FLOWX: TOWARDS EXPLAINABLE GRAPH NEURAL NETWORKS VIA MESSAGE FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the explainability of graph neural networks (GNNs) as a step towards 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 employ the concept of Shapley values from cooperative game theory. To tackle the complexity of computing Shapley values, we propose an approximation scheme to compute Shapley values as initial assessments of flow importance. We then propose a learning algorithm to refine scores and improve explainability. Experimental studies on both synthetic and real-world datasets demonstrate that our proposed FlowX leads to improved explainability of GNNs.

1 INTRODUCTION

With the advances of deep learning, graph neural networks (GNNs) are achieving promising performance on many graph tasks, including graph classification (Xu et al., 2019; Gao & Ji, 2019; Chen et al., 2020), node classification (Kipf & Welling, 2017; Veličković et al., 2018; Wu et al., 2019), and graph generation (Luo et al., 2021; You et al., 2018). Many research efforts have been made to develop advanced graph operations, such as graph message passing (Kipf & Welling, 2017; Veličković et al., 2018; Li et al., 2019), graph pooling (Yuan & Ji, 2020; Zhang et al., 2018; Ying et al., 2018), and 3D graph operations (Schütt et al., 2017; Klicpera et al., 2020). Deep graph models usually consist of many layers of these operations stacked on top of each other interspersed with nonlinear functions. The resulting deep models are usually deep and 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 (Simonyan et al., 2013; Smilkov et al., 2017; Yang et al., 2019), mapping hidden features to input space (Zhou et al., 2016; Selvaraju et al., 2017), occluding different input features (Yuan et al., 2020a; Dabkowski & Gal, 2017; Chen et al., 2018), and studying the meaning of hidden layers (Yuan et al., 2019; Olah et al., 2018; Du et al., 2018), 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 XGNN (Yuan et al., 2020b), GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), and SubgraphX (Yuan et al., 2021), etc. These methods mainly focus on explaining graph nodes, edges, features, or subgraphs.

In this work, we observe and argue that message flows are the inherent functional mechanism of GNNs and thus are more natural and intuitive for studying the explainability of GNNs. To this end, we propose a message flow based explanation method, known as FlowX, to explain GNNs. FlowX attributes GNN predictions to message flows and studies the importance of different message flows. We first develop a systematic framework that lays the foundation on message flows for naturally explaining the message passing in GNNs. With our framework, the FlowX first quantifies the importance of flows by employing the concept of Shapley values. Since message flows cannot

be directly quantified using Shapley values, we propose an approximation scheme as the initial assessments of different flows. We then propose a learning-based algorithm to refine the initial assessments to capture important flows in predictions. We conduct extensive experiments on both synthetic and real-world datasets. Experimental results show that our proposed FlowX outperforms existing methods significantly and consistently. Both quantitative and qualitative studies demonstrate that our proposed FlowX leads to improved explainability of GNNs.

2 Related Work

2.1 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) (Kipf & Welling, 2017), graph attention networks (GATs) (Veličković et al., 2018), and graph isomorphism networks (GINs) (Xu et al., 2019), 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 messages 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 $v_i \to v_j$. 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 a_{ij} in A represents the weight of e_{ij}, and $a_{ij} = 0$ indicates e_{ij} does not exist. For the t-th layer in GNNs, the 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 $\hat{a}^t_{jk}$ indicates the layer edge connecting node j and k 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

$$f(\mathcal{G}) = g(M^T(M^{T-1}(\cdots M^1(X^0\hat{A}^1)\cdots)\hat{A}^{T-1})\hat{A}^T).$$

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 a MLP node classifier.

2.2 Explainability of Graph Neural Networks

A major limitation of GNNs is their lack of explainability. Thus, different methods have been proposed to explain the predictions of GNNs, such as GraphLime (Huang et al., 2020), GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), PGMExplainer (Vu & Thai, 2020), SubgraphX (Yuan et al., 2021), XGNN (Yuan et al., 2020b), and GraphSVX (Duval & Malliaros, 2021). These methods can be mainly grouped into four categories based on the views of their explanations. First, several techniques provide explanations by identifying important nodes in the input graph. For example, GradCAM (Pope et al., 2019) measures node importance by combining the hidden features and gradients; LRP (Baldassarre & Azizpour, 2019) and Excitation BP (Pope et al., 2019) decompose the predictions into several terms and assign these terms to different nodes; PGM-Explainer (Vu & Thai, 2020) builds a probabilistic graphical model by randomly perturbing the node features and employs an interpretable Bayesian network to generate explanations. Second, several existing methods, such as GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), and GraphMask (Schlichtkrull et al., 2021), explain GNNs by studying the importance of different graph edges. These methods

follow a similar high-level idea that learns masks to identify important edges while maximizing the mutual information. Next, the recent study SubgraphX (Yuan et al., 2021) proposes to explain GNNs via subgraphs. It incorporates the Monte Carlo tree search algorithm to explore subgraphs and employs Shapley values to measure the importance. Finally, XGNN (Yuan et al., 2020b) focuses on model-level explanations, which can provide high-level insights and general understanding. It proposes to generate graph patterns that can maximize a certain model prediction. While those methods explain GNNs from different views, none of them can provide explanations in terms of message flows and we believe message flows are more natural for performing explainability.

To the best of our knowledge, GNN-LRP (Schnake et al., 2020) is the only algorithm that explains GNNs by relevant walks. The relevant walk is defined as a T-length ordered edge sequence that corresponds to a T-step directed path on the input graph. To study walk explainability, GNN-LRP considers the GNN prediction as a function and decomposes it using higher-order Taylor expansions to distribute prediction scores to relevant walks. Specifically, by using T-order Taylor expansion with a proper root, each term in the Taylor expansion corresponds to a relevant walk and is regarded as the importance score. While our proposed FlowX shares a similar explanation target, i.e., flow/walk, with GNN-LRP, our method is fundamentally different. The GNN-LRP is developed based on score decomposition while our method employs the concept of Shapley values from cooperative game theory as initial assessments and proposes a learning-based algorithm for refinement. In addition, GNN-LRP has several constraints on the activation function and bias term used in GNNs while our method can be applied to general GNN models. Furthermore, as the GNN-LRP follows the Gradient $\times$ Input scheme, it may not pass the model parameter randomization test and maybe not sensitive to model parameters (Adebayo et al., 2018).

Differences with Other Methods Using Shapley Values: Shapley values are commonly used in explaining machine learning methods. In particular, a recent study proposes a surrogate method, known as GraphSVX (Duval & Malliaros, 2021), to explain GNNs with both node and feature masks. Another recent study proposes SubgraphX (Yuan et al., 2021), which employs a search algorithm to explore and identify subgraphs with high Shapley scores. While both methods use Shapley values, there are several fundamental differences. First, our proposed method focuses on explaining message flows, which are the most basic and natural units for explanations as GNNs are based on message passing schemes. Second, we only use Shapley values as initial approximations, which are subsequently refined by our proposed learning based methods. Experiments show that the refinement step is very important. Furthermore, the fundamental difference with GraphSVX is reflected in the fact that, while our method is a perturbation-based method, GraphSVX is a surrogate method (Yuan et al., 2020c). Due to these differences, we show in experiments that our FlowX achieves more natural and improved performance as compared with other methods.

