FlowX: Towards Explainable Graph Neural Networks via Message Flows

Shurui Gui, Hao Yuan, Jie Wang, Senior Member, IEEE, Qicheng Lao, Kang Li, and Shuiwang Ji, Fellow, IEEE

Abstract—We investigate the explainability of graph neural networks (GNNs) as a step toward elucidating their working mechanisms. While most current methods focus on explaining graph nodes, edges, or features, we argue that, as the inherent functional mechanism of GNNs, message flows are more natural for performing explainability. To this end, we propose a novel method here, known as FlowX, to explain GNNs by identifying important message flows. To quantify the importance of flows, we propose to follow the philosophy of Shapley values from cooperative game theory. To tackle the complexity of computing all coalitions’ marginal contributions, we propose a flow sampling scheme to compute Shapley value approximations as initial assessments of further training. We then propose an information-controlled learning algorithm to train flow scores toward diverse explanation targets: necessary or sufficient explanations. Experimental studies on both synthetic and real-world datasets demonstrate that our proposed FlowX and its variants lead to improved explainability of GNNs.

Index Terms—Deep learning, explainability, graph neural networks, message passing neural networks.

I. INTRODUCTION

W ith the advances of deep learning, graph neural networks (GNNs) are achieving promising performance on many graph tasks, including graph classification [1], [2], [3], node classification [4], [5], [6], and graph generation [7], [8]. Many research efforts have been made to develop advanced graph operations, such as graph message passing [4], [5], [9], graph pooling [10], [11], [12], and 3D graph operations [13], [14]. Deep graph models usually consist of many layers of these operations stacked on top of each other and interspersed with nonlinear functions. The resulting deep models are usually deep, highly nonlinear, and complex. While these complex systems allow for accurate modeling, their decision mechanisms are highly elusive and not human-intelligible. Given the increasing importance and demand for trustworthy and fair artificial intelligence, it is imperative to develop methods to open the black box and explain these highly complex deep models. Driven by these needs, significant efforts have been made to investigate the explainability of deep models on images and texts. These methods are developed from different perspectives, including studying the gradients of models [15], [16], [17], mapping hidden features to input space [18], [19], occluding different input features [20], [21], [22], and studying the meaning of hidden layers [23], [24], [25], etc. In contrast, the explainability of deep graph models is still less explored. Since graph data contain limited locality information but have important structural information, it is usually not natural to directly extend image or text based methods to graphs. Recently, several techniques have been proposed to explain GNNs, such as GNNExplainer [26], PGExplainer [27], RCExplainer [28], etc. These methods mainly focus on explaining graph nodes, edges, features, or subgraphs.

In this study, we posit that message flows serve as the intrinsic operational mechanism of Graph Neural Networks (GNNs), offering a more intuitive and natural lens through which to investigate their explainability. Specifically, traditional edge-based explanation approaches encounter limitations when required to account for continuous messages or multi-hop correlations, which cannot be adequately represented by individual, discrete edges, as illustrated in Fig. 1. In this challenge, we need a fine-grained explanation to build a new overpass to solve traffic jams, but edge explanations fail to provide such information. To address this obstacle, we introduce a suite of message flow-based explanation techniques, with the primary methodology designated as FlowX. FlowX attributes GNN predictions to message flows and studies the importance of different message flows. We first develop a systematic framework that incorporates flow-based graph modeling as a foundational layer for message flows. With our framework, FlowX begins with quantifying the importance of flows by following the principle of Shapley value [29], conceptualizing flows as collaborative players that collectively contribute to the model predictions. Given that message flows cannot be straightforwardly quantified for marginal contributions, we propose a fair flow sampling scheme as the initial assessment of different flows. Building upon this initial assessment, we propose a coalition-level learning-based algorithm with information sparsity controls. Our experimental
message aggregation procedures can be mathematically written as a two-step computation as
\[
\text{Aggregate: } S^t = X^{t-1} \hat{A}^t, \tag{1}
\]
\[
\text{Combine: } X^t = M^t(S^t), \tag{2}
\]
where \(X^t \in \mathbb{R}^{d_t \times n}\) denotes the node feature matrix computed by the \(t\)-th GNN layer and \(X^0 = X\). Here \(M^t(\cdot)\) denotes the node feature transformation function at layer \(t\) and \(\hat{A}^t\) is the connectivity matrix at layer \(t\). Note that we name the elements in \(\hat{A}^t\) as layer edges and \(\hat{a}^{t}_{ij}\) indicates the layer edge connecting node \(v_j\) and \(v_i\) in layer \(t\). For example, in GCNs, the transformations are defined as \(\sigma(W^t S^t)\) and \(\hat{A}^t = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}\) where \(W^t \in \mathbb{R}^{d_t \times d_t - 1}\) is a trainable weight matrix, \(\sigma(\cdot)\) denotes the activation function, \(I\) is an identity matrix to add self-loops to the adjacency matrix, and \(D\) denotes the diagonal node degree matrix. We can stack \(T\) GNN layers on top of each other to form a \(T\)-layer network, and the network function can be expressed as
\[
f(\mathcal{G}) = g(M^T (M^{T-1} \cdots M^1 (X^0 A^1) \cdots) \hat{A}^{T-1}) \hat{A}^T).
\]
When \(f(\mathcal{G})\) is a graph classification model, \(g(\cdot)\) generally consists of a readout function, such as global mean pooling, and a multi-layer perceptron (MLP) graph classifier. Meanwhile, when \(f(\mathcal{G})\) is a node classification model, \(g(\cdot)\) represents an MLP node classifier.

\[\text{evaluations encompass a range of tests: necessary and sufficient explanations [30] comparisons, target edge/flow retrieval ability test, mutual information training comparisons between flow-based and edge-based methods, and visualizations. Experimental results on necessary and sufficient explanations [30] show that our proposed FlowX outperforms existing methods significantly and consistently from multiple perspectives. Additional experiments validate the advantages of flow-based methods compared with other approaches. Both quantitative and qualitative studies demonstrate that the proposed FlowX and its variants lead to novel and improved explanations of GNNs.}\]

II. PRELIMINARIES & RELATED WORK

A. Graph Neural Networks

With the advances of deep learning, several graph neural network approaches have been proposed to solve graph tasks, including graph convolutional networks (GCNs) [4], graph attention networks (GATs) [5], and graph isomorphism networks (GINs) [1], etc. They generally follow a message-passing framework to learn graph node features. Specifically, the new features of a target node are learned by aggregating message flows passed from its neighboring nodes. Without loss of generality, we consider the input graph as a directed graph with \(n\) nodes and \(m\) edges. The graph is denoted as \(\mathcal{G} = (V, E)\), where \(V = \{v_1, \ldots, v_n\}\) denotes nodes, and \(E = \{e_{ij}\}\) represents edges in which \(e_{ij}\) is the directed edge \(v_i \rightarrow v_j\). Then it can be represented by a feature matrix \(X \in \mathbb{R}^{d \times n}\) and an adjacency matrix \(A \in \mathbb{R}^{n \times n}\). Each node \(v_i\) is associated with a \(d\)-dimensional feature vector \(x_i\) corresponding to the \(i\)-th column of \(X\). The element \(x_{ij}\) in \(X\) represents the weight of \(e_{ij}\), and \(x_{ij} = 0\) indicates \(e_{ij}\) does not exist. For the \(t\)-th layer in GNNs, the

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Monte Carlo tree search algorithm to explore subgraphs and employs Shapley values to measure the importance. Fourth, XGNN [35] focuses on model-level explanations by generating graph patterns that can maximize a certain model prediction. In addition, Refine [37] targets explaining from both class and instance levels. RGExplainer [38] focuses on generative reinforcement learning explanations. V Gib [39] employs the recent information bottleneck technique but it is intrinsically an interpretable method, instead of an explainer given fixed models.

