

Leveraging internal representations of GNNs with Shapley Values

Ataollah Kamal^{1*}, Alessio Ragno¹, Marc Plantevit²,
Céline Robardet¹

¹INSA Lyon, LIRIS UMR 5205, F-69621 Villeurbanne, France.

²EPITA Lyon, EPITA Research Laboratory (LRE), F-94276 Le Kremlin-Bicêtre, France.

*Corresponding author(s). E-mail(s): ataollah.kamal@insa-lyon.fr;

Contributing authors: alessio.ragno@insa-lyon.fr;

marc.plantevit@epita.fr ; celine.robardet@insa-lyon.fr;

Abstract

We address the challenge of identifying the most influential graph structures in the decisions of Graph Neural Networks (GNNs). To tackle this, we propose a novel approach for evaluating the importance of subgraphs in GNN decisions, with a particular emphasis on calculating Shapley values. Unlike existing methods that impose rigid, predefined constraints on subgraph shapes (e.g., egographs or individual nodes), our approach remains flexible, accommodating arbitrary subgraph structures. Our method begins by analyzing activation patterns within the representation spaces generated by the GNN, followed by computing Shapley values for these patterns to quantify their contributions to model decisions. Using these Shapley values, we produce both instance-level and model-level explanations, offering deeper insights into the reasoning processes of GNNs. Extensive empirical studies across diverse datasets and comparisons with state-of-the-art methods highlight the effectiveness of our approach in delivering interpretable and robust explanations.

Keywords: Explainable Artificial Intelligence, Graph Neural Networks, Shapley values.

1 Introduction

Explainable Artificial Intelligence (XAI) for graph neural network (GNN) models is an emerging field dedicated to demystifying the decision-making processes of AI models applied to graph-structured data. As AI systems increasingly use graphs to model complex relationships and interactions, the demand for transparent and interpretable explanations of their behavior has grown significantly. By revealing the reasoning behind model predictions and identifying key graph features driving these outcomes, graph XAI can foster trust, improve accountability, and unlock the full potential of AI in graph-centric domains such as social networks (Fan et al., 2019), biological networks (Z. Wu et al., 2021), and recommender systems (S. Wu et al., 2022).

Most existing XAI methods rely on model-agnostic procedures, where explanations are generated based on changes in the model output after input perturbations (Pereira et al., 2023; Pope et al., 2019; Shan et al., 2021; Yuan et al., 2020; S. Zhang et al., 2022). However, these approaches have two main drawbacks. First, they heavily depend on the inputs considered to derive explanations, often leading to input-dependent explanations rather than model-dependent ones (Ahmed et al., 2024). Second, most of these techniques focus on instance-level explanations. Although instance-level explanations are valuable for analyzing individual cases, due to the potential variety of the instance explanations and noisy inputs, the general behavior of the model might not be discovered. In such cases, model-level explanations are essential (Azzolin et al., 2023).

To address these challenges, Veyrin-Forrer et al. (2024) introduced INSIDE, a method that explains GNN predictions by extracting activation rules from the latent spaces of the GNN. This approach identifies relevant subgraphs by analyzing the hidden neuron activation matrix, providing both instance-level and model-level explanations. Despite its innovative design, INSIDE has two significant limitations. First, it associates a single activation rule with each instance by selecting the one that maximizes fidelity, thereby limiting its ability to account for the collaborative contributions of multiple rules learned by the model. Second, it assumes that an activation rule is supported exclusively by a set of homogeneous graphs, which constrains its generalizability and applicability to more diverse graph datasets.

We aim to further leverage cooperative game theory techniques, which have recently gained prominence for effectively capturing collaborative effects (S. Zhang et al., 2022). Among these, the calculation of Shapley values stands out as a prominent method for generating explanations, thanks to its axiomatic guarantees that ensure fairness and consistency. In this work, we compute the Shapley values of activation rules for each graph, quantifying their contribution to the model’s decision. By incorporating the collaborative effects of these rules, our approach provides more accurate instance-level explanations than those generated by INSIDE. Moreover, we use Shapley values to identify homogeneous subgroups within the support of activation rules, enhancing the interpretability of model decisions. While the computation of Shapley values is typically exponential in complexity, approximation techniques such as KernelShap (Lundberg & Lee, 2017) offer practical alternatives. However, these methods remain sensitive to the number of players (i.e., features) they aim to assess. This is where our approach holds a distinct advantage. Unlike other Shapley-value-based explainers (Duval & Malliaros, 2021; Perotti et al., 2022), which impose restrictions on

the form of players (e.g., individual nodes, edges, or frequent subgraphs), our method offers greater flexibility by operating on latent representations. This approach enables the inclusion of more sophisticated players (i.e., subgraphs of arbitrary size) while maintaining scalability and robustness.

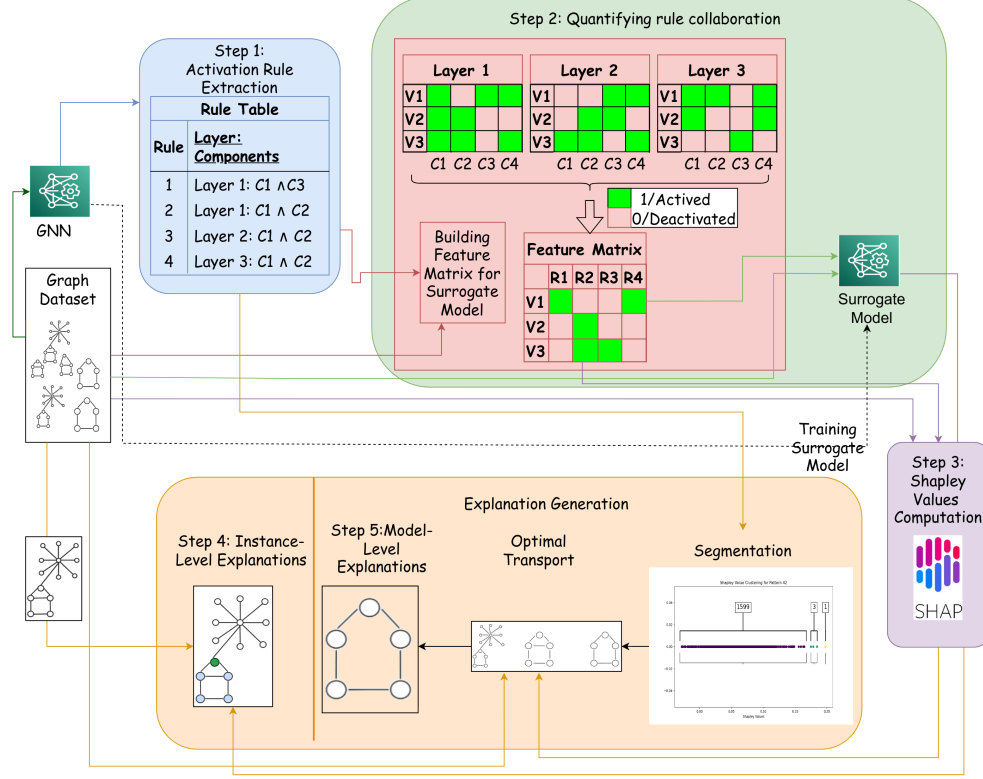


Fig. 1: Overview of INSIDE-SHAP. Step 1: Extract activation rules from the GNN’s latent space. Step 2: Train a surrogate model using rule activations to approximate the black-box model. Step 3: Compute Shapley values to quantify each rule’s individual and collaborative contributions. Step 4: Construct instance-level explanations by propagating rule contributions to activated nodes. Step 5: Derive global explanations by segmenting the rule’s support based on Shapley values and selecting representative subgraphs via optimal transport.

In this paper, we introduce INSIDE-SHAP, an extension of INSIDE that incorporates Shapley values to quantify the importance of activation rules and capture their collaborative effects on model decisions. Our contributions are fourfold: (i) Shapley values enable precise quantification of each rule’s individual importance in graph classification; (ii) they facilitate the identification of collaborative interactions between rules within a single instance; (iii) they allow differentiation of the contexts in which rules are activated; and (iv) they enable the association of representative graphs

with a rule based on its activation context. Unlike prior game-theoretic-based methods such as SubgraphX (Yuan et al., 2021a), GStarX (S. Zhang et al., 2022), and EdgeShaper (Mastropietro et al., 2022), which define players as input-level elements like nodes or edges and suffer from high computational cost as graph size increases, INSIDE-SHAP uses activation rules derived from the GNN’s latent space as players. Since the number of rules is independent of input size, our method achieves more stable and scalable Shapley value approximations across diverse datasets. Furthermore, unlike GraphSVX (Duval & Malliaros, 2021), which focuses solely on instance-level explanations, INSIDE-SHAP provides both instance-level and model-level insights, enabling a more comprehensive understanding of GNN behavior.

The methodology of INSIDE-SHAP is illustrated in Figure 1:

- **Step 1 – Extraction of activation rules:** Similar to INSIDE, our method starts by identifying activation rules as the foundation for generating explanations.
- **Step 2 – Quantifying rule collaboration:** To evaluate the cooperative impact of rules on decisions, a surrogate model is trained to mimic the GNN’s behavior using the activation status of the rules. This enables the calculation of Shapley values over the rules.
- **Step 3 – Shapley value computation and context differentiation:** Using KernelShap (Lundberg & Lee, 2017), Shapley values are computed for each rule. To capture the diverse contexts in which a rule may be activated, potentially contributing to different decisions of the GNN model, INSIDE-SHAP identifies subgroups within the rule’s graph support based on their Shapley values.
- **Step 4 – Instance-level explanations:** INSIDE-SHAP leverages the collaborative effects of rules, quantified via Shapley values, to provide detailed instance-level explanations.
- **Step 5 – Global-model explanations:** Finally, INSIDE-SHAP generates representative graphs for the discovered rules, enabling a human-interpretable understanding of the model’s learned representations. Specifically, we tackle two key challenges: selecting the most relevant rules to reduce interpretative complexity arising from an overabundance of rules, and effectively managing rules with heterogeneous support.

The remainder of this article is structured as follows: Section 2 reviews recent advances in graph explanation methods; Section ?? outlines the foundational concepts necessary to contextualize our work; Section 3 introduces our proposed method INSIDE-SHAP, focusing on the computation and application of Shapley values to explore the latent space of GNNs; Section 4 presents extensive experiments that validate the effectiveness of our approach; Finally, Section 4.4.2 summarizes the findings and outlines potential directions for future research.

2 Related Works

Explainable AI (XAI) approaches can be broadly categorized into factual and counterfactual explanations. Factual explanations identify the most important parts of the input contributing to the model’s decision, such as subgraphs or feature subsets. In

contrast, counterfactual explanations highlight the minimal changes needed to alter the model’s decision. Both perspectives offer complementary insights into model behavior.

Our focus is on factual explanations, which can be further divided into self-explainable models and post-hoc methods. Self-explainable models (Proietti et al., 2024; Ragno et al., 2022; T. Wu et al., 2020; Y.-X. Wu et al., 2022) provide built-in interpretability but often come at the cost of reduced accuracy (Kakkad et al., 2023). Post-hoc methods, which generate explanations for pre-trained models, can be classified into five categories: perturbation-based, surrogate, gradient-based, decomposition-based, and generation-based methods.

Perturbation-based methods modify the input and measure the impact on predictions. For instance GStarX (S. Zhang et al., 2022) perturbs the graph structure and evaluates node importance using Hamiache-Navarro (HN) values. Surrogate methods approximate complex models with simpler ones for interpretability. DnX (Pereira et al., 2023) trains linear GNNs to mimic black-box models and derives explanations via convex optimization. Gradient-based methods compute gradients of the output with respect to the input to assess feature importance. Examples include contrastive gradient-based saliency maps (Pope et al., 2019) and class activation mapping (CAM) for GNNs (Jung & Oh, 2021). INSIDE (Veyrin-Forrer et al., 2024) follows a similar approach by explaining GNNs through the identification of activation rules derived from the model’s latent spaces, thereby revealing its internal decision-making process. Decomposition-based methods aim to break down model decisions into interpretable components by attributing different parts of the model’s output to specific features or aspects of the input. In the context of GNNs, these methods decompose graph-level or node-level decisions into interpretable substructures, making it easier to understand the contribution of various elements to the final prediction. GNN-LRP (Schnake et al., 2022) uses layer-wise relevance propagation to attribute predictions to graph structures. Generation-based methods employ generative models for explanation. XGNN (Yuan et al., 2020) generates global explanations via reinforcement learning, while RG-Explainer (Shan et al., 2021) produces instance-level explanations.

