# REVELIO: Revealing Important Message Flows in Graph Neural Networks

Haoyu He
*GraphLab*
*The George Washington University*
haoyuhe@gwu.edu

Isaiah J. King
*GraphLab*
*The George Washington University*
iking5@gwu.edu

H. Howie Huang
*GraphLab*
*The George Washington University*
howie@gwu.edu

***Abstract*—Explainability is crucial for the deployment of Graph Neural Networks (GNNs) in real-world applications. Unfortunately, existing explanation methods primarily focus on identifying important graph components, such as nodes and edges, rather than providing insights into the fundamental message passing mechanisms of GNNs. This shortcoming impedes our understanding of how GNNs make predictions and limits their deployment in critical applications. In this paper, we introduce REVELIO, a novel method to provide faithful explanations of message flows in GNNs. REVELIO leverages a learning-based approach to quantify the importance of message flows, excelling in terms of faithfulness, compatibility, and efficiency. Our extensive experiments on both synthetic and real-world datasets demonstrate the superiority of REVELIO through quantitative and qualitative assessments.**

***Index Terms*—Explainability, graph neural networks, message passing neural networks**

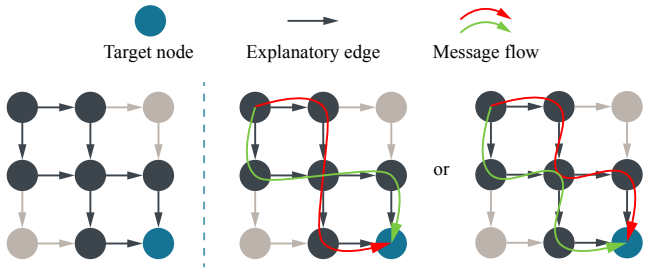



Fig. 1. Limitations of edge-based explanations. Suppose we aim to identify the top-2 important message flows from the top-left node to the bottom-right target node with a 4-layer GNN. Given a valid edge explanation (left), two distinct yet equally valid assumptions about the message flows (shown in different colors) can be made (middle and right), as both combinations align with the explanatory edges.

## I. INTRODUCTION

Graph Neural Networks (GNNs) [33, 43] have showcased their efficacy in modeling graph-structured data across diverse domains, including recommender system [45], chemical analysis [17] and natural language processing [44]. Despite their remarkable advantages, GNNs still remain black boxes, making it difficult to understand how they make predictions. This lack of transparency and explanability raises concerns about the trustworthiness of GNNs and constrains their deployment in critical applications, e.g., fair criminal justice and financial lending analysis [2]. Furthermore, the need for explainability is particularly evident in specific domains, such as drug discovery [17], where reasoning about the candidates is a crucial part of the process.

Given these imperative demands, there has been a growing focus on the explainability of GNNs [53, 54]. However, explaining GNNs presents challenges compared to traditional deep learning models. These challenges stem from GNNs' unique utilization of feature and structural information within a graph and their distinctive message-passing mechanism. To address these challenges, recent studies have introduced GNN-specialized explanation methods, such as GNNExplainer [50], PGExplainer [26] and PGM-Explainer [41]. Most of the methods identify specific graph components, such as nodes [56], edges [22], features [15] and subgraphs [52], that contribute to the predictions. Unfortunately, the existing methods fall short in providing a comprehensive understanding of the intricate message passing mechanism of GNNs, as they primarily focus on identifying a subset of important graph components.

In a typical GNN [13, 20, 40, 48], each layer passes information from one node to its neighbor nodes, and the received information is then used to update the node embeddings at that layer. In an $L$-layer GNN, information from a source node reaches a target node through $L$ connected edges (including $L+1$ nodes considering the source and target nodes). This process of transmitting information across $L$ steps is called a **message flow**. In other words, the paths of message flows towards a node can be distinguished by $L$ consecutive incoming edges, where a path may reuse the same edges multiple times, as long as they form an $L$-hop path to that node. The final embedding of a node can be regarded as the result of combining all the message flows directed towards it.

This perspective on message flow is pivotal to understanding how GNNs make predictions, benefiting both data comprehension and model development. The explanation of message flows aligns more intuitively with the fundamental message passing mechanism of GNNs. Moreover, this explanation is more fine-grained compared to edge explanation, as an individual edge within a GNN layer may carry numerous message flows. Particularly in node classification tasks, deeper GNN layers tend to use fewer edges for message transmission than a shallow layer. In other words, while the number of message flows is constant, edges in deeper layers tend to carry

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00067

more message flows. Consequently, relying solely on edge explanations may lead to confusion in intricate networks. For example, consider the example shown in Fig. 1. Given only the top-$k$ edges, we cannot determine which of several possible message flows lead to the prediction on the target node. Message flow explanation is crucial in real-world applications, such as understanding the decision-making processes and user behaviors in a recommender knowledge graph [45], as well as identifying the network flow that causes malware in a network traffic graph [6]. On the other hand, to overcome challenges such as the over-squashing problem [39] and ensure the applicability on heterophilic graphs [58], there has been recent interest in approaches that adaptively enhance specific message flows [4, 7], which indicates the need for the explanation of message flows.

Unfortunately, prior works on message flow explanation predominantly rely on traditional machine learning explanation techniques. Specifically, GNN-LRP [35] utilizes Layer-wise Relevance Propagation (LRP) to assess the importance of each message flow individually, while FlowX [12] employs Shapley values to approximate the importance of individual message flows through iterative edge removal. Both traditional techniques suffer from unfaithfulness [1, 19, 21] and face challenges when applied to the graph domain due to the unique structure of graphs and the specific architecture of GNNs [18]. Consequently, these methods often yield inaccurate explanations. In addition, both incur substantial time complexity, as their calculations highly depend on the number of message flows in the graph. Although the FlowX GPU implementation alleviates this issue through graph duplication and concurrent processing, it still requires considerable time and memory resources.

To address these challenges, we design and implement a faithful and efficient explanation method, named REVELIO. Given a pretrained GNN and an input graph with its prediction, REVELIO computes the importance scores for message flows, which can subsequently be transferred into the importance scores for edges within individual GNN layers or across the entire GNN. Specifically, REVELIO first allocates message flow masks to each message flow. During each learning epoch, REVELIO distributes these masks to the corresponding edges across GNN layers and refines them based on a predefined objective function. Notably, REVELIO is a model-agnostic approach, which is applicable to a wide range of GNNs for various tasks, including node classification and graph classification. Our experimental results consistently highlight REVELIO's superior performance in terms of faithfulness[1], compatibility and efficiency. Contributions of this paper can be summarized as follows:

- **Innovative method.** We introduce REVELIO which addresses the limitations of existing methods and excels at identifying important message flows of GNN predictions.
- **Efficient approach.** REVELIO leverages a learning-based approach to simultaneously learn the importance scores of message flows without task duplication, ensuring efficiency in the explanation process.
- **Extensive evaluation.** We evaluate REVELIO on both synthetic and real-world datasets, demonstrating its superiority over baselines through both quantitative and qualitative assessments.

[1]In this paper, faithfulness refers to the extent to which an explanation accurately reflects the model's actual reasoning process behind its predictions [27, 16]. This concept is often defined contextually based on specific explanation objectives and evaluation metrics. We adhere to the established notions of faithfulness in related works [53] and follow the evaluation guidelines in study [16].

## II. RELATED WORK

**Graph Neural Networks.** Graph Neural Networks (GNNs) are a class of deep learning models on graph-structured data [43, 20, 40, 48], the notion of which was first introduced by Gori et al. [11]. They take a graph's structure and its corresponding features as input and output meaningful node representations through a message passing mechanism. These node representations are used for various downstream tasks, which can be broadly categorized into node-focused tasks and graph-focused tasks [57]. For example, node-focused tasks include node classification [30] and link prediction [55]; graph-focused tasks include graph classification [8]. With the remarkable advancement of GNNs, they can be applied to a variety of domains, extending beyond tasks for which they were originally envisioned. There have been explorations of GNNs in computer vision [14], natural language processing [44] and recommendation systems [45]. With this wide adoption of the approach, the demand for explainable GNNs has only grown.

**Explainability of GNNs.** Driven by the demand for transparency, several approaches exist to attempt to explain the predictions of GNNs. Note that the *explainability* of GNNs provides post-hoc explanations for the outputs of pretrained GNNs, in contrast to *interpretability*, which is the ability to intrinsically understand how a model would reason about a problem [28]. We can group the explanation methods into three main classes based on granularity [54]: instance-level, group-level and class-level methods. Instance-level methods, such as GNNExplainer [50] and PGM-Explainer [41], will have different explanations for different input graphs, as they are sample-dependent methods. PGExplainer [26] and GraphMask [34] are group-level methods: they take a group of graphs as input and generate or select the important graph structures that explain the group. XGNN [51], as a class-level method, explains GNNs for specified classes by generating the most representative graph pattern for a given label. Given this taxonomy, ours is an instance-level approach; given several different graphs, REVELIO will provide a unique explanation for why the GNN classified each instance.