Furthermore, as the emergence of causality works [43], [44], [45], [46], [47], [48], causal-based explanations including counterfactual explanations also become a valuable avenue [49]. Among these methods, [50] and CLEAR [51] focus on counterfactual explanations where CLEAR can generate new counterfactual edges that do not exist in the original graph. RC-Explainer [28] combines both a causality principle and reinforcement learning to search for edge explanations and serves as a new interesting and comprehensive causal baseline. While these methods explain GNNs from different views, they cannot provide explanations to solve the unspecified challenge shown in Fig. 1.

In prior work, GNN-LRP [52] employs LRP with respect to graph adjacency matrices, resulting in explanations referred to as relevant walks. However, these gradient-based methods follow the Gradient × Input scheme that fails the model parameter randomization test and might not be sensitive to model parameters [53]. Since relevant walks are highly inherent to this gradient-based method, their explanation potentials are largely constrained. This limitation prompts us to introduce a higher-level abstraction, termed message flows, which broadens the explanation scope from subgraphs, nodes, and edges to flows. In Section III-B, we compare the explanation modeling through the lens of causality. Within our flow-based graph modeling framework, the concept of a relevant walk serves merely as a specific instance of our more expansive flow explanations. Furthermore, based on our flow framework, GNN-LRP, along with the proposed FlowX, FlowXrev, FlowXsup, and FlowMask offer a comprehensive suite of flow-based explanations that includes gradient-based, Shapley-based, surrogate, and hybrid methods.

Differences with Related Methods. While Shapley value has been studied under a node-level method GraphSVX [36] and a subgraph-level method SubgraphX [33], it has never been applied under a flow level. Applying Shapley value to flows introduces different challenges and insights, e.g., flows cannot be extracted directly to serve as individual players in marginal contribution calculations (Section III-C1). Moreover, there are two distinctions between the sampling processes of SubgraphX and FlowX. First, the sampling process in SubgraphX is a tree search process with a pruning strategy, while the sampling process in this work is a fair permutation-based sampling process as described in Section III-C2. Second, because of the pruning strategy, the sampling process in SubgraphX is sequential. In contrast, our flow sampling process is nearly fully paralleled. Specifically, with enough GPU memory, we only require one pass model forward calculation for each graph. Furthermore, in addition to the sampling process, we propose an innovative Shapley coalition-level training tailoring the pure Shapley value to diverse explanation targets: necessary or sufficient explanations [30], which makes our method unique.

III. THE PROPOSED FLOWX

While existing methods mainly focus on explaining GNNs with graph nodes, edges, or subgraphs, we propose to study the explainability of GNNs from the view of message flows. We argue that message flows are the fundamental building blocks of GNNs and it is natural to study their contributions towards GNN predictions. With our message flow framework, we propose a novel method, known as FlowX, to investigate the importance of different message flows. Specifically, we follow the philosophy of Shapley values [29] from game theory and propose a marginal contribution approximation scheme for them. In addition, a learning-based algorithm is proposed to improve the explainability of message flows.

A. A Message Flow View of GNNs

We consider a deep graph model with $T$ GNN layers. Each GNN layer aggregates 1-hop neighboring information to learn new node embeddings. Hence, for any node, the outgoing messages are transmitted within its $T$-hop neighbors. Then the outputs of GNNs can be regarded as a function of such transmitted $T$-step messages, which are named as message flows in this work. Formally, we introduce the concept of message flows and message carriers as follows:

1) Definition 1: Message Carrier: We use the connectivity matrix to represent the carriers for message flows. Given the connectivity matrix $\tilde{A}^t$ at layer $t$, the layer edge $a_{ij}^t$ in this matrix represents the message carrier with which the message passes from node $v_i$ to $v_j$ at layer $t$.

Note that we use the superscript $t$ to distinguish the message carriers in different layers since their corresponding message flows are different. Then the set of all message carriers, i.e., all layer edges, is defined as $\mathcal{A} = \{\vdots, \tilde{a}_{uv}^1, \tilde{a}_{uv}^2, \ldots, \tilde{a}_{uv}^T, \ldots\}$ and $|\mathcal{A}| = |E| \times T$.

2) Definition 2: Message Flow: In a $T$-layer GNN model, we use $F_{ijk\ldots}$ to denote the message flow that starts from node $v_i$ in the input layer, and sequentially passes the message to node $v_j$, $v_k$, $v_l$, $v_r$ until to node $v_m$ in the final layer $T$. The corresponding message carriers can be represented as $\{\tilde{a}_{ij}^1, \tilde{a}_{jk}^2, \ldots, \tilde{a}_{lm}^T\}$.

Notation of flows: In a $T$-layer GNN model, all message flows start from the input layer and end with the final layer so that their lengths are equal to $T$. For ease of notation, we introduce the wildcard “*” to represent any valid node sequence and $\mathcal{F}$ to denote the message flow set. For example, we can use $\mathcal{F}_{ij\ldots}, \mathcal{F}_{\ell m}, \text{and } \mathcal{F}_{ij\ldots}$ to denote the message flow sets that share the same message carrier $a_{ij}^1, a_{\ell m}^T$, and both of them, respectively. In addition, we employ another wildcard “?” to denote any single node and $\{t\}$ to represent any valid node sequence with $t$ nodes. For example, $\mathcal{F}_{\ell[3]}$ means the set of valid 2-step message flow with 3 nodes. Note that the following property of message flow

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
sets also holds:
\[ \mathcal{F}_{ij+\ell|m} = \mathcal{F}_{ij} \cap \mathcal{F}_{\ell|m}. \]  

Intuitively, Property 3 indicates that given two sets of flows where the first set includes all flows starting from \( v_i \rightarrow v_j \), and the second set contains all flows ending with \( v_\ell \rightarrow v_m \), the intersection flow set of these two flow sets consists of all flows starting from \( v_i \rightarrow v_j \) and ending with \( v_\ell \rightarrow v_m \).

The final embedding of node \( v_m \) is determined by all incoming message flows to node \( v_m \), which can be denoted as \( \mathcal{F}_{\ell|m} \). Since the output of the GNN model is obtained based on the final node embeddings, it is reasonable to treat the GNN output as the combination of different message flows. Hence, it is natural to explain GNN models by studying the importance of different message flows towards GNN predictions.

### B. Flow-Based Graph Modeling

To intuitively demonstrate the correlations among node-based, edge-based, subgraph-based, and flow-based explanations, we propose to model graph generation processes through the lens of causality [43], [54]. Generally, the commonly used graphon model [55], [56] \( W \) is a measurable function modeling the relations/edges between nodes, i.e., it maps two node hidden variables \( U_i \) and \( U_j \) to one adjacency matrix element \( A_{ij} \), as shown in Fig. 2. When dealing with identical nodes, the graphon model \( W \) produces the observable node signals/features \( X_i \).

Acting as a confounder, \( W \) creates explainable correlations between \( X_i/A_{ij} \) and \( Y \), enabling us to use observed variables \( X_i \) and \( A_{ij} \) to explain the model prediction \( Y \). Depending on the variables employed, explanations can fall into categories such as node-based/node-feature-based (\( X_i \)), edge-based (\( A_{ij} \)), or subgraph-based (hybrid).

