparent Explainable Logic Layer is used to approximate the MLP of a GNN. Subsequently, logic rules are extracted, and representative graphs are obtained. Finally. instance-level explanations are obtained by identifying the contributions of the nodes to the logic rules.

specialized architectures and complex training, post-hoc methods can be applied to any pre-trained GNN. Existing post-hoc methods typically rely on input perturbation [11], optimization of masks to select most influential subgraphs [16, 27], and extraction of gradients for the model outputs with respect to the inputs [23, 24]. These methods identify the key nodes, edges, or substructures driving a model's prediction. While effective, these methods often suffer from high computational costs or lack inherent interpretability.

In this paper, we introduce LogiX, a novel approach to graph classification explainability based on a surrogate model for logic-based analysis of the GNNs. Contrary to existing surrogate methods, which utilize simple and explainable models in the locality of a single data point, we utilize a transparent explainable logic layer (TELL) [20] to globally approximate the decisions of the last layer of the GNN. This enables the identification of rules in the embeddings that drive class predictions. Once these rules are identified, we can trace them back through the GNN architecture to pinpoint the nodes responsible for activating specific rules. This approach allows us to generate explanations at both the instance level, using node attribution masks, and logic level, by visualizing the learned rules. Fig. 1 reports a general scheme of the procedure we use: given a GNN for graph classification, we approximate its final classifier using TELL; successively we extract logic rules to explain the classes; finally, we generate node attribution masks to highlight the nodes that contribute to a specific class. Through extensive experimentation, we demonstrate that LogiX achieves comparable or superior explanation performances compared to state-of-the-art post-hoc techniques for producing instance-level explanations. In particular, we show that LogiX, compared to others provides significant improvements in identifying the most important nodes and the least important ones. To demonstrate this, we propose an additional experiment to analyzing the fidelity and the stability of the explanations, when removing the most relevant nodes or the least relevant ones.

Beyond the post-hoc instance-level explanations, we show that it is also possible to use the trained logic layer to visualize rules over the graphs for a broader understanding of the model behavior. We compare our approach with a recent state-of-the-art method, GraphTrail [2]. While GraphTrail can only be applied in the case of discrete node features, our approach is also applicable to graphs with all node feature types.

In summary, our work makes the following contributions: we introduce a logic-based post-hoc explanation method for graph classification; we extensively analyze our proposed method's explanations quality alongside state-of-the-art post-hoc methods for instance-level explanations, specifically focusing on the identification of most important nodes; we compare our approach against logic-based explainers; we provide an open-source implementation of our proposed approach and the conducted experiments[3].

The remainder of this work is structured as follows: We begin by surveying the current literature on XAI for GNNs and logic explainability; subsequently, we detail the mathematical foundations needed to comprehend our approach; we then present our proposed approach to derive logic-based explanations for graph classification; we follow with an experimental evaluation of the explanations on several datasets; finally, we summarize our findings and suggest potential directions for future research.

2 Related Work

We review related literature, focusing on post-hoc techniques for explaining graph classification models and logic-based XAI approaches that enhance interpretability.

2.1 Explaining GNNs

Explanations for graph classifications are generally obtained by attributing importance scores to nodes, edges, or features in a given graph. Many existing methods achieve this by computing importance scores using gradients, optimization, or perturbations.

Gradient-based approaches like GradCAM [23] and Integrated Gradients [24] adapt techniques from convolutional neural networks to GNNs, computing gradients to highlight influential node features and providing saliency-based explanations. Despite the adaptability and the speed of these methods, they often provide noisy explanations due to gradient saturation and may struggle to correctly determine node importance.

Optimization-based approaches, instead, try to optimize masks to determine the nodes that are most important for a certain prediction. GNNExplainer [27] is the seminal method, formulating the explanation process as an optimization problem where a soft mask is applied to node features and edges to maximize

[3] Public GitHub repository: https://github.com/spideralessio/logix

the mutual information between the masked graph and the model's original prediction. PGExplainer [16] builds upon this idea but introduces a probabilistic approach to generate multiple discrete explanations rather than a single deterministic one, training an edge mask predictor to sample different possible explanations.

Another line of work explores perturbation-based approaches. These approaches assess the importance of graph components by analyzing how structural modifications impact model predictions. SubGraphX [28], for instance, evaluates the collective contribution of subgraphs rather than isolated nodes or edges, by employing Shapley values. To efficiently navigate the vast space of possible subgraphs, it uses Monte Carlo Tree Search (MCTS), balancing exploration and exploitation during the search process. Despite its effectiveness, the MCTS-based search can become computationally intensive for large graphs. GraphSVX [11] introduces a more efficient approach by approximating Shapley values through a combination of graph perturbations and surrogate modeling. Instead of exhaustively exploring subgraph combinations, GraphSVX leverages a sampling strategy that reduces computational complexity while still providing high-fidelity explanations. This method balances the trade-off between accuracy and scalability, making it suitable for larger graph datasets. To address the remaining limitations in scalability, GStarX [29] refines the search strategy by focusing on star-shaped subgraphs centered around key nodes. This targeted exploration reduces computational overhead while maintaining the quality and fidelity of the explanations, making it more scalable to complex graphs.