A distinct line of work leverages cooperative game theory for explainability, primarily through Shapley values. SubgraphX (Yuan et al., 2021a) finds the most contributing subgraph leveraging the Shapley values in a Monte Carlo Tree Search algorithm. GraphShap (Perotti et al., 2022) assigns Shapley values to predefined motifs. GraphSVX (Duval & Malliaros, 2021) computes Shapley values for nodes and features. GStarX (S. Zhang et al., 2022) extends this approach using HN values for node-level explanations. EdgeShaper (Mastropietro et al., 2022) and GNNShap (Akkas & Azad, 2024) focus on Shapley values for edges, while FlowX (Gui et al., 2023) evaluates them for message flows. GraphTrail (Armgaan et al., 2024) is another noteworthy approach that provides model-level explanations for GNNs using Shapley values. This method derives logical formulas for the k-best concepts associated with each class, using computation trees as the underlying structures. While these formulas offer detailed insights, their reliance on computation trees poses interpretability challenges, particularly for non-expert users. The complexity of these logical expressions may further limit the accessibility of the explanations.

Among the aforementioned explainers, GNNShap provides only explanations for node classification tasks and GraphTrail does not offer instance-level explanations. Therefore, in this work, we concentrate on the limitations associated with explainers such as SubgraphX, GraphSVX, GStarX, and EdgeShaper which provide instance-level explanations for graph classification tasks.

3 Explaining GNNs using activation rules and Shapley values

In this section, we introduce our approach to overcome the limitations of existing methods based on INSIDE. As outlined in Section ??, INSIDE (Veyrin-Forrer et al., 2024) provides explanations at both the model and instance levels. However, the method presents two key drawbacks: at the instance level, it assumes that a single rule is responsible for the decision, while in reality, multiple rules may combine to influence the prediction; at the model level, it assumes that each rule can be represented by a single graph. This assumption becomes problematic when a rule has heterogeneous support, as one graph may not adequately represent the entire support.

To address these issues, we propose the use of Shapley values to quantify the contribution of each rule to the model’s predictions. Shapley values allow us to allocate the prediction contribution to each rule based on the nodes in its support, enabling a more granular understanding of their impact. Additionally, by identifying homogeneous clusters within the rule supports, we can construct more accurate and representative graphs for each cluster. However, this process is not trivial, as rules often involve nested activations, which complicates the direct calculation of Shapley values on the original model. To overcome this challenge, we employ a surrogate model to approximate the class distributions from the rule activations. This allows us to calculate the Shapley values directly on the surrogate model, avoiding the complexities of the original model’s direct interpretation.

In the following sections, we detail our approach, covering the Shapley value computation, the model-level explanations, and the instance-level masking procedure.

3.1 Steps 2 and 3: Computing Shapley values of activation rules

We aim to define a cooperative game where the players are the rules, represented as indivisible atomic units. Although rules consist of activated components that may overlap with those of other rules, it is critical to treat each rule as a single, cohesive entity that cannot be decomposed further. The cooperative game is defined as a pair (R, λ) , where R is the set of rules and λ is the characteristic function that approximates the model decision f (the GNN) based on the rules activated by the nodes in the graph:

$$\lambda : 2^R \rightarrow \mathbb{R} \tag{1}$$

$$\lambda (\{r \in S \mid \exists v \in V_G, \text{active}(v, r)\}) \approx f(G) \tag{2}$$

where $S \subseteq R$ represents a subset of the rules.

To construct λ , we introduce a vector $\pi_v(S)$ of size $|R|$ for each node v , where the j -th component indicates whether the corresponding rule $r_j \in R$ is activated by v and belongs to S . Formally:

$$\pi_v(S)[j] := \begin{cases} 1 & \text{if } \mathbf{active}(v, r_j) \text{ and } r_j \in S, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Using these vectors, we define a mapping m_G that associates a subset S of rules with a graph whose nodes are labeled by the activation vectors $\pi_v(S)$:

$$m_G(S) = (V_G, E_G, \pi(S)), \quad (4)$$

where V_G and E_G represent the node and edge sets of the original graph G , and $\pi(S)$ is the collection of activation vectors for the rules in S associated with the graph's nodes.

Next, we define a function γ to approximate the model decision $f(G)$ using the graph representation $m_G(R)$. It is a surrogate model of the GNN:

$$f(G) \approx \gamma(m_G(R)). \quad (5)$$

By structuring the cooperative game in this manner, we can compute the Shapley values for the rules that contribute to the predictions made by f . These Shapley values are obtained by calculating them directly through the characteristic function λ , defined as:

$$\lambda_G := \gamma \circ m_G. \quad (6)$$

Given the approximation capabilities of neural networks, multi-layer perceptrons (MLPs), and GNNs, both are viable choices for the surrogate model γ . However, it is important to note that when using MLPs or other tabular-based models, rule activations must be aggregated over the graph. In this regard, we observe that GNNs offer advantages over MLPs in our experiments. Since our goal is to minimize the discrepancy between the distribution learned by the surrogate model γ and the target distribution derived from the original GNN f , we employ the Kullback-Leibler (KL) divergence as the loss function for our task. More formally, we aim to minimize the following loss function during the training of γ :

$$\mathcal{L}_\gamma = \frac{1}{|\mathcal{D}|} \sum_{c \in C} \sum_{G \in \mathcal{D}_c} f_c(G) \log \frac{f_c(G)}{\gamma_c(m_G(R))}. \quad (7)$$

In our implementation, we employ the KernelShap method (Lundberg & Lee, 2017) to efficiently approximate Shapley values. KernelShap is a powerful approximation technique for quantifying the contribution of each feature to a model's output, making it especially suited for complex models and large feature sets. The method operates by sampling subsets of input features, referred to as coalitions, and calculating the corresponding model predictions. For each coalition, the influence of a feature is determined by assessing the change in model output when the feature is included versus excluded. These coalitions are then used in a weighted linear regression, where the

weights are computed using a kernel function specifically designed to satisfy the fundamental axioms of Shapley values. The kernel function assigns higher importance to coalitions whose sizes are closer to the midpoint of the total feature set. This weighting ensures that the contributions of features are evaluated fairly across all possible coalitions, maintaining consistency with the Shapley axioms of efficiency, symmetry, null player, and additivity. By leveraging these properties, KernelShap provides a computationally efficient and principled way to approximate Shapley values for feature contributions in machine learning models.

3.2 Step 4: Instance-level explanation based on Shapley values

In previous sections, we discussed how INSIDE generates instance-level explanations by identifying a single responsible rule for a given prediction. While effective in some cases, this approach has significant limitations, as multiple rules can be present in a single graph, each contributing differently to the decision. To overcome this limitation, we propose a novel method that incorporates rule collaboration for instance-level explanations, leveraging Shapley values to quantify the contributions of all activated rules.

Specifically, given a graph G , we compute the Shapley values for all rules activated within the graph. Using these values, we generate explanation masks by averaging the contributions across the nodes and edges. The final importance of a node v in G is determined by the following formula:

$$\text{soft_mask}_G[v] = \sum_{r \in R} \mathbb{1}_{\text{active}(v,r)} \cdot \frac{\phi_r(\lambda_G)}{|SuppV(r, G)|} \quad (8)$$

where $\phi_r(\lambda_G)$ is the Shapley value of the rule r associated to the decision of the model on the graph G , and $|SuppV(r, G)|$ is the number of nodes in the support of r within G .

It is worth noting that, unlike traditional feature attribution methods, our approach generates a soft mask on the nodes without requiring computational resources that scale exponentially with input size. This ensures scalability and efficiency while maintaining interpretability.

3.3 Step 5: Graph representation based on rules and their Shapley values

We now describe the process of obtaining representative graphs for the discovered rules, a crucial step in making the model’s learned patterns more interpretable. This involves addressing two key challenges: selecting the most relevant rules to prevent the difficulty of interpreting the XAI result due to rule abundance, and effectively handling rules with heterogeneous support.

3.3.1 Rule selection

To ensure global explanations remain interpretable, it is crucial to select the most significant rules, as the total number of rules can exceed an end-user’s capacity for

interpretation. To identify these key rules, we adapt the formula from Eq. 8, originally designed for instance-level explanations, to quantify the importance of each rule. A rule is deemed significant if it plays a decisive role in explaining the graphs within its support.

Consider a graph G within the support of a rule r , where $\phi_r(\lambda_G)$ represents the contribution of r to G 's predicted class. If r is activated across a large number of nodes, its contrastive effect on node selection diminishes compared to situations where it is activated in fewer nodes. Furthermore, when $\phi_r(\lambda_G)$ is significantly higher than the contribution values of other activated rules in G , the nodes influenced by r are more likely to rank as top contributors in the explanation.

Therefore, the importance of a rule in explaining a graph within its support is determined by the following key factors:

1. Relative contribution value: The rule's contribution $\phi_r(\lambda_G)$ is compared to the contributions of other activated rules in the graph. A higher relative value indicates that the rule has a more substantial influence on the predicted class.
2. Number of nodes activated: A rule that activates fewer nodes can exert a stronger contrastive effect, making it more significant in the explanation.

To quantify the importance of rule r in graph G , we use the formula:

$$\text{Imp}(r, G) = \frac{\phi_r(\lambda_G)}{|\text{Supp}V(r, G)| \times \sum_{r' \in R, G \in \text{Supp}_{\mathcal{D}}(r')} \mathbf{abs}(\phi_{r'}(\lambda_G))} \quad (9)$$

with $\mathbf{abs}(x)$ the absolute value of x .

For a rule to be considered important for model-level explanations, it must influence the explanations of multiple graphs within the dataset, rather than having a significant effect on just a single graph. Moreover, a rule mined for a specific class should effectively distinguish graphs belonging to that class from those of other classes. Thus, the evaluation of a rule's importance must account for its discriminative power across the entire dataset. Additionally, to ensure fairness and accuracy, the evaluation process must address potential dataset imbalances, which could otherwise skew the results. Taking these factors into consideration, we first define the rule's effectiveness for a specific class and subsequently generalize this to derive its global effectiveness across the dataset.

$$\text{Effect}_c(r, \mathcal{D}_c) = \sum_{G \in \text{Supp}_{\mathcal{D}_c}} \text{Imp}(r, G). \quad (10)$$

Assuming the target class of r is c , its effectiveness over the dataset \mathcal{D} is defined as follows:

$$\text{Effect}(r, \mathcal{D}) = w_c \times \text{Effect}_c(r, \mathcal{D}_c) - \sum_{c' \in C} w_{c'} \times \text{Effect}_{c'}(r, \mathcal{D}_{c'}). \quad (11)$$

Here, w_c represents the same weight defined in Eq. ???. In this formula, Imp ensures the within-graph effectiveness of a rule, capturing its contribution within individual graphs. Meanwhile, Effect quantifies the rule's contribution to the explanations of graphs belonging to a specific class. Finally, Effect measures the rule's overall discriminability across classes, and by incorporating the defined weights for each class, the effect of dataset imbalance is mitigated.

3.3.2 Rule representation

A limitation of rule-based methods is that when a rule spans multiple classes, a single representation may not adequately capture its entire support. This limitation arises from the inherent heterogeneity within the rule’s support, even though the rule remains vital for the model’s decision-making. To tackle this issue, Shapley values computed for a rule within its support are used to identify subgroups within the support.

Segmentation of the Shapley values

To segment the Shapley values associated with a rule r , we first eliminate outliers using the Local Outlier Factor (LOF) method (Breunig et al., 2000). This step ensures that the segmentation process focuses only on the meaningful data points, improving robustness and accuracy. To analyze a given rule r , we focus on the graphs that support it and their associated Shapley values. Let Φ_r represent the ordered list of Shapley values corresponding to r :

$$\Phi_r = [\phi_r(\lambda_{G_1}), \dots, \phi_r(\lambda_{G_n})] \quad (12)$$

with $G_i \in \text{Supp}_{\mathcal{D}}(r)$ and $\forall i < j \implies \phi_r(\lambda_{G_i}) \leq \phi_r(\lambda_{G_j})$.

