Efficient Higher-order Subgraph Attribution via Message Passing

Ping Xiong [1] **Thomas Schnake** [1 2] **Grégoire Montavon** [1 2] **Klaus-Robert Müller** [1 2 3 4] **Shinichi Nakajima** [1 2 5]

Abstract

Explaining graph neural networks (GNNs) has become more and more important recently. Higher-order interpretation schemes, such as GNN-LRP (layer-wise relevance propagation for GNN), emerged as powerful tools for unraveling how different features interact thereby contributing to explaining GNNs. GNN-LRP gives a relevance attribution of walks between nodes at each layer, and the subgraph attribution is expressed as a sum over exponentially many such walks. In this work, we demonstrate that such exponential complexity can be avoided. In particular, we propose novel algorithms that enable to attribute subgraphs with GNN-LRP in linear-time (w.r.t. the network depth). Our algorithms are derived via message passing techniques that make use of the distributive property, thereby directly computing quantities for higher-order explanations. We further adapt our efficient algorithms to compute a generalization of subgraph attributions that also takes into account the neighboring graph features. Experimental results show the significant acceleration of the proposed algorithms and demonstrate the high usefulness and scalability of our novel generalized subgraph attribution method.

1. Introduction

In recent years, there has been an increasing interest in Graph Neural Networks (GNNs) because of their ability to incorporate the intrinsic structure of data and their state-of-the-art performance on graph-structured data, e.g., social networks (Chen et al., 2018; Hamilton et al., 2017; Kipf & Welling, 2017) and molecules (Schütt et al., 2018; Domingue et al., 2019). The learning tasks on graphs include node, edge or graph classification, link prediction and others (Wu et al., 2021; Hu et al., 2020). However, since the prediction strategy of a GNN is in general not comprehensible for humans, GNN models are still treated as black-boxes, which prevents applications in some crucial areas where trustworthiness or safety is required. In the recent literature, various methods for explaining GNNs have been developed (Yuan et al., 2020b). Methods like GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020) and PGM-explainer (Vu & Thai, 2020) allow importance analysis of nodes and edges within the input data sample. GNN-LRP (layer-wise relevance propagation for GNN; Schnake et al. (2021)) aims at explaining GNNs at the level of *walks*, which reflect the practically relevant higher-order interactions of features. To obtain such walk relevances, higher-order deep Taylor decomposition is applied to a GNN, from which we get independent feature components that only depend on bag-of-edges. These bag-of-edge components are then recovered by backward massages from one node representation to the next within the GNN interaction layers, and

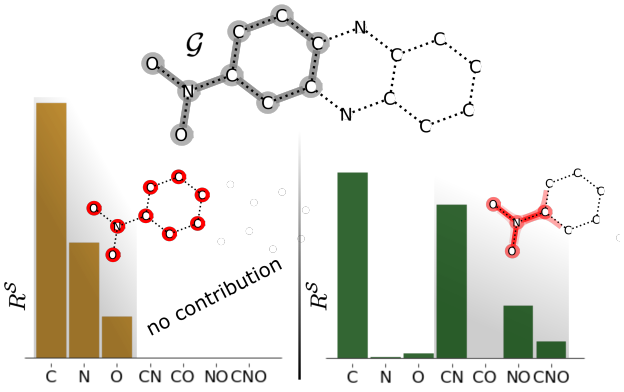


Figure 1: Visualisation for the joint contribution of atom subsets when predicting the mutagenicity of a molecule. The gray sub-molecule $\mathcal{G}$ is a strong indicator for the mutagenicity of the full molecule. The orange and green bars show the lower- and higher-order relevance scores for the subset composed of all atoms which have the denoted atomic number, respectively. The heat maps in the sub-molecules, close to the orange and green bars, show the relevance scores of lower- and higher-order attribution methods, respectively.

[1]Technische Universität Berlin (TU Berlin) [2]BIFOLD – Berlin Institute for the Foundations of Learning and Data [3]Department of Artificial Intelligence, Korea University, Seoul 136-713, Korea [4]Max Planck Institut für Informatik, 66123 Saarbrücken, Germany [5]RIKEN Center for AIP, Japan. Correspondence to: Shinichi Nakajima <nakajima@tu-berlin.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

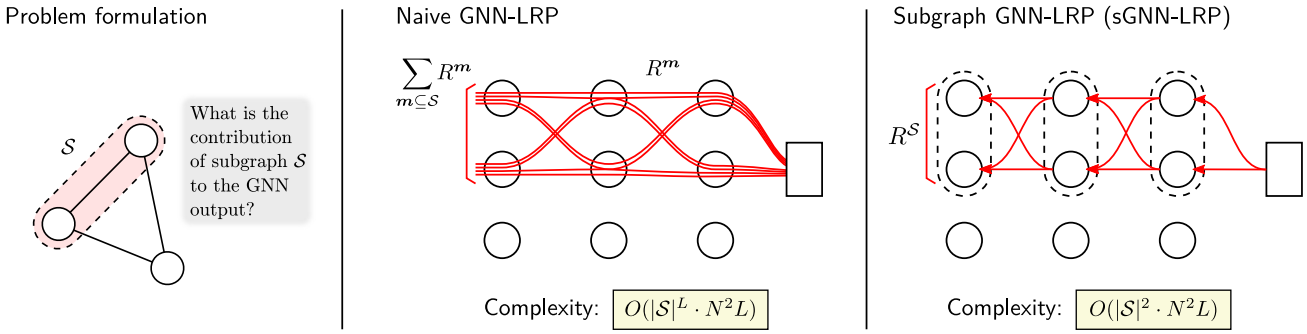


Figure 2: Computation of the subgraph relevance (left). A naive implementation of GNN-LRP computes the sum of the relevances over exponentially many walks (middle), while our proposed subgraph GNN-LRP aggregates the contributions *layer-wise* (right), allowing linear time computation with respect to the network depth L.

provide the relevance resolution of walks (Schnake et al., 2021).

Higher-order interpretation methods can give important insights into the prediction strategies employed by neural networks. In Figure 1 we see how lower- and higher- order interpretation methods assign contributions to different subsets of atoms when predicting the mutagenicity of molecules (Kazius et al., 2005b). Using the higher order scheme, we find that the NO_2 group combined with the neighboring aromatic carbon ring (highlighted in gray in Figure 1), is a strong indicator for mutagenicity. This finding is also supported by experimental studies (Kazius et al., 2005a). The lower order attribution method is unable to capture the joint contributions of atoms, and therefore have to rely on single atom contributions exclusively, inappropriately assuming that the individual atoms contribute to the mutagenicity of the molecule. On the other hand, the attribution scheme that takes into account the higher-order structure of the model reflects how sets of different atoms, particularly the combination of C and N, or N and O atoms, have a strong joint contribution to the prediction task. This aligns with chemical intuition in general, since the property of mutagenicity is not due to the effect of single atoms but arises from the higher-order interactions present in the highlighted functional group. More details can be found in Appendix A.

Although GNN-LRP has shown to be highly effective in interpreting GNNs with respect to feature interactions, its computation was so far limited to relatively small graphs and shallow networks, due to the exponential complexity with respect to the network depth. We would like to note that this complexity issue is not specific to GNN-LRP but rather it is present for general higher-order feature attribution methods beyond additive or linear explanation (Lundberg & Lee, 2017; Samek et al., 2021). This is because L-th order interpretation methods for N input features need to take N^L different feature combinations into account. In the task of extracting the joint relevance of a collection of graph features, namely the relevance of a subgraph in the input data point (Schnake et al., 2021; Yuan et al., 2021), the incorporation of higher-order feature attribution is essential, yet without finding an efficient way of computation that remedies the exponential complexity, it is unfeasible for the general case even with moderately large L.

In this work, we propose a novel propagation rule, called subgraph GNN-LRP (sGNN-LRP), that directly computes the relevance of a subgraph in a single backpropagation pass. Comparing with a naive application of GNN-LRP that sums up walk relevances, the computational complexity reduces from exponential to linear with respect to the network depth L (see Figure 2). The forward-hook trick (Schnake et al., 2021; Samek et al., 2021) allows a simple, fast, and less memory intensive implementation of sGNN-LRP.

A novel aspect of this work also exists in the way of developing the new propagation rule: sGNN-LRP is derived as a sum-product message passing algorithm, a.k.a., belief propagation (Bishop, 2006; Pearl, 1982), to compute an explicitly defined target quantity—the sum of relevances of all walks that stay within a given subgraph. We explain why message passing is applicable to the relevance computation by pointing out its mathematical similarity to the marginal probability computation of a Markov chain process, and discuss its generality by deriving existing LRP rules as message passing algorithms.

The message passing framework allows us to easily adapt the propagation rule to another target quantity. We demonstrate this benefit by deriving a variant of sGNN-LRP for a generalized definition of the subgraph relevance, which takes into account the walks outside the subgraph with discounted contributions according to how many times the walk steps out of the subgraph. Our experiments show that our generalized subgraph relevance quantitatively improves the explanation of GNNs in terms of node-ordering performance.

Table 1: Notation.

$h, \mathbf{h}, \mathbf{H}, H_{m,m'}$	scalar, vector, matrix, matrix entry
$m_{l:l'}$	partial vector with indices $(l, \ldots, l')$
$\mathcal{G}$ and $\mathcal{S}$	graph and subgraph
$\mathbf{m}$ and $\mathbf{n}$	sequence of nodes and neurons
m, m_l	integers for node identifications
n, n_l	integers for neuron identifications
R, $\mathbf{r}$	relevance
$\check{\mathbf{r}}$	propagated relevance, message, or belief
$\mathbf{T}$	propagation matrix

2. Background and Related Work

2.1. Graph Neural Networks

Graph Neural Networks (GNNs) (Scarselli et al., 2009; Wu et al., 2021) is a class of neural networks that receive a graph as an input. In a GNN, node embeddings are learned in multiple *interaction blocks*, where the interaction is defined by the given graph. In most GNN architectures, the interaction block can be divided into *aggregate* and *combine* steps (Gilmer et al., 2017), which can be expressed by

$$\begin{aligned} \text{Aggregate: } \mathbf{Z}^{(l)} &= \mathbf{\Lambda}\mathbf{H}^{(l-1)}, \\ \text{Combine: } \mathbf{H}^{(l)} &= \mathcal{C}^{(l)}\left(\mathbf{Z}^{(l)}\right). \end{aligned} \tag{1}$$

Here $\mathbf{H}^{(l)} \in \mathbb{R}^{M \times N^{(l)}}$ denotes the feature (activation) matrix of the l-th layer, which consists of the $N^{(l)}$-dimensional feature vectors for all M nodes (notation is summarized in Table 1). In the aggregate step, the features $\mathbf{H}^{(l-1)}$ from the last layer are aggregated using a modified (e.g., normalized with self-loops) adjacency matrix $\mathbf{\Lambda} \in \mathbb{R}^{M \times M}$ and stored in $\mathbf{Z}^{(l)} \in \mathbb{R}^{M \times N^{(l-1)}}$. In the combine step, a non-linear function $\mathcal{C}^{(l)} : \mathbb{R}^{N^{(l-1)}} \mapsto \mathbb{R}^{N^{(l)}}$ is applied to $\mathbf{Z}^{(l)}$ column-wise, to transform $\mathbf{Z}^{(l)}$ into the new node features $\mathbf{H}^{(l)}$ for this layer. For the combine function, common choices are a one-layer perceptron $\mathcal{C}^{(l)}(\mathbf{Z}) = \sigma(\mathbf{Z}\mathbf{W}^{(l)})$ with trainable weights $\mathbf{W}^{(l)} \in \mathbb{R}^{N^{(l-1)} \times N^{(l)}}$ and a non-linear activation $\sigma(\cdot)$ as in Graph Convolution Network (GCN) (Kipf & Welling, 2017), and a multilayer perceptron (MLP) as in Graph Isomorphism Network (GIN) (Xu et al., 2019). After the node feature of the last layer is computed, a read-out function (e.g., average or maximum over all nodes, followed by an MLP) is applied to generate the required graph-level predictions, depending on the prediction task.