In this paper, we propose a novel approach which does not fall within traditional techniques. Our aim is instead to integrate a logic-based layer that approximates the model. This provides an explanation process that is directly tied to the reasoning mechanisms of the GNN rather than being an external attribution process. This allows to attribute the nodes using the specific logics of the models. With this aim, in the next section, we present related works regarding the use of logic in XAI.

2.2 Logic-based XAI

Logic-based explainability methods [8] aim to enhance model interpretability by embedding logical constraints or symbolic reasoning into the learning process. Unlike post-hoc explanation techniques, which analyze model predictions after training, logic-based approaches integrate interpretability directly into the model's architecture, ensuring that explanations are inherently aligned with the model's decision-making process. This allows for the development of self-explainable techniques that provide explanations along with predictions [15, 21].

Classic machine learning models, such as decision trees and rule-based classifiers, naturally provide logical explanations by representing decision boundaries in an explicit and interpretable manner. However, while these models offer transparency, their performance is often limited when dealing with complex, high-dimensional data. Additionally, they cannot be easily integrated within deep learning architectures due to their lack of differentiability, preventing them from benefiting from gradient-based optimization. This challenge has motivated the development of neural models that embed logical reasoning while remaining trainable via backpropagation.

One notable example is logic-explained networks (LENs) [5], which introduce a family of neural networks that can be explained through logic rules. These models operate under the assumption of binary data and apply strong regularization, enabling the construction of truth tables to derive logic rules. However, this approach does not guarantee that the logic explanations accurately reflect the model's behavior [20]. To address this limitation, the transparent explainable logic layer (TELL) was introduced—a novel architecture constrained by positive weights that can be directly converted into logic rules. These constraints ensure a direct alignment between the extracted logic rules and the model's actual behavior. Additionally, TELL is also designed to work with continuous data thanks to a preprocessing function that automatically learns thresholds over input features.

In the context of GNNs, logic-based explainability is still an emerging field and is mainly used in model-level explanations rather than instance-level ones. GLGExplainer [3] is one of the first attempts to introduce logical rules into GNN explanations. It is an architecture that takes explanations generated by another post-hoc method and combines them into a logic formula using LENs. Another recent approach, GraphTrail [2], derives logic-based explanations on the computation trees of the graphs. With GraphTrail, the authors show that they are able to produce high faithful explanations compared to the ones of GLGExplainer, that depend on other post-hoc approaches. However, the calculation of computation trees requires that the node are either of specific types or contain discrete features.

In this work, we differentiate from these methods by proposing a novel instance-level approach that approximates the last layer of a pre-trained GNN architecture with TELL. This procedure allows for the extraction of model-level rules over the embedding activations. Although the nature of TELL is self-explainable, in this work we use it to approximate a GNN model that is not inherently trained with this objective. For this reason, the rules extracted by TELL might not share a human-comprehensible semantic meaning. However, we show that it is possible to derive model-level explanations by retrieving representative sub-graphs of the nodes that activate for a certain rule. Most importantly, we show that we can ultimately use the rules to identify the most important nodes involved in the prediction of a certain class, allowing us to generate instance-level explanations.

3 Background

As LogiX deeply relies on GNN internal mechanisms and on Transparent Explainable Logic Layer, we introduce below the necessary notation.

3.1 Graph Neural Networks

GNNs are a class of neural networks specifically designed to process data structured as graphs. They operate using a message-passing framework, which consists of two core operations: aggregation and update. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ the set of edges, the aggregation step gathers information from neighboring nodes, while the update step refines the representation of the current node based on this information. A general GNN layer is defined as follows:

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{1}$$

$$h_v^{(k)} = \text{UPDATE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right) \tag{2}$$

where $v \in \mathcal{V}$ is the target node, $h_v^{(k)}$ represents its feature vector at layer k, and $\mathcal{N}(v)$ denotes its neighboring nodes.

Different GNN variants implement this framework with diverse aggregation strategies. In this work, we use the Graph Isomorphism Network (GIN) layer [26], which derives its aggregation mechanism from the Weisfeiler-Lehman test for graph isomorphism. The representation of a node in the GIN layer is obtained by summing the representations of its neighbors from the previous layer, adding its own representation, and then passing the result through a multi-layer perceptron (MLP):

$$h_{v,\text{GIN}}^{(k)} = \text{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right). \tag{3}$$

GNNs typically stack multiple layers to learn hierarchical node representations, which can be utilized for tasks such as node classification. For tasks at the graph level, such as graph classification, a readout layer (e.g., global pooling) is applied to the node embeddings to obtain a single representation for the entire graph. This readout operation can involve summing, averaging, or computing the maximum of the node embeddings. Finally, a classification layer maps this representation to a predicted class label.