3 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 adopt the concept of Shapley values (Kuhn & Tucker, 1953) from game theory and propose an approximation scheme for them. In addition, a learning-based algorithm is proposed to improve the explainability of message flows.

3.1 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 follow:

Definition 1: Message Carrier. We use the connectivity matrix to represent the carriers for message flows. Given the connectivity matrix $\hat{A}^t$ at layer t, the layer edge $\hat{a}_{ij}^t$ 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 $\boldsymbol{\mathcal{A}} = \{\cdots, \hat{a}_{uv}^1, \cdots, \hat{a}_{uv}^t, \cdots, \hat{a}_{uv}^T, \cdots\}$ and $|\boldsymbol{\mathcal{A}}| = |E| \times T$.

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

In a T-layer GNN model, all message flows start from the input layer and end with the final layer so that their lengths equal to T. For the ease of notations, we introduce the wildcard $*$ to represent any valid node sequence and $\boldsymbol{\mathcal{F}}$ to denote the message flow set. For example, we can use $\boldsymbol{\mathcal{F}}_{ij*}$, $\boldsymbol{\mathcal{F}}_{*\ell m}$, and $\boldsymbol{\mathcal{F}}_{ij*\ell m}$ to denote the message flow sets that share the same message carrier $\hat{a}_{ij}^1$, $\hat{a}_{\ell m}^T$, or 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, $\boldsymbol{\mathcal{F}}_{?\{3\}}$ means the set of valid 2-step message flow with 3 nodes. Note that the following property of message flow sets also holds:

$$\boldsymbol{\mathcal{F}}_{ij*\ell m} = \boldsymbol{\mathcal{F}}_{ij*} \cap \boldsymbol{\mathcal{F}}_{*\ell m}. \tag{3}$$

The final embeddings of node v_m are determined by all incoming message flows to node v_m, which can be denoted as $\boldsymbol{\mathcal{F}}_{*m}$. Since the output of the GNN model is obtained based on the final node embeddings, then it is reasonable to treat the GNN output as the combinations of different message flows. Hence, it is natural to demystify GNN models by studying the importance of different message flows towards GNN predictions.

3.2 Shapley Values as Initial Assessments

While explaining GNNs with message flows seems to be promising, it is still crucial to properly measure the importance of those message flows. Hence, in this work, our FlowX proposes to adopt the concept of Shapley value (Kuhn & Tucker, 1953) as the initial assessments of flow importance. Note that Shapley value is a solution concept in cooperative game theory and used to fairly assign the game gain to different players. When applying it to 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 $\mathcal{G}$, we use $\boldsymbol{\mathcal{F}}_*$ to denote the set of all valid flows, *i.e.*, all players in the game. Then the Shapley value of flow $\mathcal{F}^k$ can be mathematically written as

$$\phi(\mathcal{F}^k) = \sum_{P \subseteq \boldsymbol{\mathcal{F}}_* \setminus \{\mathcal{F}^k\}} \frac{|P|!\,(|\boldsymbol{\mathcal{F}}_*| - |P| - 1)!}{|\boldsymbol{\mathcal{F}}_*|!} \left(f\left(P \cup \{\mathcal{F}^k\}\right) - f(P)\right), \tag{4}$$

where P denotes the possible coalition group of players, $|\cdot|$ is the set size, and $\phi(\cdot)$ denotes the Shapley score. In addition, $f\left(P \cup \{\mathcal{F}^k\}\right) - f(P)$ 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. To compute the Shapley score $\phi(\mathcal{F}^k)$, we need to enumerate all possible coalition groups and considers different interactions among players. Note that Shapley value is a fair solution since it is the only solution satisfying four desirable properties, including efficiency, symmetry, linearity, and dummy (Lundberg & Lee, 2017). However, it is not possible to compute Shapley values with Eq. (4) for message flows since we cannot remove individual message flows from the GNN model and the finest component we can directly remove is the layer edge. For example, when removing the layer edge $\hat{a}_{ij}^1$, the whole flow set $\boldsymbol{\mathcal{F}}_{ij*}$ is removed from the model. In addition, enumerating all possible coalition groups is time-consuming when the input graph is large-scale and the GNN model is deep. Hence, in our FlowX, we propose an approximation scheme to compute Shapley values based on Monte Carlo (MC) sampling (Štrumbelj & Kononenko, 2014).

Formally, let M denote the total number of MC sampling steps. For the i-th sampling step, we use $\pi(i)$ to represent a random permutation of $|E| \times T$ elements. Then the set of all layer edge $\boldsymbol{\mathcal{A}}$ is permuted based on $\pi(i)$, denoted as $\boldsymbol{\mathcal{A}}_{\pi(i)}$ that $\boldsymbol{\mathcal{A}}_{\pi(i)}[j] = \boldsymbol{\mathcal{A}}[\pi(i)[j]]$. In each step i, we iteratively

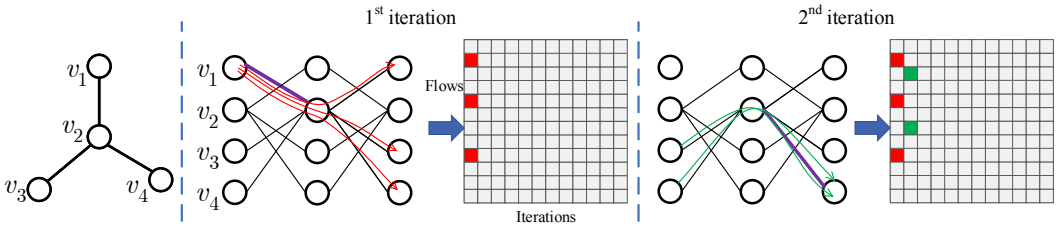

Figure 1: An illustration of our initial assessments via Shapley value approximations. For each MC sampling step, we iteratively remove one layer edge until all layer edges are removed. In this example, the removed layer edges are shown in bold and purple lines while the corresponding message flows are shown in arrow lines. For example, in the first iteration, we remove the layer edge between v_1 and v_2 from the first GNN layer and compute the marginal contribution. Then three message flows are removed and the contribution scores are averaged and assigned to these three message flows. Note that we hide the self-loops in this figure for clarity.

Algorithm 1 INITIAL APPROXIMATIONS OF FLOW IMPORTANCE SCORES.