2.2. Explainability for Graph Neural Networks

In recent years multiple explainability methods for GNNs have been proposed. GNNExplainer (Ying et al., 2019) and PGExplainer (Luo et al., 2020) explain GNNs by finding masks that maximize the mutual information between the predictions of the original graph and a masked graph. GNNExplainer learns soft masks for edges or node features, while PGExplainer trains a parametric predictor to determine if an edge should be masked out. The predicted mask is an approximate discrete mask and is known to alleviate the 'introduced evidence' problem that soft mask faces. Such masks can be used to extract the most important subgraphs, for example, we can identify the subgraph consisting of all nodes with soft mask values above a threshold as the most important subgraph. PGM-Explainer (Vu & Thai, 2020) trains a well explainable probabilistic graphical model (PGM) as a surrogate method of the GNN, and then substitute the explanation of the GNN with the explanation of the PGM. Unlike all the instance-level methods above, XGNN (Yuan et al., 2020a) is a model-level explainer, which generates a representative graph of every target class using reinforcement learning.

Most explainability techniques (Pope et al., 2019; Ying et al., 2019; Luo et al., 2020) for GNNs explains the model at the level of nodes, edges and node features, while a few of them, including SubgraphX (Yuan et al., 2021) and GNN-LRP (Schnake et al., 2021), analyze the relevance of subgraphs as higher-order features. SubgraphX identifies the most important subgraphs based on the Shapley value (Lundberg & Lee, 2017) with Monte-Carlo Tree Search (MCTS). The Shapley value is computed by perturbing the input graph and comparing the change of the model output. GNN-LRP is an LRP-based method, which scores bag-of-edges by decomposing and backpropagating the output to the input layer. Since our proposed algorithm is based on GNN-LRP, we give its detailed description in the next subsection.

Unlike the above methods, GNES (Gao et al., 2021) provides a general framework that trains the GNN model and optimizes the explanation model simultaneously with regularizations, so that the explanation is reasonable and stable. GNES can handle many attribution methods including Gradient-based, Grad-CAM and LRP.

2.3. GNN-LRP

GNN-LRP (Schnake et al., 2021) aims to explain the prediction strategy of GNNs with respect to higher-order feature interactions, by tracking the information that were passed through the internal dependency graph for making a prediction. The basic unit of explanation is therefore the relevance of a *walk*, which is defined as an ordered sequence of nodes connected from layer to layer. Assume that the whole graph $\mathcal{G}$ consists of M nodes. Then, a walk can be denoted by $\mathbf{m} \in \mathbb{M}$ with $\mathbb{M} = \{1, \ldots, M\}^{L+1}$, meaning that the walk starts from the m_0-th node at the input layer, goes through the m_l-th node at the l-th layer, and reaches the m_L-th node in the last layer. We also denote a partial walk by $m_{l:l'}$ for $0 \leq l < l' \leq L$. We identify a node and its index, and

denote, e.g., by $m \in \mathcal{G}$ that the node m is a member of $\mathcal{G}$.

The GNN-LRP rule for a simple GCN (Kipf & Welling, 2017) with the combine function $\mathcal{C}^{(l)}(\mathbf{Z}) = \text{ReLU}\left(\mathbf{Z}\mathbf{W}^{(l)}\right)$ in Eq.(1) is given as

$$\breve{\mathbf{r}}^{(l,m_l)} = \mathbf{T}^{l,m_l,m_{l+1}} \breve{\mathbf{r}}^{(l+1,m_{l+1})}. \tag{2}$$

Here $\breve{\mathbf{r}}^{(l,m_l)} \in \mathbb{R}^{N^{(l)}}$ is the *propagated relevance* at the node m_l in the l-th layer, and $\mathbf{T}^{l,m,m'} \in \mathbb{R}^{N^{(l)} \times N^{(l+1)}}$ is the propagation matrix whose entries are given as

$$T^{l,m,m'}_{n,n'} = \frac{\Lambda_{m,m'} H^{(l)}_{m,n} W^{(l)\uparrow}_{n,n'}}{\sum_{m'',n''} \Lambda_{m'',m'} H^{(l)}_{m'',n''} W^{(l)\uparrow}_{n'',n'}}.$$

Here $\mathbf{W}^{\uparrow}$ is a modified weight parameter depending on the choice of LRP rules (Bach et al., 2015; Montavon et al., 2018; Samek et al., 2021), e.g., $\mathbf{W}^{\uparrow} := \mathbf{W} + \gamma \cdot \max(0, \mathbf{W})$ for the LRP-γ rule with $\gamma \geq 0$, where the max operator applies entry-wise. Note that we mostly use subscripts to specify the entry of a matrix or vector, while superscripts for distinguishing different matrices or vectors. Note also that the propagated relevance $\breve{\mathbf{r}}^{(l,m_l)}$ does not denote a particular quantity but a variable that depends on the propagation rule.

The GNN-LRP rule depends on the network structure, and we refer to Schnake et al. (2021) for the GNN-LRP rules for other GNN variants.

3. Efficient Computation of Subgraph Attribution

Understanding the relevance contribution of subgraphs in the input graph to the model prediction is a key challenge when explaining models on graphs (Yuan et al., 2021; Luo et al., 2020; Schnake et al., 2021). As a higher-order interpretation method, Schnake et al. (2021) proposed a definition of subgraph relevance as the sum over relevance scores of all walks inside the subgraph $\mathcal{S} \subseteq \mathcal{G}$, i.e.,

$$R^{\mathcal{S}} = \textstyle\sum_{\mathbf{m} \subseteq \mathcal{S}} R^{\mathbf{m}}. \tag{3}$$

Here, with slight abuse of notation, we denote by $\mathbf{m} \subseteq \mathcal{S}$ that the walk $\mathbf{m}$ stays inside $\mathcal{S}$, i.e., $m_l \in \mathcal{S}$ for all $l = 0, \dots, L$. Unfortunately, performing the sum in Eq.(3), which we call a *naive* application of GNN-LRP for subgraph attribution (Naive GNN-LRP), over exponentially many $\sim \mathcal{O}(|\mathcal{S}|^L)$ walks is limited to small subgraphs (nodes or edges) for state-of-the-art GNNs as they are typically deep.

However, we will show in Section 4 that there is a much more efficient way of achieving subgraph attribution: the exponential sum in Eq.(3) can be computed by $R^{\mathcal{S}} = \sum_{m_0 \in \mathcal{S}} \sum_{n=1}^{N^{(0)}} \breve{r}^{(0,m_0)}_n$ after applying a single pass of our novel subgraph GNN-LRP (sGNN-LRP) rule

$$\breve{\mathbf{r}}^{(l,m_l)} = \textstyle\sum_{m_{l+1} \in \mathcal{S}} \mathbf{T}^{l,m_l,m_{l+1}} \breve{\mathbf{r}}^{(l+1,m_{l+1})} \tag{4}$$

for $l = L-1, \dots, 0$. Since a single backward pass directly yields the subgraph attribution, the computational advantage is evident and massive, as shown graphically in Figure 2. Moreover, applying the *forward-hook* trick (Schnake et al., 2021; Samek et al., 2021) to sGNN-LRP gives a simpler, faster, and less memory intensive implementation than directly implementing the sGNN-LRP rule (see Section 5).

Our novel sGNN-LRP rule (4) only differs from the GNN-LRP rule (2) for computing the walk relevance, by introducing an additional summation that pools the propagated relevances over all nodes belonging to the subgraph $\mathcal{S}$. However, we emphasize that we obtained this new rule by a novel, systematic procedure: we explicitly define the target quantity to be computed, and then derive a propagation rule as a message passing algorithm. We detail this procedure and its generality in Section 4.

4. LRP as Message Passing

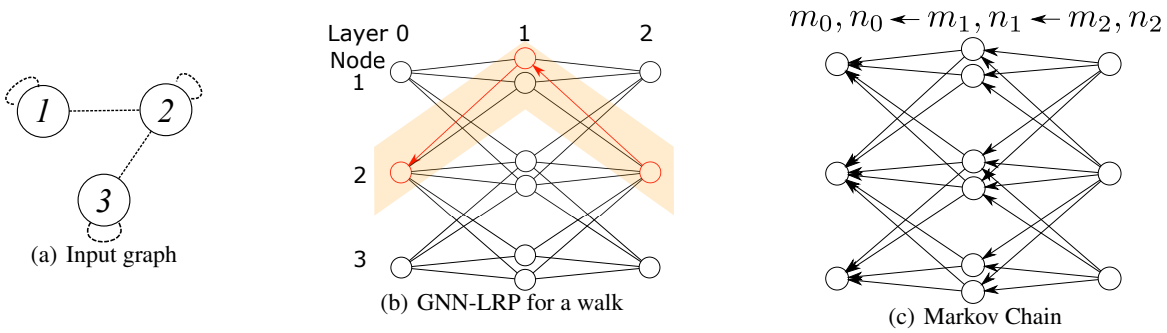
In this section, we first point out the similarity between the relevance computation and the marginal probability computation of a Markov chain, which implies the applicability of *sum-product* (a.k.a. belief propagation) algorithm (Bishop, 2006; Pearl, 1982), a message passing algorithm for marginalizing over random variables. Then, we apply the sum-product decomposition to the target quantity (3), and derive our sGNN-LRP rule (4). We also discuss the generality of our approach, and derive existing LRP rules. This novel procedure allows us to systematically derive new propagation rules by defining or modifying the target quantity, which will be further demonstrated in Section 6.

