PAGE: Prototype-Based Model-Level Explanations for Graph Neural Networks

Yong-Min Shin, *Student Member, IEEE*, Sun-Woo Kim, and Won-Yong Shin, *Senior Member, IEEE*

***Abstract*—Aside from graph neural networks (GNNs) attracting significant attention as a powerful framework revolutionizing graph representation learning, there has been an increasing demand for explaining GNN models. Although various explanation methods for GNNs have been developed, most studies have focused on *instance-level* explanations, which produce explanations tailored to a given graph instance. In our study, we propose *Prototype-bAsed GNN-Explainer* (PAGE), a novel *model-level* GNN explanation method that explains what the underlying GNN model has learned for graph classification by discovering human-interpretable *prototype graphs*. Our method produces explanations for a given *class*, thus being capable of offering more concise and comprehensive explanations than those of instance-level explanations. First, PAGE selects embeddings of class-discriminative input graphs on the graph-level embedding space after clustering them. Then, PAGE discovers a common subgraph pattern by iteratively searching for high matching node tuples using node-level embeddings via a *prototype scoring* function, thereby yielding a prototype graph as our explanation. Using six graph classification datasets, we demonstrate that PAGE qualitatively and quantitatively outperforms the state-of-the-art model-level explanation method. We also carry out systematic experimental studies by demonstrating the relationship between PAGE and instance-level explanation methods, the robustness of PAGE to input data scarce environments, and the computational efficiency of the proposed prototype scoring function in PAGE.**

***Index Terms*—Embedding, model-level explanation, graph classification, graph neural network (GNN), prototype graph.**

I. Introduction

A. Background and Motivation

GRAPHS are a ubiquitous way to organize a diverse set of complex real-world data such as social networks and molecular structures. Graph neural networks (GNNs) have been widely studied as a powerful means to extract useful features from underlying graphs while performing various downstream graph-related tasks [1]. GNNs are known to have high expressive capability via message passing to effectively learn representations of both nodes and graphs [2], while adopting neural networks as a basic building block to learn the graph structure and node features (attributes).

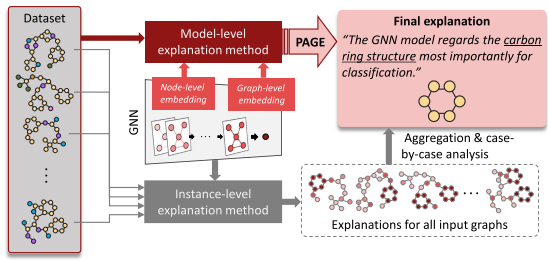


Fig. 1. Comparison between instance-level and model-level explanation methods to capture the general behavior of GNNs.

Despites their strengths, such GNN models lack transparency since they do not offer any human-interpretable explanations of GNN's predictions for a variety of downstream tasks. Thus, fostering *explainability* for GNN models has become of recent interest as it enables a thorough understanding of the model's behavior as well as trust and transparency [3]. Recent attempts to explain GNN models mostly highlight a subgraph structure within a given input graph that contributed most towards the underlying GNN model's prediction [4], [5], [6], [7], [8], [9], [10], [11], [12]. These so-called *instance-level* explanation methods for GNNs provide an in-depth analysis for a given graph instance [13].

Nevertheless, instance-level GNN explanations do not reveal the *general* behavior of the underlying model that has already been trained over an entire dataset, consisting of numerous graphs. In some real-world scenarios, explanations that showcase the decision-making process of GNN models are indeed beneficial. For example, when a black-box GNN model is employed in a real-world setting, it is likely that the model will encounter various instances unobserved during training. In such a case, understanding the general decision-making process of the GNN model will help predict the GNN's behavior in new environments. This motivates us to study *model-level* explanations that aim to interpret GNNs at the model-level.

Fig. 1 visualizes how instance-level explanation methods differ from model-level explanation methods in order to understand the general behavior of the pre-trained GNN model.

Manuscript received 20 July 2023; revised 29 February 2024; accepted 15 March 2024. Date of publication 19 March 2024; date of current version 5 September 2024. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by Korea government (MSIT) under Grant 2021R1A2C3004345 and under Grant RS-2023-00220762. Recommended for acceptance by M. Cheng. *(Corresponding author: Won-Yong Shin.)*

Yong-Min Shin is with the School of Mathematics and Computing (Computational Science and Engineering), Yonsei University, Seoul 03722, South Korea (e-mail: jordan3414@yonsei.ac.kr).

Sun-Woo Kim is with the Kim Jaechul Graduate School of AI, Korea Advanced Institute of Science and Technology (KAIST), Seoul 02455, South Korea (e-mail: kswoo97@kaist.ac.kr).