However, the inherent limitation of the graphon function \( W \) lies in its capacity to model only pairwise relations between nodes, i.e., edges. To explain messages that transmit across multiple nodes, we extend the \( W \) to model the relations of multiple nodes in sequences. For a \( T \)-hop message passing network, \( W \) maps \( T + 1 \) input node hidden variables to one message flow. Specifically, as shown in Fig. 2, the flow model \( W \) associates node sequences, including \( U_i \) and \( U_j \), to the set of flows \( \mathcal{F}_{ij} \). Then, the relations between two nodes \( A_{ij} \) can be modeled by the combinations of these message flows \( \mathcal{F}_{ij} \). Note that, we abuse the notation \( W \) only in this subsection to maintain consistency with the previous work [56].

### C. Marginal Contribution Calculation by Flow Samplings

While explaining GNNs with message flows seems to be promising, it is still crucial to properly measure the importance of those message flows. Our FlowX proposes to follow the philosophy of Shapley value [29] to use the marginal contributions in different flow sets as the initial assessments of flow importance. Shapley value is a solution concept in cooperative game theory and is used to fairly assign the game gain to different players. When considering marginal contributions in GNN explanation tasks, we treat the message flows as different players and the GNN prediction score as the total game gain.

Formally, given the trained GNN model \( f(\cdot) \) and the input graph \( G \), we use \( \mathcal{F} \) to denote the set of all valid flows, i.e., all players in the game. Then given any flow \( \mathcal{F}^k \), we mathematically define the contribution as

\[
\phi(\mathcal{F}^k) = \sum_{P \subseteq \mathcal{F}_X,|f^k|} \mathcal{W}(|P|)(f(P \cup \{\mathcal{F}^k\}) - f(P)),
\]

where \( \phi(\cdot) \) denotes the flow score of \( \mathcal{F}^k \); \( P \) denotes the possible coalition group/set of players; \(|\cdot|\) is the set size; \( \mathcal{W}(\cdot) \) is a weight function assigned according to the size of the group \( P \). Here \( f(P \cup \{\mathcal{F}^k\}) - f(P) \) is the marginal contribution of flow \( \mathcal{F}^k \) for a particular coalition group, which can be computed by the prediction difference between combining \( \mathcal{F}^k \) with the coalition group \( P \) and only using \( P \). Note that (4) is equivalent to the classic Shapley value when \( \mathcal{W}(|P|) = \left[ |P|/|\mathcal{F}_X| \right]^{-|P|-1} \).

To compute the flow score \( \phi(\mathcal{F}^k) \), we need to enumerate all possible coalition groups and considers different interactions among players. However, calculating flow Shapley values introduces two challenges.

1) **Challenge 1: Indirect Intervention Targets:** The first challenge is message flows cannot be removed/intervened independently from both the input level and the model level, which is also implied by Fig. 2. Therefore, it is impossible to directly compute their importance scores using (4). To overcome this challenge, we consider the finest component we can intervene (remove) to calculate marginal contributions, i.e., layer edges. For example, since the flow set \( \mathcal{F}_{ij} \) depends on layer edge \( A_{ij} \), when we remove \( A_{ij} \), all flows in flow set \( \mathcal{F}_{ij} \) will be simultaneously removed from the model. Fig. 3 illustrates another example, where removing layer edge \( A_{ij} \) is equivalent to removing flows \( \mathcal{F}_{123} \). This scenario, since multiple flows are removed at the same time, the obtained marginal contributions should be distributed to these flows, e.g., even distributions.

2) **Challenge 2: Fair Flow Sampling:** Since enumerating all possible coalition groups is time-consuming especially when the input graph is large-scale and the GNN model is deep, a sampling process is required. However, applying simple Monte Carlo (MC) sampling [57] cannot guarantee a fair sampling for each flow. Therefore, as shown in Algorithm 1, we propose an innovative flow sampling scheme that divides the sampling process into two levels and ensures each flow can be and only be
and

Formally, let each flow $A = \{A_1, A_2, \ldots, A_L\}$ denote the set of all layer edges in the GNN model, and $s^k$ denote the importance scores of each flow. In addition, $\hat{A}$ denotes the set of removed layer edges and $\hat{F}$ is the set of removed message flows.

1: Given a trained GNN model $f()$ and an input graph, the set of all layer edges is represented by $\mathcal{A}$. For each message flow $F^k$, two $|\mathcal{A}|$-dimensional vectors $s^{\ell_m}$ and $c^{\ell_m}$ denote its importance scores and removing index counts respectively.

In addition, $\mathcal{A}$ denotes the set of removed layer edges and $\hat{F}$ is the set of removed message flows.

2: For each flow $F^k$, initialize $s^{\ell_m}$ and $c^{\ell_m}$ as zero vectors.

3: for iteration $i$ from 1 to flow sampling iteration $M$ do

4: Initialize the removed sets as empty that $\hat{A} = \emptyset$ and $\hat{F} = \emptyset$.

5: Randomly shuffle and permute the layer edge set $\mathcal{A}$, denoted as $\mathcal{A}^\pi$.

6: for $j$ from 1 to $|\mathcal{A}|$ do

7: Select the $j$-th layer edge in $\mathcal{A}^\pi$, denoted as $\hat{a}^{\ell_m}_t$.

8: Block the layer edge $\hat{a}^{\ell_m}_t$ in GNN model $f()$, then the removed flows are $\hat{F}^j = F^j - \{A_{\{t-1\}^{\ell_m} \cap (T-t)} \setminus (F^j \setminus F^j_{(t-1)\ell_m} \setminus (T-t))\}$.

9: Compute the prediction difference that $s_j = f(\mathcal{A} \setminus \hat{A}) - f(\mathcal{A} \setminus (\hat{A} \cup \hat{a}^{\ell_m}_t*))$.

10: Update $\hat{A} = \hat{A} \cup \hat{a}^{\ell_m}_t$ and $\hat{F} = \hat{F} \cup \hat{F}^j_{(t-1)\ell_m} \setminus (T-t)$.

11: Compute averaged score that $\hat{s}_j = s_j / |\hat{F}|$.

12: for each flow $F^k$ in $\hat{F}$ do

13: Update $s^{\ell_m}_j = s^{\ell_m}_j + \hat{s}_j$.

14: Update $c^{\ell_m}_j = c^{\ell_m}_j + 1$.

15: end for

16: end for

17: end for

18: For each message flow $F^k$, compute the marginal contribution vector $s^{\ell_m} = s^{\ell_m} / c^{\ell_m}$.

Fig. 3. Illustration of our initial assessments via sampling marginal contributions. For each sampling iteration, we iteratively remove one layer edge until all layer edges are removed. In this example, the removed layer edges are shown in bold and purple lines while the corresponding message flows are shown in arrow lines. In the first step, we remove the layer edge between $v_3$ and $v_4$ from the first GNN layer and compute the marginal contribution. Then four message flows (red) are removed and the contribution scores are averaged and assigned to these four message flows. In the second step, we remove the layer edge $\hat{a}^{\ell_m}_{24}$ and distribute the marginal contribution to the corresponding three flows (green).

Algorithm 1: Initial Approximations of Flow Importance Scores.

---

sampled one time per iteration, leading to a fair flow sampling process.