Let us define the relevance of a partial walk $m_{l:l'}$ as the sum of relevances of all walks going through the specified nodes from the l-th to the l'-th layers, i.e.,

$$R^{m_{l:l'}} = \textstyle\sum_{\mathbf{m}' \in \mathbb{M}: m'_{l:l'} = m_{l:l'}} R^{\mathbf{m}'},$$

and its neuron-level counterpart $\mathbf{r}^{l,m_{l:l'}} \in \mathbb{R}^{N^{(l)}}$ whose entry $r^{l,m_{l:l'}}_n$ is the relevance of a partial walk limited to a particular neuron specified by n at the l-th layer. We denote by $m_{l:l'} \subseteq \mathcal{S}$ that the partial walk is within the subgraph, i.e., $m_{l''} \in \mathcal{S}$ for $l'' = l, \dots, l'$, and by $\mathbf{r}^{l,m_l,m_{l+1:L} \subseteq \mathcal{S}}$ the sum of relevances of the partial walks that go through the node m_l at layer l and any node in $\mathcal{S}$ at layers $l+1, \dots, L$.

GNNs can be unfolded into a feed forward neural network (FFNN) (Fig.3(b)). Based on the unfolded network, consider a virtual stochastic system where a particle exists at the L-th layer at time $t = 0$, stays at the l-th layer at time $t = L - l$, and arrives at the input layer at time $t = L$. At each layer, the particle stochastically chooses a particular neuron in a particular node. Then, its trajectory can be denoted by $(\mathbf{m}, \mathbf{n})$, where $\mathbf{n} \in \mathbb{N} \equiv \{1, \dots, \max_l(N^{(l)})\}^{L+1}$ specifies the choice of neuron at each layer. Assume that



(a) Input graph (b) GNN-LRP for a walk (c) Markov Chain

Figure 3: LRP as message passing. (a) An example graph consisting of 3 nodes (with self-connections). (b) Unfolded feed-forward network of a 2-layer GCN (with the node feature dimension in each layer being $N^{(0:2)} = [1, 2, 1]$). A walk ($\mathbf{m} = [2, 1, 2]$) is marked with yellow background, and a neuron-level walk ($\mathbf{m} = [2, 1, 2], \mathbf{n} = [1, 1, 1]$) is marked as red arrows. (c) A Markov chain process of which the joint distribution has the same decomposable structure as the relevance of a neuron-level walk. This implies that message passing techniques for computing various marginal probabilities of Markov chain can be used for computing the sum of relevances over various sets of walks.

the probability of the choice of node-neuron pair at the l-th layer only depends on the choice at the $(l+1)$-th layer. Then, the joint probability can be written as

$$p(\mathbf{m}, \mathbf{n}) = \left(\prod_{l=0}^{L-1} p(m_l, n_l | m_{l+1}, n_{l+1})\right) p(m_L, n_L), \tag{5}$$

which is a simple Markov chain (Fig.3 (c)). If we formally assume that $p(m_l, n_l | m_{l+1}, n_{l+1}) = T^{l,m_l,m_{l+1}}_{n_l,n_{l+1}}$ and $p(m_L, n_L) = r^{L,m_L}_{n_L}$, the joint distribution (5) coincides with the relevance of a *neuron-level* walk, a walk specifying not only the node but also the neuron inside the node at each layer:

$$R^{\mathbf{m},\mathbf{n}} = \left(\prod_{l=0}^{L-1} T^{l,m_l,m_{l+1}}_{n_l,n_{l+1}}\right) r^{L,m_L}_{n_L} = p(\mathbf{m}, \mathbf{n}). \tag{6}$$

Importantly, the relevance has the same decomposable structure as the joint distribution of the Markov chain. Therefore, we can use the sum-product algorithm—which allows efficient computation for various marginal distributions of a Markov chain—for computing relevances that require summation over different sets of walks.

Since the propagation matrices $\{\mathbf{T}^{l,m,m'}\}$ and the partial walk relevance $\mathbf{r}^{L,m_L}$ are not probabilities, they can have negative entries, and *not* necessarily normalized. These differences do not affect the applicability of the sum-product decomposition, and we can derive LRP rules as message passing for any propagation matrices and any definition of relevance. However, we restrict our theoretical analysis in this paper to the case where the propagation matrix is normalized, i.e., $\sum_{n,m} T^{l,m,m'}_{n,n'} = 1 \ \forall n', m'$ for simplicity. This allows us to precisely and concisely describe what quantity is carried by the propagated relevance $\breve{\mathbf{r}}^{(l,m_l)}$ (which corresponds to the message/belief in the terminology for message passing/belief propagation), and makes the derived LRP rules transparent. For unnormalized propagation matrices, the messages are scaled by layer-dependent constants, which are practically irrelevant and (if necessary) can be computed by another pass of messages.

Now let us derive our sGNN-LRP rule (4). Setting Eq.(3) as the target quantity,[1] we apply the sum-product decomposition, and obtain the following theorem (the proof is given in Appendix B):

Theorem 1 *Assume that the sGNN-LRP rule* (4) *is applied for* $l = L-1, \dots, 0$ *with the initial message* $\breve{\mathbf{r}}^{(L,m_L)} = \mathbf{r}^{L,m_L}$. *Then,* $\breve{\mathbf{r}}^{(l,m_l)} = \mathbf{r}^{l,m_l,m_{l+1:L} \subseteq \mathcal{S}} \ \forall l \in \{0, \dots, L\}$, *and* $R^{\mathcal{S}} = \sum_{m_0 \in \mathcal{S}} \sum_{n=1}^{N^{(0)}} r^{0,m_0,m_{1:L} \subseteq \mathcal{S}}_{n}$.

Our novel procedure—deriving LRP rules as message passing for computing explicitly defined target quantities—is general, and one can derive many existing LRP rules by defining the corresponding target values, as summarized in Table 2 (see Appendices C and D for derivations, and Appendix E for the same rules using the notation of Schnake et al. (2021)). This procedure allows systematic derivation of propagation, and will facilitate future development of LRP methods.

5. Forward-hook Trick

We can implement our sGNN-LRP by slightly modifying the forward-hook trick (Schnake et al., 2021; Samek et al., 2021), developed for the original GNN-LRP (see Appendix F). We implement the forward combine step in Eq.(1) as

$$\begin{aligned} \mathbf{P}^{(l)} &\leftarrow \mathbf{Z}^{(l)} \mathbf{W}^{(l)\uparrow}, \\ \mathbf{Q}^{(l)} &\leftarrow \mathbf{P}^{(l)} \odot [\sigma(\mathbf{Z}^{(l)} \mathbf{W}^{(l)}) \oslash \mathbf{P}^{(l)}]_{\mathtt{cst.}}, \\ \mathbf{H}^{(l)} &\leftarrow \mathbf{Q}^{(l)} \odot \mathbf{M}^{(l)} + [\mathbf{Q}^{(l)}]_{\mathtt{cst.}} \odot (1 - \mathbf{M}^{(l)}), \end{aligned} \tag{7}$$

[1] The subgraph relevance (3) corresponds to the marginal probability that a particle never steps out of the subgraph $\mathcal{S}$ in the virtual Makov chain process shown in Fig.3(c).

Table 2: Target quantities and the corresponding propagation rules derived as message passing. The propagation matrices $\{\mathbf{T}^l \in \mathbb{R}^{N^{(l)} \times N^{(l+1)}}\}$ can be arbitrary, and therefore this table applies for all ($\epsilon, \gamma, \alpha\beta$, etc.) propagation matrices.

	Target quantity	Propagation rule
LRP for general FFNN	$\mathbf{r}^0 = (\prod_{l=0}^{L-1} \mathbf{T}^l)\mathbf{r}^L$	$\breve{\mathbf{r}}^{(l)} = \mathbf{T}^l \breve{\mathbf{r}}^{(l+1)}$
GNN-LRP	$R^{\mathbf{m}}$	$\breve{\mathbf{r}}^{(l,m_l)} = \mathbf{T}^{l,m_l,m_{l+1}} \breve{\mathbf{r}}^{(l+1,m_{l+1})}$
sGNN-LRP	$R^{\mathcal{S}} = \sum_{\mathbf{m} \subseteq \mathcal{S}} R^{\mathbf{m}}$	$\breve{\mathbf{r}}^{(l,m_l)} = \sum_{m_{l+1} \in \mathcal{S}} \mathbf{T}^{l,m_l,m_{l+1}} \breve{\mathbf{r}}^{(l+1,m_{l+1})}$
Generalized sGNN-LRP	$\widetilde{R}_\alpha^{\mathcal{S}} = \sum_{\mathbf{m} \subseteq \mathcal{G}} \alpha^{\sum_{l=0}^{L} \mathbb{1}(m_l \notin \mathcal{S})} R^{\mathbf{m}}$	$\breve{\mathbf{r}}^{(l,m_l)} = \sum_{m_{l+1} \in \mathcal{G}} \alpha^{\mathbb{1}(m_{l+1} \notin \mathcal{S})} \mathbf{T}^{l,m_l,m_{l+1}} \breve{\mathbf{r}}^{(l+1,m_{l+1})}$

where $\mathbf{P}^{(l)}, \mathbf{M}^{(l)} \in \mathbb{R}^{M \times N}$, and $\odot$ and $\oslash$ denote the entry-wise multiplication and division, respectively. The operator $[\cdot]_{\texttt{cst}}$. *detaches* the quantity to which it applies from the gradient computation. Then, the target quantity can be computed by the `Autograd` function:

Theorem 2 *Assume that we applied a complete forward prediction with the modified combine step* (7) *with the constant mask matrix $\mathbf{M}^{(l)} = \mathbf{M}^{\mathcal{S}}$ for all l, where $\mathbf{M}^{\mathcal{S}}$ masks the columns that correspond to the nodes outside the subgraph, i.e., the m-th columns for $m \in \mathcal{S}$ are all-one vectors, and the other columns are all-zero vectors. Then, we get $R^{\mathcal{S}} = \left\langle \texttt{Autograd}(y, \mathbf{H}^{(0)}), \mathbf{M}^{\mathcal{S}} \mathbf{H}^{(0)} \right\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product.*

The proof (given in Appendix G) is similar to the one for the relevance of walk in Schnake et al. (2021). This implementation is simpler, faster, and less memory intensive than the direct implementation of the sGNN-LRP rule (4).

6. Generalized Subgraph Attribution

In this section, we propose a novel definition of subgraph attribution by generalizing Eq.(3), and derive the corresponding LRP rule. The proposed definition itself will be shown to be useful in Section 7, and our derivation of a new propagation rule demonstrates the utility of the message passing framework, introduced in Section 4.

We consider the following two properties important to fulfill for subgraph attributions: A subgraph $\mathcal{S}$ is important if and only if

1. the model makes almost the same predictions for the input graphs $\mathcal{G}$ and $\mathcal{S}$, and
2. the model predictions for its complement $\mathcal{G} \setminus \mathcal{S}$ and the full input graph $\mathcal{G}$ diverge drastically.

However, the original definition (3) of the subgraph attribution completely ignores the walks that step out of the subgraph even only once, and thus only considers the first property.