According to the study proposed by Yuan et al. [53], GNN explanation methods can be further categorized into five major families: gradient/feature-based methods [31], perturbation-based methods [42], surrogate methods [15], decomposition methods [9], and generation methods [51]. The categorization of explanation methods is a complex task, as it necessitates the consideration of various critical attributes. For example,

model-specific [35] vs. model-agnostic methods [12] based on design, black-box [41] vs. white-box methods [26] based on the prerequisite knowledge, etc. Moreover, recent efforts have expanded the spectrum of explanatory perspectives, encompassing counterfactual explanation [25] and causal explanation [22, 23].

**Message flow explanation.** The majority of the existing methods are limited to identifying a subset of important graph components (nodes, edges and node features). To the best of our knowledge, GNN-LRP [35] and FlowX [12] are the most closely related methods to ours as they explain GNNs by message flows. GNN-LRP is a decomposition method while FLowX is a perturbation-based method. GNN-LRP uses iterative LRP operations to trace the relevance of node features to the output class by following the backward paths of message flows. Specifically, the importance score for each message flow is derived using an $L$-order Taylor expansion of the model with respect to GNN layers, with each term approximated via backpropagation. FlowX, on the other hand, applies Shapley values from the game theory to approximate the importance of individual message flows. It does this by iteratively removing edges and refining the results through learning. FlowX calculates the marginal contribution of each message flow by removing the edge that carries it and then dividing the resulting prediction difference by the number of removed message flows.

Both methods are subject to issues of unfaithfulness and significant time and memory costs. First, regarding unfaithfulness, GNN-LRP is constrained to activation functions such as ReLU [3], and simple neural networks. Studies [1, 19] also indicate that LRP may yield incorrect explanations even in the simplest setting. As a model-specific method, GNN-LRP is only applicable to certain GNN architectures and its implementation has to be specifically adapted, which is challenging to deploy for non-experts. On the other hand, FlowX also struggles with unfaithfulness, as Shapley values may introduce unintended, mathematically formalizable properties in feature importance explanations [21]. Additionally, removing an edge in a given GNN layer impacts all message flows involving that edge, which limits the choice of message flows to remove and may lead to inaccurate Shapley value estimation.

Second, in terms of computational complexity, GNN-LRP calculates the importance of each message flow individually, while FlowX conducts sampling and computes the marginal contributions of message flows separately, resulting in significant time overhead. Even though FlowX achieves outstanding results, we argue that the trade-off between performance and cost is impractical, especially with large networks and deep GNNs; additionally, using a learning-based approach solely is enough to yield comparable or superior results with significantly reduced costs. While parallelization can mitigate this overhead, it comes at the cost of considerable memory usage due to task duplication.

It is worth mentioning that efforts have been made to reduce the time complexity of GNN-LRP. Specifically, sGNN-LRP [46] reduces the complexity from exponential to linear with respect to the number of GNN layers; EMP-neu and AMP-ave [47] find the top-$k$ relevant walks within polynomial time complexity.

TABLE I
SUMMARY OF NOTATIONS.

| Notation | Description |
|---|---|
| $G$ | A graph with nodes and edges |
| $V$ | Node set |
| $E$ | Edge set |
| $X$ | Node features |
| $Y$ | Node/graph labels |
| $\Phi$ | GNN model |
| $\mathcal{F}$ | Message flows |
| $M$ | Learnable message flow masks |
| $\omega[\cdot]$ | The ultimate importance scores |

## III. PRELIMINARIES

We summarize the frequently used notations of this paper in Table I. In general, capital letters are used to denote sets or matrices, while their corresponding lowercase represent individual elements; bold lowercase letters represent vectors.

**Graphs.** Let $G = (V, E)$ represent a graph with $V$ denoting the nodes and $E$ denoting the edges. The variable $e_{ij}$ represents a directed edge from node $v_i$ to $v_j$. To avoid confusion, for specific node id's we add comma delimeters. For example, we write $e_{1,23}$ to represent an edge from $v_1$ to $v_{23}$, instead of $e_{123}$. In this paper, we consider nodes to be associated with node features $X$. We also assume there exists a set of labels $Y$ either for each node, or for each graph.

**Graph Neural Networks.** Graph Neural Networks (GNNs), denoted as $\Phi$, are specialized neural networks that process graph-structured data. These networks take a graph and node features as input and aim to produce a meaningful representation, commonly a set of node embeddings $Z$. Node information is transmitted to neighbor nodes through connected edges, via a message passing mechanism: the fundamental building block of GNNs. We consider three essential steps in a GNN layer: message calculation, message aggregation, and node update. In the first step, for each edge $e_{ij}$, the GNN calculates the message to be transmitted: $\boldsymbol{m}_{ij}^l = \textsc{Msg}(\boldsymbol{h}_i^{l-1}, \boldsymbol{h}_j^{l-1}, e_{ij})$, where $l$ is the current GNN layer and $\boldsymbol{h}_i^{l-1}$ is the node representation of $v_i$ from the previous layer. Next, for each node $v_j \in V$, the GNN aggregates all the received messages from its neighbor nodes $\mathcal{N}_j$: $\boldsymbol{m}_j^l = \textsc{Agg}(\{\boldsymbol{m}_{ij}^l \mid v_j \in \mathcal{N}_j\})$. Last, the GNN updates the node representation from the previous result, which serves as the output of this layer: $\boldsymbol{h}_j^l = \textsc{Update}(\boldsymbol{m}_j^l, \boldsymbol{h}_j^{l-1})$. The final representations are used for tasks such as node classification [30], link prediction [55] and graph classification [8].

**Message flow.** A message flow in a GNN model is the sequential transmission of information over $L$ consecutive steps. A message flow can be uniquely identified by a sequence of nodes or edges. We denote flows as $\mathcal{F}_*$, where the subscript identifies the ordered sequence of nodes that defines that flow. For example, in a 2-layer GNN, $\mathcal{F}_{ijk}$ represents the message

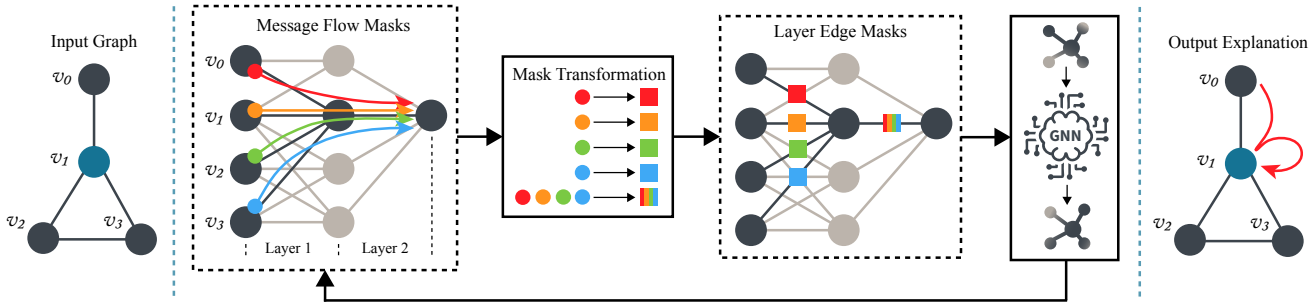



Fig. 2. The Workflow of REVELIO. In this example, we consider node classification task on $v_1$ with a 2-layer GNN. We illustrate all the valid layer edges but only show the message flows on layer edge $e^1_{1,1}$ (i.e., $\mathcal{F}_{?1,1}$) for simplicity. We treat the message flow masks as learnable parameters. For each learning epoch, we transform the message flow masks to layer edge masks, then apply them to the input graph and each GNN layer, finally we update the message flow masks based on the feedback of GNN.

flow that starts from $v_i$, passes through $v_j$, and ends at $v_k$. Alternatively, it goes through the path $\{e_{ij}, e_{jk}\}$. In an $L$-layer GNN, all the message flows start from the input layer and end in the $L$-th layer, therefore, their paths are all constructed with a sequence of $L$ edges. Following Gui et al. [12], for the ease of notations, we use "$*$" to denote a node sequence of any length and "?" to represent any single node. Additionally, "?$\{n\}$" represents a sequence of $n$ nodes. For example, $\mathcal{F}_{i*j}$ represents any message flows that start from $v_i$ and end at $v_j$; $\mathcal{F}_{?\{2\}ij*}$ represents any message flows that take the third step on $e_{ij}$.

## IV. METHODOLOGY

In this paper, we aim to explain GNNs from the perspective of message flows. We propose a perturbation-based and instance-level method, named REVELIO, to effectively identify the important message flows. REVELIO utilizes masking to measure the importance of message flows in GNN predictions. However, due to the common aggregation operations in GNNs, it is challenging to explicitly distinguish the importance score of each message flow. To address this issue, REVELIO effectively transforms message flow masks into corresponding edge masks at each GNN layer, thereby enabling the direct optimization of message flow masks. The output of REVELIO is the importance scores of message flows, which can subsequently be translated into important graph structures. Fig. 2 shows the overall workflow of REVELIO, which we will elaborate on in this section.