Sorting these values allows for a systematic examination of the differences between consecutive Shapley values. To achieve this, we compute the differences as follows:

$$\text{Dist}_r = [d_i \mid d_i = \phi_r(\lambda_{G_{i+1}}) - \phi_r(\lambda_{G_i}), \forall i < |\Phi_r| - 1]. \quad (13)$$

Here, Dist_r represents the differences, where each d_i quantifies the change between consecutive Shapley values in the ordered list Φ_r . This provides insights into the relative importance of the rule r across the graphs that support it.

A value $\phi_r(\lambda_{G_i})$ is identified as a *cut point* if its corresponding distance $d_i \in \text{Dist}_r$ exceeds the 99.9th quantile of Dist_r . These cut points represent significant gaps in the sorted Shapley values, effectively dividing them into meaningful segments: we assign each graph G_i to a segment based on the identified cut points. A graph belongs to the first segment if its Shapley value is less than the first cut point. For $j > 1$, a graph belongs to segment j if its Shapley value lies between the $(j - 1)$ -th and j -th cut points. This method systematically segments the Shapley values, capturing meaningful changes in their distribution and avoiding arbitrary thresholds.

Representative graph computing

The process of computing a representative graph begins by isolating the most relevant parts of the graphs within each segment. For each graph in a given segment, a subgraph called the *mask* is extracted. This mask is defined as the ℓ -ego network of the activated nodes. The parameter ℓ corresponds to the GNN layer where the rule was discovered. By constructing this mask, the method ensures that the resulting subgraph focuses on the portions of the graph most pertinent to the rule, while excluding unrelated nodes and edges.

After obtaining masks for all graphs in the segment, the next step is to identify a single representative graph that captures the core substructure shared across the

segment. This is achieved by computing pairwise distances between the masks using a metric derived from optimal transport (OT). OT provides a mathematically principled framework to compare distributions by minimizing the cost of transporting mass from one distribution to another.

To apply optimal transport as a similarity measure between graphs, we first represent each graph as a probability distribution. Consider a graph $G = (V_G, E_G, X)$. We define a uniform probability distribution μ supported on the node set V_G , so that

$$\forall v \in V_G, \quad \mu(v) = \frac{1}{|V_G|}. \quad (14)$$

Let $G = (V_G, E_G, X)$ and $H = (V_H, E_H, X')$ be two graphs with associated distributions μ_G and μ_H . The Fused Gromov-Wasserstein (FGW) distance (Vayer et al., 2019) defines a transportation cost from μ_G to μ_H that jointly accounts for the structural information of G and H , as well as their node feature matrices X and X' . While a full discussion of FGW is beyond the scope of this paper, we briefly outline how it is used to compute distances between graphs:

Define the set of probabilistic couplings between the nodes of G and H as

$$\Pi = \left\{ \pi : V_G \times V_H \rightarrow \mathbb{R}^+ \mid \sum_{i \in V_G} \pi(i, j) = \mu_H(j), \quad \sum_{j \in V_H} \pi(i, j) = \mu_G(i) \right\}. \quad (15)$$

The FGW distance with parameter $\alpha \in [0, 1]$ is then defined as

$$\begin{aligned} FGW_\alpha(\mu_G, \mu_H) = \min_{\pi \in \Pi} & (1 - \alpha) \sum_{i, j \in V_G \times V_H} d(X[i], X'[j]) \pi(i, j) \\ & + \alpha \sum_{\substack{i, k \in V_G \\ j, l \in V_H}} |D_{i, k} - D_{j, l}| \pi(i, j) \pi(k, l), \end{aligned} \quad (16)$$

where $D_{i, k}$ and $D_{j, l}$ denote the distances between nodes i, k in G and j, l in H respectively, and $d(\cdot, \cdot)$ measures the distance between node features.

Once pairwise FGW distances between all masks in the segment are computed, the representative graph G^* is selected as the one minimizing the sum of distances to all other graphs, i.e., the FGW median:

$$G^* = \arg \min_{G_i} \sum_j FGW_\alpha(\mu_{G_i}, \mu_{G_j}). \quad (17)$$

By leveraging FGW, this approach reduces bias and ensures that the selected representative is a central exemplar of the shared substructure, rather than an outlier.

To enhance the interpretability of the representative graph, the contributions of its individual nodes to the rule are analyzed and visually highlighted. This is achieved by systematically removing nodes from the mask, one at a time, and observing the impact of their removal. Specifically, the removal of a node is quantified by the number of other nodes that become deactivated as a result, i.e. nodes that lose their contribution to the rule. This step assigns a clear importance score to each node in the representative

graph, making it possible to visually and intuitively understand which parts of the graph play the most significant roles in supporting the rule.

Overall, this multi-step process ensures that the representative graph not only captures the essential elements of the discovered rule but also provides insights into how individual nodes contribute to the explanation. By isolating relevant substructures, selecting a median graph, and highlighting node contributions, the method offers a comprehensive and interpretable representation of the rule’s impact across the dataset.

4 Experiments

In this section, we present a comprehensive evaluation of our proposed method, INSIDE-SHAP, through a series of experiments. Our aim is to demonstrate the effectiveness and robustness of INSIDE-SHAP in generating high-quality explanations for graph classification tasks. First, we outline the experimental setup, detailing the datasets utilized and the configurations applied (Section 4.1). This provides the foundation for a fair and transparent evaluation. Next, we address a critical component of our methodology: the selection of a surrogate model to compute Shapley values for the rules (Section 4.2). Once the surrogate model is established, we focus on evaluating the explanations generated by INSIDE-SHAP. In Section 4.3, we analyze the rules discovered by our method and their corresponding visualizations. In Section 4.4, we turn our attention to instance-level explanations and benchmark INSIDE-SHAP against state-of-the-art explainers.

Through this structured evaluation, we provide a detailed understanding of the capabilities and advantages of INSIDE-SHAP across multiple dimensions¹.

4.1 Experiment settings and aims

We perform the experiments over several benchmark datasets:

- BA2-Motifs (Luo et al., 2020) is a synthetic dataset where graphs are labeled into positive and negative classes depending on the presence of a 5-node cycle or a house motif, respectively;
- AIDS (B. Wu et al., 2017), BBBP (Z. Wu et al., 2017) and Mutagenicity (Z. Zhang et al., 2019) are three datasets containing molecular graphs associated with their biological activities;
- Benzene and Alkane-Carbonyl (Agarwal et al., 2023) are two molecular datasets where molecules are divided into two classes, depending on the presence of particular functional groups: the benzene ring and the carbonyl group with an unbranched alkane, respectively.

These datasets are often used as benchmarks in XAI for GNNs due to the presence of known data rules, which can help during the analysis of the explanations. For instance, BA2-Motifs and Benzene present “ground-truth” masks for explanations that highlight the subgraphs which are determinant for the class. However, there is currently an active debate on the role of these “ground-truths” as they cannot be used to assess the correctness of the explanations, as this property should only be checked

¹To reproduce the results, the code is accessible at the following link: https://github.com/atakml/INSIDE_SHAP

with respect to the models’ predictions (Faber et al., 2021). In this case, we utilize such motifs to compare them with the rules representation and have an understanding of whether the model is identifying them.

We divide each dataset into three partitions: training (80%), validation (10%), and testing (10%). On each dataset, we train a 3-convolutional layer GNN² with an embedding size of 20. The main characteristics of the datasets and the performance of the associated black-box GNN model are summarized in Table 1. In all the experiments, we use the best-performing black box (using the validation accuracy as reference) to obtain the explanations.

Table 1: Statistics of the datasets and the performance metrics of the models trained on these datasets, expressed as mean \pm standard deviation over 5 seeds, with the best results, according to the validation accuracy, reported in parentheses.

Dataset	# Graphs	Avg # Nodes	Train Acc (%)	Validation Acc (%)	Test Acc (%)
AIDS	2000	15.69	99.19 \pm 0.41 (99.56)	98.50 \pm 0.45 (99.00)	99.30 \pm 0.25 (99.50)
BA2-Motifs	1000	25	99.90 \pm 0.20 (100.00)	100.00 \pm 0.00 (100.00)	100.00 \pm 0.00 (100.00)
BBBP	1640	24.08	89.27 \pm 0.80 (90.55)	81.46 \pm 0.83 (82.93)	81.22 \pm 0.98 (82.93)
Mutagenicity	4337	30.32	84.63 \pm 0.84 (85.90)	81.43 \pm 1.06 (82.72)	79.54 \pm 1.36 (81.34)
Benzene	4000	20.58	93.71 \pm 0.62 (94.56)	91.65 \pm 0.64 (92.49)	90.21 \pm 0.45 (90.95)
Alkane-Carbonyl	1125	21.13	99.00 \pm 0.07 (99.11)	99.11 \pm 0.00 (99.11)	100.00 \pm 0.00 (100.00)

We explore the following research questions:

- What surrogate model should be used, and how effectively does it capture the behavior of the original model by leveraging the activation rules?
- Do the segments obtained from the Shapley value-based rule support segmentation capture meaningful structural differences, and how well do their representative graphs reflect the underlying rules?
- How does INSIDE-SHAP compare to other state-of-the-art explainers, and does the incorporation of Shapley values enhance instance-level explanations compared to INSIDE? To answer this, we evaluate the impact of multiple rules on instance-level explanations and conduct a thorough comparison with INSIDE and other baseline methods, demonstrating the overall performance and versatility of our approach.

²To demonstrate that the experimental results are not specific to the GCN architecture, we include additional experiments using alternative GNN architectures in the Appendix B.

4.2 Evaluation of the surrogate model

We evaluate several options for the surrogate model to determine the most effective architecture for capturing the behavior of the original black-box model. Specifically, we compare the following approaches:

- A GCN model with the same architecture as the black-box model, but instead of raw graph data, it takes the rules’ activations as input node features. This approach ensures that the surrogate model leverages the same structural inductive biases as the original model while focusing on the rules’ activation.
- An MLP that operates on aggregated rule activations. Here, the activations are summed over the nodes of the graph, resulting in a vector representation used as input to the MLP. This method simplifies the input representation while maintaining essential information about the rule activations.
- Two traditional machine learning models: decision tree regression and linear regression. Both models also use the summed rule activations as input. The decision tree regression offers non-linear interpretability, while linear regression provides a simpler, more transparent mapping to assess the rules’ contribution.

These diverse architectures allow us to explore the trade-offs between complexity, interpretability, and fidelity to the original model in our surrogate modeling approach.

Dataset	Metric	Decision Tree	Linear Model	MLP	GCN
AIDS	Acc. (%)	97.70 \pm 1.80	98.50 \pm 0.20	98.50 \pm 0.20	98.60 \pm 0.58
	KL-Div.	0.0654 \pm 0.0166	0.0320 \pm 0.0008	0.0320 \pm 0.0008	0.0154 \pm 0.0017
BA2-Motifs	Acc. (%)	98.00 \pm 0.80	97.00 \pm 0.00	97.00 \pm 0.00	99.80 \pm 0.40
	KL-Div.	0.0044 \pm 0.0010	0.0037 \pm 0.00015	0.0037 \pm 0.0002	0.0003 \pm 0.0001
BBBP	Acc. (%)	91.80 \pm 1.40	94.51 \pm 0.00	94.51 \pm 0.00	95.85 \pm 1.18
	KL-Div.	0.0444 \pm 0.0060	0.0296 \pm 0.0004	0.0296 \pm 0.0004	0.0107 \pm 0.0014
Mutagenicity	Acc. (%)	86.20 \pm 1.00	87.52 \pm 0.26	87.52 \pm 0.26	93.27 \pm 0.59
	KL-Div.	0.0817 \pm 0.0045	0.0769 \pm 0.0003	0.0769 \pm 0.0003	0.0216 \pm 0.0017
Benzene	Acc. (%)	92.60 \pm 1.20	94.38 \pm 0.31	94.38 \pm 0.31	96.97 \pm 0.54
	KL-Div.	0.0774 \pm 0.0099	0.0413 \pm 0.0002	0.0413 \pm 0.0002	0.0158 \pm 0.0013
Alkane-Carbonyl	Acc. (%)	100.00 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	99.82 \pm 0.35
	KL-Div.	0.0014 \pm 0.0000	0.0011 \pm 0.0000	0.0011 \pm 0.0000	0.0005 \pm 0.0002

Table 2: Surrogate models evaluation. Test accuracy and KL-Divergence between surrogates and the black-box expressed in mean \pm std.