3.2 Transparent Explainable Logic Layer

TELL is a particular layer designed to be directly translated into logic rules by constraining a feed-forward transformation with non-negative weights and a specific thresholding mechanism. Given an input $X \in \mathbb{R}^I$ and an output $y \in \mathbb{R}^O$, the transformation in TELL follows:

$$y = \sigma(XW^+ + b) \tag{4}$$

where $W^+ \in \mathbb{R}_{\geq 0}^{I \times O}$ is a weight matrix constrained to be non-negative, $b \in \mathbb{R}^O$ is a bias vector, and σ is the sigmoid activation function. The primary characteristic of TELL is that its outputs can be directly interpreted as logical conditions. By defining the binary activation of the output neurons as

$$y_k^{\text{bin}} = \begin{cases} 1 & \text{if } y_k > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

we can express the decision boundary for each output neuron as a disjunctive normal form (DNF) rule:

$$y_k^{\text{bin}} = 1 \iff \sum_{i \in S} W_{ik}^{+} > -b_k, \quad S \subseteq \{1, \dots, I\} \tag{6}$$

where S represents the subsets of input features, minimal w.r.t set inclusion, that activate y_k. The logical formula E_k associated with y_k is then given by:

$$E_k = \bigvee_{S \in \mathcal{S}_k} \bigwedge_{i \in S} x_i \tag{7}$$

where $\mathcal{S}_k$ is the set of all minimal subsets satisfying the threshold condition. This formulation ensures that each neuron in TELL corresponds to a logic rule, enabling a direct mapping from neural parameters to symbolic expressions.

TELL can also be extended to handle real-valued inputs by incorporating an automatic thresholding mechanism. Given a preprocessing function:

$$X' = \sigma(X \odot \exp(\tilde{W}_i) + \tilde{b}_i) \tag{8}$$

where $\odot$ is the Hadamard element-wise product, and $\tilde{W}_i$ and $\tilde{b}_i$ are learnable parameters. With this approach, the thresholded input features X' remain interpretable as binary predicates:

$$x_i' = 1 \iff x_i > -\frac{\tilde{b}_i}{\exp(\tilde{W}_i)} \tag{9}$$

which are subsequently used within the logic rule extraction process. This allows TELL to process and explain real-valued inputs while maintaining the interpretability constraints.

4 Explaining GNNs with LogiX

Let f be a Graph Neural Network (GNN) composed of the following three functions:

- A set of L GNN layers $g = \{g_1, \dots, g_L\}$, where each layer g_ℓ maps node embeddings from one representation space to another:

$$g_\ell : \mathbb{R}^{N \times D_\ell} \to \mathbb{R}^{N \times D_{\ell+1}} \tag{10}$$

  where N is the number of nodes in the graph, and D_ℓ is the embedding dimension at layer ℓ. For simplicity we set $D_\ell = D$ for all the layers.

- A readout function implemented through a global pooling operator p that aggregates the node embeddings into a graph-level embedding:

$$p : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{D}. \tag{11}$$

- A classifier c that maps the graph embedding to class probabilities:

$$c : \mathbb{R}^{d} \rightarrow \mathbb{R}^{K} \tag{12}$$

  where K is the number of classes, typically implemented as a Multi-Layer Perceptron (MLP) followed by a softmax function.

LogiX uses TELL to approximate c and identify a set of interpretable logic rules that approximate the decision boundaries. For each class k, the corresponding logical explanations E_k are extracted, identifying the activation ranges of specific dimensions in $\mathbb{R}^d$ that contribute to predicting class k:

$$E_k : \mathbb{R}^{d} \rightarrow \{0, 1\}. \tag{13}$$

Thus, by tracing back the contributions of node embeddings to these activations, we obtain node-level explanations for the GNN predictions. In the remainder of this section, we first show how we can identify most important nodes extracted by the logic layer, then we present how we can also obtain logic-based explanations at a global level for the whole model.