### A. Explanation Objective

In this work, we consider two types of explanations: factual and counterfactual. Factual explanations provide the important input components with the maximum contribution to the prediction. On the other hand, counterfactual explanations aim to find the smallest possible components that, when removed, cause the prediction to change[2]. In other words, factual explanations aim to identify *sufficient* components, whereas counterfactual explanations focus on identifying *necessary* components. For example, in a traffic network, a factual explanation seeks to answer the question: *"Which traffic flows are sufficient to trigger or exacerbate a traffic jam?"*, while a counterfactual explanation asks, *"Which traffic flows, if removed, would prevent a traffic jam?"*.

**Factual explanation.** Our objective can be formalized as finding the message flows that obtain the maximum prediction probability. This can be formulated as minimizing the conditional entropy:

$$\mathcal{L}_{obj} = -\log P_\Phi(Y = c|G, \hat{\mathcal{F}}), \tag{1}$$

where $P_\Phi$ is the probability distribution that the GNN outputs, $c$ is the class to be explained and $\hat{\mathcal{F}}$ is reduced message flows. The objective function maximizes mutual information between the target label distribution and the explanation. This suggests that the explanatory message flows, $\hat{\mathcal{F}}$, should be sufficient to generate similar or higher predicted probabilities. Meanwhile, the excluded (unexplanatory) components should make minimal or negative contributions to the prediction. Because the original prediction is constant, Eq. (1) can be interpreted as a measurement of prediction drop after removing the excluded components.

**Counterfactual explanation.** Unlike prior works [12, 25] that reverse the sign of Eq. (1) as their objective function for counterfactual explanation, we use:

$$\mathcal{L}_{obj} = -\log\left(1 - P_\Phi(Y = c|G, \hat{\mathcal{F}})\right). \tag{2}$$

This function is equivalent to binary cross entropy with the target set to label 0, while the class being explained is designated as label 1. Compared to negated Eq. (1), it ensures a larger penalty when the prediction deviates from the target class and a smaller penalty when it is close to the target class. Opposite to factual explanation, the excluded message flows are necessary to the prediction and hence are our explanatory components.

### B. Design Details

**Challenge and solution.** Although the established masking approach has shown its effectiveness in explaining edges [26, 34, 50], it cannot be directly extended to explain message flows

[2]Following existing works [5, 25, 38], we adapt the definition of counterfactual explanation to the graph domain. In this paper, we restrict the perturbation to deletions, meaning that explanatory subgraphs are only obtained by removing components from the original graph.

because message flows cannot be accessed as individual units. Notably, the path of a message flow consists of $L$ sequential edges, in the order of the GNN layers. For example, $\mathcal{F}_{?\{2\}ij*}$ represents a set of message flows that pass $e_{ij}$ in the third layer of a GNN. To distinguish edges from different layers, we use $e_{ij}^l$ to represent an edge from $v_i$ to $v_j$ that is processed in the $l$-th layer of a GNN. We will refer to this as a *layer edge*. For example, the path $\mathcal{F}_{ijk}$ can instead be represented by a sequence of layer edges, $\{e_{ij}^1, e_{jk}^2\}$.

Our approach to masking message flows involves transforming message flow masks so that they can be directly applied to individual layer edges. Specifically, we distribute the message flow masks to their corresponding layer edges and aggregate the received masks for each layer edge, which is formulated as:

$$\omega\left[e_{ij}^l\right] = f\left(\omega\left[\mathcal{F}_{?\{l-1\}ij*}\right]\right), \tag{3}$$

where $\omega[\cdot]$ represents the importance scores as well as masks, and $f(\cdot)$ is the summation function. A message flow mask collectively controls the importance of a distinct sequence of layer edges and serves as the measurement of importance for the entire sequence as a cohesive unit. While the reduction of a message flow mask can affect other message flows, since each path of message flows consists of unique layer edges, only the corresponding message flow goes through exponential reduction. In a 3-layer GNN, for example, when $\omega[\mathcal{F}_{ijkl}]$ decreases, only the contribution of $\mathcal{F}_{ijkl}$ will drop three times consecutively. Note that we do not use individual layer edge masks to estimate message flow importance. Each layer edge mask reflects only the importance of its corresponding edge. In contrast, a message flow mask collectively controls the contribution of a distinct sequence of layer edges and represents the importance of that specific layer edge sequence.

**Message flow masks.** We first initialize $M \in \mathbb{R}^{|\mathcal{F}|}$ as the message flow masks that we need to learn. To limit the upper and lower bound of the masks, we map them to importance scores via $\tanh$:

$$\omega\left[\mathcal{F}\right] = \tanh\left(M\right). \tag{4}$$

Here, we incorporate negative scores to avoid excessive accumulation in Eq. (3), as these scores will be aggregated as layer edge masks in the subsequent step. Compared with sigmoid, using $\tanh$ also prevents the problem where layer edges with more message flows tend to gain higher mask values, even if these edges do not carry important message flows.

**Mask transformation.** In this stage, the importance scores are satisfactory for ranking edge importance within a given layer. However, the impact of message flow masks may vary across different GNN layers. For node classification tasks, deeper layer edges tend to carry a higher number of message flows. For example, in Fig. 2, the importance scores of edges in the first layer are controlled by individual message flows; while in layer 2, the contribution of edges is quantified by an accumulated score of their corresponding message flows. Additionally, the combinations of message flow masks are different across layers. Thus, the distribution of accumulated scores of a certain layer can be significantly different from other layers.

TABLE II
TIME COMPLEXITY OF BASELINES.

| Methods | Time Complexity |
|---|---|
| GNNExplainer | $O(T(\lvert E\rvert + \mathcal{T}_\Phi))$ |
| GNN-LRP | $O(\lvert\mathcal{F}\rvert\,(\lvert\mathbf{x}\rvert + L\lvert\mathbf{h}\rvert + \mathcal{T}_\Phi))$ |
| FlowX | $O(S(\lvert\mathcal{F}\rvert + L\,\lvert E\rvert\,\mathcal{T}_\Phi))$ |
| REVELIO | $O(T(L\,\lvert\mathcal{F}\rvert + \mathcal{T}_\Phi))$ |

To better align the accumulated results with each layer, as well as to consider the different impact across layers, we introduce a weight vector $\boldsymbol{w} \in \mathbb{R}^L$ to adjust the accumulated scores for each layer. For example, $w_l$ is shared across all the accumulated scores within the $l$-th layer. However, directly multiplying this term with the previous results in Eq. (3) is not feasible due to its uncertain signs. To ensure that higher message flow scores indicate greater contribution, it is crucial to use an activation function that guarantees the output values of $\boldsymbol{w}$ are positive. Considering the distribution of aggregated scores and the learning process of the weight vector, an optimal activation function applied to $\boldsymbol{w}$ should ideally have a low gradient in the interval $(0, 1)$, which is common in dense graphs within deep layers, and a high gradient in the interval $(1, +\infty)$. Activation functions such as exp and softplus are good candidates. Through empirical evaluation, we have found that exp performs most effectively under these conditions. We avoid using ReLU due to its potential to invalidate masks and disrupt the entire GNN when values reach zero. Therefore, the final importance score (mask) of a layer edge can be calculated by:

$$\omega\left[e_{ij}^l\right] = \sigma\left(\sum \omega\left[\mathcal{F}_{?\{l-1\}ij*}\right] \cdot \exp(w_l)\right), \tag{5}$$

where $\sigma(\cdot)$ is sigmoid function to limit the aggregated scores within $(0, 1)$.

**Layer edge masks.** Similar to other perturbation-based methods [50, 26], the importance scores of layer edges serve as additional edge attentions through the message passing mechanism, which indicate the portion of a message to be transmitted. Specifically, we rewrite the first step of a GNN layer as:

$$\boldsymbol{m}_{ij}^l = \textsc{Msg}(\boldsymbol{h}_i^{l-1}, \boldsymbol{h}_j^{l-1}, e_{ij}^l) \cdot \omega\left[e_{ij}^l\right]. \tag{6}$$

The message flow masks are learned through Eq. (1).