3) Flow Sampling: Formally, let $M$ denote the total number of sampling iterations. In the $i$-th sampling iteration, we generate a random permutation vector $\pi^i$ of $|\mathcal{A}| = |E| \times T$ elements. Then all layer edges in $\mathcal{A}$ will be permuted based on $\pi^i$, denoted as $\mathcal{A}^\pi$ that $\mathcal{A}^\pi_{\{t\}^{\ell_m}} = \mathcal{A}_{\{t\}^{\ell_m}}$. Subsequently, in each iteration, there are $|\mathcal{A}^\pi|$ steps. In each step, we remove one layer edge from $\mathcal{A}^\pi$ in order, and compute the marginal contributions of corresponding removed flows as shown in Fig. 3. We use $\hat{A}$ to denote the set of layer edges that have been removed in the current iteration and $\hat{F}$ to denote the set of corresponding removed message flows. These two sets are initialized as $\hat{A} = \emptyset$ and $\hat{F} = \emptyset$ in the beginning of each sampling iteration. Specifically, at step $j$ in the current iteration, the $j$-th element of $\mathcal{A}^\pi$ is removed. Assuming the removed layer edge is $\hat{a}^{\ell_m}_t$, the computation operations can be mathematically written as

$$s_j = f(\mathcal{A} \setminus \hat{A}) - f(\mathcal{A} \setminus (\hat{A} \cup \hat{a}^{\ell_m}_t)),$$

$$\hat{A} = \hat{A} \cup \hat{a}^{\ell_m}_t,$$

$$\hat{F}^j = F^j_{(t-1)\ell_m} \setminus (\hat{F} \setminus F^j_{(t-1)\ell_m} \setminus (T-t)),$$

$$\hat{F} = \hat{F} \cup F^j_{(t-1)\ell_m} \setminus (T-t),$$

where $\hat{F}^j$ denotes the removed message flows at step $j$ by removing $\hat{a}^{\ell_m}_t$; similar to the set difference operation, "\" denotes the removal of layer edges or flows. Note that $\hat{F}^j$ is not equivalent to $\hat{F}$.
to $\mathcal{F}_{\{t-1\} \cup \mathcal{m} \cup \{T-t\}}$ since parts of the flows in $\mathcal{F}_{\{t-1\} \cup \mathcal{m} \cup \{T-t\}}$ may have been already removed in the previous steps. Given these removed flows $\bar{\mathcal{F}}$, the score is averaged by the number of flows, i.e., $\bar{s}_j = s_j / |\bar{\mathcal{F}}|$, and we distribute $\bar{s}_j$ to each flow in $\bar{\mathcal{F}}$. By repeating such operations until all layer edges are removed/calculated, each flow ends up obtaining one distributed score. Note that the order information in $\mathcal{A}^c$ is important since in the earlier steps, the removed flows $\bar{\mathcal{F}}$ are interacting with a larger coalition group $\mathcal{A} \setminus (\hat{\mathcal{A}} \cup \hat{\mathcal{A}}_{T_m})$, while in the later steps the coalition groups are generally smaller. Altogether, the steps of our marginal contribution sampling are shown in Algorithm 1.

After $M$ flow sampling iterations, our method explores $M$ permutations of layer edge set and each flow is sampled to obtain $M$ marginal contributions. For each flow $\mathcal{F}_k$, we use a $|\mathcal{A}|$-dimensional vector to store its marginal contributions, denoted as $s^{F_k}$, where $s^{F_k}_j$ is the importance score obtained when $\mathcal{F}_k$ is removed at step $j$. To obtain $s^{F_k}$, valid scores at the same step and different sampling iterations are averaged, where a score of a flow is valid at step $j$ if the flow is removed at this step. The concrete operation is shown in line 18 of the algorithm 1, where “)” denotes an element-wise division. To avoid zero-divisions, we add a small value $10^{-10}$ to every element in $s^{F_k}$. The importance score will be a flow Shapley value approximation if we compute the summation of the elements in $s^{F_k}$.

### D. Trainable FlowX for Diverse Explanations

While Shapley values are promising, there are different types of explanations we can achieve, e.g., counterfactual explanations [49], or sufficient explanations that maintain model performances. Making an explainer trainable can not only refine importance scores but also enable the explainer to serve different purposes, e.g., searching for necessary components or sufficient components. These two aspects are first proposed by [58] and further characterized by [30]. Specifically, we relate selecting necessary components as a process of finding counterfactual subgraphs that can produce the most significant different performance. In this process, the complementary subgraphs of the counterfactual subgraphs are necessary components of the predictions. On the other hand, selecting sufficient components is the process of finding subgraphs that can maintain similar or even better model performances.

Different from optimizing classification tasks, training such an explainer is intrinsically a harder task as described in Section III-E. Specifically, with our obtained score vectors $s^{F_k}$, we propose to consider these values as initial assessments of flow importance and learn associated importance scores by injecting randomness. In this training process, instead of retraining individual flow scores, we only learn weights of coalitions $\mathbf{w}$ that are used to sum the elements in $s^{F_k}$ to obtain the final importance score for each flow. Note that learning $\mathbf{w}$ can be considered as learning $W(P)$ in (4).

1) From Flow Scores to Layer Edge Scores: Formally, given the input graph $\mathcal{G}$ and GNN model $f(\cdot)$, we first obtain marginal contribution vector $s^{F_k}$ for each message flow. Then we apply a dot product between the only trainable weight vector $\mathbf{w}$ and the flow score vector $s^{F_k}$ to attain the final flow score $s^{\hat{F}_k}$. Since layer edges are the finest components we can operate in an MPNN, we convert these flow scores to layer edge scores, where summations are applied to the flows that share the same layer edges as follows. Note that the weight vector $\mathbf{w}$ is initialized by value 0.5 and is shared across all flows in the same graph, which explains our training intuition, i.e., coalitions in different steps are not equally important. These processes can be written as

$$s^{\hat{F}_k} = s^{F_k} \cdot \mathbf{w}$$

$$s^{\hat{A}_{uv}} = \sum_{\mathcal{F} \in \mathcal{F}} s^{\hat{F}_k}$$

where $s$ denotes the score of layer edges or flows; $\mathcal{F} = \mathcal{F}_{\{t-1\} \cup \mathcal{m} \cup \{T-t\}}$ denotes the flows that share layer edge $\hat{A}_{uv}$. Then based on $s^{\hat{A}_{uv}}$, we can obtain a mask vector indicating the importance of different layer edges, denoted as $\mathbf{m}$:

$$\mathbf{m} = g(s^A),$$

where $s^A$ denotes all layer edges’ importance scores. $g(\cdot)$ is defined as an operation that sequentially includes an InstanceNorm [59] and a Gumbel-Sigmoid [60] to inject randomness and normalize the mask. We provide more analyses in Section II-I-E.

2) Necessary Explanations: By applying the mask to layer edges, important layer edges are restricted and the model prediction becomes

$$\hat{y} = f(\text{Combine}(\mathcal{G}, 1 - \mathbf{m})),$$

where $\hat{y}$ is the prediction vector and $\text{Combine}(\mathcal{G}, 1 - \mathbf{m})$ denotes a modified graph where layer edges are masked out from $\mathcal{G}$ based on the values of $\mathbf{m}$. Intuitively, if important layer edges are restricted, then the prediction should change significantly. Hence, $\hat{y}$ is encouraged to be different from the original prediction by learning proper weights $\mathbf{w}$. This encouragement may be also considered as searching a counterfactual graph [49], while the distinction resides at that we focus on explanations in original graphs, but counterfactual explanations generate new graphs/edges. Specifically, we employ the log-likelihood loss as

$$\mathcal{L}(\hat{y}, y) = \sum_{c=1}^{|\mathcal{Y}|} 1\{y = c\} \log \hat{y}_c,$$

where $y \in \mathcal{Y}$ is the label representing the predicted class of the original graph $\mathcal{G}$; $|\mathcal{Y}|$ is the number of classes; $1\{\cdot\}$ denotes the indicator function, and $\hat{y}_c$ is the predicted probability for class $c$.