4.1 Identifying important nodes

Let $H = [h_1, h_2, \ldots, h_N] \in \mathbb{R}^{N\times D}$ be the matrix of node embeddings, where $h_i \in \mathbb{R}^D$ is the embedding of node i in the final GNN layer. The graph embedding is then computed as:

$$h_G = p(H) \in \mathbb{R}^{D} \tag{14}$$

For a specific dimension $d \in [1, ..., D]$, we denote the corresponding embedding value as $h_G^{(d)}$. Given that p is a global pooling operator (max, mean, or sum), we compute the contribution of each node i to $h_G^{(d)}$ as follows:

- *max* pooling. Given $h_G^{(d)} = \max_{i=1,\ldots,N} h_i^{(d)}$:

$$c_i^{(d)} = \begin{cases} h_i^{(d)}, & \text{if } i = \arg\max_i h_i^{(d)} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

- *mean* pooling. Given $h_G^{(d)} = \frac{1}{N}\sum_{i=1}^{N} h_i^{(d)}$:

$$c_i^{(d)} = \frac{1}{N} h_i^{(d)} \tag{16}$$

– *sum* pooling. Given $h_G^{(d)} = \sum_{i=1}^{N} h_i^{(d)}$:

$$c_i^{(d)} = h_i^{(d)} \tag{17}$$

Finally, since the class prediction is determined by multiple dimensions of the graph embedding h_G, the node attributions a_i are obtained by aggregating the contributions of each node across all dimensions involved in the logic rules E_k for the predicted class k:

$$a_i = \sum_{d \in \mathcal{D}_k} w_d \cdot c_i^{(d)} \tag{18}$$

where $\mathcal{D}_k$ is the subset of embedding dimensions that influence the classification of G into class k, and w_d represents the importance weight associated with dimension d, derived from the activation strength of the corresponding rule in E_k. The resulting attributions a_i quantify the influence of each node i on the final classification. A higher value of a_i indicates a stronger contribution of node i to the predicted class k, thereby providing an interpretable, rule-based explanation for the GNN's decision.

Once determined how it is possible to obtain node attributions for a prediction, our explanation pipeline follows the scheme proposed in Fig. 1: Given a GNN explainer, we first train TELL to replicate the results of the MLP; we then obtain logic rules for the explanations which enable understanding the global behavior of the model under a logical perspective; finally, given an instance, we use the above-defined rules to obtain explanation masks for the nodes' contributions towards the extracted rules.

4.2 Obtaining global logic rules

The extraction of the logic rules allows for an inspection of the behavior of the model. In our case, rules are directly obtained from the TELL that is used to approximate the MLP. For each class k, TELL generates a DNF rule E_k:

$$E_k = \bigvee_i e_{ik}. \tag{19}$$

Each conjunctive clause e_{ik} is then defined as:

$$e_{ik} = \bigwedge_j l_{ijk} \tag{20}$$

where l_{ijk} represents individual literals that operate on the activations of the GNN embeddings. Once these rules are obtained, we identify the nodes that specifically activate each of the conjunctive components e_{ik}. This allows us to determine the patterns that the model has learned. The process is as follows:

– Mask construction. For each conjunctive component e_{ik}, we construct a binary mask that selects only the nodes whose embeddings activate for all the literals l_{ijk}. A node satisfies e_{ik} if and only if all the conditions imposed by its literals hold.

– Subgraph extraction. Using the computed node masks, we extract all the connected components of the subgraphs that activate according to the rule E_k. These connected components components serve as interpretable structures that contribute to the model's decision for class k.
– To prevent redundant extractions of equivalent subgraphs, we perform graph isomorphism checks [6]. If a newly extracted subgraph is isomorphic to an already stored one, it is merged by increasing its occurrence count rather than adding a duplicate.

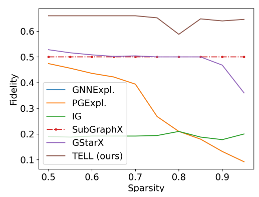
It is important to note that the rule extraction process from the TELL layer, while enabling faithful and interpretable explanations, can in principle involve exponential complexity with respect to the number of embedding dimensions, due to the enumeration of minimal activating feature subsets. However, prior work has shown that this process can be made tractable in practice through regularization techniques that promote sparse and thresholded activations. Moreover, and crucially for graph applications, the rule extraction complexity depends only on the dimensionality of the graph embeddings and not on the number of nodes in the input graph. This is in contrast to other approaches such as GStarX, whose complexity grows with graph size due to subgraph-level combinatorics.