### C. Implementation

**Matrix implementation.** In practice, we can leverage the efficiency and convenience of matrix multiplication to calculate the layer edge importance. Here, we introduce a sparse binary matrix $I \in \{0, 1\}^{L \times |E| \times |\mathcal{F}|}$, indicating whether a layer edge carries a specific message flow. For example, $I_{ijk} = 1$ indicates that the $j$-th edge in the $i$-th layer carries the $k$-th message flow. The matrix of layer edge importance can be computed by:

$$\omega\left[\boldsymbol{E}\right] = \sigma\left(I \cdot \omega\left[\mathcal{F}\right] \odot \exp(\boldsymbol{w})\right). \tag{7}$$

TABLE III
STATISTICS OF DATASETS. HERE, ASTERISKS* DENOTE SYNTHETIC DATASETS AND UNDERLINES DENOTE GRAPH CLASSIFICATION DATASETS.

| Dataset | Reference | # graphs | # nodes | # edges | # features | # classes | GCN Acc. | GIN Acc. | GAT Acc. |
|---|---|---|---|---|---|---|---|---|---|
| Cora | [49] | 1 | 2,708 | 10,556 | 1,433 | 7 | 86.0% | 83.6% | 85.8% |
| Citeseer | | 1 | 3,327 | 9,104 | 3,703 | 6 | 75.2% | 69.8% | 73.9% |
| PubMed | | 1 | 19,717 | 88,648 | 500 | 3 | 87.2% | 87.3% | 85.5% |
| BA-Shapes* | [50] | 1 | 700 | 4,110 | 10 | 4 | 95.7% | 92.1% | N/A |
| Tree-Cycles* | | 1 | 871 | 1,942 | 10 | 2 | 82.3% | 92.6% | N/A |
| MUTAG | [29] | 188 | 17.9 | 39.6 | 7 | 2 | 81.1% | 86.5% | 75.7% |
| BBBP | [32] | 2,039 | 24.1 | 51.9 | 9 | 2 | 81.8% | 80.6% | 85.7% |
| BA-2motifs* | [26] | 1,000 | 25.0 | 50.9 | 10 | 2 | 98.0% | 99.0% | N/A |

Here, $\boldsymbol{E}$ is the set of layer edges and $\odot$ is element-wise multiplication. When performing element-wise multiplication between a matrix and a vector, the vector's dimension should match the first dimension of the matrix, so that the vector can be duplicated to match the matrix's size. As a result, the size of $\omega[\boldsymbol{E}]$ is $L \times |E|$, as it stores the importance scores of layer edges.

**Counterfactual explanation.** We replace Eq. (1) with Eq. (2) for counterfactual explanation. The importance scores of message flows are learned in the same way as before, then we assign the negative of these scores as the final values: $\omega'[\mathcal{F}] = -\omega[\mathcal{F}]$. As before, the importance score of a layer edge is calculated using Eq. (5), which can also be reduced to $\omega'\left[e_{ij}^l\right] = 1 - \omega\left[e_{ij}^l\right]$. As a result, the values remain consistent with those from factual explanation, meaning that they fall within the same ranges, and higher values indicate greater importance.

**Additional constraints** are deployed during mask learning to serve desired properties, e.g., controlling the sparsity of the explanatory subgraph. By doing that, we add a regularization term to penalize the density of the subgraph. Particularly, we average the importance scores of edges across different layers, while skipping those that are unused by GNN layers. Then we add the result to Eq. (1) as the final objective function:

$$\mathcal{L} = \mathcal{L}_{obj} + \alpha \cdot \text{mean}\left(\omega[\boldsymbol{E}]\right), \tag{8}$$

where $\alpha$ is the hyperparameter indicating the strength of penalty. For counterfactual explanation, we adjust Eq. (8) to:

$$\mathcal{L} = \mathcal{L}_{obj} + \alpha \cdot \text{mean}\left(1 - \omega[\boldsymbol{E}]\right), \tag{9}$$

with Eq. (2) being the objective function.

### D. Complexity Analysis

In Table II, we summarize the time complexity of REVELIO and other representative instance-level methods. Here, we do not consider parallel implementation. For better analysis, we include the time complexity of the forward operation of GNNs, denoted as $O(\mathcal{T}_\Phi)$, which is considered the most time-consuming operation of the process. Here, $T$ represents the number of training epochs and $S$ represents the number of flow sampling iterations (only used by FlowX). In one training epoch of REVELIO, each message flow is distributed to $L$ layer edges. Taking parameter updates into account, the time complexity per epoch can be written as $O(L|\mathcal{F}|)$, excluding the model's forward operation.

REVELIO requires more running time than GNNExplainer due to the mask update for message flows. The upper bound of the number of message flows is $O((d_-)^L)$ for node classification tasks and $O(|E|(d_-)^{L-1})$ for graph classification tasks, where $d_-$ is the largest incoming degree of nodes. However, the dominant term for the complexity of REVELIO is $O(T\mathcal{T}_\Phi)$, regardless of the graph size and GNN depth (i.e., regardless of the number of message flows). In contrast, the dominant term for GNN-LRP is $O(|\mathcal{F}|\mathcal{T}_\Phi)$, and $O(SL|E|\mathcal{T}_\Phi)$ for FlowX, where $S$ increases with the number of message flows. In practice, one would likely find that $T \ll L|E| \ll |\mathcal{F}|$, making REVELIO the only feasible option in real-world scenarios.

## V. EXPERIMENT

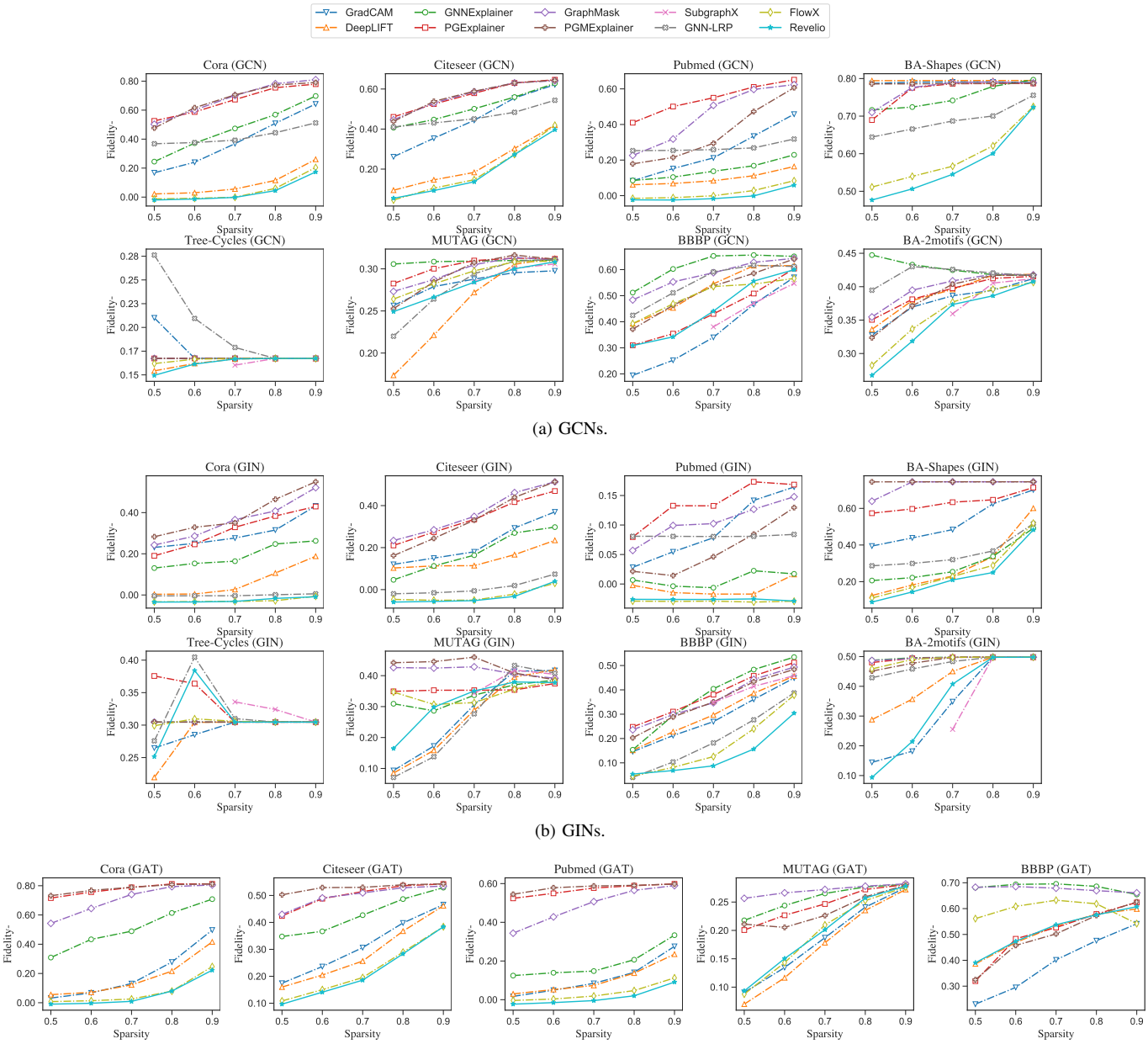
### A. Experimental Setup

**Datasets.** We evaluate the methods on five real-world datasets and three synthetic datasets. Table III shows the metadata and more details of these datasets. Here, edges are considered as directed without self-loops. For graph classification datasets, the numbers of nodes and edges are the average across each graph.

**Models.** We use three GNNs with different structures as the target models: Graph Convolutional Networks (GCN) [20], Graph Isomorphism Network (GIN) [48] and Graph Attention Network (GAT) [40]. All models use three layers, and GATs use 8 attention heads. We present the performance accuracy of each trained model on each dataset in Table III.

**Baselines.** For fair comparison, we use baselines that can generate edge explanation. We compare REVELIO with two traditional methods (GradCAM [36] and DeepLIFT [37]), three edge-centric methods (GNNExplainer [50], PGExplainer [26] and GraphMask [34]), one node-centric method (PGMExplainer [41]), one subgraph-centric method (SubgraphX [52]), and two flow-based methods (GNN-LRP [35] and FlowX [12]). Only GNNExplainer, PGMExplainer, SubgraphX, FlowX and REVELIO do not require full access to the GNN architecture.