3) Sufficient Explanations: Considering sufficient explanations, we retain the layer edges denoted by $\mathbf{m}$ and train to maintain model performances. Therefore, the model prediction and the loss in (12) and 13 are replaced by

$$\hat{y} = f(\text{Combine}(\mathcal{G}, \mathbf{m})).$$
and
\[
\mathcal{L}(\hat{y}, y) = -\sum_{c=1}^{y} \mathbf{1}\{y = c\} \log \hat{y}_c. \tag{15}
\]

Note that (13) can be interpreted as a counterfactual training while training negative log-likelihood (15) can be interpreted as maximizing a lower bound of mutual information \(I(\hat{Y}; \text{Combine}(G, m))\) according to (6) of [61]. After training, the importance scores of necessary/sufficient explanations for different flows can be obtained via (9).

To distinguish the notations of different training strategies, we denote the necessary-oriented FlowX as FlowX\(_{nec}\), the sufficient-oriented FlowX as FlowX\(_{suf}\), and the pure flow sampling as FlowX. In addition, we introduce a training-based flow explainer variant in Section IV-A4, denoted as FlowMask.

### E. Training Dilemma & Marginal Distribution Control

Training importance scores is essential for a ranking/sorting process that conceptually requires pair-wise importance comparisons in simple \(O(n^2)\) sorting algorithms. This problem can be considered as a regression task without direct ground-truth values, making this task more challenging than both the classification and regression tasks with explicit ground-truth. Empirically, we observe that a naive approach to this problem often results in deteriorated performance compared to the initial results obtained through Algorithm 1. Therefore, we strive to reformat this problem into a binary classification task with the help of the Sparsity requirement Section II-B3. This Sparsity requirement introduces a percentage threshold, \(\gamma\), which effectively designates the top \(1 - \gamma\) layer edges as important (label 1) while relegating the remainder to the category of unimportant (label 0).

Given this reformatted binary classification, there are two challenges remaining, i.e., discretizing \(m\) and the enforcement of the selection ratio \(1 - \gamma\). To be specific, we tackle the first challenge by adopting a Bernoulli sampling process that incorporates stochasticity, thereby facilitating escape from local minima. The sampling process \(g(\cdot)\), as outlined in (11), employs an InstanceNorm and a reparameterization trick Gumbel-Sigmoid [60] to make the process differentiable, where each value after the InstanceNorm layer are considered as the log-probability in a Bernoulli distribution. For the second challenge, we propose to combine the Sparsity and the marginal distribution control [61] to set soft thresholds. Essentially, considering every element in \(m\) as a random variable, the soft \(1 - \gamma\) threshold is interpreted as a prior that every element has a \(1 - \gamma\) probability of being labeled as important (1), otherwise, unimportant (0). Statistically, when the length of \(m\) reaches infinity, the ratio of selected elements asymptotically approaches \(1 - \gamma\). Therefore, this marginal distribution control is a regularization that applies a penalty on the difference (KL divergence) between \(m\) and a vector of random variables with probability \(1 - \gamma\) to be 1.

Formally, the regularizer \(L_\gamma\) can be written as
\[
L_\gamma = \sum_{m \in m} m \log \frac{m}{1 - \gamma} + (1 - m) \log \frac{1 - m}{\gamma}. \tag{16}
\]

Therefore, our final optimization objective is
\[
w = \arg \min_w \mathcal{L}(y, y) + L_\gamma. \tag{17}
\]

After training, the flow scores are obtained by applying (9).

### F. Complexity Analysis

1) **Message Flows:** Considering the total number of message flows \(|\mathcal{F}|\), its loose upper bound is \(|\mathcal{F}| = O(|E|^T)\). With the consideration of connectivity, given the largest outgoing degree of nodes in the graph \(d_L\), the tighter upper bound of the number of message flows is \(|\mathcal{F}| = O(|E|(d_L)^{T-1})\).

2) **Marginal Contribution Sampling:** It is noticeable that the deep model’s forward operations are the most time-consuming operations, denoted as \(O(T_f)\). Without any parallel consideration, the time complexity of marginal contribution sampling is \(T_{mcs} = O(M|A|T_f)\). When we consider parallel implementations, the most simple improvement is to use the GPU acceleration to move lines 9–15 out of the two loops and to make line 8 executed in \(O(1)\), which leads to the time complexity \(T_{mcs} = O(M|A|S_f)\). The space complexity should be considered because there is a big flow score matrix \(S_{gpu} = O(M|A||\mathcal{F}|)\). As for the graphic memory requirement, this parallel implementation consumes \(S_{gpu} = O(M|A|S_f)\), where \(S_f = O(|V| + |E|)dT + S_p\) is the deep model memory complexity; here, \(|V| = n\) denotes the number of nodes; \(S_p\) is the model’s parameter memory complexity. In our implementation, there is a trade-off that we only parallely calculate the model outside the innermost loop, so that our time complexity is \(T_{mcs} = O(M(|A| + T_f))\) and the space complexity of GPUs is \(S_{gpu} = O(|A|S_f)\). Generally, we set \(M = 50\) which is enough for stable explanations.

### IV. EXPERIMENTAL STUDIES

We conduct experiments to validate the effectiveness of our explainer FlowX. Specifically, we target answering the following research questions. **RQ1:** How do FlowX and FlowX\(_{nec}\) perform in terms of explanations of necessity? **RQ2:** How do FlowX and FlowX\(_{suf}\) perform in terms of explanations of sufficiency? **RQ3:** What are the benefits brought by flow-based methods?

#### A. Datasets and Baselines

1) **Datasets:** We employ nine different datasets to demonstrate the effectiveness of our proposed FlowX with both quantitative studies and qualitative visualization results. These datasets are BA-Infection, BA-Traffic, BA-Shapes [26], BA-LRP [52], ClinTox [62], Tox21 [62], BBBP [62], BACE [62], and GraphSST2 [58], which include both synthetic and real-world data. Specifically, BA-Infection and BA-Traffic are datasets proposed by this work, where BA-Traffic (Section IV-E1) simulates the traffic jam challenge, as shown in Fig. 1, to emphasize the benefits of flow-based explanations. Detailed descriptions of BA-Infection can be found in the supplementary material. BA-Shapes is a node-classification synthetic dataset that is built by attaching house-like motifs to the base Barabási-Albert...
In our experiments, we consider GCNs as our graph models for all datasets. We adopt the trained graph models, which can be considered as special cases of our flow explanations, converting its node explanations to edge explanations, and node features, where node feature explanations are out of the scope of our work. In addition to the loss of node feature explanations, converting its node explanations to edge explanations introduces another level of information loss, rendering the comparisons hard to be fair. Furthermore, if we consider an edge-based Shapley value method, because of the additive property of the Shapley method, edge-based Shapley values can be equivalent to the summations of finer-grained Shapley values (naive flow sampling scores), leading to an uninteresting comparison when comparing under edge-based settings. More implementation details about explanation methods setting and GNN models can be found in the supplementary material 2.2. We used the datasets and implementations of the comparing algorithms in the DIG library [64].