5 Experiments

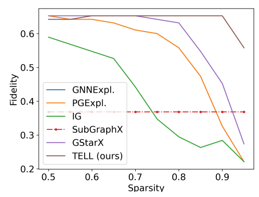
In this section, we perform extensive experiments to evaluate the explanations produced by LogiX in comparison with state-of-the-art post-hoc XAI methods for GNNs. For all the experiments we utilize a GNN formed of 5 GIN layers [26] followed by a max, sum or mean readout function which aggregates graph embeddings for each of the 5 layers. Finally, a 2-layer MLP predicts the class probabilities from the graph embeddings. We perform our analysis on the following datasets: BA2Motifs [16], a synthetic dataset where negative and positive classes are determined by the presence of a cycle or 5-node house motif, respectively; MUTAG [9], Mutagenicity [14], NCI1 [25], and BBBP [17], four molecular graph datasets where graphs are categorized depending on molecular properties; PROTEINS [10], a dataset containing graphs that represent proteins classified as enzymes or non-enzymes. To ensure reproducibility, we report the hyperparameters and the accuracy of the black-box models in the Supplementary Material. Additionally, since our method utilizes a surrogate logic layer to approximate the classifier of the GNN, we report in Table 1, the alignment between the surrogate and the clsifier.

Table 1: Alignment between LogiX surrogate and the initial GNN over the different datasets. Results report mean and standard deviations over 5 seeds.

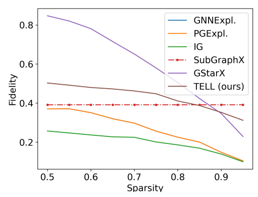
BA2Motifs	**MUTAG**	**Mutagenicity**	**NCI1**	**BBBP**	**PROTEINS**
1.000 ± 0.000	0.926 ± 0.071	0.901 ± 0.059	0.878 ± 0.089	0.886 ± 0.118	0.956 ± 0.059



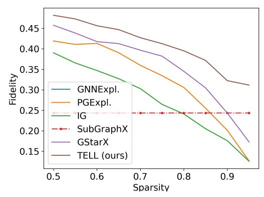
(a) BA2-Motifs



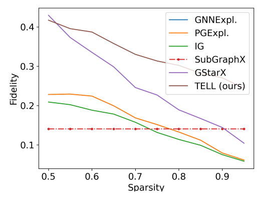
(b) MUTAG



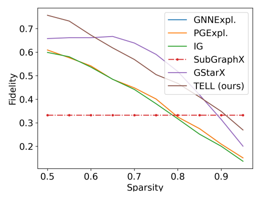
(c) Mutagenicity



(d) NCI1



(e) BBBP



(f) PROTEINS

Fig. 2: Fidelity scores of the different explanations ranging from a Sparsity of 0.5 to 0.95 using a step 0.05. Plots report the mean values over 5 seeds.

As our approach allows obtaining both instance-level explanations in the form of attribution masks and logic-based explanations in the form of rules, we perform comparisons with both instance-level techniques and logic-based approaches. On the instance-level side, we evaluate our method against several state-of-the-art techniques, each leveraging distinct explanation strategies (perturbation, optimization, and gradient-based approaches). Specifically, we select GNNExplainer [27], PGExplainer [16], Integrated Gradients [24], SubgraphX [28], and GStarX [29] as benchmarks. Finally we analyze the logic explanations produced by our approach alongside with the ones of GraphTrail [2].

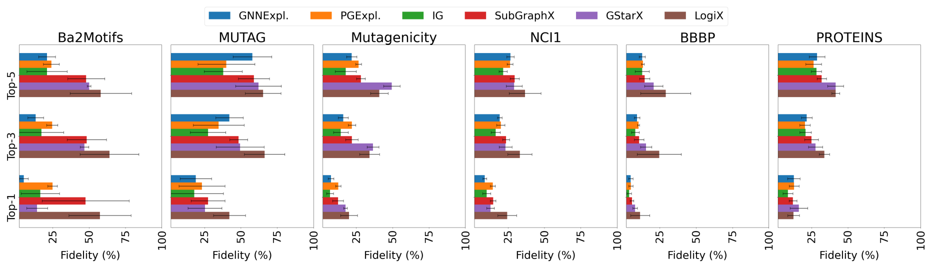
5.1 Instance-level Explanations

Instance-level explanations consist of attribution scores for the nodes (as defined in Eq. 18), indicating their importance towards a prediction. To evaluate the explanations, we convert the scores into hard masks using a threshold over the attribution scores. We obtain different hard masks for each graph by selecting thresholds that produce varying Sparsity values. Sparsity is measured as the proportion of non-important nodes over the total number of nodes in a graph:

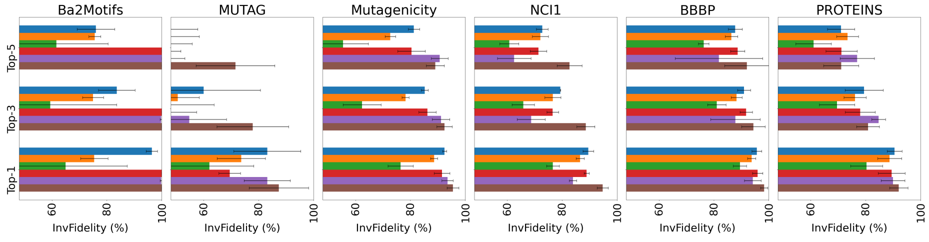
$$\text{Sparsity} = \frac{|\{v \in V : s_v < \tau\}|}{|V|} \tag{21}$$

where V is the set of all nodes in the graph, s_v is the attribution score of node v, and τ is the threshold.