We use PyG implementation [10] for the edge-centric methods and the DIG implementation [24] for the others. We use the same learning rates as in the original implementations for GNNExplainer, PGExplainer and GraphMask, which are 1e-2, 3e-3 and 1e-2 respectively. We assigned 500, 500,




(a) GCNs.

(b) GINs.

(c) GATs.

Fig. 3. Comparison of $Fidelity-$ under different sparsity rate with GNNs.

and 200 learning epochs to GNNExplainer, PGExplainer and GraphMask. Otherwise, We adopt the original settings for the baseline methods. For REVELIO, we set the learning rate to be 1e-2 and conduct 500 learning epochs for each instance and adaptively adjust $\alpha$ for each dataset.

### B. Quantitative Study

**Metrics.** For factual explanation, to evaluate how well the explanatory graphs can retain the original predictions, we employ the metric $Fidelity-$ as follows [53]:

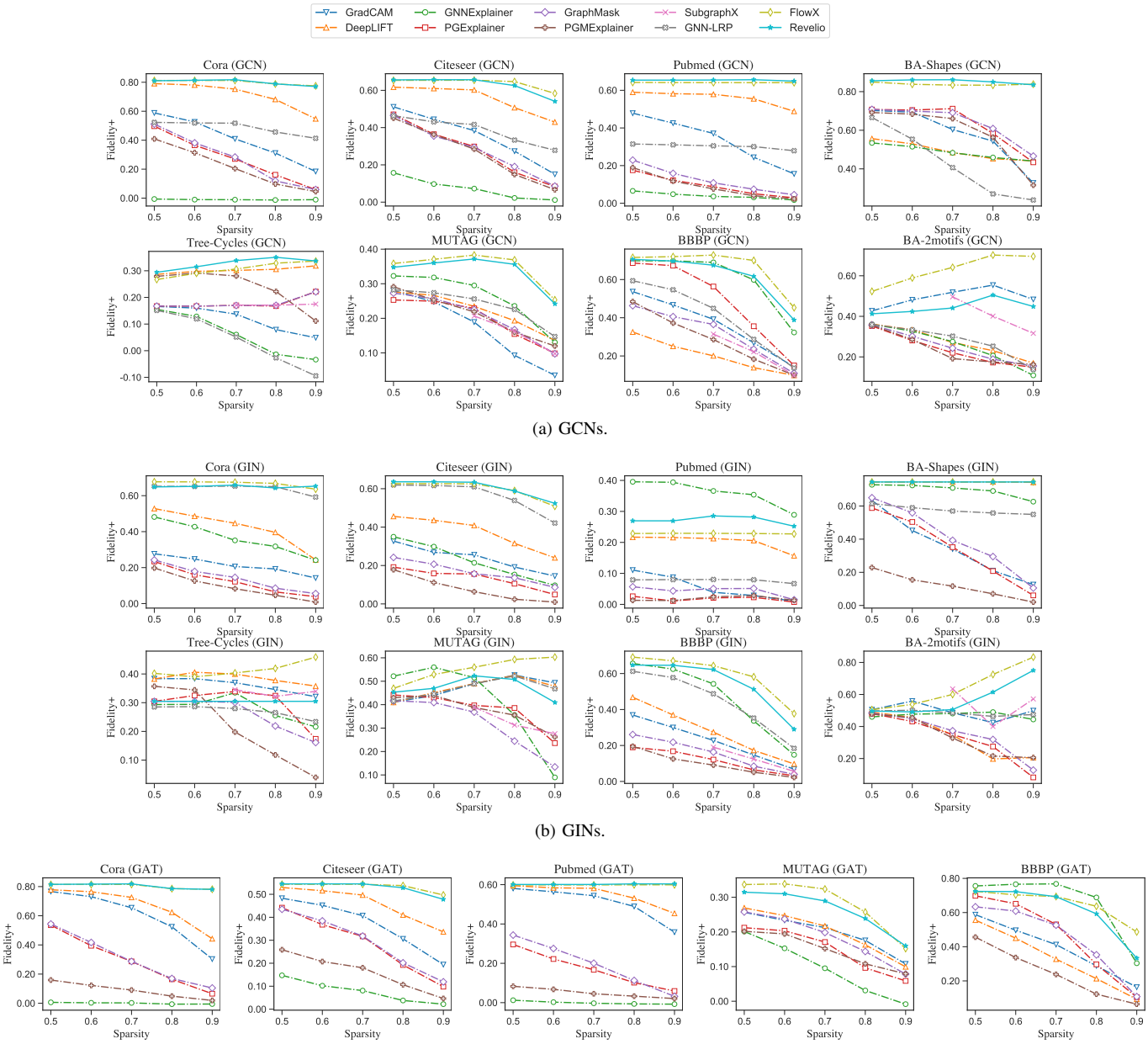
$$Fidelity- = \frac{1}{N}\sum_{i=1}^{N}\left(P_\Phi(y_i|G_i) - P_\Phi(y_i|G_i^{(s)})\right), \quad (10)$$

where $N$ is the number of instances and $G^{(s)}$ is the explanatory subgraph. This metric measures the reduction in predicted probabilities by keeping the important graph components while removing the unimportant ones, aligning with our explanation objective.

For counterfactual explanation, we use $Fidelity+$ to assess performance:

$$Fidelity+ = \frac{1}{N}\sum_{i=1}^{N}\left(P_\Phi(y_i|G_i) - P_\Phi(y_i|G_i^{(\bar{s})})\right), \quad (11)$$

where $G^{(\bar{s})}$ is the unexplanatory subgraph formed by removing the important components. This metric evaluates the probabil-




(a) GCNs.

(b) GINs.

(c) GATs.

Fig. 4. Comparison of $Fidelity+$ under different sparsity rate with GNNs.

ity drop without the explanatory components, which is also consistent with our explanation objective.

A smaller $Fidelity-$ signifies better performance in factual explanation, while a bigger $Fidelity+$ indicates better performance in counterfactual explanation. The value of both $Fidelity$s falls within $(1/C-1, 1)$ theoretically, where $C$ is the number of classes. The negative value indicates that removing the input components can increase the predicted probability.

**Specification.** We randomly select 50 target instances from each dataset for evaluation, regardless of their ground-truth labels and predicted labels. We assess the performance of different methods by conducting a comparison of $Fidelity$s under identical sparsity levels. The sparsity ratio denotes the proportion of edges removed from the original graph. That is, higher sparsity indicates a smaller number of important edges are retained. Specifically, we remove an equivalent number of explanatory/unexplanatory edges from the graph and calculate the $Fidelity-$/$Fidelity+$ for each method. Please note that GATs do not work on synthetic datasets and GNN-LRP is not compatible with GATs. Due to significant time consumption, we evaluated SubgraphX on the last four datasets using a subset of sparsity values with GCNs and GINs.

**Factual explanation.** We report the factual explanation results with various GNNs in Fig. 3, where we show the plots of $Fidelity-$ with respect to different sparsity values.

TABLE IV
EXPLANATION AUC ON SYNTHETIC DATASETS WITH GCNS AND GINS. WE HIGHLIGHT THE BEST RESULT IN **BOLD**, AND THE RUNNER-UPS ARE UNDERLINED.

| | | GCNs | | | GINs | | |
|---|---|---|---|---|---|---|---|
| | Method | BA-Shapes | Tree-Cycles | BA-2motifs | BA-Shapes | Tree-Cycles | BA-2motifs |
| General | GradCAM | 0.246 | 0.548 | 0.267 | 0.560 | **0.954** | **0.978** |
| | DeepLIFT | 0.003 | **0.889** | **0.842** | 0.601 | 0.857 | 0.789 |
| | PGMExplainer | 0.127 | 0.155 | 0.641 | 0.120 | 0.053 | 0.630 |
| | SubgraphX | N/A | 0.720 | 0.586 | N/A | 0.720 | 0.819 |
| | GNN-LRP | 0.706 | 0.153 | 0.472 | 0.157 | 0.764 | 0.787 |
| Factual explanation | GNNExplainer | 0.657 | 0.401 | 0.453 | 0.719 | 0.447 | 0.473 |
| | PGExplainer | 0.500 | 0.500 | 0.184 | **0.747** | 0.502 | 0.494 |
| | GraphMask | 0.499 | 0.500 | 0.485 | 0.520 | 0.508 | 0.536 |
| | FlowX | **0.919** | 0.742 | 0.590 | 0.633 | 0.734 | 0.317 |
| | REVELIO | 0.783 | 0.792 | 0.746 | 0.719 | 0.836 | **0.978** |
| Counterfactual explanation | GNNExplainer | 0.112 | 0.232 | 0.787 | 0.480 | 0.410 | 0.407 |
| | PGExplainer | 0.500 | 0.500 | 0.500 | 0.667 | 0.185 | 0.504 |
| | GraphMask | 0.501 | 0.500 | 0.494 | 0.513 | 0.492 | 0.469 |
| | FlowX | 0.804 | 0.716 | 0.414 | 0.652 | 0.777 | 0.410 |
| | REVELIO | 0.570 | 0.628 | 0.629 | 0.746 | 0.631 | 0.686 |

Generally, REVELIO surpasses other baseline methods. In particular, REVELIO demonstrates superior performance in node classification tasks, where fewer message flows overlap within a single layer edge. While some baselines, e.g., GradCAM, outperform REVELIO on specific datasets, they often decline significantly on others. In contrast, REVELIO consistently provides stable and reliable explanations.