The performance of the surrogate models is summarized in Table 2. Each surrogate was trained to optimize for the KL-Divergence between its predictions and those of the black-box model. To comprehensively evaluate their effectiveness, we report both the KL-Divergence and accuracy of the surrogates relative to the black-box model. Among the tested architectures, the GCN model demonstrates superior performance, achieving the lowest KL-Divergence and the highest accuracy. This result can be attributed to its ability to operate on rule activations at the node level, thereby preserving the granular structural information inherent to the graph. In contrast, the other surrogates, which rely on aggregated rule activations at the graph level, may

lose critical information, leading to reduced fidelity when approximating the behavior of the original black-box model.

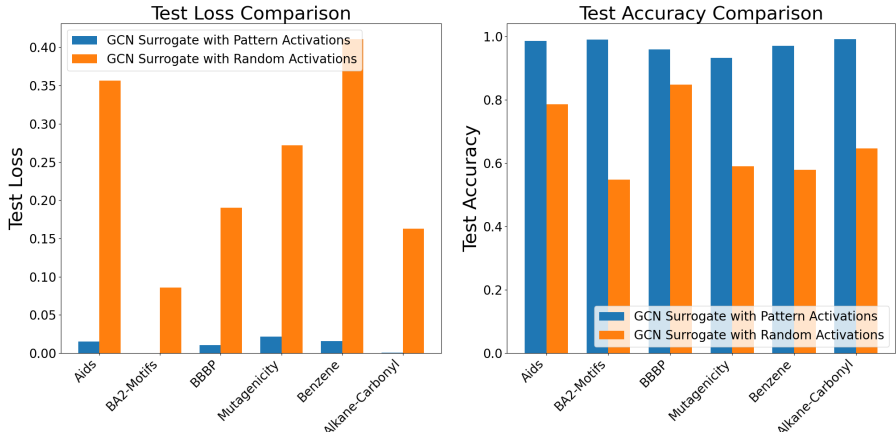


Fig. 2: Loss and accuracy comparison of the GCN surrogate model using random features and rules activation status matrix as features across different datasets. High accuracy and low loss are desired.

To dispel any doubts that the superior performance of the GCN surrogate might be due to its use of graph structure rather than actual rule activations, we include a comparison in Figure 2 between the GCN and another GCN trained on random rule activations. This comparison shows that the GCN trained on random rule activations fails to approximate the black-box model, confirming our claim that the GCN’s performance is tied to its ability to leverage node-level rule activations rather than relying solely on graph structure.

Additionally, the superior performance of the GCN with respect to the other models suggests that the mere presence of a rule is not sufficient to make a decision. This implies that a rule could potentially recognize multiple motifs, even those belonging to different classes. In the remainder of this work, we utilize the GCN to obtain instance-level explanations, given its best ability to approximate the black-box-model.

4.3 Global view of activation rule importance and their representative graphs

To analyze the rules, we present visualizations of the representative graphs for the most relevant rules identified across several datasets. Specifically, we focus on the BA2-Motifs, Alkane-Carbonyl, and Benzene datasets, as they provide a rich source for understanding the types of structures that the black-box models are capable of recognizing. Our goal is to investigate whether these identified structures align with the relevant motifs present in the ground truth.

It is important to note that the lack of correlation between the discovered rules and the ground-truth motifs does not necessarily imply poor quality of the rules. On the contrary, this discrepancy can be insightful in uncovering rules that, while not explicitly part of the ground truth, offer new perspectives on the model’s behavior. This approach allows us to identify exemplary structures that contribute to a deeper understanding of the underlying model.

For each dataset, we examine the rules with the highest effectiveness by the Effect measure introduced in Equation 11. In particular, for each identified rule, we report the median graph of the clusters obtained by analyzing the Shapley values associated with that rule, following the procedure outlined in Section 3.3. This detailed examination of the rules aims to provide valuable insights into how the model processes and classifies different structures within the data.

4.3.1 BA2-Motifs

On BA2-Motifs, rules 6, 16, 14, and 12 achieve the highest Effect scores (see Equation 11). The median graphs for these rules are displayed in Figure 3.

Rule 6 is predominantly activated in the negative class, which is characterized by the presence of a 5-node loop. Notably, the Shapley values for this rule exhibit two primary segments, both of which highlight nodes within the loop structure with a strong negative contribution. However, in the first segment, the importance of the cycle motif diminishes when a node outside the loop is activated, indicating a contextual shift in rule activation.

Rule 16 is primarily activated by the cycle motif when the Shapley value is negative, whereas a positive Shapley value indicates activation by the house motif. A similar pattern is observed in rule 14, though with a slightly lower magnitude. This analysis of Shapley values enables us to cluster activations into distinct structural motifs, such as loops and houses, offering deeper insights into how the model distinguishes between these patterns.

Rule 12, in contrast, is mostly associated with the positive class and can be divided into two segments. In this case, the segment with higher importance aligns with the motif characteristic of the positive class, while the segment with lower Shapley values corresponds to activations of the 5-node loop, which defines the negative class.

Overall, our findings demonstrate that the learned rules successfully capture the two ground-truth motifs, confirming that the model integrates these structural patterns within its activations. Moreover, by leveraging Shapley values, we can differentiate between heterogeneous activations of the same rule, further refining our understanding of the model’s decision-making process.

4.3.2 Alkane-Carbonyl

The results for Alkane-Carbonyl are presented in Figure 4, where rules 35, 23, 26, and 38 exhibit the highest Effect values on this dataset. This case is particularly noteworthy, as experiments indicate that the model predominantly relies on a shortcut. While the classes are theoretically defined by the presence of a carbonyl functional group (C=O bond) and an unbranched alkane chain (a sequence of carbon atoms), the model instead tends to use fluorine atoms as a key determinant for the positive class.

This reliance on fluorine is consistently observed across all analyzed rules. Specifically, these rules primarily activate on fluorine atoms, assigning them high importance, while the contributions of their neighboring atoms remain relatively minor. This finding highlights a potential bias in the model’s decision-making process, where an unintended feature plays a dominant role in classification.

4.3.3 Benzene

The results for Benzene are shown in Figure 5, where rules 41, 42, 10, and 32 exhibit the highest Effect values in this dataset.

Most of these rules primarily identify six-carbon rings, a common structural feature in chemistry. Notably, a six-carbon ring can correspond to either cyclohexane or benzene, depending on the bond types between the carbon atoms. However, since the black-box model does not have direct access to bond type information, it relies on the presence of surrounding atoms to differentiate between these structures. This is particularly evident in rule 42, where the neighboring atom arrangements help determine the ring type. Additionally, some rules capture rings with varying numbers of atoms, which may assist the model in distinguishing benzenes from other cyclic structures, making these rules critical for accurate predictions.

4.3.4 General considerations

When analyzing a rule within its segment, we observe a clear relationship between its Shapley value and its contribution to the model’s decision. When the Shapley value is weak (i.e., close to zero), the rule does not focus on any particular region of the graph and thus has little to no impact on the prediction. Conversely, when the Shapley value is strong, the rule selectively activates specific graph components that are critical to the model’s decision-making. This confirms that the Shapley value serves as a valid and reliable metric for assessing a rule’s influence on a graph’s classification.

This effect is explicitly incorporated in Equation 8, which generates soft masks for instance-level explanations. The formulation ensures that nodes activated by rules with near-zero Shapley values receive minimal weight, while nodes influenced by strong Shapley values are assigned higher importance. Consequently, Equation 8 guarantees that highly weighted nodes in the soft mask correspond to activations with strong Shapley contributions, reinforcing their interpretability and reliability.

At a global level, we find that the identified rules provide meaningful insights into the model’s reasoning by capturing key structural components. However, in datasets such as Alkane-Carbonyl and Benzene, some rules do not always align perfectly with the expected chemical reasoning behind positive-class predictions. For example, in the Benzene dataset, the rules do not activate all nodes corresponding to atoms in the aromatic functional group. This suggests that rather than relying on full structural activation, the model may be identifying specific motifs that serve as class indicators. Indeed, removing the activated nodes generally leads the model to shift its prediction from the positive to the negative class. This indicates that the model primarily depends on these rules for classification, as when a crucial activation is removed, the decision flips. For instance, in the case of benzene, removing nodes from the aromatic cycle disrupts the motif, causing the model to classify the graph as negative. However, the

reverse does not hold: removing nodes from a negative-class graph does not cause the model to switch its prediction to positive.

4.4 Evaluation of the instance-level explanations

We evaluate our instance-level explanations using both quantitative and qualitative methods. In the quantitative analysis, we compare INSIDE-SHAP with its competitors using the Fidelity, Inv-Fidelity, and Sparsity metrics, in addition to their combination by leveraging H-Fidelity (S. Zhang et al., 2022). To isolate the effect of the black-box model architecture, we further compare INSIDE-SHAP with cooperative game-theoretic explainers based solely on the H-Fidelity metric.

For the qualitative evaluation, we conduct two experiments. First, we randomly select instances from the BA2-Motifs and Benzene datasets to assess the human interpretability of INSIDE-SHAP’s explanations in comparison to INSIDE and the top-performing competitor. Then, we present specific examples where INSIDE-SHAP significantly outperforms other methods.

4.4.1 Quantitative Results

We evaluate the instance-level explanations generated by INSIDE-SHAP by providing explanations as masks over relevant nodes. In the literature, Fidelity and Inv-Fidelity (see Equation ??) are commonly used to assess how well an explanation method identifies important portions of the graph by measuring the variation in prediction probability when removing important or non-important nodes. Drawing from the precision-recall analogy, S. Zhang et al. (2022) introduced a single scalar metric called harmonic fidelity (H-Fidelity), which combines Fidelity and Inv-Fidelity by normalizing them with Sparsity and calculating their harmonic mean.

Evaluation of Explainability Metrics on INSIDE-SHAP Against Competitors

In this study, we compare INSIDE-SHAP with state-of-the-art methods using H-Fidelity as the evaluation metric, as it offers a balanced view across all key evaluation criteria. However, some methods generate explanations in the form of soft masks. To ensure a fair comparison, we discretize the soft masks of all methods by selecting the sparsity that maximizes H-Fidelity.

Table 3 presents the results for INSIDE-SHAP, INSIDE, and other state-of-the-art techniques based on cooperative game theory, including GStarX (S. Zhang et al., 2022), SubGraphX (Yuan et al., 2021b), GraphSVX (Duval & Malliaros, 2021), and Edge-Shaper (Mastropietro et al., 2022). Compared to INSIDE, our method consistently achieves superior results. As discussed in previous sections, INSIDE relies on a single rule for explanations, significantly limiting its effectiveness. In contrast, INSIDE-SHAP aggregates contributions from multiple rules, weighting them using Shapley values. This approach not only enhances explanation accuracy but also provides insights into the relative influence of different rules on a given prediction.

Compared to other state-of-the-art approaches, INSIDE-SHAP outperforms them in four out of six cases and achieves the second-best performance in another, demonstrating its effectiveness in generating high-quality explanations.

Method	AIDS	BA2-Motifs	BBBP	Mutagenicity	Alkane-Carbonyl	Benzene
INSIDE-SHAP	0.5157	0.5579	<u>0.5440</u>	0.5845	0.5415	0.6043
INSIDE	0.4765	0.5421	0.5368	<u>0.5837</u>	<u>0.5411</u>	<u>0.6034</u>
SubgraphX	0.4627	0.5190	0.4963	0.5001	0.5237	0.5394
GStarX	0.5100	0.5519	0.5430	-	0.5382	0.5822
GraphSVX	<u>0.5109</u>	0.5774	0.5334	0.5467	0.5367	0.5728
EdgeShaper	0.5055	<u>0.5613</u>	0.5535	-	0.5236	0.5836

Table 3: Evaluation of the instance-level explanations in terms of H-Fidelity. We conducted experiments with a timeout limit of 60 hours for computation. “-” indicates that the results were not available within this timeframe. The best value is highlighted in bold, and the second-best value is underlined.

Figure 6 presents the Fidelity, Inv-Fidelity, and Sparsity metrics for six different explanation methods. Notably, these metrics are derived from explanations optimized using H-Fidelity as the objective function. Despite this optimization, our method consistently performs on par with the best-performing approach for each individual metric.

4.4.2 Qualitative Evaluation of INSIDE-SHAP’s Explanations

To complement the quantitative results, we conduct two qualitative experiments aimed at understanding the strengths of INSIDE-SHAP in different scenarios. The first experiment focuses on instances where INSIDE fails to produce sparse explanations, allowing us to assess how INSIDE-SHAP performs in such challenging cases. The second experiment highlights examples where INSIDE-SHAP outperforms other game-theoretic methods in terms of H-Fidelity, providing insight into the conditions under which it delivers the most faithful and concise explanations.