We propose a generalized version of subgraph attribution that trades-off both properties with a *discounting* parameter $\alpha \in [0, 1]$:

$$R_\alpha^{\mathcal{S}} = \sum_{\mathbf{m} \in \mathcal{G}} g_\alpha^{\mathcal{S}}(\mathbf{m}) R^{\mathbf{m}}, \tag{8}$$

where

$$g_\alpha^{\mathcal{S}}(\mathbf{m}) = \begin{cases} 0 & \text{if } m_l \notin \mathcal{S},\ \forall\, l = 0, \dots, L, \\ \alpha^{\sum_{l=0}^{L} \mathbb{1}(m_l \notin \mathcal{S})} & \text{otherwise.} \end{cases} \tag{9}$$

Here we used the indicator function $\mathbb{1}(\cdot)$ equal to one if the event is true and zero otherwise. The generalized subgraph attribution (8) counts all walks that go through a node in $\mathcal{S}$ at least once with their discounted contributions according to how many times the walk steps outside $\mathcal{S}$. For $\alpha = 0$, it reduces to the original definition, i.e., $R^{\mathcal{S}} = R_0^{\mathcal{S}}$.

We can efficiently compute the generalized subgraph attribution (8) by decomposing it as

$$R_\alpha^{\mathcal{S}} = \widetilde{R}_\alpha^{\mathcal{S}} - \alpha^{L+1} R_0^{\mathcal{G} \setminus \mathcal{S}}, \tag{10}$$

where

$$\widetilde{R}_\alpha^{\mathcal{S}} = \sum_{\mathbf{m} \in \mathcal{G}} \alpha^{\sum_{l=0}^{L} \mathbb{1}(m_l \notin \mathcal{S})} R^{\mathbf{m}}, \tag{11}$$

and applying a message passing algorithm for Eq.(11). Note that the second term in Eq.(10) is the original ($\alpha = 0$) subgraph attribution to the complementary set of $\mathcal{S}$ weighted by α^{L+1}, which can be efficiently computed by sGNN-LRP, described in Section 3. For the first term (or Eq.(11)), our message passing framework, introduced in Section 4, gives the following theorem (the proof is given in Appendix B):

Theorem 3 *Assume that we apply the LRP rule*

$$\breve{\mathbf{r}}^{(l,m_l)} = \sum_{m_{l+1} \in \mathcal{G}} \alpha^{\mathbb{1}(m_{l+1} \notin \mathcal{S})} \mathbf{T}^{l,m_l,m_{l+1}} \breve{\mathbf{r}}^{(l+1,m_{l+1})} \tag{12}$$

for $l = L-1, \dots, 0$ with the initial message $\breve{\mathbf{r}}^{(L,m_L)} = \mathbf{r}^{L,m_L}$. Then, $\breve{\mathbf{r}}^{(l,m_l)} = \widetilde{\mathbf{r}}^{\,l,m_l,m_{l+1:L} \subseteq \mathcal{G}}\ \forall l \in \{0, \dots, L\}$, where $\widetilde{\mathbf{r}}^{\,l,m_l,m_{l+1:L} \subseteq \mathcal{G}} = \alpha^{\sum_{l'=l+1}^{L} \mathbb{1}(m_{l'} \notin \mathcal{S})} \mathbf{r}^{\,l,m_l,m_{l+1:L} \subseteq \mathcal{G}}$, and

$$\widetilde{R}_\alpha^{\mathcal{S}} = \sum_{m_0 \in \mathcal{G}} \alpha^{\mathbb{1}(m_0 \notin \mathcal{S})} \sum_{n=1}^{N^{(0)}} \widetilde{r}_n^{0,m_0,m_{1:L} \subseteq \mathcal{G}}.$$

The forward-hook trick is also applicable (the proof is given in Appendix G):

Table 3: Computation time (in msec) comparison on 5 datasets. '—' means 'failed'.The subgraph size is $|\mathcal{S}| = 5$.

DATASET	BA-2MOTIF			MUTAG	MUTAGENICITY	REDDIT-B	GRAPH-SST2
MODEL-L (DEPTH)	GIN-3	GIN-5	GIN-7	GIN-3	GIN-3	GIN-5	GCN-3
NAIVE GNN-LRP	224.22	6.07×10^3	1.42×10^5	4.23×10^3	4.28×10^3	—	3.16×10^5
SGNN-LRP (OURS)	**4.22**	**6.44**	**9.81**	**28.90**	**26.68**	**195.43**	**29.94**

Theorem 4 *Assume that we applied a complete forward prediction with the modified combine step* (7) *with the constant mask matrix* $\mathbf{M}^{(l)} = \mathbf{M}_\alpha^{\mathcal{S}}$ *for all* l*, where* $\mathbf{M}_\alpha^{\mathcal{S}}$ softly *masks the nodes outside the subgraph, i.e., the columns corresponding to the nodes in the subgraph* $\mathcal{S}$ *are all-one vectors, and all the other entries are equal to* α*. Then, we get* $\widetilde{R}_\alpha^{\mathcal{S}} = \left\langle \texttt{Autograd}(y, \mathbf{H}^{(0)}), \mathbf{M}_\alpha^{\mathcal{S}}\mathbf{H}^{(0)} \right\rangle$.

We will show the usefulness of the generalized subgraph attribution in Section 7.

7. Experiment

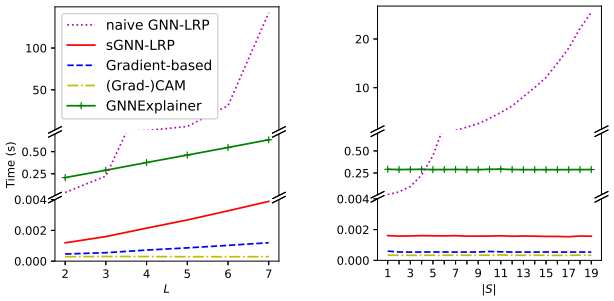
In this section, we conduct two experiments demonstrating (1) the massive gain in computation time by our efficient sGNN-LRP, and (2) the usefulness of the generalized subgraph attribution in relevant node-ordering tasks. For GNN models, we used GIN and GCN with different depths L. We used the following five popular datasets: **BA-2motif** (Luo et al., 2020), **MUTAG** (Debnath et al., 1991), **Mutagenicity** (Kazius et al., 2005b), **REDDIT-BINARY** (Yanardag & Vishwanathan, 2015), and **Graph-SST2** (Yuan et al., 2020b). Detailed experimental setting is given in Appendix H, and our implementation is available at our GitHub repository.[2]

7.1. Computational Efficiency Evaluation

Here we show computational advantages of sGNN-LRP over the Naive GNN-LRP (Schnake et al., 2021) as a baseline on different scales of models and subgraph sizes. Experiments were performed on a Xeon E5-2620 CPU with 8GB memory.

Figure 4 shows the computation time for subgraph attribution on BA-2motif, as functions of (a) the network depth L and (b) the subgraph size $|\mathcal{S}|$, respectively. We clearly observe the (a) exponential and the (b) cubic ($L = 3$) complexity, $O(|\mathcal{S}|^L)$, of Naive GNN-LRP. Our proposed sGNN-LRP with its complexity $O(L|\mathcal{S}|^2)$ is drastically faster. Table 3 summarizes the computation time for various GNNs and datasets. The reported computation time is the average over three trials for randomly chosen 50 samples in each dataset. We report 'fail' if out-of-memory error occurs or the computation does not finish within 360 sec. Again we

[2] https://github.com/xiong-ping/sgnn_lrp_via_mp.



(a) Network depth dependence (b) Subgraph size dependence

Figure 4: Computation time on BA-2motif dataset. Note the different vertical scales in the top, middle, and bottom parts. (a) GIN-L for $L = 1, \ldots, 6$ with $|\mathcal{S}| = 5$. (b) GIN-3 with $|\mathcal{S}| = 1, \ldots, 19$.

observe from the table a significant computational gain by sGNN-LRP.

We also compared computation time with other baseline methods in Figure 4, and observed that our sGNN-LRP is significantly faster than GNNExplainer, and even comparable with very simple baselines, Gradient-based heatmap and (Grad-)CAM. Notably the complexity bounds $O(L|\mathcal{S}|^2)$ of sGNN-LRP is the same as those for a single forward/backward pass of GNNs, and therefore, sGNN-LRP can be applied to deeper GNNs for larger graphs at a similar computational cost to prediction.

7.2. Node Ordering Performance by Generalized Subgraph Attribution

Here, we demonstrate the high usefulness of our generalized definition of subgraph attribution. Specifically, we show that the optimal discounting parameter, α in Eq.(9), is not always zero, and depends on the evaluation task. We first evaluate the node ordering performance on the BA-2motif dataset, for which the ground truth is available. Then, we evaluate the performance in the model activation and the model pruning tasks (Schnake et al., 2021) on other datasets.

7.2.1. NODE ORDERING

We use the generalized subgraph attribution for providing node ordering in two modes, *activation* and *pruning*.

Activation mode: In this mode, we obtain the node or-

dering $\left[m^{(1)}, ..., m^{(M)}\right]$ such that $\sum_{i=1}^{M} R_\alpha^{\{m^{(1)},...,m^{(i)}\}}$ is maximized. Since this maximization is infeasible, we perform the following greedy search for $i = 1, \cdots, M$: $m^{(i)} = \underset{m \in \mathcal{G} \setminus \{m^{(1)},...,m^{(i-1)}\}}{\operatorname{argmax}} R_\alpha^{\{m^{(1)},...,m^{(i-1)},m\}}$.

Pruning mode: In this mode, we obtain the node ordering $\left[m^{(1)}, ..., m^{(M)}\right]$ such that $\sum_{i=1}^{M} |R_\alpha^{\mathcal{G}\setminus\{m^{(1)},...,m^{(i)}\}} - R^{\mathcal{G}}|$ is minimized. This minimization is again infeasible, and therefore, we perform the following greedy search for $i = 1, \cdots, M$: $m^{(i)} = \underset{m \in \mathcal{G} \setminus \{m^{(1)},...,m^{(i-1)}\}}{\operatorname{argmin}} |R_\alpha^{\mathcal{G}\setminus\{m^{(1)},...,m^{(i-1)},m\}} - R^{\mathcal{G}}|$.

Since the activation mode focuses on the high relevance subgraphs themselves and the pruning mode focuses on their complement, we expect that they respectively tend to satisfy the first and the second properties that good subgraph attribution should fulfill (see Section 6). We investigate how the discounting parameter α of the generalized subgraph attribution affects the activation and the pruning performance.