We observe that FlowX and REVELIO perform better on most datasets than other methods, which supports our claim that fine-grained message flow explanation is advantageous. However, REVELIO does not always achieve the lowest $Fidelity-$ scores for several reasons: 1) We used universal hyperparameter values (e.g., $\alpha$ and learning rate) for all sparsity values to find the overall best result, rather than fine-tuning for each sparsity value or each dataset; 2) Graph classification involves more complex message flow dynamics; 3) We expect occasional fluctuations due to the nature of the task, e.g., the ReLU function can suppress gradients for negative inputs. Despite these factors, REVELIO still achieves the best results in most cases, showing its effectiveness in message flow explanation.

PGExplainer and GraphMask aim to offer a global insight into the GNNs; however, their ability to handle complex datasets, such as Cora, falls short compared to other methods. Due to the model-specific nature of GNN-LRP, it is not applicable to GATs. REVELIO, on the other hand, can be applied to any GNNs with the fundamental message passing architecture. For synthetic datasets such as Tree-Cycles, removing edges may not necessarily decrease prediction probabilities, especially for edges outside the motif (the influence of which can be arbitrary), because the models focus solely on learning the distinctive motif pattern, whereas the remaining portions of the graph are generated randomly.

**Counterfactual explanation.** To generate counterfactual explanations, we adopt Eqs. (2) and (9) as our objective function with constraint. We use the original explanations provided by GradCAM, DeepLIFT, PGMExplainer, SubgraphX and GNN-LRP. The results of $Fidelity+$ are presented in Fig. 4.

In general, both FlowX and REVELIO achieve leading results, with FlowX performing better on several datasets. However, REVELIO exhibits an advantage on more complex datasets, particularly in node classification tasks. Given the minor performance gap between FlowX and REVELIO, and considering time complexity, REVELIO is the preferable choice for practical applications, where the slight gap in performance can be offset by introducing a few additional explanatory components.

Although factual and counterfactual explanations have opposite objectives, we observe that their results are not necessarily contrary. Removing key components from a factual explanation does not always significantly impact the prediction. For example, in Fig. 3, GradCAM achieves the best factual explanation result in BA-2motifs (GIN) and BBBP (GAT). In Fig. 4, on the other hand, the performance drops significantly from both settings even with the same results from edge importance. The importance scores generated by FlowX and REVELIO are specifically trained and adjusted according to the objective of counterfactual explanation; thus, these methods tend to achieve better results.

**AUC on synthetic datasets.** Following the related studies [50], we conduct experiments to compare AUC on synthetic datasets with both GCN and GIN. These datasets are generated with specific motifs, such as a house motif and a cycle motif. For example, BA-Shapes dataset is created by connecting house-like motifs to a random Barabási-Albert base graph, with the node labels indicating their positions within the motif. An effective GNN is expected to detect the house motif and accurately predict node labels within the motif. Consequently, a reliable explanation method should be able to identify the edges within the motif as the top explanatory edges. A higher AUC value suggests a closer alignment between the explana-

TABLE V

THE AVERAGE RUNNING TIME (IN SECONDS) OF EXPLANATION METHODS WITH GCNS AND GINS. THE RUNNING TIMES OF METHODS WITH AN ASTERISK (*) ARE EXPECTED TO BE LONGER IN COMPARABLE SITUATIONS. WE HIGHLIGHT THE OVERALL BEST RESULT IN **BOLD**, AND THE BEST RESULT WITHIN FLOW-BASED EXPLANATION METHODS IS UNDERLINED.

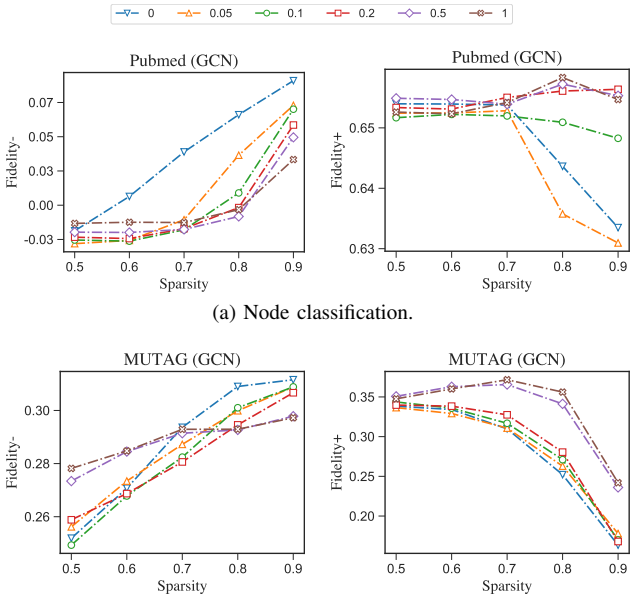
| Method | Cora | Citeseer | Pubmed | BA-Shapes | Tree-Cycles | MUTAG | BBBP | BA-2motifs |
|---|---|---|---|---|---|---|---|---|
| GradCAM | **0.18** | **0.18** | **0.18** | **0.18** | **0.18** | **0.14** | **0.17** | **0.14** |
| DeepLIFT | 0.26 | 0.28 | 0.27 | 0.26 | 0.25 | 0.22 | 0.22 | 0.22 |
| GNNExplainer | 87.41 | 79.76 | 93.55 | 83.93 | 77.77 | 80.55 | 78.51 | 81.85 |
| PGExplainer | 116.64 (0.13) | 149.88 (0.12) | 214.09 (0.14) | 74.07 (0.12) | 55.27 (0.12) | 166.32 (0.09) | 160.26 (0.08) | 159.95 (0.08) |
| GraphMask | 193.65 | 188.08 | 192.65 | 195.31 | 184.81 | 190.72 | 209.89 | 196.08 |
| PGMExplainer | 92.65 | 97.03 | 103.47 | 109.36 | 81.59 | 164.57 | 162.02 | 164.82 |
| SubgraphX* | N/A | N/A | N/A | N/A | 1,173.44 | 5,632.29 | 7,083.89 | 11,173.81 |
| GNN-LRP | 221.17 | 109.25 | 4,210.53 | 1,497.13 | 9.81 | 52.92 | 66.41 | 104.73 |
| FlowX* | 312.92 | 200.70 | 4,024.77 | 1,398.58 | 117.80 | 169.24 | 183.22 | 177.96 |
| REVELIO | 108.86 | 112.35 | 147.16 | 132.21 | 101.37 | 111.08 | 108.83 | 109.43 |

tion and human understanding. In our evaluation, we focus on instances associated with motifs and correct predictions. We then gather the importance scores each methods assigned to the edges, and calculate the AUC for each instance, using the motif edges as the ground truth. The average AUC results are presented in Table IV.

We acknowledge that AUC analysis is a valuable tool for assessing whether a GNN's predictions align with expectations (i.e., for plausibility evaluation). However, AUC is not an appropriate metric for faithfulness evaluation, as discussed in recent works [16, 53]. This metric works only on datasets with ground truth and assumes that GNNs base their predictions solely on this ground truth. AUC is reliable only under specific, rigid conditions as discussed above. A method achieving superior $Fidelity$ might exhibit lower AUC if the GNN itself does not perform well or make predictions as expected. In other words, the correlation between these metrics is not straightforward. This discrepancy comes from the difference between *faithfulness* and *plausibility*; explanations that are convincing to humans (or, plausible) are not necessarily descriptive of a model's true reasoning process (or, faithful) [16]. For instance, DeepLIFT achieves the highest AUC on BA-2motifs dataset with GCN but performs poorly in both factual and counterfactual explanations. Conversely, both FlowX and REVELIO excel in explaining the BA-Shapes dataset with GCN. Given their AUC values and the model's high prediction accuracy, we can infer that the GCN's predictions align with human understanding. When GNNs are trained on diverse instances and achieve high accuracy, using AUC as an evaluation metric becomes more convincing.

### C. Sensitivity Study

We select Pubmed and MUTAG datasets to demonstrate the effectiveness of sparsity constraint parameter $\alpha$ in Eqs. (8) and (9), with results presented in Fig. 5. Similar results can also be obtained using other datasets and GNN models as well. We vary the value of $\alpha$ from 0 to 1 and repeat the experiments in Figs. 3 and 4. A larger $\alpha$ value means a stronger constraint, leading to a smaller explanatory subgraph. Therefore, within a reasonable constraint range, we expect better results with




(a) Node classification.