Experiment 1: Performance Where INSIDE Fails

In this experiment, we analyze instances where INSIDE fails to generate sparse explanations—specifically, those with a sparsity score below 0.5. From this subset, we randomly sample examples to compare the quality of explanations produced by INSIDE-SHAP, INSIDE, and the best-performing game-theoretic competitor. While our goal is not to align with the ground truth, it serves as a useful reference for evaluating the plausibility and focus of the explanations.

A key observation is that, compared to INSIDE, INSIDE-SHAP consistently produces sparser and more focused explanations. This is because INSIDE relies on a single rule, which limits its ability to distribute importance across multiple contributing factors. As a result, it often highlights many nodes as important, even when their actual relevance to the prediction is minimal.

To further illustrate this comparison, we present example explanations for the BA2-Motifs and Benzene datasets in Table 4. These examples are drawn from the subset of instances where INSIDE fails to achieve a sparsity score of at least 0.5. For the Benzene dataset, we additionally restrict the selection to positively predicted graphs to facilitate comparison with the ground truth.

In the BA2-Motifs example, INSIDE-SHAP mostly avoids highlighting redundant nodes and focuses on those within the cycle motifs, the structure known to be influential. Compared to GraphSVX, INSIDE-SHAP provides a sharper contrast between important and unimportant nodes, enhancing interpretability.

In the Benzene example, while INSIDE highlights two entire cycles, INSIDE-SHAP distinguishes between individual nodes within those cycles. As shown in Section 4.3, for over 90% of positively predicted graphs (covered by rule 41), the GNN’s decision is influenced by a partial rather than a complete cycle. This makes INSIDE-SHAP’s explanation more faithful to the model’s actual behavior. Additionally, EdgeShaper fails to correctly identify the cycle structure, further underscoring the advantages of INSIDE-SHAP in capturing meaningful and precise explanations.

These findings reinforce that INSIDE-SHAP is particularly effective in scenarios where other methods—especially rule-based ones like INSIDE—struggle to isolate the most relevant components. Its ability to generate sparse, focused, and structurally meaningful explanations makes it a robust choice for interpreting complex graph-based decisions.

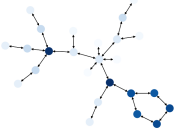
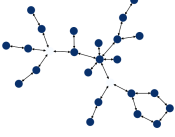
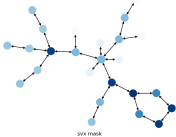
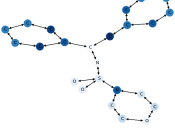

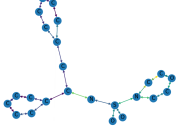
Dataset	INSIDE-SHAP	INSIDE	Best Method
BA2-Motifs			
Benzene			

Table 4: Comparison of the explanations between INSIDE-SHAP, INSIDE, and the best method among others (GraphSVX for BA2-motifs and EdgeShaper for Benzene). The stronger the color of a node/edge is, the more important it is.

Experiment 2: Performance Where INSIDE-SHAP Outperforms Competitors

While the first experiment focuses on challenging cases for INSIDE, the second experiment investigates instances where INSIDE-SHAP achieves the highest H-Fidelity among all methods. These examples help us understand the conditions

under which INSIDE-SHAP provides superior explanations and also serve to validate H-Fidelity as a meaningful quantitative metric.

Table 5 presents the results for two datasets: Benzene and BA2-Motifs. Below, we examine each case in detail.

In the BA2-Motifs dataset, all methods except INSIDE-SHAP and EdgeShaper select nodes outside the main motif, suggesting redundancy in their explanations. EdgeShaper identifies only a single important edge, whereas INSIDE-SHAP highlights one highly important node and three additional, less critical nodes, all within the motif. Although the Inv-Fidelity scores for INSIDE-SHAP and EdgeShaper may be comparable, EdgeShaper’s failure to identify multiple relevant components negatively impacts its Fidelity, resulting in a lower overall H-Fidelity compared to INSIDE-SHAP.

In the Benzene dataset, SubgraphX fails to detect any atoms within the two rings, resulting in the lowest H-Fidelity among all methods. GStarX identifies one atom inside a ring and one atom outside, but fails to capture any atoms in the upper ring. Consequently, its explanation lacks fidelity, as removing its mask still leaves a complete ring structure, preventing a change in the prediction. GraphSVX includes atoms from both rings; however, the most important atoms in its explanation lie outside the rings. As a result, each mask of this explanation includes non-ring atoms, reducing the sparsity of the explanation and introducing redundancy. EdgeShaper identifies several relevant edges within the rings, but also includes an edge that connects the upper ring to the remainder of the graph, which diminishes its precision.

By contrast, INSIDE-SHAP uniquely identifies the most influential atoms as those within the rings—capturing atoms from both rings while maintaining a minimal and concise explanation. This means that removing these specific atoms is both necessary and sufficient to alter the model’s prediction. Importantly, INSIDE-SHAP does not select any atoms outside the rings as significant, resulting in both high fidelity and high sparsity in its explanation.

5 Conclusion and Directions for Future Works

In this study, we presented INSIDE-SHAP, a novel framework for explaining GNN decisions that effectively provides instance-level and model-level explanations, addressing a key limitation of many existing methods that typically focus on only one of these levels. By leveraging Shapley values to analyze activation rules, our approach enables the collaborative assessment of rules at each input, resulting in improved instance-level explanations compared to INSIDE. Furthermore, INSIDE-SHAP demonstrated superior performance against state-of-the-art methods across multiple benchmarks.

Unlike other cooperative game theory-based methods that compute Shapley values for input features (where the number of players scales with input size) our approach calculates Shapley values for rules. These rules are independent of input size, and their number remains fixed. This reduction in the number of players significantly enhances the accuracy and efficiency of the Shapley value approximations. Additionally, we introduced a methodology to calculate Shapley values for players with potential overlaps, addressing a critical challenge in cooperative game-based explanations.

Another key contribution is our solution to the representation problem for heterogeneous rule supports. By utilizing Shapley values in conjunction with optimal transport

techniques, we created meaningful and interpretable graph representations even when the rule supports were diverse. This advancement enhances the interpretability of GNNs, providing clearer insights into their decision-making processes.

Future Directions. While INSIDE-SHAP builds on INSIDE’s rule mining framework, which can generate a large number of patterns, our global explanation results in Section 4.3 show that only a small subset of these rules is typically needed to capture the GNN’s decision logic. This opens up opportunities to integrate INSIDE-SHAP with alternative rule extraction techniques that are both more selective and computationally efficient.

Additionally, INSIDE-SHAP currently relies on KernelSHAP, a general-purpose method for approximating Shapley. While effective, its accuracy and stability can be sensitive to the number of samples. Future work could explore dedicated methods for computing Shapley values in games where the players are rules or patterns. Such methods could provide more accurate and confident attributions by directly modeling the contribution of rules with more reliable approximations designed for rules as players.

References

- Agarwal, C., Queen, O., Lakkaraju, H., Zitnik, M. (2023, March). Evaluating explainability for graph neural networks. *Scientific Data*, 10(1), , <https://doi.org/10.1038/s41597-023-01974-x>
- Ahmed, S., Shamim Kaiser, M., Hossain, M.S., Andersson, K. (2024). A comparative analysis of lime and shap interpreters with explainable ml-based diabetes predictions. *IEEE Access*, 1-1, <https://doi.org/10.1109/ACCESS.2024.3422319>
- Akkas, S., & Azad, A. (2024). GNNShap: Scalable and Accurate GNN Explanation using Shapley Values. *The acm web conference*.
- Armgaan, B., Dalmia, M., Medya, S., Ranu, S. (2024). Graphtrail: Translating GNN predictions into human-interpretable logical rules. *The thirty-eighth annual conference on neural information processing systems*.
- Azzolin, S., Longa, A., Barbiero, P., Lio, P., Passerini, A. (2023). Global explainability of GNNs via logic combination of learned concepts. *The eleventh international conference on learning representations*.
- Breunig, M.M., Kriegel, H.-P., Ng, R.T., Sander, J. (2000). Lof: identifying density-based local outliers. *Proceedings of the 2000 acm sigmod international conference on management of data* (p. 93–104). New York, NY, USA: Association for Computing Machinery.
- Chataing, T., Perez, J., Plantevit, M., Robardet, C. (2024). DiffVerify: a Scalable Approach to Differentiable Pattern Mining with Coverage Regularization.

- A. Bifet, J. Davis, T. Krilavicius, M. Kull, E. Ntoutsi, & I. Zliobaite (Eds.), *Machine learning and knowledge discovery in databases. research track - european conference, ECML PKDD 2024, vilnius, lithuania, september 9-13, 2024, proceedings, part VI* (Vol. 14946, pp. 407–422). Springer. Retrieved from https://doi.org/10.1007/978-3-031-70365-2_24
- De Bie, T. (2011). Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Discov.*, 23(3), 407–446, <https://doi.org/10.1007/S10618-010-0209-3>
- Duval, A., & Malliaros, F.D. (2021). *Graphsvx: Shapley value explanations for graph neural networks*.
- Faber, L., Moghaddam, A.K., Wattenhofer, R. (2021, August). When comparing to ground truth is wrong. *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. ACM.
- Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D. (2019). Graph neural networks for social recommendation. *The world wide web conference* (p. 417–426). New York, NY, USA: Association for Computing Machinery.
- Gui, S., Yuan, H., Wang, J., Lao, Q., Li, K., Ji, S. (2023). *Flowx: Towards explainable graph neural networks via message flows*.
- Jung, H., & Oh, Y. (2021, October). Towards better explanations of class activation mapping. *Proceedings of the IEEE/CVF international conference on computer vision (iccv)* (p. 1336–1344).
- Kakkad, J., Jannu, J., Sharma, K., Aggarwal, C., Medya, S. (2023). *A survey on explainability of graph neural networks*.
- Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., ... Lee, S.-I. (2020, Jan 01). From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1), 56–67, <https://doi.org/10.1038/s42256-019-0138-9>
- Lundberg, S.M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.
- Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X. (2020). *Parameterized explainer for graph neural network*.
- Mastropietro, A., Pasculli, G., Feldmann, C., Rodríguez-Pérez, R., Bajorath, J. (2022, October). EdgeSHAPer: Bond-centric shapley value-based explanation method

for graph neural networks. *iScience*, 25(10), 105043,

- Pereira, T.A., Nascimento, E., Resck, L.E., Mesquita, D., de Souza, A.H. (2023). Distill n' explain: explaining graph neural networks using simple surrogates. *International conference on artificial intelligence and statistics*.
- Perotti, A., Bajardi, P., Bonchi, F., Panisson, A. (2022). *Graphshap: Motif-based explanations for black-box graph classifiers*.
- Pope, P.E., Kolouri, S., Rostami, M., Martin, C.E., Hoffmann, H. (2019). Explainability methods for GCN. *IEEE CVPR 2019* (pp. 10772–10781).
- Proietti, M., Ragno, A., Rosa, B.L., Ragno, R., Capobianco, R. (2024). Explainable ai in drug discovery: self-interpretable graph neural network for molecular property prediction using concept whitening. *Machine Learning*, 113(4), 2013–2044,
- Ragno, A., La Rosa, B., Capobianco, R. (2022). Prototype-based interpretable graph neural networks. *IEEE Transactions on Artificial Intelligence*, 5(4), 1486–1495,
- Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K.T., Müller, K.-R., Montavon, G. (2022). Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11), 7581–7596, <https://doi.org/10.1109/TPAMI.2021.3115452>
- Shan, C., Shen, Y., Zhang, Y., Li, X., Li, D. (2021). Reinforcement learning enhanced explainer for graph neural networks. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J.W. Vaughan (Eds.), *Advances in neural information processing systems* (Vol. 34, pp. 22523–22533). Curran Associates, Inc.
- Vayer, T., Courty, N., Tavenard, R., Chapel, L., Flamary, R. (2019). Optimal transport for structured data with application on graphs. *ICML* (pp. 6275–6284).
- Veyrin-Forrer, L., Kamal, A., Duffner, S., Plantevit, M., Robardet, C. (2024). On GNN explainability with activation rules. *Data Min. Knowl. Discov.*, 38(5), 3227–3261, <https://doi.org/10.1007/S10618-022-00870-Z>
- Wu, B., Liu, Y., Lang, B., Huang, L. (2017). *Dgcnn: Disordered graph convolutional neural network based on the gaussian mixture model*.
- Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B. (2022, December). Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), , <https://doi.org/10.1145/3535101>

- Wu, T., Ren, H., Li, P., Leskovec, J. (2020). *Graph information bottleneck*.
- Wu, Y.-X., Wang, X., Zhang, A., He, X., Chua, T.-S. (2022). *Discovering invariant rationales for graph neural networks*.
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4-24, <https://doi.org/10.1109/TNNLS.2020.2978386>
- Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., ... Pande, V. (2017). *Moleculenet: A benchmark for molecular machine learning*. arXiv.
- Yuan, H., Tang, J., Hu, X., Ji, S. (2020). Xgmn: Towards model-level explanations of graph neural networks. *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining* (p. 430-438). New York, NY, USA: Association for Computing Machinery.
- Yuan, H., Yu, H., Wang, J., Li, K., Ji, S. (2021a). On Explainability of Graph Neural Networks via Subgraph Explorations. M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event* (Vol. 139, pp. 12241-12252). PMLR. Retrieved from <http://proceedings.mlr.press/v139/yuan21c.html>
- Yuan, H., Yu, H., Wang, J., Li, K., Ji, S. (2021b, 18-24 Jul). On explainability of graph neural networks via subgraph explorations. M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (Vol. 139, pp. 12241-12252). PMLR.
- Zhang, S., Liu, Y., Shah, N., Sun, Y. (2022). Gstarx: Explaining graph neural networks with structure-aware cooperative games. *Advances in neural information processing systems*.
- Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., Wang, C. (2019). *Hierarchical graph pooling with structure learning*.