7.2.2. BA-2motif Benchmark Experiment

Here, we use the BA-2motif dataset to evaluate the node ordering performance of subgraph attribution. Since this dataset is synthetic and the ground truth *motif* is available, we can simply compare the node ordering obtained by the subgraph attribution with the ground truth. All sample graphs have 25 nodes, of which 5 nodes are specified as the motif. Fig. 5 shows the accuracy—the proportion that the subgraph attribution in the activation mode gives the ordering such that the top 5 nodes match the ground truth motif—with its dependence on α. The figure also shows the area under the receiver operating characteristic curve (AUROC), where the threshold of motif detection is scanned. We found that the best performance is achieved around $\alpha \sim 0$, and therefore, the original subgraph attribution is almost optimal. This is not surprising because, for this dataset, the nodes outside the motifs are completely random, and therefore, considering the outside nodes gives no useful information.

7.2.3. Model Activation and Pruning Experiments

Next we evaluate the subgraph attribution performance in model activation and model pruning tasks on MUTAG and Graph-SST2 datasets, for which the ground truth explanation is not available. These tasks measure the correlation between the relevance attribution and the model output in two ways.

Let $f(\cdot)$ be the GNN model output for the correct label. The goal of the model activation task is to find the node ordering $\left[m^{(1)}, ..., m^{(M)}\right]$ such that the area under the activation curve, AUAC $= \frac{1}{M}\sum_{i=1}^{M} f(\{m^{(1)}, ..., m^{(i)}\})$, is maximized. This task evaluates how many subgraphs formed by the top predicted nodes recover the correct prediction, and therefore, measures how well the attribution fulfills Property 1 in Section 6. On the other hand, the goal of the pruning task is to find the node ordering $\left[m^{(1)}, ..., m^{(M)}\right]$ such that the area under the pruning curve, AUPC $= \frac{1}{M}\sum_{i=1}^{M} |f(\mathcal{G}\setminus\{m^{(1)}, ..., m^{(i)}\}) - f(\mathcal{G})|$, is minimized. This task evaluates how much the complement of subgraphs formed by the top predicted nodes retains the predictive performance, and therefore, can be a performance measure on Property 2 in Section 6. Further details are given in Appendix J. We should naturally use the activation and the pruning modes, respectively, for node ordering in the model activation and the model pruning tasks (see Section 7.2.1).

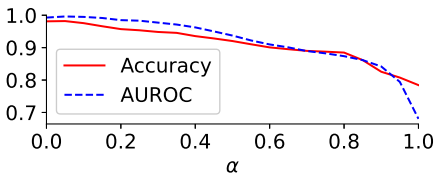


Figure 5: Accuracy and AUROC of node ordering task on the BA-2motif dataset with different discounting parameter α. The best performance is achieved around $\alpha \sim 0$.

Figure 6 shows AUAC and AUPC with their dependence on the discounting parameter α. Because AUAC, as well as AUPC, differs largely between positive samples and negative samples, we plotted them separately. We speculate that this is due to the different predictive capability of the model on positive and negative samples. The (red) curves for sGNN-LRP imply that the original definition of the subgraph attribution, i.e., $\alpha = 0$, is not always optimal, and tuning α can improve the node ordering performance. This applies not only in the pruning task but also in the activation task. We will further investigate the performance dependence on α, and develop tuning procedures in our future work.

To show the superiority of the attribution via sGNN-LRP, we also compared the node ordering performance with three baselines, GNNExplainer, Gradient-based heatmap, and (Grad-)CAM, and plotted the results in Figure 6. The subgraph relevance by GNNExplainer for $\alpha = 0$ is given by the sum of the relevances over the edges between the nodes both inside the subgraph. For $\alpha > 0$, the sum over the edges connecting inside and outside nodes is also added with the discounting factor α (see Appendix K for more detail). Assuming that the parameter α is optimized for each method, sGNN-LRP outperforms the baselines in most of the cases (6 out of 8).

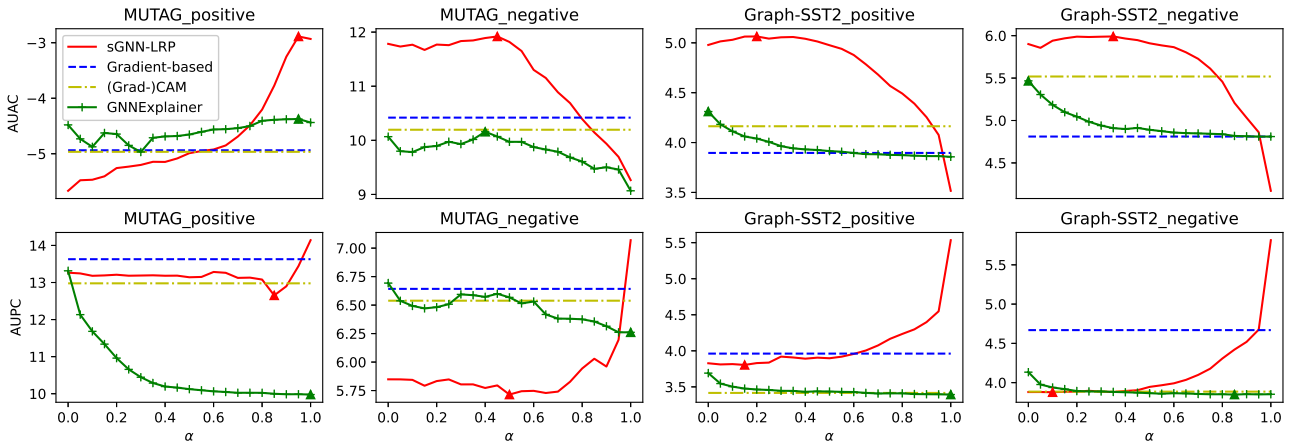


Figure 6: AUAC (top row, higher is better) and AUPC (bottom row, lower is better) on MUTAG and Graph-SST2 datasets of our sGNN-LRP and baseline methods, GNNExplainer, Gradient-based heatmap, and (Grad-)CAM. The triangles mark the best performance points for sGNN-LRP and GNNExplainer.

8. Conclusion

Layer-wise relevance propagation for Graph neural networks (GNN-LRP) is a higher-order explainability method for GNNs, which provides attributes of the GNN models at the level of walks. Specifically, it supports subgraph-level attribution by summing over the walks inside a given subgraph, which however suffers from *exponential* complexity and thus has computational limits in application. In this paper, we have overcome this issue by proposing a *polynomial-time* algorithm (sGNN-LRP) that directly computes the subgraph GNN-LRP attribution. Notably, our development of sGNN-LRP has been conducted by a novel procedure: unlike previous work, we developed sGNN-LRP by first defining the target quantity to be computed and then deriving a propagation rule as a message passing algorithm. This novel procedure is general, rediscovering many existing LRP rules, and thus expected to facilitate future development of new LRP methods. We have demonstrated the utility of this procedure by deriving another LRP rule for computing a generalized definition of the subgraph relevance that takes into account the partly-outside walks. Experimental results showed that our proposed sGNN-LRP is significantly faster than the naive application of GNN-LRP, and that the generalized subgraph relevance definition can more robustly attribute GNNs at the subgraph level.

Future research will address the broad application of the novel algorithms, as now novel 'deeper' insights (manifested in longer walks or deeper GNNs) have become possible for learning problems that possess a significant amount of higher-order and long range nonlinear interactions, such as in the sciences, e.g., for neuroimaging (see, e.g., Rubinov & Sporns (2010); Shine et al. (2019)) or quantum chemistry (see, e.g., Gilmer et al. (2017); Schütt et al. (2018); Unke et al. (2021)). Beyond applications in the sciences, we consider the novel efficient GNN-LRP algorithms as promising for NLP applications where assessing deeper higher-order interactions may be helpful for assessing trustworthiness, fairness, and unbiasedness of SOTA systems.

Acknowledgments

We would like to thank Michael Gastegger for helpful discussion and his significant support in preparing the manuscript after the abstract submission deadline. This work was supported by the German Ministry for Education and Research (BMBF) as BIFOLD - Berlin Institute for the Foundations of Learning and Data under grants 01IS18025A and 01IS18037A.

References

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*, pp. 402–411. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G.,

Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34 (2):786–797, 1991. doi: 10.1021/jm00106a046.

Domingue, M., Dhamdhere, R., Harish Kanamarlapudi, N. D., Raghupathi, S., and Ptucha, R. Evolution of graph classifiers. In *2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pp. 1–5, 2019.

Gao, Y., Sun, T., Bhatt, R., Yu, D., Hong, S., and Zhao, L. GNES: learning to explain graph neural networks. In Bailey, J., Miettinen, P., Koh, Y. S., Tao, D., and Wu, X. (eds.), *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, pp. 131–140. IEEE, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Kazius, J., McGuire, R., and Bursi, R. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005a.

Kazius, J., McGuire, R., and Bursi, R. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005b. doi: 10.1021/jm040835a. PMID: 15634026.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 4768–4777. Curran Associates, Inc., 2017.

Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., and Zhang, X. Parameterized explainer for graph neural network. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Montavon, G., Samek, W., and Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

Pearl, J. Reverend Bayes on inference engines: A distributed hierarchical approach. In Waltz, D. L. (ed.), *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, August 18-20, 1982*, pp. 133–136. AAAI Press, 1982.

Pope, P. E., Kolouri, S., Rostami, M., Martin, C. E., and Hoffmann, H. Explainability methods for graph convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 10772–10781. Computer Vision Foundation / IEEE, 2019.

Rubinov, M. and Sporns, O. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. Explaining deep neural networks and beyond: A review of methods and applications. *Proc. IEEE*, 109(3):247–278, 2021.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K. T., Müller, K.-R., and Montavon, G. Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

Shine, J. M., Breakspear, M., Bell, P. T., Martens, K. A. E., Shine, R., Koyejo, O., Sporns, O., and Poldrack, R. A. Human cognition involves the dynamic integration of neural activity and neuromodulatory systems. *Nature neuroscience*, 22(2):289–296, 2019.

Unke, O. T., Chmiela, S., Gastegger, M., Schütt, K. T., Sauceda, H. E., and Müller, K.-R. Spookynet: Learning force fields with electronic degrees of freedom and nonlocal effects. *Nature Communications*, 12(1):7273, 2021. doi: 10.1038/s41467-021-27504-0. URL https://doi.org/10.1038/s41467-021-27504-0.

Vu, M. N. and Thai, M. T. PGM-Explainer: Probabilistic graphical model explanations for graph neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G. (eds.), *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 1365–1374. ACM, 2015. doi: 10.1145/2783258.2783417.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9240–9251, 2019.

Yuan, H., Tang, J., Hu, X., and Ji, S. XGNN: towards model-level explanations of graph neural networks. In Gupta, R., Liu, Y., Tang, J., and Prakash, B. A. (eds.), *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 430–438. ACM, 2020a.

Yuan, H., Yu, H., Gui, S., and Ji, S. Explainability in graph neural networks: A taxonomic survey. *CoRR*, abs/2012.15445, 2020b. URL https://arxiv.org/abs/2012.15445.

Yuan, H., Yu, H., Wang, J., Li, K., and Ji, S. On explainability of graph neural networks via subgraph explorations. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12241–12252. PMLR, 2021.