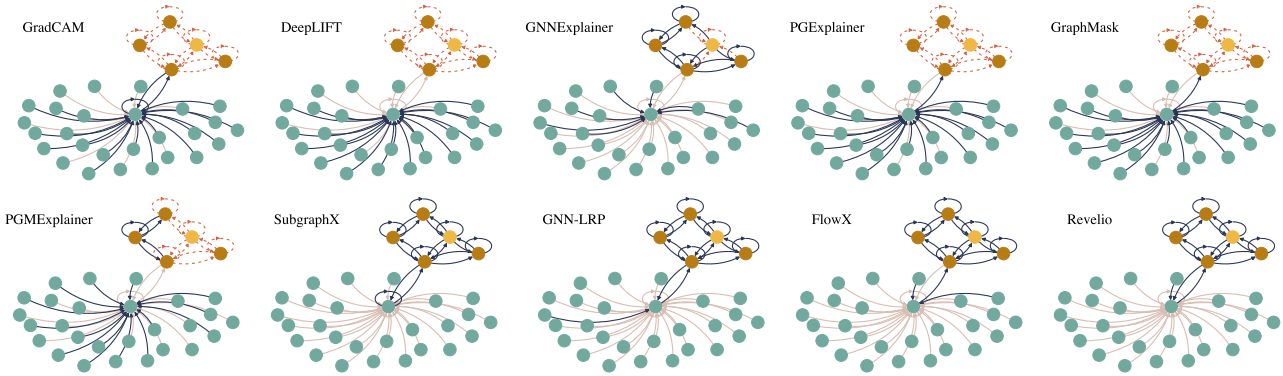
(b) Graph classification.

Fig. 5. Sensitivity study.

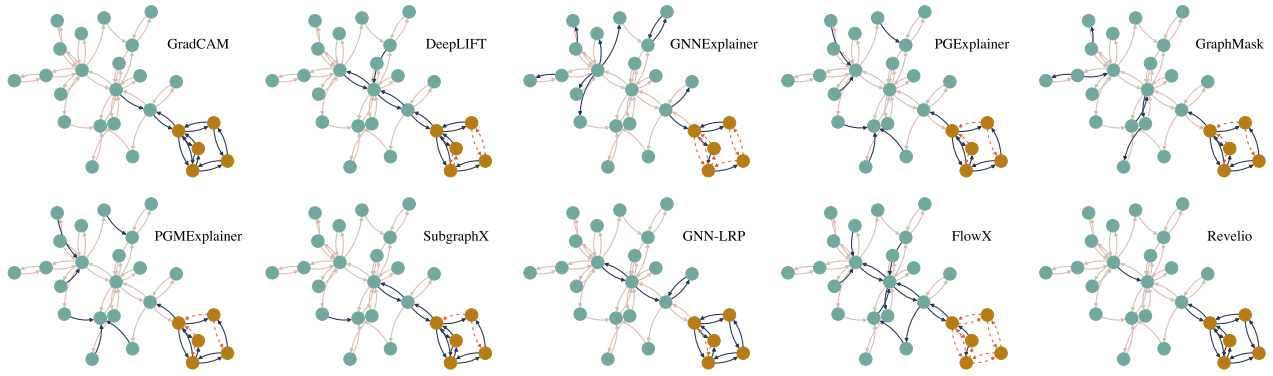
larger $\alpha$ values when dealing with higher sparsity (i.e., smaller graphs). As shown in Fig. 5, we can manually adjust the constraint value for the optimal result under different sparsity. A good constraint value can provide the optimal explanation results within a certain sparsity range, e.g., the factual explanation result in Fig. 5b. In the experiments from Figs. 3 and 4, we simply provide the overall best result by a single constraint value, which still outperforms other baseline methods.

### D. Efficiency Study

We report the running times on each dataset by different methods in Table V. Specifically, we present the training and inference times for PGExplainer in the format "training (inference)." Additionally, SubgraphX is only conducted on four datasets with only three sparsity values and the FlowX implementation operates in parallel with graph duplication. The running times of these methods are expected to be longer




(a) BA-Shapes with GCN.




(b) BA-2motifs with GIN.

Fig. 6. Result visualization. Nodes within the motifs are colored differently, as well as target nodes of node classification task. The explanatory edges are highlighted in dark. Dashed red edges represent the ground truth connections that methods fail to recognize.

in comparable situations. Although we share the running times of all methods for reference and analysis, it is important to note that REVELIO should be compared strictly to flow-based methods. Other methods, such as GradCAM—a white-box method that requests full access to GNN models and only provides node/edge explanations—are not directly comparable to REVELIO due to their substantially different scopes and capabilities.

Compared with GNNExplainer, REVELIO requires more running time due to the mask update for message flows. However, this extra computation enables REVELIO to produce more fine-grained explanations, which can be further translated into edge-level insights. We selected GNNExplainer as a baseline because it shares the same level of access to the target model. The additional overhead introduced by REVELIO is justified by the significant performance improvements observed in Figs. 3 and 4, along with the enhanced granularity of the information it provides. For denser graphs with more message flows, REVELIO shows substantial efficiency improvements compared to other flow-based methods. The relatively inefficient results stem from small or sparse graphs and shallow GNNs. The graphs we tested are relatively small, which results in $T > |\mathcal{F}|$ according to Table II, particularly in Tree-Cycles dataset. However, in real-world situations, often $T \ll |\mathcal{F}|$. In such graphs—Pubmed and BA-Shapes—REVELIO significantly reduces running time. When we tested datasets with hundreds of thousands of edges, other flow-based methods required substantial time and memory, often exceeding GPU capacity and expecting days to complete. In contrast, REVELIO completed these tasks in minutes or hours, making it the only option in practice.

### E. Result Visualization

**Graph structure.** Here, we use the experimental results conducted in Fig. 3. For each method, we display the target graphs in Fig. 6, where both explanatory and false negative edges are highlighted. For flexibility, we report additional explanatory edges identified by each method. The datasets used are well-established benchmarks in the GNN explainability domain, containing ground truth structures. We specifically selected these datasets because they not only allow GNNs to perform well but also enable explanation methods to achieve low $Fidelity-$ and high AUC scores. This ensures that the explanation results align with both the model's performance and human interpretability. Both BA-Shapes and BA-2motifs datasets contain a "house" motif, where the nodes are colored yellow in Fig. 6. The model's objective is to determine either the node's position within the motif or the existence of the "house" motif within the graph.

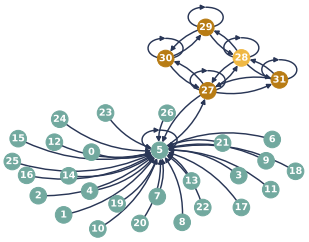


Fig. 7. The target graph from Fig. 6a. Top-10 message flows are shown in Table VI.

TABLE VI
TOP-10 MESSAGE FLOWS FROM FIG. 7 BY DIFFERENT METHODS.

(a) GNN-LRP.

| Message Flow | Score |
|---|---|
| $31 \to 31 \to 31 \to 28$ | 102.632 |
| $29 \to 30 \to 29 \to 28$ | 95.124 |
| $30 \to 30 \to 29 \to 28$ | 95.124 |
| $30 \to 29 \to 29 \to 28$ | 91.046 |
| $29 \to 29 \to 29 \to 28$ | 91.046 |
| $28 \to 31 \to 31 \to 28$ | 88.882 |
| $31 \to 31 \to 28 \to 28$ | 80.993 |
| $27 \to 31 \to 31 \to 28$ | 79.498 |
| $28 \to 29 \to 29 \to 28$ | 78.848 |
| $27 \to 30 \to 29 \to 28$ | 73.683 |

(b) FlowX.

| Message Flow | Score |
|---|---|
| $30 \to 29 \to 29 \to 28$ | 0.008 |
| $29 \to 29 \to 29 \to 28$ | 0.008 |
| $31 \to 31 \to 28 \to 28$ | 0.006 |
| $30 \to 29 \to 28 \to 28$ | 0.006 |
| $30 \to 30 \to 29 \to 28$ | 0.006 |
| $29 \to 30 \to 29 \to 28$ | 0.006 |
| $29 \to 29 \to 28 \to 28$ | 0.006 |
| $27 \to 30 \to 29 \to 28$ | 0.006 |
| $28 \to 28 \to 31 \to 28$ | 0.006 |
| $28 \to 29 \to 29 \to 28$ | 0.005 |

(c) REVELIO.

| Message Flow | Score |
|---|---|
| $27 \to 30 \to 27 \to 28$ | 0.681 |
| $30 \to 30 \to 27 \to 28$ | 0.650 |
| $29 \to 30 \to 27 \to 28$ | 0.641 |
| $31 \to 31 \to 27 \to 28$ | 0.640 |
| $29 \to 29 \to 28 \to 28$ | 0.631 |
| $28 \to 29 \to 29 \to 28$ | 0.628 |
| $30 \to 29 \to 29 \to 28$ | 0.626 |
| $30 \to 29 \to 28 \to 28$ | 0.626 |
| $29 \to 29 \to 29 \to 28$ | 0.620 |
| $27 \to 27 \to 31 \to 28$ | 0.606 |

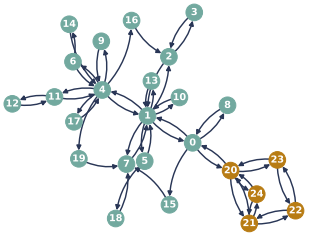


Fig. 8. The target graph from Fig. 6b. Top-10 message flows are shown in Table VII.