4) Flow-Based Explainer Variant: FlowMask: In order to make this work comprehensive, we incorporate a mutual information training based flow-based method FlowMask, which sets $s^F_k$ in (9), as a training parameter for each flow $F_k$. These trainable parameters for flows form a flow mask that indicates flow importance scores. Following (10) and (11), we then obtain a layer edge mask. Given the output from (14) using the layer edge mask, we employ the exact mutual information optimization objective as used in [26], [28] to train the flow mask. Further comparisons with GNNExplainer are described in Section IV-E2.

B. Evaluation Metrics

The main metrics we apply are Fidelity+, Fidelity- [33], [58], and Sparsity [58]. According to the interpretation of [30], we use Fidelity+ and Sparsity to evaluate the effectiveness of necessary explanations and evaluate sufficient explanations with Fidelity- and Sparsity. Furthermore, we use full recall sparsity (Section IV-E1) to evaluate explainer performances on flow-related tasks with human-design ground-truths, i.e., BA-Traffic. It is critical to note that the Accuracy used in [28] is equivalent to Fidelityacc [58] that is similar to Fidelity$_{prob}$. To keep consistency, the Fidelity+ and Fidelity- we adopt are the probability version, i.e., Fidelity$_{prob}^{+}$ and Fidelity$_{prob}^{-}$ in [58]. We provide detailed introductions and explanations of these metrics in the supplementary materials.

C. Necessary Explanation Comparison

We first quantitatively compare different explanation methods on Fidelity+ and Sparsity to answer the first research question. Good necessary explanations should be faithful to the model and capture comprehensive structures for the predictions. When such structures are removed, the original predictions should change significantly. In addition, a GNN model generally uses

| Datasets          | Task      | # graph | # nodes | GCN Acc. | GIN Acc. |
|-------------------|-----------|---------|---------|----------|---------|
| BA-Infection      | Syn./GC   | 2000    | 39      | 99.00%   | 99.50%  |
| BA-Traffic        | Syn./GC   | 2000    | 20      | 100.00%  | 100.00% |
| BA-Shapes         | Syn./NC   | 1       | 700     | 90.29%   | 85.57%  |
| BA-LRP            | Syn./GC   | 20000   | 20      | 97.95%   | 100%    |
| ClinTox           | Real./GC  | 1478    | 136     | 93.96%   | 93.96%  |
| Tox21             | Real./GC  | 7831    | 58      | 88.66%   | 91.02%  |
| BBBP              | Real./GC  | 2039    | 100     | 87.80%   | 86.34%  |
| BACE              | Real./GC  | 1513    | 73      | 78.29%   | 80.26%  |
| Graph-SST2        | Real./GC  | 70042   | 36      | 90.84%   | 90.91%  |

Note that "NC" denotes node classification, and "GC" denotes graph classification. # nodes denotes the number of nodes of the largest graph in the datasets for the split of explanations. Acc. represents test accuracy. Syn. denotes Synthesis datasets; while Real denotes real-world datasets;

graph where the node labels are determined by their own identifies and localizations in motifs. Then, BA-LRP is a graph-classification synthetic dataset that includes Barabási-Albert graphs and the two classes are node-degree concentrated graph and evenly graph. Next, ClinTox, Tox21, BBBP, and BACE are real-world molecular datasets for graph classification. The chemical molecular graphs in these datasets are labeled according to their chemical properties, such as whether the molecule can penetrate a blood-brain barrier. Finally, Graph-SST2 is a natural language sentimental analysis dataset that converts text data to graphs. These graphs are labeled by their sentiment meanings. The properties and statistics of these datasets are included in Table I.

2) GNN Models: In our experiments, we consider GCNs [4] and GINs [1] as our graph models for all datasets. We adopt 2-layer GNNs for node classification and 3-layer GNNs for graph classification. The graph models are trained to achieve competitive performance and the details are reported in Table I.

3) Baselines: With the trained graph models, we quantitatively and qualitatively compare our methods with ten baselines. Specifically, we provide a justification for the choice of baselines as follows. We adopt GradCAM [40] and DeepLIFT [63] as traditional explanation baselines; GNNExplainer [26], PGMExplainer [27], PGMExplainer [32], and SubgraphX [33] as pioneer GNN explanation baselines. Since GNN-GI and GNN-LRP [52] can be considered as special cases of our flow modeling Section III-B, they serve as flow-based explanation baselines. Considering recent new techniques, we employ RC-Explainer [28] as our causal baseline and VGIN [39] as an information bottleneck baseline. Since these baselines have different explanation targets, we set the explanation target to graph edges for fair comparisons, i.e., the explanations of different methods are converted to edge importance scores if needed. It is critical to note that GraphSVX [36] focuses on explanations of node and node features, where node feature explanations are out of the scope of our work. In addition to the loss of node feature explanations, converting its node explanations to edge explanations introduces another level of information loss, rendering the comparisons hard to be fair. Furthermore, if we consider an edge-based Shapley value method, because of the additive property of the Shapley method, edge-based Shapley values can be equivalent to the summations of finer-grained Shapley values (naive flow sampling scores), leading to an uninteresting comparison when comparing under edge-based settings. More implementation details about explanation methods setting and GNN models can be found in the supplementary material 2.2. We used the datasets and implementations of the comparing algorithms in the DIG library [64].

4) Flow-Based Explainer Variant: FlowMask: In order to make this work comprehensive, we incorporate a mutual information training based flow-based method FlowMask, which sets $s^F_k$ in (9), as a training parameter for each flow $F_k$. These trainable parameters for flows form a flow mask that indicates flow importance scores. Following (10) and (11), we then obtain a layer edge mask. Given the output from (14) using the layer edge mask, we employ the exact mutual information optimization objective as used in [26], [28] to train the flow mask. Further comparisons with GNNExplainer are described in Section IV-E2.

B. Evaluation Metrics

The main metrics we apply are Fidelity+, Fidelity- [33], [58], and Sparsity [58]. According to the interpretation of [30], we use Fidelity+ and Sparsity to evaluate the effectiveness of necessary explanations and evaluate sufficient explanations with Fidelity- and Sparsity. Furthermore, we use full recall sparsity (Section IV-E1) to evaluate explainer performances on flow-related tasks with human-design ground-truths, i.e., BA-Traffic. It is critical to note that the Accuracy used in [28] is equivalent to Fidelityacc [58] that is similar to Fidelity$_{prob}$. To keep consistency, the Fidelity+ and Fidelity- we adopt are the probability version, i.e., Fidelity$_{prob}^{+}$ and Fidelity$_{prob}^{-}$ in [58]. We provide detailed introductions and explanations of these metrics in the supplementary materials.
redundant information to make predictions. Therefore, achieving high Fidelity+ scores indicates the removed structures are comprehensive to the predictions. When comparing using Fidelity+, another desired property of explanations is Sparsity. To encourage the explanations to be more human-intelligible, they should contain fewer but more important features. Higher Sparsity scores indicate that fewer structures are identified as important in the explanations. Hence, we control the Sparsity scores of explanations and compare the corresponding Fidelity+ scores. For each dataset, we use samples from the test set and conduct such quantitative evaluations. The results are reported in Fig. 4 where we show the plots of Fidelity+ scores with respect to different Sparsity levels. As shown in the figure, while FlowX (blue thin X) performs better than most methods, FlowX_{nec} (orange hollow X) outperforms them significantly across all Sparsity levels. These stable performances on different datasets indicate the promising ability of our methods for various applications and generalization scenarios. For synthetic datasets, the comparisons are more clear under a high Sparsity level.