A. Details of Figure 1

In Figure 1 we consider the GIN model trained on the Mutagenicity dataset (Kazius et al., 2005b). We use the same model architecture and training procedure as described in Appendix H.3. The visualized molecule is one sample of the Mutagenicity dataset which is classified as mutagenic. The bars in each graph reflect the interaction relevance of the subgraph consisting of all atoms with the described atomic number at the x-axis. In order to reflect the relevance of the interactions of a set $\mathcal{S}$, we use the subgraph relevance definition in equation (3) and substract the relevance scores of all features that are not exclusively composed of all atoms types in S. For example, in the case where we measure the interaction between the atoms C, N and O we define the interaction relevance $R^{CNO}_{\leftrightarrow}$ to be

$$R^{CNO}_{\leftrightarrow} = R^{CNO} - R^{CN} - R^{CO} - R^{NO} + R^{C} + R^{N} + R^{O}.$$

It is important to see that we add the relevance scores with only one atom (such as R^C), because by definition of the subraph relevance in (3) they occur in the scores which consists of two atoms (as in R^{CO} and R^{CN}), which we already subtract twice. This definition ensures the conservation of the interaction relevance, i.e. it ensures $R^{CNO} = \sum_{\mathcal{S} \subset CNO} R^{\mathcal{S}}_{\leftrightarrow}$.

B. Proof of Theorem 1 and Theorem 3

We prove Theorem 3, which covers Theorem 1 as a special case for $\alpha = 0$. The required quantity is decomposed as

$$\widetilde{R}^{\mathcal{S}}_{\alpha} = \sum_{\mathbf{m} \in \mathcal{G}} \alpha^{\sum_{l=0}^{L} \mathbb{1}(m_l \notin \mathcal{S})} R^{\mathbf{m}} = \sum_{\mathbf{m} \in \mathcal{G}} \alpha^{\sum_{l=0}^{L} \mathbb{1}(m_l \notin \mathcal{S})} \sum_{n=1}^{N^{(0)}} r_n^{0,\mathbf{m}} = \sum_{m_0 \in \mathcal{G}} \alpha^{\mathbb{1}(m_0 \notin \mathcal{S})} \sum_{n=1}^{N^{(0)}} \widetilde{r}_n^{0,m_0,m_{1:L} \subseteq \mathcal{G}}, \tag{13}$$

where

$$\begin{aligned}
\widetilde{\mathbf{r}}^{\,0,m_0,m_{1:L} \subseteq \mathcal{G}} &= \sum_{m_{1:L} \in \mathcal{G}} \alpha^{\sum_{l=1}^{L} \mathbb{1}(m_l \notin \mathcal{S})} \mathbf{T}^{0,m_0,m_1} \mathbf{T}^{1,m_1,m_2} \cdots \mathbf{T}^{L-1,m_{L-1},m_L} \mathbf{r}^{\,L,m_L} \\
&= \sum_{m_1 \in \mathcal{G}} \alpha^{\mathbb{1}(m_1 \notin \mathcal{S})} \mathbf{T}^{0,m_0,m_1} \underbrace{\sum_{m_2 \in \mathcal{G}} \alpha^{\mathbb{1}(m_2 \notin \mathcal{S})} \mathbf{T}^{1,m_1,m_2} \quad \cdots \quad \underbrace{\sum_{m_L \in \mathcal{G}} \alpha^{\mathbb{1}(m_L \notin \mathcal{S})} \mathbf{T}^{L-1,m_{L-1},m_L} \mathbf{r}^{\,L,m_L}}_{=\widetilde{\mathbf{r}}^{\,L-1,m_{L-1},m_L \subseteq \mathcal{G}}}}_{=\widetilde{\mathbf{r}}^{\,1,m_1,m_{2:L} \subseteq \mathcal{G}}}.
\end{aligned} \tag{14}$$

This decomposition gives the LRP rule (12) as a sum-product message passing with the propagated relevance $\breve{\mathbf{r}}^{(l,m_l)} = \widetilde{\mathbf{r}}^{\,l,m_l,m_{l+1:L} \subseteq \mathcal{G}}$, and proves Theorem 3. Noting that

$$\widetilde{\mathbf{r}}^{\,l,m_l,m_{l+1:L} \subseteq \mathcal{G}} = \mathbf{r}^{\,l,m_l,m_{l+1:L} \subseteq \mathcal{S}} \qquad \text{for} \qquad \alpha = 0 \tag{15}$$

proves Theorem 1. □

C. Stardard LRP as Message Passing

Here, we derive the standard LRP rule as a sum-product message passing. Consider a plain feed forward neural networks with $N^{(l)}$ neurons at layer $l = 0, \ldots, L$. The relevance of the (neuron-level) walk $\mathbf{n} = (n_0, \ldots, n_L)$ is then given as

$$R^{\mathbf{n}} = T^0_{n_0,n_1} T^1_{n_1,n_2} \quad \cdots \quad T^{L-1}_{n_{L-1},n_L} r^L_{n_L} = \left(\prod_{l=0}^{L-1} T^l_{n_l,n_{l+1}} \right) r^L_{n_L}, \tag{16}$$

where $\mathbf{r}^l \in \mathbb{R}^{N^{(l)}}$ is such that r^l_n denotes the sum of relevances over all walks going through the neuron n at layer l. Collecting the relevances of all walks starting from the input node n_0 amounts to

$$r^0_{n_0} = \sum_{n_1=1}^{N^{(1)}} \sum_{n_2=1}^{N^{(2)}} \cdots \sum_{n_L=1}^{N^{(L)}} T^0_{n_0,n_1} T^1_{n_1,n_2} \quad \cdots \quad T^{L-1}_{n_{L-1},n_L} r^L_{n_L}. \tag{17}$$

This summation can be decomposed as

$$r^0_{n_0} = \sum_{n_1=1}^{N^{(1)}} T^0_{n_0,n_1} \underbrace{\sum_{n_2=1}^{N^{(2)}} T^1_{n_1,n_2} \quad \cdots \quad \underbrace{\sum_{n_L=1}^{N^{(L)}} T^{L-1}_{n_{L-1},n_L} r^L_{n_L}}_{=r^{L-1}_{n_{L-1}}}}_{=r^1_{n_1}}, \tag{18}$$

giving a sum-product message passing:

$$r^l_{n_l} = \sum_{n_{l+1}} T^l_{n_l,n_{l+1}} r^{l+1}_{n_{l+1}} \qquad \text{or} \qquad \mathbf{r}^l = \mathbf{T}^l \mathbf{r}^{l+1}. \tag{19}$$

This coincides with the standard LRP (Bach et al., 2015).

D. GNN-LRP as Message Passing

We can drive the original GNN-LRP rule as message passing, giving another proof of the following theorem:

Theorem 5 *(Schnake et al., 2021) Assume that the GNN-LRP rule* (2) *is applied for* $l = L-1, \dots, 0$ *with the initial message* $\breve{\mathbf{r}}^{(L,m_L)} = \mathbf{r}^{L,m_L}$*. Then,* $\breve{\mathbf{r}}^{(l,m_l)} = \mathbf{r}^{l,m_{l:L}}$ $\forall l \in \{0,\dots,L\}$*, and* $R^{\mathbf{m}} = \sum_{n=1}^{N^{(0)}} r_n^{0,\mathbf{m}}$.

(Proof) The required quantity is decomposed as

$$R^{\mathbf{m}} = \sum_{n=1}^{N^{(0)}} r_n^{0,\mathbf{m}}, \tag{20}$$

where

$$\mathbf{r}^{0,\mathbf{m}} = \mathbf{T}^{0,m_0,m_1} \underbrace{\mathbf{T}^{1,m_1,m_2} \quad \cdots \quad \underbrace{\mathbf{T}^{L-1,m_{L-1},m_L} \mathbf{r}^{L,m_L}}_{=\mathbf{r}^{L-1,m_{L-1:L}}}}_{=\mathbf{r}^{1,m_{1:L}}}. \tag{21}$$

This decomposition gives the LRP rule (2) as a sum-product message passing with the propagated relevance $\breve{\mathbf{r}}^{(l,m_l)} = \mathbf{r}^{l,m_{l:L}}$, and proves the theorem. □

E. Propagation Rules of Table 2 for GCNs with the Notation in Schnake et al. (2021)

To bring further intuition on the proposed propagation rules, we rewrite all propagation rules of Table 2 in a notation similar to that of the original paper on GNN-LRP (Schnake et al., 2021). Note that the defined notation here applies only in this appendix. We show the propagation rules specifically for the Graph Convolution Networks (Kipf & Welling, 2017), with network connectivity encoded in the matrix Λ, weights at a given layer stored in a matrix W, and h^a_J denoting the activation of neuron a in node J. We denote by $\dots JKL \dots$ node indices in successive layers and jointly forming a walk. We denote by a and b two neurons indices (within their corresponding nodes) in successive layers. With this notation, the GCN equation for a given layer can be written as

$$\forall_{K,b}: \quad h^b_K = \max\Big(0, \sum_{J,a} \lambda_{JK} h^a_J w_{ab}\Big).$$

Furthermore, we denote by $\mathcal{S}$ the set of nodes composing the subgraph of interest $\mathcal{S}$. It can also be interpreted as a coarse-graining of the member nodes into a new single node denoted by $\mathcal{S}$. We denote by $\sum_K$ the sum over all nodes in the given layer, and $\sum_b$ the sum over all neuron indices of a node in the given layer. The propagation rules can then be written

as follows:

$$R_J^a = \sum_{K,b} \frac{\lambda_{JK} h_J^a w_{ab}^{\uparrow}}{\sum_{J,a} \lambda_{JK} h_J^a w_{ab}^{\uparrow}} R_K^b, \qquad \text{(LRP for general FFNN)}$$

$$R_{JKL\ldots}^a = \sum_{b} \frac{\lambda_{JK} h_J^a w_{ab}^{\uparrow}}{\sum_{J,a} \lambda_{JK} h_J^a w_{ab}^{\uparrow}} R_{KL\ldots}^b, \qquad \text{(GNN-LRP)}$$

$$R_{J\mathcal{S}\mathcal{S}\ldots}^a = \sum_{K\in\mathcal{S},b} \frac{\lambda_{JK} h_J^a w_{ab}^{\uparrow}}{\sum_{J,a} \lambda_{JK} h_J^a w_{ab}^{\uparrow}} R_{K\mathcal{S}\ldots}^b, \qquad \text{(sGNN-LRP)}$$

$$R_{J\mathcal{S}\mathcal{S}\ldots}^{a,\alpha} = \sum_{K,b} \alpha^{\mathbb{1}(K\in\mathcal{S})} \frac{\lambda_{JK} h_J^a w_{ab}^{\uparrow}}{\sum_{J,a} \lambda_{JK} h_J^a w_{ab}^{\uparrow}} R_{K\mathcal{S}\ldots}^{b,\alpha}. \qquad \text{(Generalized sGNN-LRP)}$$