A Time Complexity and Efficiency

In this section, first we theoretically discuss about the time complexity of INSIDE-SHAP. Second, we compare its time with the competitors to assess its time efficiency.

A.1 Computational Complexity Analysis

INSIDE-SHAP comprises multiple stages. To demonstrate its practical feasibility, we present a theoretical analysis of its computational complexity. Specifically, we evaluate the complexity of each component and derive the overall computational cost for both global and instance-level explanations using Big O notation. To avoid confusion from the variety of variables used in this section, we summarize all of them in Table 6.

Step 1: Rule Discovery

The rule extraction process in INSIDE-SHAP leverages the INSIDE framework, which consists of two principal phases:

- **Maximum Entropy Model Construction:** This phase employs the FOR-SIED method to build a probabilistic model. The corresponding time complexity is given by:

$$O(t_{\text{maxent}} \cdot \sqrt{m \cdot n}),$$

where n denotes the number of nodes in the training set, m represents the embedding dimension, and t_{maxent} is the number of iterations required to optimize the entropy (De Bie, 2011).

- **Rule Mining:** This stage utilizes a branch-and-bound strategy to identify patterns that maximize subjective interestingness. While the worst-case complexity is exponential, empirical observations suggest that the algorithm generally terminates in polynomial time.

Step 2: Surrogate Model Training

A three-layer GCN is trained as a surrogate model. Given the sparsity typical of real-world graphs, the per-iteration time complexity is:

$$O(n \cdot m^2 + e \cdot m),$$

where e denotes the number of edges. Since e is often linear in n , this simplifies to:

$$O(n \cdot m^2).$$

Step 3: Shapley Value Approximation

This step applies Kernel-Shap to approximate Shapley values:

- **Shapley Value Computation:** The time complexity is:

$$O(|R|(t_{\text{shap}} + |R|^2)),$$

where $|R|$ is the number of rules and t_{shap} denotes the number of samples used in the approximation (Lundberg et al., 2020).

- **Rule Matrix Construction:** For each instance, constructing the rule activation matrix incurs a cost of:

$$O(|V_G| \cdot |R|),$$

where $|V_G|$ is the number of nodes in the graph.

Instance-Level Explanation

Shapley values attributed to rules are propagated to the corresponding supporting nodes:

- In the worst case, where all rules apply to all nodes, the cost is:

$$O(|R| \cdot |V_G|).$$

- A tighter bound, reflecting practical conditions where each graph activates only a subset of patterns, is:

$$O(T_{\text{pattern}} \cdot |V_G|),$$

where T_{pattern} is the number of patterns activated in a specific input graph.

Global-Level Explanation

This component includes segmentation of explanations and aggregation into a summary representation.

- **Outlier Removal and Segmentation:** Outliers are removed using the LOF method, which has a complexity of:

$$O(S_r \cdot \log S_r + k \cdot S_r),$$

where $S_r = |\text{Supp}_{\mathcal{D}}(r)|$ is the number of graphs supporting rule r , and k is the number of nearest neighbors in LOF. The segmentation process has an additional cost of:

$$O(S_r \cdot \log S_r).$$

- **Mask Generation:** For each supporting graph, generating the mask for the activated subgraph has complexity:

$$O(|V_G|),$$

resulting in a total cost of $O(n)$ across the rule support.

- **FGW Distance Computation:** The pairwise distance between graphs is computed using the FGW metric (Vayer et al., 2019), with the complexity:

$$O(n_1^2 \cdot n_2 + n_1 \cdot n_2^2),$$

where n_1 and n_2 denote the number of nodes in two input graphs. Assuming n_i nodes in the i -th graph, the overall cost is:

$$\begin{aligned}
& \sum_j \sum_i O(n_i^2 \cdot n_j + n_i \cdot n_j^2) = \\
& \sum_j \sum_i O(n_i^2 \cdot n_j) + O(n_i \cdot n_j^2) = \\
& \sum_j O(\sum_i n_i^2 \cdot n_j) + O(\sum_i n_i \cdot n_j^2) = \\
& \sum_j O(\sum_i n'' \cdot n_j) + O(n'' \cdot n_j^2) = O(n'' n_s'') = O(n''^3),
\end{aligned} \tag{18}$$

where $n'' = \sum_i n_i$ and $n_s'' = \sum_i n_i^2$.

- **Heatmap Generation:** The final step involves projecting the aggregated values onto the summary graph:

$$O(n'),$$

with n' being the number of nodes in the representation.

Overall Complexity of Global-Level Explanation

Summing the complexities of the contributing components, the total cost is:

$$\begin{aligned}
& O(t_{\maxent} \cdot \sqrt{m \cdot n}) + O(\text{Pattern Mining}) + O(n \cdot m^2) + \\
& O(\mathcal{D}_T \cdot |R|(t_{\text{shap}} + |R|^2)) + O(n \cdot |R|) + O(S_r \log S_r) + O(n_r^3),
\end{aligned} \tag{19}$$

where n_r is the total number of nodes of the masks generated for the graphs within the support of r .

It is worth noting that the term $O(|\mathcal{D}_T| \cdot |R|(t_{\text{shap}} + |R|^2))$ is computed once across all rules. Hence, the total asymptotic complexity becomes:

$$\begin{aligned}
& O(t_{\maxent} \cdot \sqrt{m \cdot n}) + O(\text{Pattern Mining}) + O(n \cdot m^2) + \\
& O(\mathcal{D}_T \cdot (t_{\text{shap}} + |R|^2)) + O(n \cdot |R|) + O(S_r \log S_r) + O(n_r^3).
\end{aligned} \tag{20}$$

Instance-Level Explanation Complexity

Since both the pattern mining and surrogate model training are performed offline, their computational cost does not contribute to the per-instance complexity. Given a graph instance $G = (V_G, E_G)$, the explanation generation incurs the following cost:

$$O(|R| \cdot (t_{\text{shap}} + |R|^2)) + O(T_{\text{pattern}} \cdot |V_G|). \tag{21}$$

A.2 Evaluation of Execution Time

In this section, we evaluate the time required by each method to generate explanations.

All benchmarks were conducted on a machine with the following configuration: **Operating System:** Debian GNU/Linux 12, **CPU:** Intel(R) Xeon(R) Silver 4210R @ 2.40GHz, **RAM:** 251 GB, and **GPU:** NVIDIA RTX A6000.

Since INSIDE-SHAP involves two precomputation steps prior to generating instance-level explanations, we report the total runtime, including these precomputations, in Table 7. Note that the precomputations are performed only once on the training dataset.

Furthermore, we compare the average time required to obtain an explanation for a single instance, as reported in Table 8. As can be seen, while GraphSVX achieves the fastest computation time overall, INSIDE-SHAP consistently ranks as the best or second-best performer across all datasets among other methods.

At first glance, INSIDE-SHAP may appear to require more time than its competitors due to the initial precomputation phase. However, except for GraphSVX, INSIDE-SHAP’s significantly lower time per instance compensates for this upfront cost, especially in large datasets like Mutagenicity and Benzene. Furthermore, unlike GraphSVX, which lacks model-level explanations, INSIDE-SHAP provides them, offering a key advantage to GraphSVX.

B Investigation of the Role of Black-Box Model’s Architecture

In this section, we evaluate the instance-level explanations provided by INSIDE-SHAP, alongside other game-theoretic-based explainers, on models with architectures other than GCN. Specifically, for each dataset, we train two black-box models with identical data—one using a GAT architecture and the other using GIN—substituting the original GCN architecture. We then assess the explanations using H-Fidelity as the main evaluation metric to determine whether the architecture of the underlying model impacts the relative performance of INSIDE-SHAP.

It is important to note that the GAT-based model was unable to learn on the BA2-Motifs dataset; hence, we omit the evaluation for this case.

As the comparison of H-Fidelity shown in Table 9, when using GIN, INSIDE-SHAP achieves the highest performance on 4 out of 6 datasets: BA2-Motifs, BBBP, Mutagenicity, and Benzene. With GAT, INSIDE-SHAP outperforms all baselines across the evaluated datasets, except for BBBP, where it ranks as the second-best method. These observations suggest dominance of INSIDE-SHAP over other game-theoretic-based explainers is independent from the choice of the architecture of the black-box model, and INSIDE-SHAP is applicable to variety of architectures while preserving the performance.

C Evaluation of the Extracted Rules by INSIDE

To evaluate the quality of the extracted rules in Section ??, we adopt DiffVerify (Chataing et al., 2024) as a comparative baseline. DiffVerify is a pattern discovery framework tailored for binary datasets with class labels. It leverages a neural autoencoder architecture trained via a multi-objective loss function, comprising

reconstruction loss, classification loss, a regularization term to penalize excessively long patterns, and a coverage loss that encourages both pattern diversity and comprehensive data representation. We select DiffVersify over established baselines such as BinAPS and DiffNAPS due to its empirically demonstrated superiority in terms of coverage, purity, and accuracy of the patterns.

We train Diffversify on the activation matrices and extract the learned patterns. To further evaluate the quality of the rules extracted in Section ?? and by Diffversify, we introduce three evaluation metrics: purity, cover, and the weighted F1 score.

Purity: This metric measures the average label homogeneity within the support of the patterns. Specifically, purity is defined as follows:

$$\text{Purity}(R) = \frac{1}{|R|} \sum_{r \in R} \frac{\max_{c \in C} |\{G \in \text{Supp}_{\mathcal{D}}(r) : f(G) = c\}|}{|\text{Supp}_{\mathcal{D}}(r)|} \quad (22)$$

Cover: This metric measures the proportion of graphs that are contained in the support of at least one pattern. It is defined as:

$$\text{Cover}(R) = \frac{|\{G \in \mathcal{D} : \exists r \in R \text{ such that } G \in \text{Supp}_{\mathcal{D}}(r)\}|}{|\mathcal{D}|} \quad (23)$$

Weighted F1: For each graph G , we assign the label of the pattern with the highest purity in which G is included in its support. If no pattern includes G in its support, we assign the majority class label to G . The F1 score is then calculated between the predicted class by the GNN f and the assigned labels for each class, and the weighted average of the F1 scores is reported.

To compare the rules mined by INSIDE-GNN and Diffversify, we first sort the rules in descending order based on their purity. The purity of a single rule is measured when the rule set contains only that pattern. In other words, it is the proportion of the majority class in its support relative to the size of its support. Then, for each k , we evaluate the weighted F1 score and the cover of the top- k patterns. The results of this comparison are shown in Figure 7 and Figure 8. As both methods eventually reach a cover of 1, we focus on three factors at the point where full coverage is achieved: the number of rules required, the purity of the patterns, and their weighted F1 score. Across all datasets, INSIDE-GNN achieves full coverage with fewer rules compared to Diffversify. Furthermore, INSIDE-GNN consistently offers superior or comparable purity while maintaining a higher weighted F1 score. These observations provide justification for choosing INSIDE-GNN over Diffversify for the continuation of this work.