TABLE VII
TOP-10 MESSAGE FLOWS FROM FIG. 8 BY DIFFERENT METHODS.

(a) GNN-LRP.

| Message Flow | Score |
|---|---|
| $0 \to 8 \to 0 \to 20$ | 0.039 |
| $20 \to 24 \to 21 \to 20$ | 0.032 |
| $21 \to 22 \to 21 \to 20$ | 0.032 |
| $23 \to 22 \to 21 \to 20$ | 0.032 |
| $21 \to 24 \to 21 \to 20$ | 0.032 |
| $23 \to 22 \to 23 \to 20$ | 0.031 |
| $21 \to 22 \to 23 \to 20$ | 0.031 |
| $20 \to 23 \to 22 \to 23$ | 0.029 |
| $22 \to 23 \to 22 \to 23$ | 0.029 |
| $20 \to 23 \to 22 \to 21$ | 0.029 |

(b) FlowX.

| Message Flow | Score |
|---|---|
| $1 \to 5 \to 18 \to 7$ | 0.002 |
| $20 \to 0 \to 15 \to 7$ | 0.002 |
| $23 \to 20 \to 24 \to 20$ | 0.001 |
| $8 \to 0 \to 15 \to 7$ | 0.001 |
| $1 \to 0 \to 15 \to 7$ | 0.001 |
| $23 \to 20 \to 24 \to 21$ | 0.001 |
| $0 \to 15 \to 7 \to 1$ | 0.001 |
| $23 \to 20 \to 21 \to 22$ | 0.001 |
| $20 \to 21 \to 20 \to 23$ | 0.001 |
| $21 \to 20 \to 24 \to 21$ | 0.001 |

(c) REVELIO.

| Message Flow | Score |
|---|---|
| $20 \to 23 \to 20 \to 0$ | 0.927 |
| $22 \to 23 \to 20 \to 0$ | 0.927 |
| $20 \to 23 \to 20 \to 23$ | 0.927 |
| $22 \to 23 \to 20 \to 23$ | 0.927 |
| $22 \to 23 \to 20 \to 21$ | 0.926 |
| $20 \to 23 \to 20 \to 21$ | 0.926 |
| $22 \to 23 \to 20 \to 24$ | 0.926 |
| $20 \to 23 \to 20 \to 24$ | 0.926 |
| $23 \to 22 \to 21 \to 22$ | 0.886 |
| $21 \to 22 \to 21 \to 22$ | 0.886 |

By examining Fig. 6 in conjunction with the results from Fig. 3, we can conclude that the GNN models have effectively learned the motif, as these successful methods have attained the lowest $Fidelity-$ values. However, such visualization alone is not sufficient to evaluate an explanation method. This is because the ground truth does not necessarily exist in practice, models may not make predictions based on what we as humans think of as ground truth.

**Message flow.** We report the top-10 message flows by flow-based methods in Tables VI and VII, using the same instances from Fig. 6. We observe significant differences in the importance scores across these methods, which are due to their underlying mechanisms: GNN-LRP employs a Gradient $\times$ Input scheme (producing arbitrary results), FlowX utilizes Shapley values (smaller scores are expected), and REVELIO adopts a mask technique that inherently restricts scores to the range of $(-1, 1)$. In BA-Shapes dataset, all three methods successfully identify the edges within the motif. The results suggest that the prediction is primarily influenced by neighbors within two hops. The explanation is reasonable because two-hop information is enough for the model to determine the position of node 28. In BA-2motifs dataset, FlowX does not correctly identify important edges. GNN-LRP and REVELIO both have captured message flows passing through node 0, even though it is not within the motif. However, this may reflect the model's actual prediction process. The graphs are generated by connecting a random graph and a house motif with two directed edges, i.e., "$0 \leftrightarrow 20$" from Table VII. It is possible that the model learns a motif of six nodes, comprising a house motif connected to an additional node.

## VI. DISCUSSION & CONCLUSION

We have observed that REVELIO has more difficulty explaining graph classification tasks than node classification ones, likely because the message flows from graph classification tasks are much more intricate. Additionally, while our approach is the best performing flow-based approach, there is still room for improvement. For example, if one could identify the top-$k$ most important message flows before using REVELIO, and only propagate those top-$k$ flow masks, it would save a significant amount of memory, and improve running time. We leave this and other potential efficiency improvements as topics for future work.

In this work, we have presented REVELIO, a novel method to provide faithful explanations of message flows in GNNs. To our knowledge, it is the most scalable flow-based explanation method currently available. We have shown that REVELIO achieves as much as $28\times$ speedup over the state-of-the-art flow-based GNN explanation methods and can scale to graphs larger than those prior works can realistically process. Additionally, this speedup does not come with a tradeoff in faithfulness. Our extensive experiments with various GNN models on both synthetic and real-world datasets demonstrated the superiority of REVELIO through both quantitative and qualitative assessments.

## References


[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[2] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*, pages 2114–2124. PMLR, 2021.

[3] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[4] Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria M Oliver. Diffwire: Inductive graph rewiring via the lovász bound. In *Learning on Graphs Conference*, pages 15–1. PMLR, 2022.

[5] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems*, 34: 5644–5655, 2021.

[6] Julian Busch, Anton Kocheturov, Volker Tresp, and Thomas Seidl. Nf-gnn: network flow graph neural networks for malware detection and classification. In *Proceedings of the 33rd International Conference on Scientific and Statistical Database Management*, pages 121–132, 2021.

[7] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pages 7865–7885. PMLR, 2023.

[8] Federico Errica, Marco Podda, Davide Bacciu, Alessio Micheli, et al. A fair comparison of graph neural networks for graph classification. In *Proceedings of the Eighth International Conference on Learning Representations*, 2020.

[9] Qizhang Feng, Ninghao Liu, Fan Yang, Ruixiang Tang, Mengnan Du, and Xia Hu. DEGREE: Decomposition based explanation for graph neural networks. In *International Conference on Learning Representations*, 2022.

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

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[12] Shurui Gui, Hao Yuan, Jie Wang, Qicheng Lao, Kang Li, and Shuiwang Ji. Flowx: Towards explainable graph neural networks via message flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. Vision gnn: An image is worth graph of nodes. *Advances in Neural Information Processing Systems*, 35:8291–8303, 2022.

[15] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[16] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? *arXiv preprint arXiv:2004.03685*, 2020.

[17] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23, 2021.

[18] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A survey on explainability of graph neural networks. *arXiv preprint arXiv:2306.01958*, 2023.

[19] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[21] I Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. Problems with shapley-value-based explanations as feature importance measures. In *International conference on machine learning*, pages 5491–5500. PMLR, 2020.

[22] Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. In *International Conference on Machine Learning*, pages 6666–6679. PMLR, 2021.

[23] Wanyu Lin, Hao Lan, Hao Wang, and Baochun Li. Orphicx: A causality-inspired latent variable model for interpreting graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13729–13738, 2022.

[24] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, Keqiang Yan, Haoran Liu, Cong Fu, Bora M Oztekin, Xuan Zhang, and Shuiwang Ji. DIG: A turnkey library for diving into graph deep learning research. *Journal of Machine Learning Research*, 22(240):1–9, 2021.

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

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

[27] Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Towards faithful model explanation in nlp: A survey. *Computational Linguistics*, pages 1–67, 2024.

[28] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[29] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tu-dataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

[30] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2019.

[31] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10772–10781, 2019.

[32] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019.

[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[34] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. In *International Conference on Learning Representations*, 2020.

[35] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7581–7596, 2021.

[36] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[37] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[38] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM web conference 2022*, pages 1018–1027, 2022.

[39] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2021.

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

[41] Minh Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33:12225–12235, 2020.

[42] Xiang Wang, Yingxin Wu, An Zhang, Xiangnan He, and Tat-Seng Chua. Towards multi-grained explainability for graph neural networks. *Advances in Neural Information Processing Systems*, 34:18446–18458, 2021.

[43] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore, 2022.

[44] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.

[45] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[46] Ping Xiong, Thomas Schnake, Grégoire Montavon, Klaus-Robert Müller, and Shinichi Nakajima. Efficient computation of higher-order subgraph attribution via message passing. In *International Conference on Machine Learning*, pages 24478–24495. PMLR, 2022.

[47] Ping Xiong, Thomas Schnake, Michael Gastegger, Grégoire Montavon, Klaus-Robert Müller, and Shinichi Nakajima. Relevant walk search for explaining graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[49] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[50] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka

Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[51] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.

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

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

[54] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. Trustworthy graph neural networks: Aspects, methods, and trends. *Proceedings of the IEEE*, 112(2):97–139, 2024. doi: 10.1109/JPROC.2024.3369017.

[55] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[56] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. *Advances in Neural Information Processing Systems*, 35:19810–19823, 2022.

[57] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2020.

[58] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.