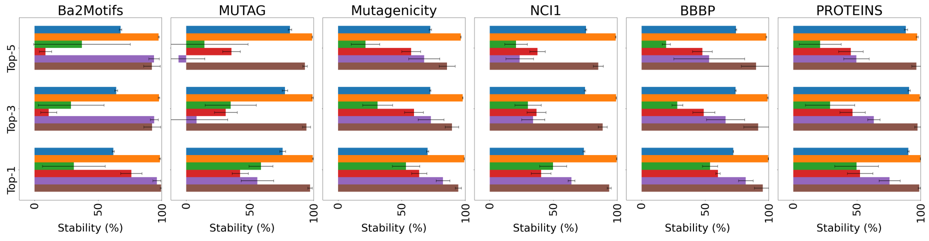
Finally, we use the Fidelity and InvFidelity as metrics to evaluate our explanations:



(a) Fidelity



(b) InvFidelity



(c) Stability

Fig. 3: Fidelity, 1-InvFidelity and Stability Scores. Fidelity is calculated removing the the top 1, top 3 and top 5 nodes. InvFidelity and Stability is calculated when keeping the top $N-1$, top $N-3$ and top $N-5$ nodes. Values are reported as average over 5 seeds, with error bars representing standard deviations. We detail the values in tabular form in the Supplementary Material.

- Fidelity measures how well the masked graph retains the original prediction: $\text{Fidelity} = f(G) - f(G_\text{m})$;
- InvFidelity measures the prediction shift when only the masked-out nodes are kept: $\text{InvFidelity} = f(G) - f(\overline{G_\text{m}})$.

$f(G)$ is the model's prediction on the original graph, G_m is the graph after removing non-relevant nodes, and $\overline{G_\text{m}}$ is the graph keeping only the non-relevant nodes.

We report in Fig. 2, the Fidelity scores of the different explanations ranging from a Sparsity of 0.5 to 0.95 using a step of 0.05. Overall, we observe that on BA2-Motifs, MUTAG and NCI1, our approach surpasses the others. In Mutagenicity and BBBP, instead, GStarX appears to provide more faithful explanations at low Sparsity values, while TELL is the best performing at higher Sparsity values. Finally, on PROTEINS, GStarX and our approach report similar performances, with the latter keeping higher scores at high sparsities. As a general observation, all the models have the general tendency of decreasing their Fidelity values when Sparsity is increased, with the sole exception of SubGraphX, which is reported in a dotted line, as it is the only method returning hard masks instead of soft masks. We attribute this behavior to the fact that LogiX, by capturing the rules that lead to the predictions, is better at identifying the most important nodes. Therefore, when only few nodes are taken out of the graphs, LogiX yields the ones that are really important for the prediction. To better study this behavior, we report in Fig. 3a an analysis of the Fidelity on the four datasets, when removing the top 1, 3 and 5 nodes from the graphs. This study confirms our previous results: when we focus on the most important node, the three most important nodes, and the five most important nodes, with LogiX, we obtain generally higher Fidelity compared to other approaches. Only on a few datasets, namely Mutagenicity, and PROTEINS, our results report comparable results with the best-performing methods.

Once we determined the capability of LogiX to better identify the most relevant nodes for the prediction, we also study its capability to discard non-important nodes. To this aim, in Fig. 3b we analyze the values of 1-InvFidelity values of the different explainers when removing the least important node, the least three important nodes or the least five important nodes. We use 1-InvFidelity to invert the scale such that higher values correspond to better quality of the explanations. On this side, we observe that most of the approaches report similar performances. In particular, we see that for BA2Motifs our method and SubGraphX obtain perfect scores. On MUTAG, NCI1, and Mutagenicity, we generally report the best results. Finally, on PROTEINS, most of the methods perform similarly, with ours and GStarX reporting the best results. Overall, we observe that LogiX is always either best-performing or comparable with the best-performing explainer even in identifying non-important nodes.