D. Sufficient Explanation Comparison

In order to answer the second research question, we further conduct experiments on these datasets with metric Fidelity- and Sparsity for sufficient explanation comparisons. Intuitively, as the Sparsity increases, the selected subgraphs become smaller. If the explanation can preserve important structures to maintain the prediction performance, the explainer can achieve a low Fidelity-. As shown in Fig. 5, while FlowX (blue thin X) with pure flow samplings performs as a strong explainer, FlowX_{suf}'s performances (green star) surpass most methods on nearly all datasets clearly. The only exception happens on Graph-SST2 compared with RC-Explainer. There are two possible reasons behind this behavior. First, Graph-SST2, as a sentimental dataset, contains redundant sentimental information in sentences. This implies that only one or two keywords can be sufficient to make predictions, while large amounts of words need to be removed to change the predictions. Second, RC-Explainer reinforcement searches subgraphs from zero nodes, in which this

Fig. 4. Necessary explanation comparison. We compare Fidelity+ values on 9 datasets with GINs under different Sparsity levels. Our methods are drawn in solid lines while baselines are drawn in dashed. Higher Fidelity+ indicates better performance.

Fig. 5. Sufficient explanation comparison. We compare Fidelity- values on 9 datasets with GINs under different Sparsity levels. Our methods are drawn in solid lines while baselines are drawn in dashed. Lower Fidelity- indicates better performance.
tree-like searching process provides exhaustive-like searching for subgraphs with few nodes because of the smaller searching space near the roots of the searching tree. This exhaustive small subgraph search behavior coincides with the need for sufficient explanations of Graph-SST2, leading to its high sufficient explanation performance. However, because of this characteristic, RC-Explainer produces suboptimal results on datasets requiring large explanations and long-term correlations in Fig. 5 and is shown hard to search for necessary explanations that target covering comprehensive important information evaluated by Fidelity+.

Note that VGIB performs as a remarkable interpretable method, but its explanation ability is limited with fixed GNN models.

E. Flow-Based Versus Non-Flow-Based Methods

To demonstrate the benefits of using fine-grained flow-based methods and answer the third research question, we conduct experiments from the following two aspects.

1) Target Edge/Flow Retrieval Ability: BA-Traffic is a dataset simulating the challenge in Fig. 1. The base graphs of samples in BA-Traffic are 20-node 19-edge Barabási-Albert graphs that are constructed as traffic maps, where nodes are intersections, and edges are roads. In each sample, the node with the highest degree is a congested intersection. There are two types of traffic flows (car flows and bike flows), indicated by node features, going across the congested intersection and causing traffic jams. The label of the task is determined by the number of flows of each type, i.e., if the number of bike flows is larger than the number of car flows, the sample has label 0, otherwise, 1. There can be one or two flows for each type of flow, but the number of them cannot be the same. We provide the flow ground truths represented by edge sequences.

The explanation task is to cover all the true traffic flows, while we convert the true flows into edges for other explainers. Specifically, we calculate the lowest ratio between the number of selected edges/flows and the total number of edges/flows such that the selected edges/flows can cover all ground-truths. We denote “1 - this ratio” as a metric: full recall sparsity. Higher full recall sparsity indicates better retrieval performance. As shown in Fig. 6, flow-based methods have the ability to retrieve the real flows with high sparsity. However, other methods fail in this task from two aspects. First, they cannot retrieve true traffic flows intrinsically leaving unspecified edge explanations as depicted in Fig. 1. Second, the edge full recall sparsity is lower than the flow full recall sparsity, which indicates the flow explanations are more specified and concrete than the edge explanations. That is, edge explanations need larger portions of graphs to convey the same or commonly less information, compared to flow explanations. Therefore, flow explanations are more concrete and informative.

Note that the relatively weak performances produced by node-based explainers indicate the non-trivial information loss by converting node explanations to edge explanations.

2) Mutual Information Training Comparisons: Except for the benefits of the flow explanation itself, flow-based explainers have a unique advantage. This advantage originally comes from the flow modeling introduced in Section III-B, in which flow-based graph modeling can model multi-hop correlations while edge-based (graphon) modeling cannot. Empirically, to explore the merit brought from the modeling, we compare our flow-based training method FlowMask (Section IV-A4) with the similar edge-based method GNNExplainer in our comprehensive explanation experiments to show its favorable explanation performance.

FlowMask has the same mutual information training objective as GNNExplainer. The only distinction between them is the granularity of the trainable mask, i.e., a flow mask or an edge mask. As shown in Figs. 4, FlowMask (red hexagon, solid line) clearly outperforms GNNExplainer (green rectangular, dashed) on 7 out of 9 datasets for necessary explanation comparisons. For sufficient explanation comparisons, FlowMask’s performances surpass GNNExplainer’s non-trivially on 6 out of 9 datasets. Except for the Fidelity- comparison on BACE, FlowMask performs similarly to GNNExplainer on the rest of 4 datasets, i.e., BA-LRP and Graph-SST2 for Fidelity+; BA-LRP, BBBP for Fidelity-. Therefore, this experiment indicates that for the same training strategy, training on finer-grained explanation targets can produce better results, which implies the effectiveness of using flow-based methods from a training aspect.

According to the visualization in the supplementary materials and previous experimental results, we can obtain an insight: multi-hop correlation modeling and explanations can provide natural, more specified, and stronger explanation results.

V. CONCLUSION

We study the explainability of deep graph models, which are generally treated as black boxes. From the inherent functional mechanism of GNNs, we propose FlowX and its variants to explain GNNs by studying message flows. This work covers flow-based explanations from a systematical framework including an innovative flow-based graph modeling to comprehensively empirical implementations. Extensive experiments not only demonstrate the effectiveness and flexibility of FlowX and its variants but also emphasize the superiority of flow explanations. We hope this work can shed light on further multi-hop correlation modeling and explanation research.
D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, “SmoothGrad: Removing noise by adding noise,” 2017, arXiv:1706.03825.

F. Yang et al., “X-Fake: Explainable fake news detector with visualizations,” in Proc. World Wide Web Conf., 2019, pp. 3600–3604.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2921–2929.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 618–626.

H. Yuan, L. Cai, X. Hu, J. Wang, and S. Ji, “Interpreting image classifiers by generating discrete masks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 4, pp. 2019–2030, Apr. 2022.

P. Dabkowski and Y. Gal, “Real time image saliency for black box classifiers,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 6967–6976.

J. Chen, L. Song, M. Wainwright, and M. Jordan, “Learning to explain: An information-theoretic perspective on model interpretation,” in Proc. Int. Conf. Mach. Learn., PMLR, 2018, pp. 883–892.

H. Yuan, Y. Chen, X. Hu, and S. Ji, “Interpreting deep models for text analysis via optimization and regularization methods,” in Proc. 3rd AAAI Conf. Artif. Intell. Assoc. Adv. Artif. Intell., 2019, pp. 5717–5724.

C. Olah et al., “The building blocks of interpretability,” Distill, 2018.[Online]. Available: https://distill.pub/2018/building-blocks

M. Du, N. Liu, Q. Song, and X. Hu, “Towards explanation of DNN-based prediction with guided feature inversion,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1358–1367.