Given a trained GNN model $f(\cdot)$ and an input graph, the set of all layer edges is represented as $\boldsymbol{\mathcal{A}}$. For each message flow $\mathcal{F}^k$, two $|\boldsymbol{\mathcal{A}}|$-dimensional vectors $S(\mathcal{F}^k)$ and $C(\mathcal{F}^k)$ denote its importance scores and removing index counts respectively. In addition, $\widehat{\boldsymbol{\mathcal{A}}}$ denotes the set of removed layer edges and $\widehat{\boldsymbol{\mathcal{F}}}$ is the set of removed message flows.
Initialize $S(\cdot)$ and $C(\cdot)$ as zeros for all message flows.
for step i from 1 to Monte Carlo sampling step M **do**
  Initialize the removed sets as empty that $\widehat{\boldsymbol{\mathcal{A}}} = \emptyset$ and $\widehat{\boldsymbol{\mathcal{F}}} = \emptyset$.
  Randomly shuffle and permute the layer edge set $\boldsymbol{\mathcal{A}}$, denoted as $\boldsymbol{\mathcal{A}}_{\pi(i)}$.
  for j from 1 to $|\boldsymbol{\mathcal{A}}|$ **do**
    Select the j-th layer edge in $\boldsymbol{\mathcal{A}}_{\pi(i)}$, denoted as $\hat{a}^t_{\ell m}$.
    Block the layer edge $\hat{a}^t_{\ell m}$ in GNN model $f(\cdot)$, then the removed flows are $\widehat{\boldsymbol{\mathcal{F}}}^j = \boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}} \setminus (\widehat{\boldsymbol{\mathcal{F}}} \cap \boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}})$.
    Compute the prediction difference that $s_j = f(\boldsymbol{\mathcal{A}} \setminus \widehat{\boldsymbol{\mathcal{A}}}) - f(\boldsymbol{\mathcal{A}} \setminus (\widehat{\boldsymbol{\mathcal{A}}} \cup \hat{a}^t_{\ell m}))$.
    Update $\widehat{\boldsymbol{\mathcal{A}}} = \widehat{\boldsymbol{\mathcal{A}}} \cup \hat{a}^t_{\ell m}$ and $\widehat{\boldsymbol{\mathcal{F}}} = \widehat{\boldsymbol{\mathcal{F}}} \cup \boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}}$.
    Compute averaged score that $\bar{s_j} = s_j / |\widehat{\boldsymbol{\mathcal{F}}}^j|$.
    for each flow $\mathcal{F}^k$ in $\widehat{\boldsymbol{\mathcal{F}}}^j$ **do**
      Update $S(\mathcal{F}^k)[j] = S(\mathcal{F}^k)[j] + \bar{s_j}$.
      Update $C(\mathcal{F}^k)[j] = C(\mathcal{F}^k)[j] + 1$
    end for
  end for
end for
For each message flow $\mathcal{F}^k$, compute our Shapley approximations $S(\mathcal{F}^k) = S(\mathcal{F}^k)/C(\mathcal{F}^k)$

remove one layer edge from $\boldsymbol{\mathcal{A}}_{\pi(i)}$ following the order, and compute the marginal contribution. An illustration of our proposed Shapley value approximations is shown in Figure 1. We use $\widehat{\boldsymbol{\mathcal{A}}}$ to denote the set of removed layer edges and $\widehat{\boldsymbol{\mathcal{F}}}$ to denote the set of removed message flows, which are initialized as $\widehat{\boldsymbol{\mathcal{A}}} = \emptyset$ and $\widehat{\boldsymbol{\mathcal{F}}} = \emptyset$ in the beginning of each MC sampling step. For the iteration j in sampling step i, the j-th element of $\boldsymbol{\mathcal{A}}_{\pi(i)}$ is removed and we assume the removed layer edge is $\hat{a}^t_{\ell m}$. Then the computation operations can be mathematically written as

$$s_j = f(\boldsymbol{\mathcal{A}} \setminus \widehat{\boldsymbol{\mathcal{A}}}) - f(\boldsymbol{\mathcal{A}} \setminus (\widehat{\boldsymbol{\mathcal{A}}} \cup \hat{a}^t_{\ell m})), \tag{5}$$

$$\widehat{\boldsymbol{\mathcal{A}}} = \widehat{\boldsymbol{\mathcal{A}}} \cup \{\hat{a}^t_{\ell m}\}, \tag{6}$$

$$\widehat{\boldsymbol{\mathcal{F}}}^j = \boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}} \setminus (\widehat{\boldsymbol{\mathcal{F}}} \cap \boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}}), \tag{7}$$

$$\widehat{\boldsymbol{\mathcal{F}}} = \widehat{\boldsymbol{\mathcal{F}}} \cup \boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}}, \tag{8}$$

where $\widehat{\boldsymbol{\mathcal{F}}}^j$ denotes the removed message flows in iteration j by removing $\hat{a}^t_{\ell m}$. Note that it is not equivalent to $\boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}}$ since the flows in $\boldsymbol{\mathcal{F}}_{?\{t-1\}\ell m?\{T-t\}}$ may be already removed in the previous iterations. Then the score is averaged that $\bar{s_j} = s_j/|\widehat{\boldsymbol{\mathcal{F}}}^j|$ and we assign $\bar{s_j}$ to each flow in $\widehat{\boldsymbol{\mathcal{F}}}^j$. By repeating such operations until all layer edges are removed, we can obtain $|\boldsymbol{\mathcal{A}}|$ marginal scores and assign them to the corresponding flows. Note that the order information in $\boldsymbol{\mathcal{A}}_{\pi(i)}$ is important since in the earlier iterations, the removed flows $\widehat{\boldsymbol{\mathcal{F}}}^j$ are interacting with a larger coalition group $\boldsymbol{\mathcal{A}} \setminus (\widehat{\boldsymbol{\mathcal{A}}} \cup \hat{a}^t_{\ell m})$ and in the later iterations the coalition groups are small. Altogether, the steps of our Shapley approximations are shown in Algorithm 1.

After M MC sampling steps, our method explores M permutations of layer edge set and each flow is assigned with M importance scores. For each flow $\mathcal{F}^k$, we use a $|\boldsymbol{\mathcal{A}}|$-dimensional vector to store its importance scores, denoted as $S(\mathcal{F}^k)$, where $S(\mathcal{F}^k)[j]$ is the importance score obtained when $\mathcal{F}^k$ is removed in the iteration j. If the flow is removed mutiple times in the iteration j for different MC sampling steps, the scores are averaged. Then it raises the question that how to convert $S(\mathcal{F}^k)$ to the final importance score of $\mathcal{F}^k$. While Shapley values are promising, our approximations may not be accurate since M is always set as $M << |\boldsymbol{\mathcal{A}}|!$ for the sake of computation efficiency. Hence, directly averaging the scores in $S(\mathcal{F}^k)$ may not suitable.

3.3 Learning to Refine Importance Scores

With our obtained Shapley value approximations $S(\mathcal{F}^k)$, we propose to regard these values as initial assessments of flow importance and learn to refine such importance scores. We propose to learn weights for combining the elements in $S(\mathcal{F}^k)$ to obtain the final importance score for each flow. Inspired by the mask-based explanation methods (Ying et al., 2019; Luo et al., 2020), we propose to obtain masks for indicating important layer edges. Note that since message flows cannot be masked out individually, we can only directly focus on the masks of layer edges and indirectly study the message flows.