Won-Yong Shin is with the School of Mathematics and Computing (Computational Science and Engineering), Yonsei University, Seoul 03722, South Korea, and also with the Graduate School of Artificial Intelligence, Pohang University of Science and Technology (POSTECH), Pohang 37673, South Korea (e-mail: wy.shin@yonsei.ac.kr).

Digital Object Identifier 10.1109/TPAMI.2024.3379251

For instance-level explanations, the explanation is provided for each graph instance. Thus, to understand the model on the whole dataset, we need further steps such as extraction of common patterns among explanations at the instance-level and aggregation of the extracted patterns [14]. However, model-level explanation methods directly capture how the pre-trained GNN model behaves by presenting what it has learned from the whole dataset, which is summarized as a form of a human-interpretable subgraph.

As a representative model-level explanation method for GNNs, XGNN [14] trains a graph generator based on reinforcement learning. Since XGNN requires domain-specific knowledge to manually design the reward function, it often fails to produce reliable explanations unless precise rewards are provided according to different types of datasets. Despite numerous practical challenges, designing model-level explanation methods for GNNs remains largely underexplored.

B. Main Contributions

In this paper, we introduce *Prototype-bAsed GNN-Explainer (*PAGE*)*, a novel *post-hoc* model-level explanation method that explains what the underlying GNN model has learned for graph classification by discovering human-interpretable *prototype graphs*. The GNN model's behavior is represented as the prototype graphs by identifying graph patterns learned by the model for a given target class. We formulate our problem guided by the following insight: the common graph pattern that we shall seek is indeed within the training set. Thus, we use the training set as well as representations generated by the pre-trained GNN model as the input to PAGE in order to precisely capture what the model has learned. Fig. 1 illustrates an example where PAGE correctly identifies the benzene structure from the set of molecules as the resulting model-level explanation, while using a set of both node-level and graph-level embeddings through the pre-trained GNN model as input.

We aim to devise our PAGE method according to the following design principles: an explanation module itself should (a) be interpretable and (b) not be accompanied with advanced learning models (i.e., black-box learning models). Towards this end, we design a two-phase method consisting of 1) clustering and selection of embeddings on the graph-level embedding space and 2) discovery of the prototype graph. In Phase 1, we select class-distinctive graphs from the training set by performing clustering on the graph-level embedding space and then choosing k embeddings near the centroid of each cluster. This phase is motivated by claims that 1) graph-level embeddings manifesting common subgraph patterns tend to be co-located on the graph-level embedding space (which is validated empirically) and 2) the selected k graphs (or equivalently, graph-level embeddings) near the centroid can be good representations of the cluster to which they belong. In Phase 2, as our explanation, we discover the prototype graph from the selected k graphs. However, representing the GNN's behavior as a precise subgraph pattern from the selected k graphs is challenging since it requires combinatorial search over all possible k nodes across k different graphs, which should collectively embody what the model has learned. To guide this search, we newly characterize a *prototype scoring* function, which is used for efficiently calculating the matching score among k nodes across k selected graphs using node-level embedding vectors. By applying the function to all possible combinations of k nodes, we iteratively search for high matching node tuples until we discover a subgraph. It is worth noting that the impacts and benefits of leveraging the two types of embedding spaces (i.e., node-level and graph-level embedding spaces) in prototype discovery are two-fold: 1) the behavior of the pre-trained GNN model is encoded to vector representations on low-dimensional embedding spaces, which can be handled readily, and 2) the rich attribute and topological information can be fused into vector representations.

To validate the quality and effectiveness of our PAGE method, we perform comprehensive empirical evaluations using various synthetic and real-world datasets. However, the evaluation of model-level explanations poses another challenge as it has yet been largely underexplored in the literature. As one of the main contributions of our study, we device systematic evaluations for model-level GNN explanations, which are summarized below. First, we perform a *qualitative* evaluation by visualizing the explanations, which demonstrates that PAGE produces prototype graphs similar to the ground truth explanations. Second, by adopting four performance metrics, accuracy, density, consistency and faithfulness [15], with modifications, we carry out the *quantitative* analysis, which (a) shows the superiority of PAGE over XGNN [14], the representative model-level GNN explanation method, and (b) is in agreement with the qualitative assessment. Third, we investigate the relationship between our PAGE method and instance-level explanation methods. To this end, we present two quantities, including the concentration score and relative training gain, and quantitatively assess the extent to which instance-level explanations agree with the prototype graph discovered by PAGE. Fourth, we analyze the robustness of PAGE by visualizing the resultant prototype graphs in a more difficult setting where each training set is composed of only a few available graphs as the input of PAGE; we demonstrate that reasonable ground truth explanations can still be produced. Finally, we empirically validate the effectiveness of our prototype scoring function, which successfully approximates computationally expensive alternatives.