To complement our analysis of explainers' ability to identify the most and least important nodes, we also examine their stability. We define stability as the cosine similarity between the node attributions of a graph and those obtained after removing the least important nodes [1]. Fig. 3c presents an analysis of explainers' stability under conditions similar to previous experiments, i.e., after removing the least important node, the three least important nodes, or the five least important nodes. Unlike Fidelity and InvFidelity, stability alone does not indicate the correctness of explanations but rather measures the consistency of importance rankings when non-important nodes are removed. Our results show that PGExplainer is the most stable method. However, it is important to note that PGExplainer performs poorly in Fidelity and InvFidelity. Conversely, our method—which excels in Fidelity and InvFidelity—is the second most stable explainer, consistently achieving over 90% stability when removing the most important node. This is because LogiX globally approximates the MLP, ensuring that if a node activates certain rules, its activation remains largely unaffected by

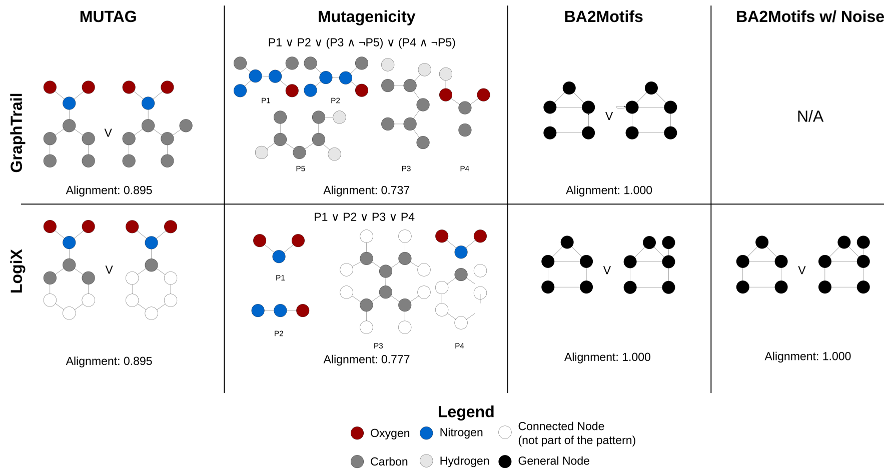


Fig. 4: Logic explanations obtained using GraphTrail and LogiX on the BA2Motifs, MUTAG and Mutagencity datasets. The figure presents a visual extract of the obtained results paired with the alignment of the rules with the model.

small perturbations. In contrast, while SubGraphX and GStarX provide faithful explanations, their reliance on Shapley value computation makes them more sensitive to perturbations, leading to variations in explanation values.

5.2 Global Inspection of the Model through Logic Rules

Here, we analyze the rules learned by LogiX in comparison to those extracted by another logic-based global explainer, GraphTrail. It is important to emphasize that these two methods serve different purposes. GraphTrail is specifically designed to determine rules over computation trees, with no focus on instance-level explanations. In contrast, our approach constructs logic rules to explain the last layer of the GNN, aiming to identify the most relevant nodes in a graph. The only commonality between the two methods is their use of logic as a means of presenting explanations. Nevertheless, comparing their outputs provides valuable insights.

Fig. 4 presents the rules extracted by our method and GraphTrail [2] on the BA2Motifs and MUTAG and Mutagenicity datasets. We observe that both methods identify similar patterns, suggesting they effectively capture the model's reasoning. Indeed, when measuring the alignment of the extracted rules with the model's predictions, both techniques achieve similar scores, with equal values on MUTAG and BA2Motifs and a slightly higher alignment for Logix on Mutagenicity. Alignment is quantified as the ratio of samples where the model and the extracted rules yield the same outcome.

However, a key limitation of GraphTrail is its reliance on computation trees, which restricts its applicability to cases where nodes have discrete features or belong to predefined categories. To illustrate this, we conduct an experiment where we replace the original features of BA2Motifs with random values and train a model on the same task. In this scenario, GraphTrail is unable to extract rules, as all nodes have unique features. In contrast, our approach remains fully applicable, producing logic rules that maintain full alignment with the model.

6 Conclusions

In this work, we introduced LogiX, a novel logic-based post-hoc explainability method for graph classification, leveraging a logic layer to directly model the decision function of a GNN. Unlike conventional post-hoc methods that rely on perturbations, gradient-based techniques, or optimization-based subgraph selection, our approach produces explanations that are inherently aligned with the model's decision process, ensuring greater faithfulness and interpretability. Through extensive experiments on synthetic and real-world datasets, we demonstrated that our approach outperforms state-of-the-art post-hoc techniques in terms of explanation fidelity, sparsity, and stability. Specifically, our method excels in identifying the most relevant nodes contributing to a prediction while maintaining consistency across perturbations. Additionally, LogiX enables global-level interpretability by extracting human-interpretable logic rules that approximate the decision boundaries of the model. Despite these advancements, our work presents certain limitations. The extracted logic rules, while faithful to the model's decision-making process, may not always be immediately human-comprehensible due to the abstract nature of learned embeddings. Future research could explore techniques to bridge this gap by integrating domain-specific knowledge into the logic extraction process. Additionally, extending TELL to other graph-based tasks beyond classification, such as link prediction and node classification, remains an interesting avenue for further investigation. Finally, other limitations might concern the execution times. Indeed, while we did not encounter timing issues in our experiments, compared to other approaches, depending on the complexity of the problem, it was shown that the enumeration of rules from TELL might entail long running times.