Formally, given the input graph $\mathcal{G}$ and GNN model $f(\cdot)$, we obtain Shapley value approximations $S(\mathcal{F}^k)$ for each message flow. Then a weight vector w is randomly initialized, which shares the same dimension of $|S(\mathcal{F}^k)|$. Note that the weights w are shared among all message flows in the same graph. Then the score vectors $S(\mathcal{F}^k)$ can be combined via element-wise multiplication with w. Next, the flow importance can be converted to layer edge importance by simply summing up the scores of all flows sharing a particular layer edge as the message carrier. Mathematically, it can be written as

$$s(\mathcal{F}^k) = S(\mathcal{F}^k) \cdot w \quad \forall \mathcal{F}^k \in \boldsymbol{\mathcal{F}}_*, \tag{9}$$

$$s(\hat{a}^t_{uv}) = \sum s(\mathcal{F}^j) \quad \forall \mathcal{F}^j \quad \textbf{with carrier} \quad \hat{a}^t_{uv}, \tag{10}$$

where $s(\cdot)$ denotes the importance score for layer edges and flows. Then based on $s(\hat{a}^t_{uv})$, we can obtain a mask indicating the importance of different layer edges, denoted as M. Note that the mask is normalized so that each element is in the range $[0, 1]$. By applying the mask to layer edges, important layer edges are blocked and the model prediction becomes

$$\hat{y} = f(\text{Combine}(\mathcal{G}, 1 - M)), \tag{11}$$

where $\hat{y}$ is the prediction vector and $\text{Combine}(\mathcal{G}, 1 - M)$ means layer edges are masked out based on the values of M. Intuitively, if important layer edges are removed, then the prediction should change significantly. Hence, $\hat{y}$ is encouraged to be different from the original prediction by learning proper weights w. Note that, the weights w are the only learnable parameters in our refining stage while all other variables are fixed. Specifically, we employ the negative cross-entropy loss to learn the weights and it can be formally expressed as

$$\mathcal{L}(\hat{y}, y) = \sum_{c=1}^{m} \mathbf{1}\{y = c\} \log \widehat{y}_c, \tag{12}$$

where y is a scalar representing the predicted class of the original graph $\mathcal{G}$, m is the number of classes, $\mathbf{1}\{\cdot\}$ denotes the indicator function, and $\widehat{y}_c$ is the new predicted probability for class c. After training, the final importance scores for different flows can be obtained via Eq. (9). Furthermore, we apply exponential scaling on the layer edge mask M to facilitate training and encourage discreteness as we observe that the training is not stable without such a scaling term.

Table 1: Statistics and properties of seven datasets. Note that "NC"denotes node classification, and "GC" denotes graph classification.

Datasets	Task	# graph	GCN Accuracy	GIN Accuracy
BA-Shapes	Synthetic/NC	1	90.29%	89.57%
BA-LRP	Synthetic/GC	20000	97.95%	100%
Clintox	Real/GC	1478	93.96%	93.96%
Tox21	Real/GC	7831	88.66%	91.02%
BBBP	Real/GC	2039	87.80%	86.34%
BACE	Real/GC	1513	78.29%	80.26%
Graph-SST2	Real/GC	70042	90.84%	90.91%

4 EXPERIMENTAL STUDIES

4.1 DATASETS AND EXPERIMENTAL SETTINGS

Datasets. We employ seven different datasets to demonstrate the effectiveness of our proposed FlowX with both quantitative studies and qualitative visualization results. These datasets are BA-Shapes (Ying et al., 2019), BA-LRP (Schnake et al., 2020), ClinTox (Wu et al., 2018), Tox21 (Wu et al., 2018), BBBP (Wu et al., 2018), BACE (Wu et al., 2018), BA-INFE, and Graph-SST2 (Yuan et al., 2020c), which include both synthetic and real-world data. First, BA-Shapes is a node-classification synthetic dataset that is built by attaching house-like motifs to the base Barabási-Albert 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 molecule datasets for graph classification. The chemical molecule 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. Note that the BA-INFE dataset is a synthetic dataset for Accuracy comparisons and the results are reported in the appendix. We also report the properties and statistics of these datasets in Table 1.

GNN Models. In our experiments, we consider GCNs (Kipf & Welling, 2017) and GINs (Xu et al., 2019) 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 1.

Baselines. With the trained graph models, we quantitatively and qualitatively compare our FlowX with eight baselines, including GradCAM (Pope et al., 2019), DeepLIFT (Shrikumar et al., 2017), GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), PGMExplainer (Vu & Thai, 2020), SubgraphX (Yuan et al., 2021), GNN-GI (Schnake et al., 2020), GNN-LRP (Schnake et al., 2020). Note that since these methods cannot be directly compared, we set the explanation target to graph edges for fair comparisons that the explanations are converted to edge importance scores if needed. More implementation details about explanation methods setting and GNN models can be found in the supplementary material. We used the datasets and implementations of the comparing algorithms in the DIG library (Liu et al., 2021). We will release our code after the anonymous review period.

4.2 QUANTITATIVE STUDIES

We first quantitatively compare different explanation methods. We follow the existing studies (Yuan et al., 2020c; Pope et al., 2019) and employ two metrics to evaluate the explanations: Fidelity and Sparsity. Good explanations should be faithful to the model and capture the discriminative features for the predictions. When such input features are removed, the original predictions should change significantly. The Fidelity score measures the change of predicted probabilities when removing important input features identified by different explanation methods. Higher Fidelity scores indicate the removed features are more important to the predictions and hence the explanations are more faithful to the model. In addition, the other desired property of explanations is sparsity. To encourage the explanations to be more human-intelligible, they should contain fewer but more important features. Hence, we also employ the Sparsity metric which measures the percentage of input features that are

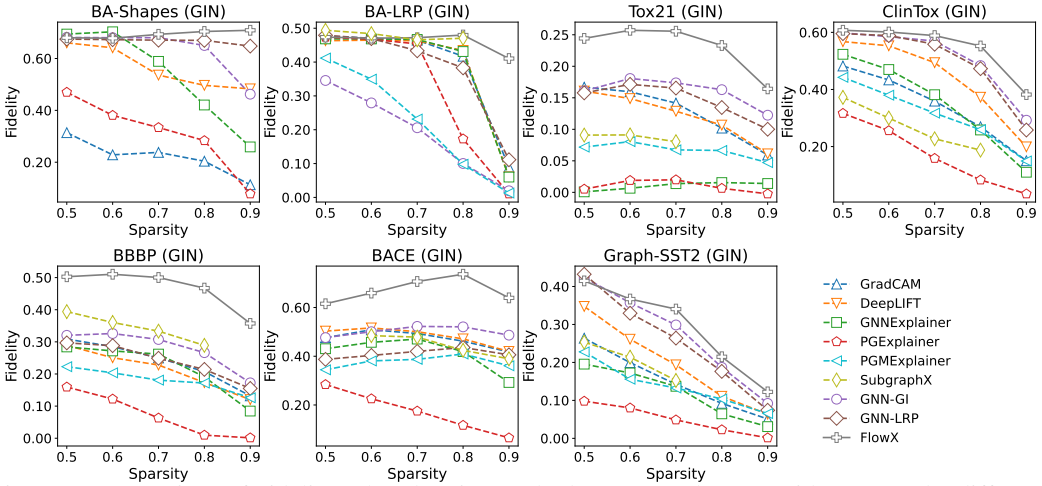

Figure 2: Comparison of Fidelity values by nine methods on seven datasets with GINs under different Sparsity levels.