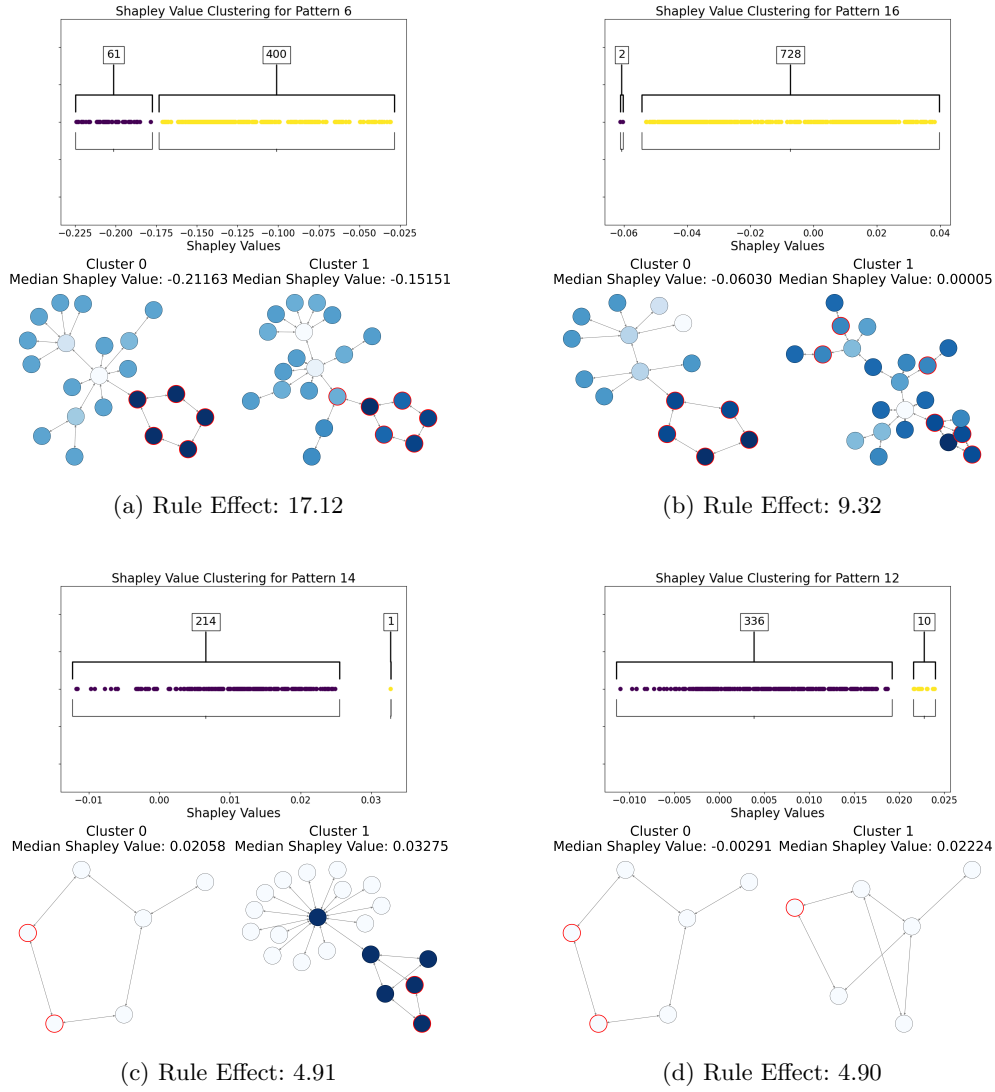


Fig. 3: BA2-Motifs Rule Representations. For each rule, the top plot displays the distribution of Shapley values for the positive class. Higher Shapley values (towards the right) indicate a stronger contribution to predicting the positive class. Each segment of the rule's support is color-coded, with the number of instances shown above each segment. The bottom figure presents a representative graph for each segment, where node importance is indicated by color intensity—the stronger the color, the greater the importance. Nodes outlined in red denote activation by the rule.

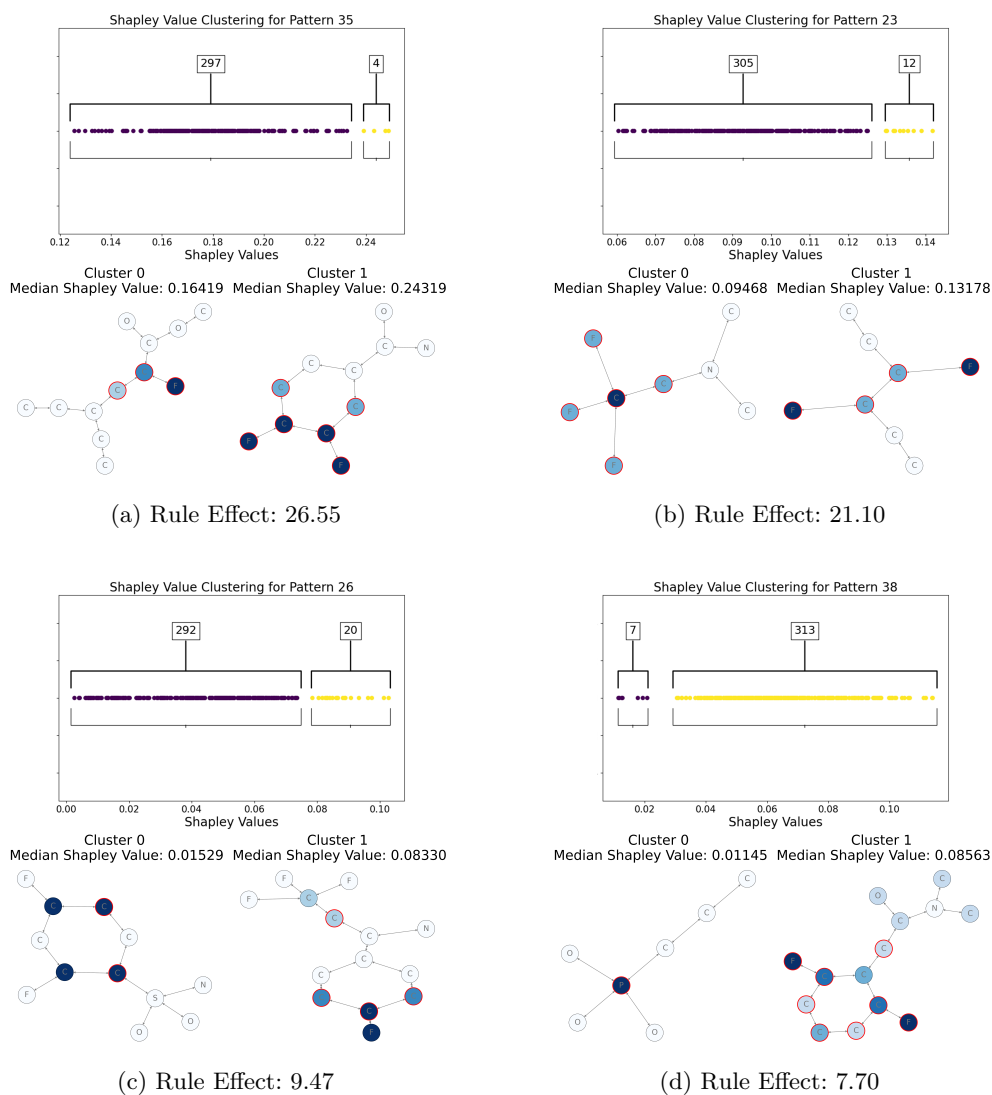


Fig. 4: AlkaneCarbonyl Rule Representations. For each rule, the top plot displays the distribution of Shapley values for the positive class. Higher Shapley values (towards the right) indicate a stronger contribution to predicting the positive class. Each segment of the rule’s support is color-coded, with the number of instances shown above each segment. The bottom figure presents a representative graph for each segment, where node importance is indicated by color intensity—the stronger the color, the greater the importance. Nodes outlined in red denote activation by the rule.

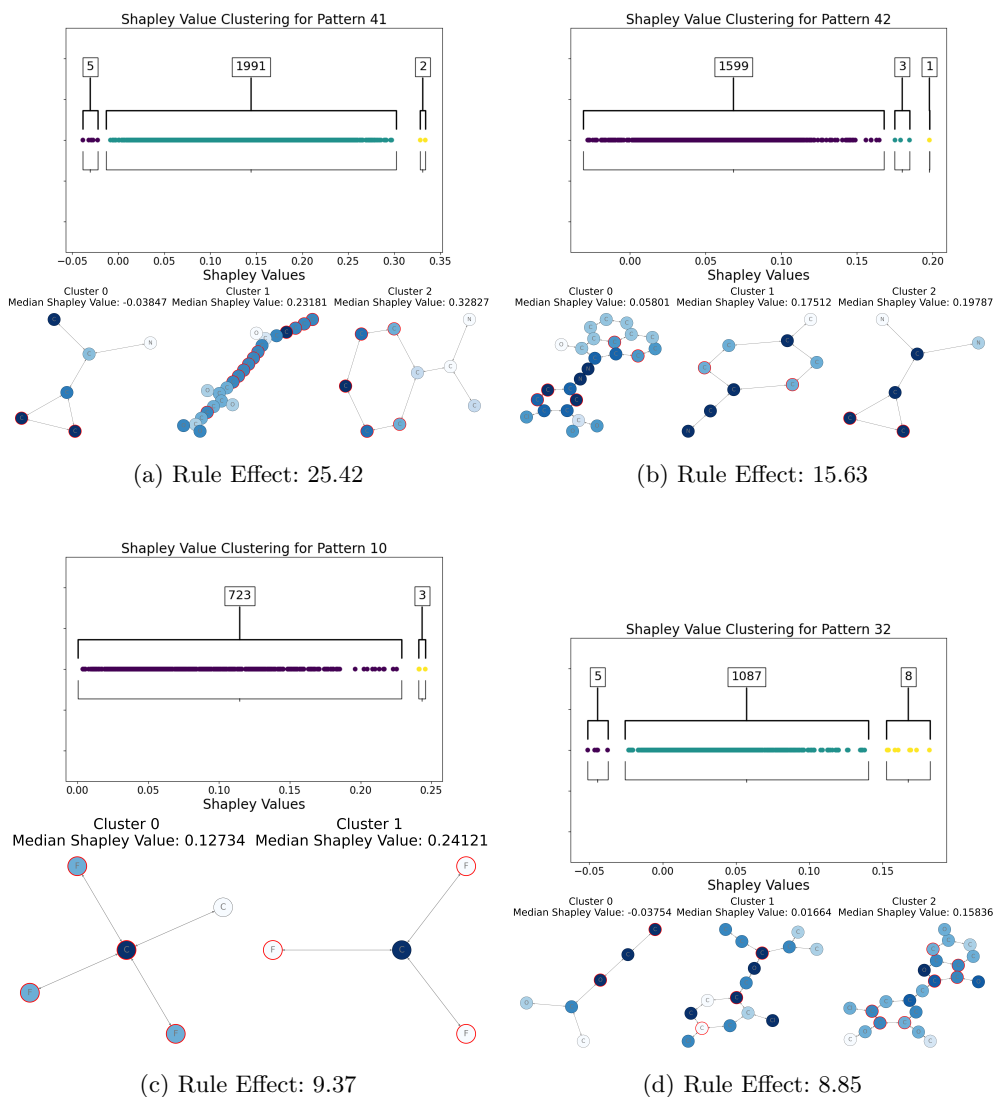
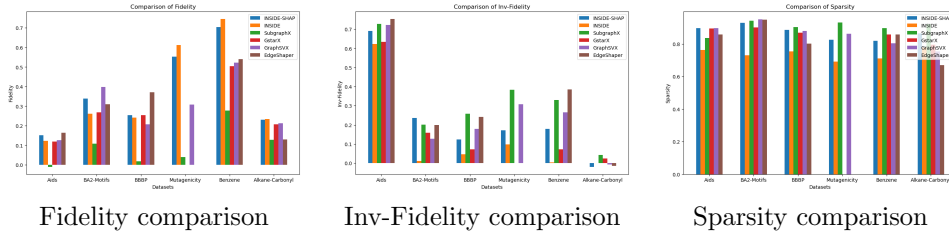


Fig. 5: Benzene Rule Representations. For each rule, the top plot displays the distribution of Shapley values for the positive class. Higher Shapley values (towards the right) indicate a stronger contribution to predicting the positive class. Each segment of the rule's support is color-coded, with the number of instances shown above each segment. The bottom plot presents a representative graph for each segment, where node importance is indicated by color intensity—the stronger the color, the greater the importance. Nodes outlined in red denote activation by the rule.