Acknowledgments. This work was supported by French state aid managed by the National Research Agency under the France 2030 program, with the references "WAIT4 ANR-22-PEAE-0008" and "PANDORA ANR-24-CE23-0950.

References

1. Agarwal, C., Queen, O., Lakkaraju, H., Zitnik, M.: Evaluating explainability for graph neural networks. Scientific Data **10**(1) (Mar 2023)
2. Armgaan, B., Dalmia, M., Medya, S., Ranu, S.: Graphtrail: Translating GNN predictions into human-interpretable logical rules. In: NeurIPS (2024)
3. Azzolin, S., Longa, A., Barbiero, P., Lio, P., Passerini, A.: Global explainability of gnns via logic combination of learned concepts. In: ICLR (2023)

4. Bongini, P., Bianchini, M., Scarselli, F.: Molecular generative graph neural networks for drug discovery. Neurocomputing **450**, 242–252 (Aug 2021)
5. Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M., Melacci, S.: Logic explained networks. Artificial Intelligence **314**, 103822 (2023)
6. Cordella, L.P., Foggia, P., Sansone, C., Vento, M., et al.: An improved algorithm for matching large graphs. In: IAPR-TC15 (2001)
7. Dai, E., Wang, S.: Towards self-explainable graph neural network. In: International Conference on Information & Knowledge Management, CIKM. p. 302–311 (2021)
8. Darwiche, A.: Logic for explainable ai. In: 2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–11. IEEE (2023)
9. Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Journal of Medicinal Chemistry **34**(2), 786–797 (1991)
10. Dobson, P.D., Doig, A.J.: Distinguishing enzyme structures from non-enzymes without alignments. Journal of Molecular Biology **330**(4), 771–783 (2003)
11. Duval, A., Malliaros, F.D.: Graphsvx: Shapley value explanations for graph neural networks. In: ECML PKDD. pp. 302–318 (2021)
12. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: WWW '19. ACM Press (2019)
13. Jiang, D., Wu, Z., Hsieh, C.Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., Hou, T.: Could graph neural networks learn better molecular representation for drug discovery? Journal of Cheminformatics **13**(1) (2021)
14. Kazius, J., McGuire, R., Bursi, R.: Derivation and validation of toxicophores for mutagenicity prediction. Journal of Medicinal Chemistry **48**(1), 312–320 (2004)
15. Lipton, Z.C.: The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue **16**(3), 31–57 (2018)
16. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. NeurIPS **33**, 19620–19631 (2020)
17. Martins, I.F., Teixeira, A.L., Pinheiro, L., Falcao, A.O.: A bayesian approach to in silico blood-brain barrier penetration modeling. Journal of Chemical Information and Modeling **52**(6), 1686–1697 (jun 2012)
18. Ragno, A., Capobianco, R.: Impo: Interpretable memory-based prototypical pooling. In: WSDM. p. 625–632 (2025)
19. Ragno, A., La Rosa, B., Capobianco, R.: Prototype-based interpretable graph neural networks. IEEE Transactions on Artificial Intelligence pp. 1–11 (2022)
20. Ragno, A., Plantevit, M., Robardet, C., Capobianco, R.: Transparent explainable logic layers. In: ECAI. vol. 392, pp. 914–921 (2024)
21. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
22. Schwarzenberg, R., Hübner, M., Harbecke, D., Alt, C., Hennig, L.: Layerwise relevance visualization in convolutional text graph classifiers. In: Workshop on Graph-Based Methods for Natural Language Processing, TextGraphs. pp. 58–62 (2019)
23. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: International conference on computer vision, ICCV. pp. 618–626 (2017)
24. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: ICML. pp. 3319–3328. PMLR (2017)
25. Wale, N., Watson, I.A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. KAIS **14**(3), 347–375 (2007)

26. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019)
27. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. NeurIPS **32** (2019)
28. Yuan, H., Yu, H., Wang, J., Li, K., Ji, S.: On explainability of graph neural networks via subgraph explorations. In: ICML. pp. 12241–12252. PMLR (2021)
29. Zhang, S., Liu, Y., Shah, N., Sun, Y.: Gstarx: Explaining graph neural networks with structure-aware cooperative games. Advances in Neural Information Processing Systems, NeurIPS **35**, 19810–19823 (2022)
30. Zhang, Z., Liu, Q., Wang, H., Lu, C., Lee, C.: ProtGNN: Towards self-explaining graph neural networks. AAAI Artificial Intelligence **36**(8), 9127–9135 (2022)