Table 2: Results of ablation studies of our method on GCNs. FlowX† denotes our FlowX without Shapley initial assessments and FlowX* denotes our FlowX without learning refinement.

	BA-Shapes	BA-LRP	ClinTox	Tox21	BBBP	BACE	Graph-SST2
FlowX†	0.3999	0.1657	0.3437	0.1662	0.3918	0.3186	0.2219
FlowX*	0.4137	**0.5153**	0.3171	0.1933	0.4929	0.5019	0.2583
FlowX	**0.4198**	0.5145	**0.3843**	**0.2243**	**0.5657**	**0.5146**	**0.3217**

identified as important. Higher Sparsity scores indicate that fewer features are identified as important in the explanations. The formulations and details of these metrics are discussed in the appendix B.

It is noteworthy that these two metrics are highly correlated since the predictions tend to change more when more input features are removed. Then explanations with higher Sparsity scores tend to have lower Fidelity scores. Hence, we argue that the Fidelity scores need to be compared under a similar Sparsity level. Specifically, we control the Sparsity scores of explanations and compare the corresponding Fidelity scores. For each dataset, we randomly select 100 samples and conduct such quantitative evaluations. The results are reported in Figure 2 where we show the plots of Fidelity scores with respect to different Sparsity levels. Obviously, FlowX performs better on all of the real-world datasets that FlowX consistently achieves higher Fidelity across all Sparsity levels. Meanwhile, the results on synthetic datasets are competitive when the Sparsity is low and our FlowX still performs significantly better with high Sparsity levels. Note that our method is shown to achieve stable performance on different datasets and GNN models, which is promising for the applications and generalizations of our FlowX. Meanwhile, our proposed method, GNN-LRP, and GNN-GI perform better than the other methods, which indicates the superiority of the methods based on flows and walks. Note that GNN-LRP achieves good results on synthetic datasets but surprisingly, it is not better than GNN-GI on real-world datasets. In addition, PGExplainer only obtains competitive results on simple synthetic datasets but not on complex real-world datasets. Note that, for SubgraphX, its subgraph explanations are not directly comparable so we convert its subgraph explanations to edge explanations. It performs well on the BA-LRP and BBBP datasets but not as expected on the other datasets. We believe the reason may be that such converting destroys the continuousness and completeness of its subgraph explanations. Last but not least, we wish to mention that the comparisons using edge actually limit the performance of the flow-based methods due to explanation granularity changes. More results and time comparisons are also reported in the appendix.

4.3 Ablation Studies

To demonstrate the effectiveness of our proposed learning-based algorithm in Section 3.3, we study the performance when only using the average of our Shapley initial assessments as the importance scores without further learning, denoted as FlowX*. In addition, to show the Shapley initial assessments are necessary, we study the method only using the learning algorithm with randomly initialized scores

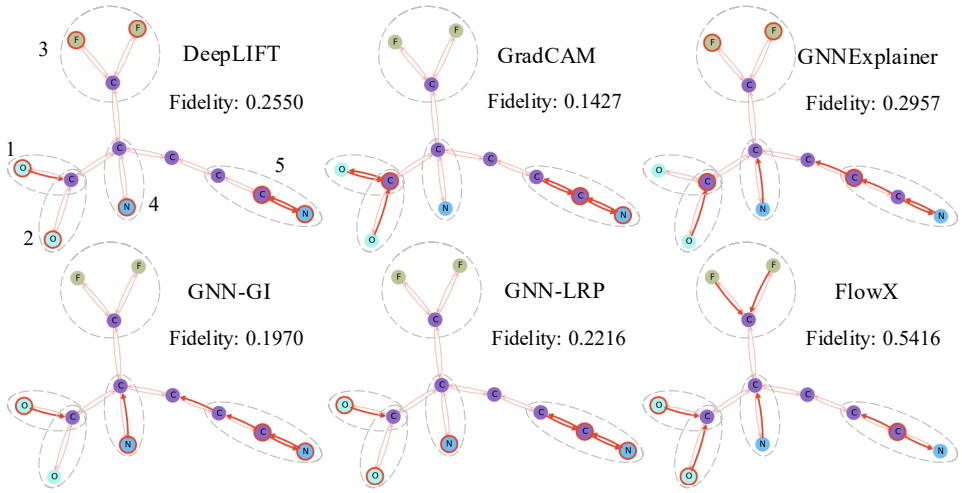

Figure 3: Sample explanation results of different methods. Note that explanations are shown as directed edges (highlighted red arrow lines) and self-loops (red circles around atoms). In addition, motifs are emphasized by dashed circles and numbered from 1 to 5.

for different flows, denoted as FlowX†. We compare our FlowX with these two baselines on GCN models, and the results are reported in Table 2. Obviously, for all datasets, the FlowX and FlowX* outperform FlowX† significantly, showing that the Shapley initial assessments are appropriate and necessary. In addition, for synthetic datasets, the FlowX and FlowX* show competitive performance, which indicates our initial assessments can obtain reasonable results for simple models and datasets. However, FlowX obtains significantly better performance on complex real-world datasets than FlowX*, showing the effectiveness of our proposed learning-based algorithm. More experimental details will be attached in the supplementary material.

4.4 Visualization of Explanation Results

Finally, we report the visualization results of different explanation methods in Figure 3. Specifically, we show the explanations of a molecule graph from ClinTox dataset. The smiles string of this input graph is C(CC(C(F)F)(C(=O)[O-])[NH3+])C[NH3+] and the model to be explained is a 3-layer GCN model. In this input graph, motifs 1 and 2 form a Carboxylate Anion; motif 4 and 5 both contain an Ammoniumyl while motif 3 corresponds to a CF2 group. It is clear that our FlowX finds all motifs in the explanation and obtains the highest Fidelity score. It indicates that those motifs are indeed important for the model to make predictions. In addition, we can conclude that the model combines these motifs for the predictions since we generally observe that explanations with more motifs chosen tend to have higher Fidelity scores. Furthermore, we find that DeepLIFT identifies all motifs but focuses more on the self-loops of atoms. However, its Fidelity score is much lower than GNNExplainer who only identifies four motifs but focuses on the information transmissions between atoms. Hence, we believe the interactions among different atoms contribute more to the model predictions, which further indicates the Shapley value is a promising solution for studying the explainability of GNN models. More visualization results are reported in the supplementary material.

5 Conclusions

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 to explain GNNs by studying message flows. Our FlowX first computes Shapley values approximations as initial assessment and then incorporates a learning refinement. Extensive experiments demonstrate our FlowX achieves significantly improved explanations.

REPRODUCIBILITY STATEMENT

Several efforts are made to ensure the reproducibility of our work. To demonstrate the details of proposed algorithm, we provide the rigorous description of our approach's first stage in Algorithm 1, the second stage learning process in Subsection 3.3 and the concrete setting in appendix A.4. For experiments, besides the detailed description of datasets and evaluation metrics in Section 4, we provide model configurations, training details, target mapping, and metrics details in appendix A and appendix B. Our implementations will be publicly available once the paper is published.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in Neural Information Processing Systems*, 31:9505–9515, 2018.

Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, 2019.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pp. 883–892. PMLR, 2018.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pp. 6967–6976, 2017.

Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. Towards explanation of dnn-based prediction with guided feature inversion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1358–1367, 2018.

Alexandre Duval and Fragkiskos D Malliaros. Graphsvx: Shapley value explanations for graph neural networks. *arXiv preprint arXiv:2104.10482*, 2021.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2083–2092, 2019.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *arXiv preprint arXiv:2001.06216*, 2020.

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

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Harold William Kuhn and Albert William Tucker. *Contributions to the Theory of Games*, volume 2. Princeton University Press, 1953.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9267–9276, 2019.

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 Oztekin, Xuan Zhang, and Shuiwang Ji. DIG: A turnkey library for diving into graph deep learning research. *arXiv preprint arXiv:2103.12608*, 2021.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pp. 4765–4774, 2017.

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, 2020.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. In *Proceedings of The 38th International Conference on Machine Learning*, 2021.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: undefined. https://distill.pub/2018/building-blocks.

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 Conference on Computer Vision and Pattern Recognition*, pp. 10772–10781, 2019.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. 2021.

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, 2020.

Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*, 2017.

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 *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626. IEEE, 2017.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. 2017.

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

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`. accepted as poster.

Minh N Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. 2020.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Fan Yang, Shiva K Pentyala, Sina Mohseni, Mengnan Du, Hao Yuan, Rhema Linder, Eric D Ragan, Shuiwang Ji, and Xia Hu. Xfake: explainable fake news detector with visualizations. In *The World Wide Web Conference*, pp. 3600–3604, 2019.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, pp. 9244–9255, 2019.

Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6412–6422, 2018.

Hao Yuan and Shuiwang Ji. StructPool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Hao Yuan, Yongjun Chen, Xia Hu, and Shuiwang Ji. Interpreting deep models for text analysis via optimization and regularization methods. In *AAAI-19: Thirty-Third AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, 2019.

Hao Yuan, Lei Cai, Xia Hu, Jie Wang, and Shuiwang Ji. Interpreting image classifiers by generating discrete masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020a.

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 and Data Mining*, pp. 430–438, 2020b.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*, 2020c.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. *arXiv preprint arXiv:2102.05152*, 2021.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, volume 18, pp. 4438–4445, 2018.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

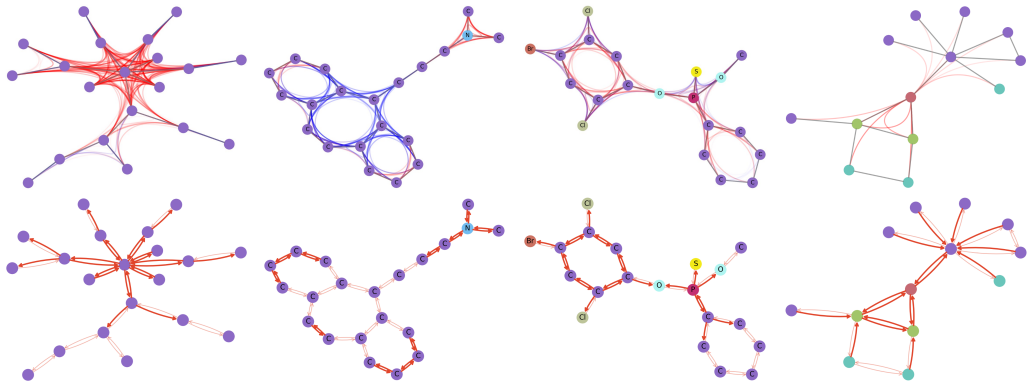

Figure 4: An illustration showing how to convert message flow contribution to edge contribution. The top row shows the the walk view of explanations while bottom row shows the corresponding edge view. In the top row, we use red and blue flows to denote the positive and negative contribution respectively. In the bottom row, bold red arrow lines denote important edges.

A Experimental Settings

A.1 Deep Graph Models

We first introduce the details of the GNN models we try to explain. For all GNN models, we employ two message passing layers for node classification (BA-Shapes (Ying et al., 2019)) and three for graph classification tasks (ClinTox, Tox21, BBBP, BACE (Wu et al., 2018), BA-LRP (Schnake et al., 2020), Graph-SST2 (Yuan et al., 2020c)). In addition, the final classifier consists of two fully-connected layers. For graph classification tasks, the average pooling is used to convert node embeddings to graph embeddings. In addition, the feature dimensions of message passing layers and fully-connected layers are set to 300. We apply the ReLU function as the activation function after each message passing layer and fully-connected layer. In addition, the dropout is applied between two fully-connected layers. Note that we consider both GCNs and GINs for all datasets. For the GCN layer, we employ the original normalized Laplacian matrix $\hat{A}^t = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$. For the GIN layer, two fully-connected layers with the ReLU function are employed as the multilayer perceptron (MLP).

A.2 Training Settings

The models on different datasets use different learning rates and decay settings. Generally, we set the learning rate to 1×10^{-3} and the learning rate decay equal to 0.5 after 500 epochs. For datasets BA-LRP and Graph-SST2, the total number of epochs is 100 while we train models for 1000 epochs on the other datasets. All datasets are split into the training set (80%), validation set (10%), and testing set (10%). All experiments are conducted using one NVIDIA 2080Ti GPU.

A.3 Targets of Explanations

Since different techniques focus on different explainable targets, such as nodes, edges, walks, flows, etc., these methods cannot be directly compared. For fair comparisons, we convert all explainable targets to graph edges. First, GNNExplainer and PGExplainer provide edge-level explanations so their results are directly used. Second, for flow-based methods and walk-based methods, including GNN-GI, GNN-LRP, and our FlowX, we convert the flow (walk) importance to edge importance by summing the total contribution of message flows (walks) that go through a particular edge. In addition, for node-based methods such as GradCAM (Pope et al., 2019) and DeepLIFT (Shrikumar et al., 2017), the node importance is mapped to edge importance that the contribution of an edge is the averaged

contribution of its connected nodes. Specially, for subgraph-based method SubgraphX (Yuan et al., 2021), we pick the explainable subgraph out, then assign the edges in this subgraph instead of nodes as the explanation. Note that the absolute values of contribution are not important since the metrics focus on their relative rankings. The Figure. 4 shows how to map from message flows to edges.

A.4 FlowX Settings

We randomly initialize the weight vector of our learning refinement from the uniform distribution $[0, 0.1]$. During the training, we apply a learning rate equal to 0.3 without learning rate decay and train the weight vector for 500 iterations.

Exponential scaling. Exponential scaling is employed to facilitate training and encourage discreteness. Formally, given a normalized vector $\mathbf{x}$, the scaled result $\mathbf{x}'$ can be expressed as:

$$\mathbf{x}' = \text{Nor}(\mathbf{x}^r), \tag{13}$$

where $\text{Nor}(\cdot)$ denotes the normalization operation; r is the exponent factor. Generally, we find that $r = 8$ is a proper choice.

FlowX* & FlowX†. FlowX* can be implemented by replacing the weight vector w in section 3.3 with a constant vector w' that $w' = 1/|w|$ where $|w|$ is the length of w. FlowX† is implemented with a similar pipeline as GNNExplainer (Ying et al., 2019). Instead of training the weights to obtain the edge mask, we use a flow mask to indicate the flow importance scores. The flow mask can be considered as the weighted flow scores in our FlowX. In addition, we do not apply exponential scaling to FlowX†. The rest of FlowX† is the same as the learning refinement part in our FlowX.