Fidelity comparison Inv-Fidelity comparison Sparsity comparison

Fig. 6: Comparison of INSIDE-SHAP in terms of Fidelity, Inv-Fidelity, and Sparsity. High values for Fidelity and Sparsity and low values for Inv-Fidelity are desired.

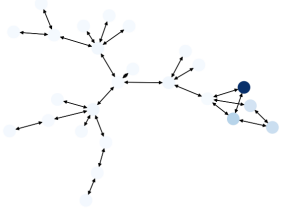
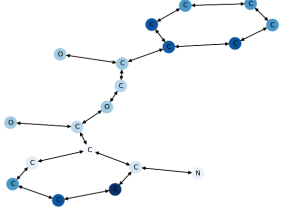
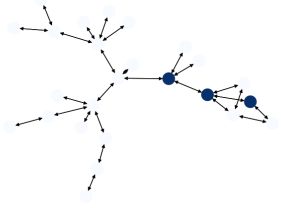
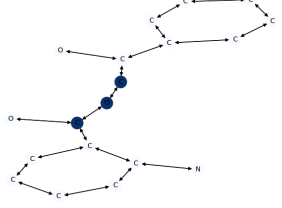
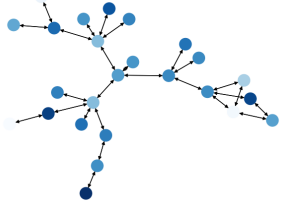
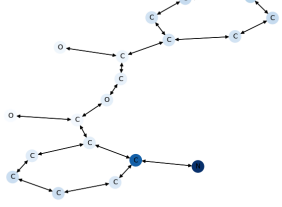
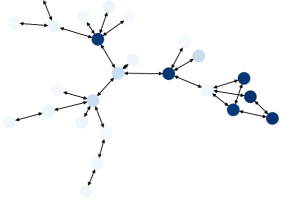
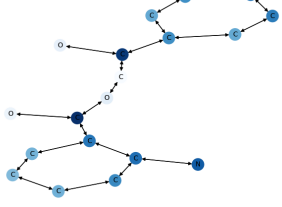
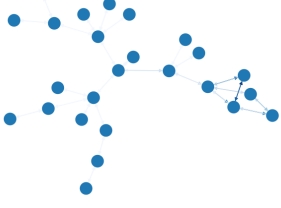
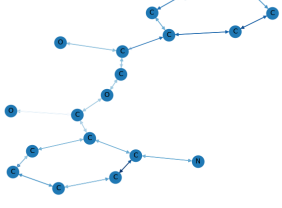
Explainer	BA2-Motifs	Benzen
INSIDE-SHAP	 <p>H-Fidelity: 0.600</p>	 <p>H-Fidelity: 0.642</p>
SubgraphX	 <p>H-Fidelity: 0.580</p>	 <p>H-Fidelity: 0.476</p>
GStartX	 <p>H-Fidelity: 0.591</p>	 <p>H-Fidelity: 0.602</p>
GraphSVX	 <p>H-Fidelity: 0.596</p>	 <p>H-Fidelity: 0.607</p>
EdgeShaper	 <p>H-Fidelity: 0.572</p>	 <p>H-Fidelity: 0.621</p>

Table 5: Comparison of explainers on BA2-Motifs and Benzene datasets with visualizations and H-Fidelity scores.

Symbol	Description
n	Number of nodes in the dataset
m	Embedding size or feature dimension
t_{maxent}	Number of iterations for maximum entropy optimization
e	Number of edges in the dataset
$ R $	Number of extracted rules
t_{shap}	Number of samples in Kernel-Shap approximation
$ V_G $	Number of nodes in a single input graph
T_{pattern}	Number of patterns activated in an instance
S_r	Support size of rule r in the dataset
k	Number of neighbors used in LOF outlier detection
n_i	Number of nodes in the i -th graph in a segment
n''	Total number of nodes across all graphs in a segment ($\sum_i n_i$)
n''_s	Sum of squared node counts across all graphs ($\sum_i n_i^2$)
n'	Number of nodes in the final representative graph
n_r	total number of nodes of the masks generated for the graphs within the support of r .
$ \mathcal{D}_T $	Number of Graphs in train dataset

Table 6: Summary of notation used in complexity analysis

Dataset	Rule Mining by INSIDE	Surrogate Model Training
AIDS	4724	731
BA2-Motifs	240	162
BBBP	5944	609
Mutagenicity	17766	1156
AlkaneCarbonyl	1513	674
Benzene	47914	1936

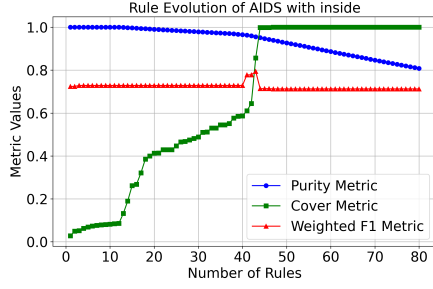
Table 7: Execution times for rule mining and surrogate model training for each dataset in seconds.

Explainer	AIDS	BA2-Motifs	BBBP	Mutagenicity	AlkaneCarbonyl	Benzene
INSIDE-SHAP	43.42	40.71	42.97	46.02	36.37	44.46
SubgraphX	35.77	224.46	52.09	145.58	71.50	89.67
GStarX	124.26	122.73	245.33	–	141.86	185.63
GraphSVX	0.76	0.60	0.53	1.53	0.47	0.58
EdgeShaper	206.45	41.75	292.27	–	45.11	69.97

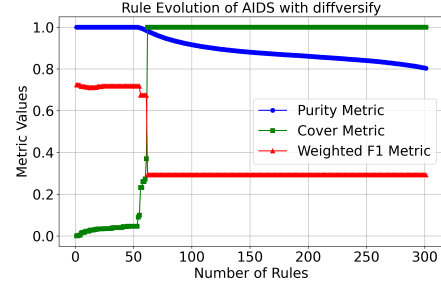
Table 8: Execution times per instance for each explainer across different datasets in seconds.

Method	AIDS	BA2-Motifs	BBBP	Mutagenicity	Alkane-Carbonyl	Benzene
GIN						
INSIDE-SHAP	0.518	0.559	0.535	0.583	0.535	0.567
SubgraphX	0.500	0.528	0.499	0.542	0.500	0.544
GStarX	0.524	0.547	0.531	–	0.546	0.549
GraphSVX	0.518	0.557	0.525	0.557	0.536	0.567
EdgeShaper	0.520	0.549	0.520	–	0.524	0.549
GAT						
INSIDE-SHAP	0.521	–	0.549	0.515	0.560	0.575
SubgraphX	0.501	–	0.502	0.508	0.499	0.504
GStarX	0.514	–	0.559	–	0.559	0.572
GraphSVX	0.515	–	0.529	0.509	0.536	0.571
EdgeShaper	0.518	–	0.528	–	0.525	0.551

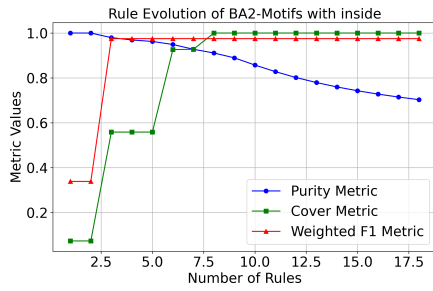
Table 9: Comparison of Explanation Methods for Black-Box GIN and GAT (Highlighted Best per Column).



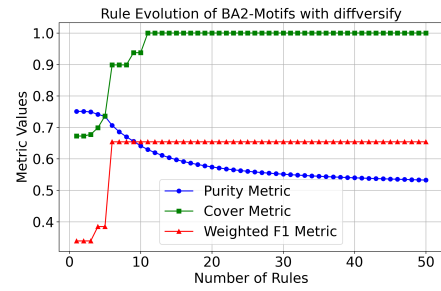
(a) AIDS - INSIDE-GNN



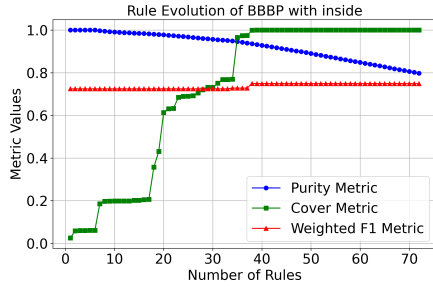
(b) AIDS - Diffversify



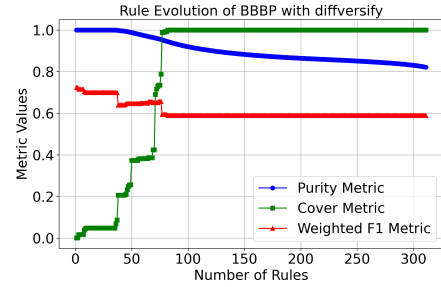
(c) BA2-Motifs - INSIDE-GNN



(d) BA2-Motifs - Diffversify

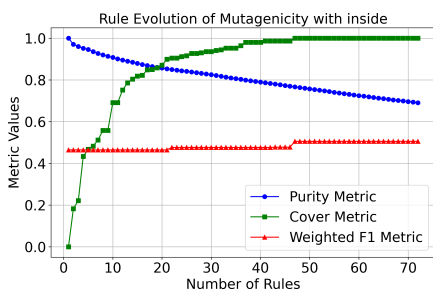


(e) BBBP - INSIDE-GNN

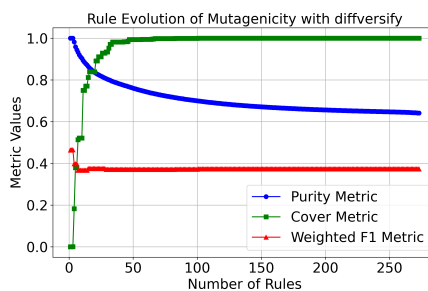


(f) BBBP - Diffversify

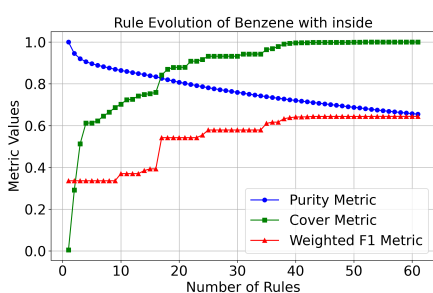
Fig. 7: Comparison of quality of the rules discovered by INSIDE-GNN and Diffversify (Part 1). High purity and weighted F1, and the low number of rules while reaching full coverage are desirable.



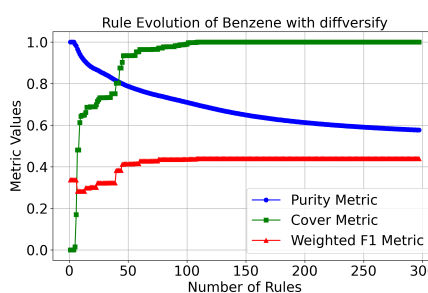
(a) Mutagenicity - INSIDE-GNN



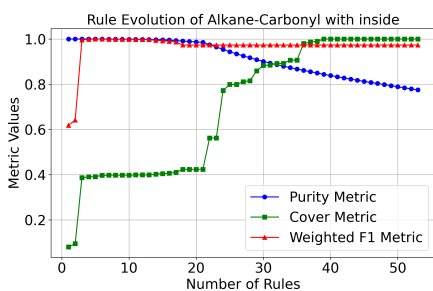
(b) Mutagenicity - Diffversify



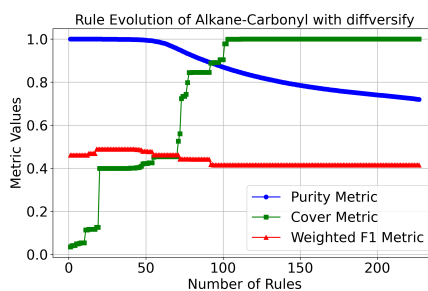
(c) Benzene - INSIDE-GNN



(d) Benzene - Diffversify



(e) Alkane-Carbonyl - INSIDE-GNN



(f) Alkane-Carbonyl - Diffversify

Fig. 8: Comparison of quality of the rules discovered by INSIDE-GNN and Diffversify (Part 2). High purity and weighted F1, and the low number of rules while reaching full coverage are desirable.