Z. Ding, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, “GNNEncoder: Generating explanations for graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 9244–9255.

D. Luo et al., “Parameterized explainer for graph neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 19620–19631.

X. Wang, Y. Wu, A. Zhang, F. Feng, X. He, and T.-S. Chua, “Reinforced causal explainer for graph neural networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 2, pp. 2297–2309, Feb. 2023.

H. Kuhn and A. W. Tucker, Contributions to the Theory of Games, vol. 2. Princeton, NJ, USA: Princeton Univ. Press, 1953.

K. Amara et al., “GraphFramex: Towards systematic evaluation of explainability methods for graph neural networks,” 2022, arXiv:2206.09677.

Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, “GraphLIME: Local interpretable model explanations for graph neural networks,” 2020, arXiv:2001.06216.

M. N. Vu and M. T. Thai, “PGM-explainer: Probabilistic graphical model explanations for graph neural networks,” Adv. Neural Inf. Process. Syst., vol. 33, pp. 12225–12235, 2020.

H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, “On explainability of graph neural networks via subgraph explanations,” in Proc. 38th Int. Conf. Mach. Learn., 2021, pp. 12241–12252.

Y. Xie et al., “Task-agnostic graph explanations,” 2022, arXiv:2202.08335.

H. Yuan, J. Tang, X. Hu, and S. Ji, “XGNN: Towards model-level explanations of graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 430–438.

A. Duval and F. Malliaros, “GraphSVX: Shapley value explanations for graph neural networks,” in Proc. Eur. Conf. Mach. Learn. Knowl. Discov. Data Min., 2021, pp. 30–39.

X. Wang, Y. Wu, A. Zhang, X. He, and T.-S. Chua, “Towards multi-grained explainability for graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 18446–18458.

C. Shan, Y. Shen, Y. Zhang, X. Li, and D. Li, “Reinforcement learning enhanced explainer for graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 22523–22533.

J. Yu, J. Cao, and R. He, “Improving subgraph recognition with variational graph information bottleneck,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 19396–19405.

P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, “Explainability methods for graph convolutional neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 10772–10781.

F. Baldassarre and H. Azizpour, “Explainability techniques for graph convolutional networks,” 2019, arXiv:1905.13686.

M. S. Schlichtkrull, N. De Cao, and I. Titov, “Interpreting graph neural networks for NLP with differentiable edge masking,” in Proc. Int. Conf. Learn. Representations, 2020.

J. Peters, P. Bühlmann, and N. Meinshausen, “Causal inference by using invariant prediction: Identification and confidence intervals,” J. Roy. Statist. Soc.: Ser. B (Statist. Methodol.), vol. 78, no. 5, pp. 947–1012, 2016.

K. Ahuja et al., “Invariance principle meets information bottleneck for out-of-distribution generalization,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 3438–3450.

J. Peters, P. Bühlmann, and N. Meinshausen, “Causal inference by using invariant prediction: Identification and confidence intervals,” J. Roy. Statist. Soc.: Ser. B (Statist. Methodol.), vol. 78, no. 5, pp. 947–1012, 2016.

K. Ahuja et al., “Invariance principle meets information bottleneck for out-of-distribution generalization,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 3438–3450.

Y. Chen et al., “Learning causally invariant representations for out-of-distribution generalization,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 22131–22148.

S. Gui, M. Liu, X. Li, Y. Luo, and S. Ji, “Joint learning of label and environment causal independence for graph out-of-distribution generalization,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 25895–25907.

S. Gui, X. Li, L. Wang, and S. Ji, “GOOD: A graph out-of-distribution benchmark,” in Proc. 36th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track, 2022, pp. 2059–2073. [Online]. Available: https://openreview.net/forum?id=8Hlg_zs-p-h

X. Li, S. Gui, Y. Luo, and S. Ji, “Graph structure and feature extrapolation out-of-distribution generalization,” 2023, arXiv:2306.00876.

Z. Guo, T. Xiao, C. Aggarwal, H. Liu, and S. Wang, “Counterfactual learning on graphs: A survey,” 2023, arXiv:2304.01391.

Z. Guo, T. Xiao, C. Aggarwal, H. Liu, and S. Wang, “Counterfactual explanations on graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 5644–5655.

J. Ma, R. Guo, S. Mishra, A. Zhang, and J. Li, “CLEAR: Generative counterfactual explanations on graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 25895–25907.

T. Schnake et al., “Higher-order explanations of graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 302–318.

M. Bajaj et al., “Robust counterfactual explanations on graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 5644–5655.

J. Ma, R. Guo, S. Mishra, A. Zhang, and J. Li, “CLEAR: Generative counterfactual explanations on graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 25895–25907.

T. Schnake et al., “Higher-order explanations of graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 302–318.
[55] L. Lovász and B. Szegedy, “Limits of dense graph sequences,” J. Combinatorial Theory, Ser. B, vol. 96, no. 6, pp. 933–957, 2006.

[56] B. Bevilacqua, Y. Zhou, and B. Ribeiro, “Size-invariant graph representations for graph classification extrapolations,” in Proc. Conf. Mach. Learn., PMLR, 2021, pp. 837–851.

[57] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” 2016, arXiv:1607.08022.

[59] E. Štrumbelj and I. Kononenko, “Explaining prediction models and individual predictions with feature contributions,” Knowl. Inf. Syst., vol. 41, no. 3, pp. 647–665, 2014.

[60] H. Yuan, H. Yu, S. Gui, and S. Ji, “Explainability in graph neural networks: A taxonomic survey,” 2020, arXiv: 2012.15445.

[61] S. Miao, M. Liu, and P. Li, “Interpretable and generalizable graph learning via stochastic attention mechanism,” in Proc. Int. Conf. Mach. Learn., PMLR, 2022, pp. 15524–15543.

[62] Z. Wu et al., “MoleculeNet: A benchmark for molecular machine learning,” Chem. Sci., vol. 9, no. 2, pp. 513–530, 2018.

[63] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” in Proc. 34th Int. Conf. Mach. Learn., 2017, pp. 3145–3153.

[64] M. Liu et al., “DIG: A Turnkey library for diving into graph deep learning research,” J. Mach. Learn. Res., vol. 22, no. 240, pp. 1–9, 2021. [Online]. Available: http://jmlr.org/papers/v22/21-0343.html

Shurui Gui received the BS degree in computer science from the University of Science and Technology of China, in 2020. He is currently working toward the PhD degree with Texas A&M University, College Station, Texas. His research interests include deep learning, explainability, OOD generalization, and causality.

Hao Yuan received the BS degree in computer science from the University of Science and Technology of China, in 2012, the MS degree in computer science from the Memorial University of Newfoundland, in 2016, and the PhD degree in computer science from Texas A&M University, College Station, Texas, in 2021. He is currently a research scientist with Meta Platforms, Inc. His research interests include machine learning, deep learning, and explainability.

Jie Wang (Senior Member, IEEE) received the BS degree in electronic information science and technology from the University of Science and Technology of China, Hefei, China, in 2005, and the PhD degree in computational science from the Florida State University, Tallahassee, FL, in 2011. He is currently a professor with the Department of Electronic Engineering and Information Science at University of Science and Technology of China. His research interests include reinforcement learning, knowledge graph, large-scale optimization, deep learning, etc.