B Evaluation Metrics

Mathematically, a K-class classification dataset with N samples can be represented as $\{((X_i, \Lambda_i), y_i) \mid i = 1, 2, \ldots, N\}$, where (X_i, Λ_i) is the input graph and y_i is the ground-truth label of the i-th sample. The predicted class of a GNN model is $\hat{y}_i = \text{argmax} f(X_i, \Lambda_i)$. Here the output of $f(X_i, \Lambda_i)$ is a K-length vector, in which each element denote the probability of the corresponding class. Formally, we define the algorithm's explanation as a column vector $\dot{m}_i$ that represents a mask on explainable targets where the number of targets is $|\dot{m}_i|$. In this mask, the explanation method marks each selected target as a value 1, otherwise a 0. The source of the mask is a vector $\dot{c}_i$ with the same shape that stores the contribution score of each explainable target. Intuitively, we prefer to assign targets with 1s in the $\dot{m}_i$ if the corresponding scores in $\dot{c}_i$ are relatively high. With these notations, we introduce the metrics employed for quantitative comparison.

Fidelity. Following the metrics mentioned in the graph explainability survey (Yuan et al., 2020c), we choose the metric Fidelity to measure the faithfulness of the explainability methods to the model. Mathematically, Fidelity is defined as:

$$\text{Fidelity} = \frac{1}{N} \sum_{i=1}^{N} (f(X_i, \Lambda_i)_{\hat{y}_i} - f^{\bar{\dot{m}}_i}(X_i, \Lambda_i)_{\hat{y}_i}), \tag{14}$$

where $f^{\dot{m}_i}(X_i, \Lambda_i)_{\hat{y}_i}$ denotes the output class $\hat{y}_i$'s probability when masking out the edges based on $\dot{m}_i$. Specifically, we keep the value unchanged for the explainable targets that are marked as 1 in $\dot{m}$, while setting the other explainable targets as 0. Note that $\bar{\dot{m}}_i = \mathbb{1} - \dot{m}$ is complementary mask where $\mathbb{1}$ is an all-one vector that has the same shape as $\dot{m}$. In the evaluation period, the higher Fidelity indicates the better performance of the explainability algorithm. Intuitively, Fidelity represents the predicted probability dropping after occluding the important features. Therefore, Fidelity can measure whether the chosen features are important to the predictions from the model's perspective.

Sparsity. In order to fairly compare different techniques, we argue that controlling the Sparsity (Pope et al., 2019; Yuan et al., 2020c) of explanations is necessary. Mathematically, Sparsity is defined as:

$$\text{Sparsity} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{\sum_{k=1}^{|\dot{m}_i|} \dot{m}_{ki}}{|\dot{m}_i|} \right). \tag{15}$$

A higher Sparsity score means fewer explainable targets are selected. Note that the selections of targets are determined by the ranking of the contribution score in $\dot{c}_i$. It is noteworthy that these two

Table 3: The averaged time cost of eight algorithms.

	GradCAM	DeepLIFT	GNNExplainer	PGExplainer	GNN-GI	GNN-LRP	SubgraphX	FlowX
Time (ms)	14	15	353	N/A	704	5993	407784	4501

Table 4: Comparisons between FlowX and other methods in terms of average Fidelity over different Sparsity levels on GCNs. **Bold** and underline scores respectively denote the best and the second best results. Note that we do not apply PGMExplainer on node-classification dataset BA-Shapes, thus leave the corresponding place with an N/A. In addition, because SubgraphX cannot control and reach all the Sparsity levels as we need, there is no way to compare it with others in this table.

	BA-Shapes	BA-LRP	ClinTox	Tox21	BBBP	BACE	Graph-SST2
GradCAM	0.3881	0.4968	0.2268	0.0989	0.2977	0.2862	0.1775
DeepLIFT	**0.4230**	0.4203	0.3082	0.1372	0.3652	0.4202	0.2209
GNNExplainer	0.4119	0.4473	0.2294	0.0868	0.3217	0.2403	0.1661
PGExplainer	0.3828	0.4319	0.2029	0.0214	0.1421	0.1044	0.0660
PGMExplainer	N/A	0.3521	0.2454	0.0769	0.2501	0.1651	0.1585
GNN-GI	0.4029	0.4462	0.3070	0.1612	0.4272	0.2851	0.3089
GNN-LRP	0.4109	**0.5151**	0.3159	0.1350	0.3535	0.2541	0.2134
FlowX	0.4198	0.5145	**0.3843**	**0.2243**	**0.5657**	**0.5146**	**0.3217**

metrics are highly correlated since the predictions tend to change more when more input features are removed. Then explanations with higher Sparsity scores tend to have lower Fidelity scores. For fair comparisons, we compare different methods with similar sparsity levels.

C Experimental Results

In this section, we provide additional quantitative comparisons and visualization results.

C.1 Quantitative Results

We report the quantitative results of GCN and GIN models in Table 4 and Table 5. Obviously, our method obtains the best averaged Fidelity score for 5 out of 7 GCN models. Note that the performance is competitive for synthetic dataset BA-Shapes and BA-LRP and our method performs significantly better on the complex real-world datasets. In addition, we show the plots of Fidelity scores with respect to different Sparsity levels for all datasets and GCN models in Figure 5. Clearly, the performance of our method is stable and better.

Accuracy comparisons: To further demonstrate the effectiveness of our proposed method, we conduct experiments to compare the explanatory accuracy of three methods that usually evaluated in terms of accuracy. The results are reported in Table 6. These results are obtained from a synthetic graph classification dataset BA-INFE. Each graph includes a base BA-graph and four types of motifs. Two classes of motifs are attached (2 motifs denote the one property, 2 for another; let's denote them as class/property A and B). We first connect the same number (1 3) of each class's motifs to the base graph. Then we attach 1 to 3 extra motifs from one of the classes (for example, class A) to the base graph; thus, the number motifs in class A is more than class B, representing the graph tends to have the specific property (property A). The ground-truths of a graph are the elements in those motifs including edges (undirected) and self-loop connections. The number of ground-truth elements that are covered by explanation edges is denoted as the hit number. The accuracy is defined as (hit number) / (total number of ground-truth elements).

We also show the time cost of different methods in Table 3. Specifically, we report the averaged computation time for 40 graphs in the ClinTox (Wu et al., 2018) dataset. It shows that our FlowX performs the best with a reasonable time cost.

Table 5: Comparisons between FlowX and other methods in terms of average Fidelity over different Sparsity levels on GINs. **Bold** and underline scores respectively denote the best and the second best results. Note that we do not apply PGMExplainer on node-classification dataset BA-Shapes, thus leave the corresponding place with an N/A. In addition, because SubgraphX cannot control and reach all the Sparsity levels as we need, there is no way to compare it with others in this table.