F. Forward-hook Trick for Walk Relevance by GNN-LRP

The forward-hook trick (Schnake et al., 2021; Samek et al., 2021) for computing the relevance of a walk works as follows. We implement the forward combine step in Eq.(1) as

$$\begin{aligned}
\mathbf{P}^{(l)} &\leftarrow \mathbf{Z}^{(l)} \mathbf{W}^{(l)\uparrow}, \\
\mathbf{Q}^{(l)} &\leftarrow \mathbf{P}^{(l)} \odot [\sigma(\mathbf{Z}^{(l)} \mathbf{W}^{(l)}) \oslash \mathbf{P}^{(l)}]_{\texttt{cst.}}, \\
\mathbf{H}^{(l)} &\leftarrow \mathbf{Q}^{(l)} \odot \mathbf{M}^{(l)} + [\mathbf{Q}^{(l)}]_{\texttt{cst.}} \odot (1 - \mathbf{M}^{(l)}),
\end{aligned} \tag{22}$$

where $\mathbf{P}^{(l)}, \mathbf{M}^{(l)} \in \mathbb{R}^{M\times N}$, and $\odot$ and $\oslash$ denote the entry-wise multiplication and division, respectively. The operator $[\cdot]_{\texttt{cst.}}$ *detaches* the quantity to which it applies from the gradient computation. Schnake et al. (2021) have shown that, if the *mask* matrices $\{\mathbf{M}^{(l)}\}$ are set such that the column corresponding to the node m_l specified by the walk is all-one vector, and the other columns are all-zero vectors, the relevance of a walk is obtained by

$$R^{\mathbf{m}} = \left\langle \texttt{Autograd}(y, \mathbf{H}^{(0)}), \mathbf{M}^{(0)} \mathbf{H}^{(0)} \right\rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product.

G. Proof of Theorem 2 and Theorem 4

We prove Theorem 4, which covers Theorem 2 as a special case for $\alpha = 0$. The adapted forward computation of the model gives

$$\begin{aligned}
Z_{m_l,n_{l-1}}^{(l)} &= \sum_{m_{l-1}\in\mathcal{G}} \Lambda_{m_{l-1},m_l} H_{m_{l-1},n_{l-1}}^{(l-1)}, \\
P_{m_l,n_l}^{(l)} &= \sum_{n_{l-1}} Z_{m_l,n_{l-1}}^{(l)} W_{n_{l-1},n_l}^{(l)\uparrow}, \\
Q_{m_l,n_l}^{(l)} &= P_{m_l,n_l}^{(l)} \left[\frac{\rho(\sum_{n_{l-1}} Z_{m_l,n_{l-1}}^{(l)} W_{n_{l-1},n_l}^{(l)})}{P_{m_l,n_l}^{(l)}} \right]_{\texttt{cst.}}, \\
H_{m_l,n_l}^{(l)} &= \alpha^{\mathbb{1}(m_l\notin\mathcal{S})} Q_{m_l,n_l}^{(l)} + (1 - \alpha^{\mathbb{1}(m_l\notin\mathcal{S})}) \left[Q_{m_l,n_l}^{(l)} \right]_{\texttt{cst.}}.
\end{aligned} \tag{23}$$

Then the gradient is

$$\frac{\partial H_{m_l,n_l}^{(l)}}{\partial H_{m_{l-1},n_{l-1}}^{(l-1)}} = \alpha^{\mathbb{1}(m_l\notin\mathcal{S})} \Lambda_{m_{l-1},m_l} W_{n_{l-1},n_l}^{(l)\uparrow} \frac{H_{m_l,n_l}^{(l)}}{P_{m_l,n_l}^{(l)}}, \tag{24}$$

where we used $H_{m_l,n_l}^{(l)} = \rho(\sum_{n_{l-1}} Z_{m_l,n_{l-1}}^{(l)} W_{n_{l-1},n_l}^{(l)})$. By using the transition coefficient

$$T_{n_{l-1},n_l}^{l-1,m_{l-1},m_l} = \frac{\Lambda_{m_{l-1},m_l} H_{m_{l-1},n_{l-1}}^{(l-1)} W_{n_{l-1},n_l}^{(l-1)\uparrow}}{\sum_{m',n'} \Lambda_{m',m_l} H_{m',n'}^{(l-1)} W_{n',n_l}^{(l-1)\uparrow}} \tag{25}$$

and Eq.(23), the gradient (24) can be written as

$$\begin{aligned}\frac{\partial H^{(l)}_{m_l,n_l}}{\partial H^{(l-1)}_{m_{l-1},n_{l-1}}} =&\alpha^{\mathbb{1}(m_l\notin\mathcal{S})}\underbrace{\frac{\Lambda_{m_{l-1},m_l}W^{(l)\uparrow}_{n_{l-1},n_l}H^{(l-1)}_{m_{l-1},n_{l-1}}}{P^{(l)}_{m_l,n_l}}}_{T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}}\frac{H^{(l)}_{m_l,n_l}}{H^{(l-1)}_{m_{l-1},n_{l-1}}}\\ =&\alpha^{\mathbb{1}(m_l\notin\mathcal{S})}T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}\frac{H^{(l)}_{m_l,n_l}}{H^{(l-1)}_{m_{l-1},n_{l-1}}}.\end{aligned} \tag{26}$$

By applying the chain rule, we have the gradient of the output layer with respect to the input layer as

$$\frac{\partial H^{(L)}_{m_L,n_L}}{\partial H^{(0)}_{m_0,n_0}} = \sum_{m_1,\cdots,m_{L-1}}\sum_{n_1,\cdots,n_{L-1}}\frac{\partial H^{(L)}_{m_L,n_L}}{\partial H^{(L-1)}_{m_{L-1},n_{L-1}}}\cdots\frac{\partial H^{(1)}_{M_1,n_1}}{\partial H^{(0)}_{M_0,n_0}}. \tag{27}$$

Substituting Eq.(26) into Eq.(27) gives

$$\frac{\partial H^{(L)}_{m_L,n_L}}{\partial H^{(0)}_{m_0,n_0}} = \sum_{m_1,\cdots,m_{L-1}}\sum_{n_1,\cdots,n_{L-1}}\alpha^{\sum_{l=1}^{L}\mathbb{1}(m_l\notin\mathcal{S})}\prod_{l=1}^{L}T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}\frac{H^{(L)}_{m_L,n_L}}{H^{(0)}_{m_0,n_0}}. \tag{28}$$

For the readout function, if we modify it according to the forward-hook trick described in Appendix B in Samek et al. (2021), we can obtain the relevance of the L-th layer of GNN as

$$r^{L,m_L}_{n_L} = H^{(L)}_{m_L,n_L}\frac{\partial y}{\partial H^{(L)}_{m_L,n_L}}. \tag{29}$$

Then, we have

$$\begin{aligned}\frac{\partial y}{\partial H^{(0)}_{m_0,n_0}} &= \sum_{m_1,\cdots,m_L}\sum_{n_1,\cdots,n_L}\alpha^{\sum_{l=1}^{L}\mathbb{1}(m_l\notin\mathcal{S})}\prod_{l=1}^{L}T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}\frac{H^{(L)}_{m_L,n_L}}{H^{(0)}_{m_0,n_0}}\frac{\partial y}{\partial H^{(L)}_{m_L,n_L}}\\ &= \sum_{m_1,\cdots,m_L}\sum_{n_1,\cdots,n_L}\alpha^{\sum_{l=1}^{L}\mathbb{1}(m_l\notin\mathcal{S})}\prod_{l=1}^{L}T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}\frac{H^{(L)}_{m_L,n_L}}{H^{(0)}_{m_0,n_0}}\frac{r^{L,m_L}_{n_L}}{H^{(L)}_{m_L,n_L}}\\ &= \sum_{m_1,\cdots,m_L}\sum_{n_1,\cdots,n_L}\alpha^{\sum_{l=1}^{L}\mathbb{1}(m_l\notin\mathcal{S})}\prod_{l=1}^{L}T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}\frac{r^{L,m_L}_{n_L}}{H^{(0)}_{m_0,n_0}}.\end{aligned} \tag{30}$$

Multiplying the gradient with the masked initial activation and summing them up gives the final inner product:

$$\begin{aligned}\sum_{m_0,n_0}\alpha^{\mathbb{1}(m_0\notin\mathcal{S})}H^{(0)}_{m_0,n_0}\frac{\partial y}{\partial H^{(0)}_{m_0,n_0}} &= \sum_{m_0,\cdots,m_L}\alpha^{\sum_{l=0}^{L}\mathbb{1}(m_l\notin\mathcal{S})}\sum_{n_0,\cdots,n_L}\prod_{l=1}^{L}T^{l-1,m_{l-1},m_l}_{n_{l-1},n_l}r^{L,m_L}_{n_L}\\ &=\sum_{\mathbf{m}}\alpha^{\sum_{l=0}^{L}\mathbb{1}(m_l\notin\mathcal{S})}R^{\mathbf{m}} = \widetilde{R}^{\mathcal{S}}_{\alpha},\end{aligned} \tag{31}$$

which proves the theorem. □

H. Details of Datasets and GNN Models used in Experiments

H.1. BA2-Motif

BA-2motif (Luo et al., 2020) is a synthetic dataset of graphs that can be classified into two classes according to the different motifs. For each sample graph, a base graph is generated by the Barabási-Albert (BA) model, and then one of two motifs is connected to it. Because this dataset is synthesized, the ground truth about which nodes build the motif is available.

Table 4: Statistics of the 5 datasets used in experiments.

	BA-2MOTIF	MUTAG	MUTAGENICITY	REDDIT-BINARY	GRAPH-SST2
# OF EDGES (AVG)	25.48	19.79	17.79	497.75	19.40
# OF NODES (AVG)	25.00	17.93	16.90	429.63	10.20
# OF GRAPHS	1000	188	4337	2000	70042

We trained GIN models with 2,3,4,5,6,7 layers, and all models has the same GIN block, which is a 2-layer MLP. The input feature dimension is 1 and the output feature dimension in the MLP blocks $N^{(l)} = 20,\ \forall l = 1, \cdots, L-1$. The activation function is ReLU. We employed the SGD optimizer with a decreasing learning rate $\gamma = 0.00001/(1.0 + (\texttt{epoch}/\texttt{epochs})$ for 10000, 10000, 5000, 1000, 1000, 1000 epochs, respectively. We downloaded the dataset from the repository of Schnake et al. (2021), and the dataset includes 1000 samples with the first 500 samples from positive class and the last 500 samples from negative class. Because all samples are randomly generated, it is unnecessary to sample randomly and we used the 0-400 and 500-900 as training dataset and the rest as testing dataset. The test accuracy of the three models are 98%, 99.50%, 100%, 100%, 100%, 100%, respectively.