	BA-Shapes	BA-LRP	ClinTox	Tox21	BBBP	BACE	Graph-SST2
GradCAM	0.2194	0.3836	0.3388	0.1258	0.2366	0.4715	0.1497
DeepLIFT	0.5634	0.3772	0.4372	0.1212	0.2122	0.4822	0.1953
GNNExplainer	0.5331	0.3789	0.3485	0.0101	0.2187	0.4140	0.1200
PGExplainer	0.3091	0.3155	0.1696	0.0095	0.0710	0.1732	0.0501
PGMExplainer	N/A	0.2206	0.3096	0.0666	0.1809	0.3764	0.1370
GNN-GI	0.6308	0.1901	0.5056	0.1600	0.2784	0.5013	0.2716
GNN-LRP	0.6671	0.3751	0.4944	0.1458	0.2406	0.4104	0.2552
FlowX	**0.6928**	**0.4613**	**0.5463**	**0.2308**	**0.4676**	**0.6715**	**0.2918**

Table 6: Comparisons between FlowX and three methods in terms of average Accuracy with 0.9 Sparsity on GCNs.

	GNNExplainer	PGExplainer	PGMExplainer	FlowX
GCN	0.2528	0.2199	0.3612	0.5266
GIN	0.2917	0.2143	0.3748	0.4265

C.2 Explanation Visualizations

We provide additional examples to visually compare the explanations by six methods. The results are reported in Figure. 6, and 7. We can conclude that the message interactions between nodes are more important than self-loops. For example, in Figure. 6, all of DeepLIFT, GNNExplainer, and FlowX can identify the motifs; however, FlowX attributes more on the node interactions and obtain a higher Fidelity score.

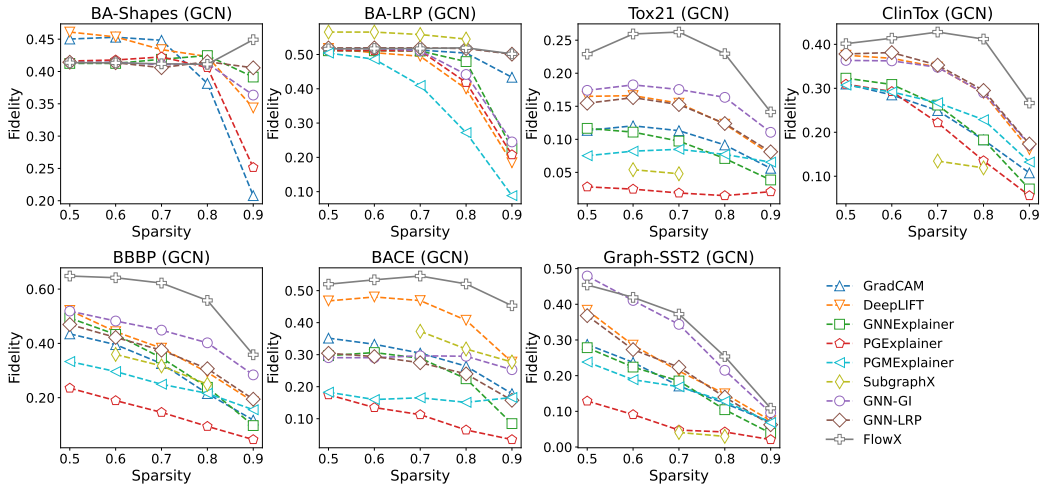

Figure 5: Comparison of Fidelity values by nine methods on seven datasets with GCNs under different Sparsity levels.

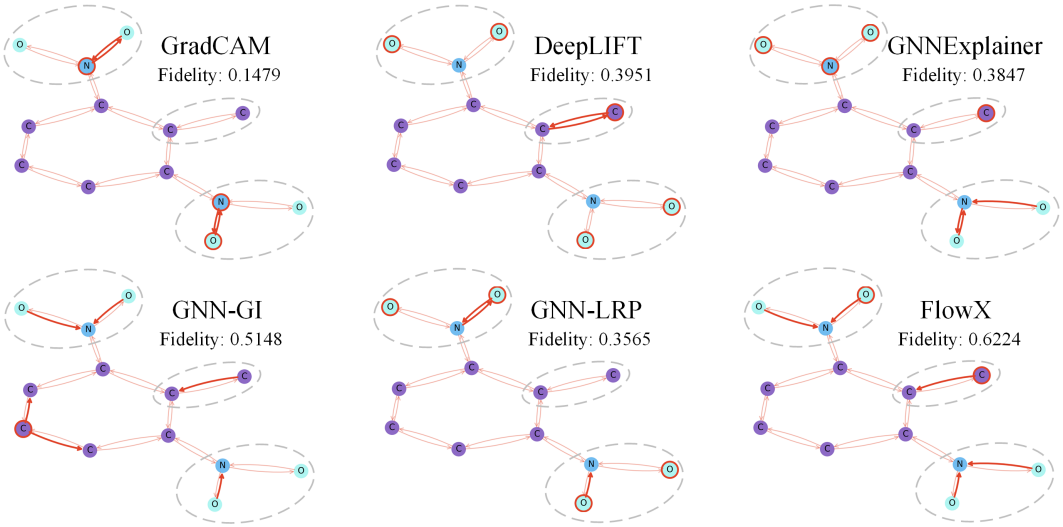

Figure 6: Sample explanation results of different methods on Tox21.

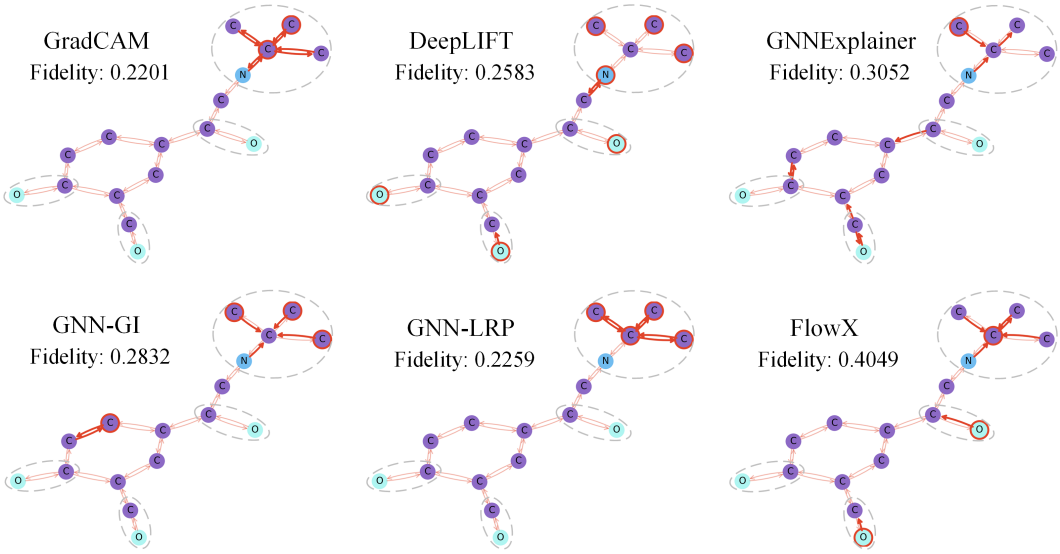

Figure 7: Sample explanation results of different methods on ClinTox.