H.2. MUTAG

MUTAG (Debnath et al., 1991) is a datasets of molecules. Every sample graph includes atoms as nodes and chemical links as edges. The molecules are labeled as mutagenic or non-mutagenic. The node feature is the node type (atom), which is represented as a one-hot vector.

The 3-layered GIN model has in all layers a 2-layer MLP as the GIN blocks. The input feature dimension is 7, which is one-hot vectors denoting different atoms. The output dimensions in the MLP blocks are 128. We used 108 samples with half positive and half negative as training dataset, and the rest samples build the testing dataset. We trained the model with SGD optimizer for 1500 epochs, and the learning rate $\gamma = 0.0005/(1.0 + (\texttt{epoch}/\texttt{epochs})$. The test accuracy is 85.00%.

H.3. Mutagenicity

Mutagenicity (Kazius et al., 2005b) is another dataset for chemical molecules, which is larger than MUTAG and contains much more variety of mutagenic molecules with different types of toxic groups.

The model setting is the same as for MUTAG, except the input feature dimension increased to 13 as there are more atoms in this dataset. We used 3096 samples with half positive and half negative as training dataset, and the rest as testing dataset. We trained the model with Adam optimizer for 25 epochs, and the initial learning rate $\gamma = 0.00005$. The test accuracy is 83.16%.

H.4. REDDIT-BINARY

REDDIT-BINARY (Yanardag & Vishwanathan, 2015) is a social network dataset, and each graphs stands for a community, with nodes being users and edges denoting that there is at least one response to the comments between the two users. The graphs are classified into two classes according to which kind of community the users build, i.e., question/answer-based community or discussion-based community. The dataset contains large graphs (> 400 nodes in average), and no node feature is provided.

Our model for this dataset is 5-layer GIN, with 2-layer MLP as GIN blocks. The input feature dimension is 1 and the output dimensions of the GIN blocks are 64. The training dataset has 1600 samples of half positive and half negative, and the rest 400 samples build the testing dataset. We trained the model with Adam optimizer for 500 epochs and the initial learning rate $\gamma = 0.00005$. The test accuracy is 82.50%.

H.5. Graph-SST2

Graph-SST2 (Yuan et al., 2020b) is a dataset of texts labeled in two sentiment classes. The text are transformed into parse tree graphs with 768-dimensional embedded vectors of the words for the initial node features.

The model is built with a node feature embedding part and a following 3-layer GCN. The input feature dimension is 768,

Algorithm 1 sGNN-LRP, $\alpha = 0$

Input: graph $\mathcal{G}$, subgraph $\mathcal{S}$, # of model layers L.
Output: subgraph relevance score $R^{\mathcal{S}}$
Initialize mask matrix $\mathbf{M}_{\mathcal{S}}$, such that $\mathbf{M}_{\mathcal{S}}$ is valued 1 on the indices of nodes inside $\mathcal{S}$ and 0 else.
for $l = 1$ **to** L **do**
  $\mathbf{Z}^{(l)} \leftarrow \mathbf{\Lambda}\mathbf{H}^{(l-1)}$
  $\mathbf{P}^{(l)} \leftarrow \mathbf{Z}^{(l)}\mathbf{W}^{(l)\uparrow}$
  $\mathbf{Q}^{(l)} \leftarrow \mathbf{P}^{(l)} \odot [\rho(\mathbf{Z}^{(l)}\mathbf{W}^{(l)}) \oslash \mathbf{P}^{(l)}]_{cst.}$
  $\mathbf{H}^{(l)} \leftarrow \mathbf{Q}^{(l)} \odot \mathbf{M}_{\mathcal{S}} + [\mathbf{Q}^{(l)}]_{cst.} \odot (1 - \mathbf{M}_{\mathcal{S}})$
end for
$y \leftarrow \text{readout}(\mathbf{H}^{(L)})$ with readout function modified according to LRP forward-hook trick.
$R^{\mathcal{S}} = \sum_{m_0 \in \mathcal{S}} \left\langle \text{Autograd}(y, \mathbf{H}^{(0)}_{m_0}), \mathbf{H}^{(0)}_{m_0} \right\rangle$

and in the middle layer of GCN the output dimension is 20. We downloaded the dataset from Yuan et al. (2020b) and used their dataset split. We trained the model with Adam optimizer for 50 epochs, and the initial learning rate $\gamma = 0.0001$. The test accuracy is 89.40%.

I. Details of sGNN-LRP Implementation

Algorithm 1 and Algorithm 2, respectively, summarize the procedures of sGNN-LRP (with the Forward-hook trick) for the original subgraph attribution ($\alpha = 0$) and for the generalized subgraph attribution ($\alpha \in (0, 1]$).

J. Details of the Model Activation and Pruning Experiments

Algorithm 3 and Algorithm 4, respectively, describe the detailed procedures of model activation and pruning experiments (node ordering and its evaluation).

K. Subgraph Attribution using GNNExplainer

In Section 7.2.3, we applied the GNNExplainer to compute the subgraph relevance according to the definition of generalized subgraph relevance definition (8). GNN-Explainer attributes the edges, and we consider an edge as a one-step walk which contains only two nodes. According to (8), the subgraph relevance is the sum of all edges that have at least one node inside the subgraph, with the partly-outside edge (one node inside and one node outside the subgraph) deweighted with α, i.e.,

$$R^{\mathcal{S}}_{\text{GNN}-\text{Exp}} = \sum_{\mathbf{m} \in \mathcal{G}} g^{\mathcal{S}}_{\alpha}(\mathbf{m}) R^{\mathbf{m}}_{\text{GNN}-\text{Exp}}, \qquad g^{\mathcal{S}}_{\alpha}(\mathbf{m}) = \begin{cases} 0 & \text{if } m_1 \notin \mathcal{S} \wedge m_2 \notin \mathcal{S}, \\ \alpha & \text{if } m_1 \notin \mathcal{S} \vee m_2 \notin \mathcal{S}, \\ 1 & \text{otherwise}, \end{cases} \tag{32}$$

where $\mathbf{m} \in \{1, \ldots, M\}^2$ denotes edges between nodes m_1 and m_2.

Algorithm 2 sGNN-LRP, $\alpha \in (0, 1]$

Input: graph $\mathcal{G}$, subgraph $\mathcal{S}$, discount factor α, # of model layers L.
Output: subgraph relevance score $R^{\mathcal{S}}$
Initialize mask vector $\mathbf{M}_{\alpha}^{\mathcal{S}}$, such that $\mathbf{M}_{\alpha}^{\mathcal{S}}$ is valued 1 on the indices of nodes inside $\mathcal{S}$ and α else.
for $l = 1$ **to** L **do**
  $\mathbf{Z}^{(l)} \leftarrow \mathbf{\Lambda}\mathbf{H}^{(l-1)}$
  $\mathbf{P}^{(l)} \leftarrow \mathbf{Z}^{(l)}\mathbf{W}^{(l)\uparrow}$
  $\mathbf{Q}^{(l)} \leftarrow \mathbf{P}^{(l)} \odot [\rho(\mathbf{Z}^{(l)}\mathbf{W}^{(l)}) \oslash \mathbf{P}^{(l)}]_{cst.}$
  $\mathbf{H}^{(l)} \leftarrow \mathbf{Q}^{(l)} \odot \mathbf{M}_{\alpha}^{\mathcal{S}} + [\mathbf{Q}^{(l)}]_{cst.} \odot (1 - \mathbf{M}_{\alpha}^{\mathcal{S}})$
end for
$y \leftarrow \text{readout}(\mathbf{H}^{(L)})$ with readout function modified according to LRP forward-hook trick.
$R_1 = \sum_{m_0 \in \mathcal{G}} \left\langle \text{Autograd}(y, \mathbf{H}_{m_0}^{(0)}), \mathbf{H}_{m_0}^{(0)} \right\rangle$
Set mask vector $\mathbf{M}_{\alpha}^{\mathcal{S}}$, such that $\mathbf{M}_{\alpha}^{\mathcal{S}}$ is valued 0 on the indices of nodes inside $\mathcal{S}$ and α else.
for $l = 1$ **to** L **do**
  $\mathbf{Z}^{(l)} \leftarrow \mathbf{\Lambda}\mathbf{H}^{(l-1)}$
  $\mathbf{P}^{(l)} \leftarrow \mathbf{Z}^{(l)}\mathbf{W}^{(l)\uparrow}$
  $\mathbf{Q}^{(l)} \leftarrow \mathbf{P}^{(l)} \odot [\rho(\mathbf{Z}^{(l)}\mathbf{W}^{(l)}) \oslash \mathbf{P}^{(l)}]_{cst.}$
  $\mathbf{H}^{(l)} \leftarrow \mathbf{Q}^{(l)} \odot \mathbf{M}_{\alpha}^{\mathcal{S}} + [\mathbf{Q}^{(l)}]_{cst.} \odot (1 - \mathbf{M}_{\alpha}^{\mathcal{S}})$
end for
$y \leftarrow \text{readout}(\mathbf{H}^{(L)})$ with readout function modified according to LRP forward-hook trick.
$R_2 = \sum_{m_0 \in \mathcal{G}} \left\langle \text{Autograd}(y, \mathbf{H}_{m_0}^{(0)}), \mathbf{H}_{m_0}^{(0)} \right\rangle$
$R^{\mathcal{S}} = R_1 - R_2$

Algorithm 3 Model Activation Task

Input: GNN model $f(\cdot)$, input graph $\mathcal{G}$
Output: AUAC
Find the node sequence $\left[m^{(1)}, ..., m^{(M)}\right]$ such that $\sum_{i=1}^{M} R^{\{m^{(1)}, ..., m^{(i)}\}}$ is maximized.
Initialize $\texttt{AUAC} = 0$.
$\mathcal{S} = \varnothing$
for $i = 1, \cdots, |\mathcal{G}|$ **do**
  $\mathcal{S} = \mathcal{S} \cup \{m^{(i)}\}$
  $\texttt{AUAC} = \texttt{AUAC} + f(\mathcal{S})$
end for
$\texttt{AUAC} = \texttt{AUAC}/|\mathcal{G}|$

Algorithm 4 Model Pruning Task

Input: GNN model $f(\cdot)$, input graph $\mathcal{G}$
Output: AUPC
Find the node sequence $\left[m^{(1)}, ..., m^{(M)}\right]$ such that $\sum_{i=1}^{M} |R^{\mathcal{G}\backslash\{m^{(1)}, ..., m^{(i)}\}} - R^{\mathcal{G}}|$ is minimized.
Initialize $\texttt{AUPC} = 0$.
$\mathcal{S} = \varnothing$
for $i = 1, \cdots, M$ **do**
  $\mathcal{S} = \mathcal{S} \cup \{m^{(i)}\}$
  $\texttt{AUPC} = \texttt{AUPC} + |f(\mathcal{G}\backslash\mathcal{S}) - f(\mathcal{G})|$
end for
$\texttt{AUPC} = \texttt{AUPC}/|\mathcal{G}|$