Additionally, we address the key differences between PAGE and XGNN [14]: 1) the use of *interpretable* components without black-box learning modules and 2) *generalization* ability in the sense of applying our method to diverse domains without domain-specific prior knowledge. The main technical contributions of this paper are three-fold and are summarized as follows:

- We propose PAGE, a novel model-level GNN explanation method for graph classification, which is comprised of two phases, 1) clustering and selection of graph-level embeddings and 2) prototype discovery;
- In PAGE, we theoretically and empirically justify the usage of clustering of graph-level embeddings and design a new prototype graph discovery algorithm to capture the general behavior of GNN models;
- Through our systematic evaluations using five graph classification datasets, we comprehensively demonstrate that

PAGE is effective, robust to incomplete datasets, and computationally efficient.

C. Organization and Notations

The remainder of this paper is organized as follows. In Section II, we present prior studies related to our work. In Section III, we explain the methodology of our study, including the basic settings, description of GNN models, our problem formulation, and an overview of our PAGE method. Section IV describes technical details of the proposed method. Comprehensive experimental results are shown in Section V. Finally, we provide a summary and concluding remarks in Section VI.

II. Related Work

We first summarize broader research lines related to explanation methods for deep neural networks, and then we focus on reviewing explanation methods for GNNs.

A. Explanation Methods for Deep Learning

Significant research efforts have been devoted to interpretation techniques that explain deep neural network models on image data. Although existing techniques are commonly partitioned into instance-level and model-level methods, considerable attention has been paid to instance-level explanations , which explain the prediction of a given input instance by discovering salient features in the input through the underlying explainable method. As one of widely used techniques, layer-wise relevance propagation (LRP) [16] proposed to redistribute the model output values towards the input in proportion to the activation values. In addition, Grad-CAM [17] was presented by producing explanations based on a course heatmap highlighting salient regions in the given image.

In contrast to the instance-level explanations, model-level explanation methods aim at offering the interpretability of the underlying model itself. These methods generally generate the optimized input to maximize a certain prediction score (e.g., a class score) produced by a given explanation model. DGN-AM [18] employed a separate neural network capable of inverting the feature representations of an arbitrary layer. Furthermore, PPGN [19] was developed by adding a learned prior to generate realistic high-resolution images exhibiting more diversity.

B. Explanation Methods for GNNs

Despite the great success of GNN models in solving various graph mining tasks, such as node/graph classification [1], much less attention has been yet paid to the study on *explaining* GNN models. For GNN models, explanations in the form of *subgraphs* rather than heatmaps would be appropriate. As a representative work, GNNExplainer [5] was developed to discover a subgraph, including the target node, to be explained by maximizing the mutual information between prediction values of the original graph and the discovered subgraph. GNN-LRP [9] extended the idea of LRP to GNN models to produce higher-order explanations via relevant walks contributing to the model decisions. In [6], SubgraphX identified the most relevant subgraph explaining the model prediction via Monte Carlo tree search using Shapley values as a measure of subgraph importance. On the other hand, instead of directly searching for subgraphs, several studies have learned parameterized models to explain GNN's predictions. PGExplainer [7] proposed a probabilistic graph generative model to collectively explain multiple instances. GraphMask [8] also proposed a post-hoc method that interprets the predictions determining whether (superfluous) edges at every GNN layer can be removed. Furthermore, PGM-Explainer [10] presented a probabilistic graphical model so as to provide an explanation by investigating predictions of GNNs when the GNN's input is perturbed. In RC-Explainer [12], a reinforcement learning agent was presented to construct an explanatory subgraph by adding a salient edge to connect the previously selected subgraph at each step, where a reward is obtained according to the causal effect for each edge addition. Most recently, CF-GNNExplainer [11] presented a counterfactual explanation in the form of minimal perturbation to the input graph such that the model prediction changes.

Although recent efforts have been made to effectively produce *instance-level* explanations for GNN models, *model-level* explanations for GNNs have been largely underexplored in the literature. XGNN [14] was developed only for model-level GNN explanations, where a subgraph pattern that maximizes the target class probability is generated via a reinforcement learning framework. It is worth mentioning that several explanation methods that provide more coarse-grained explanations were recently proposed. ReFine [20] proposed to train global attribution that captures class-wise patterns, while also training local attribution by fine-tuning on the given instance. CGE [21] jointly considered attribution on the input as well as the intermediate neural activation in the underlying GNN model for explanations, and found a large overlap of neural activations within instances from the same class. However, the class-wise patterns still require a specific instance as input to be expressed, and cannot map such explanations as a standalone structural representation, which is required in model-level explanation methods.

C. Discussion

Apart from most explanation methods that operate on the instance-level, we aim to design model-level interpretations that offers high-level insights as well as generic understandings of the underlying model mechanism. Critically, XGNN requires domain knowledge to provide appropriate rewards in the graph generation procedure based on reinforcement learning. If the reward is not adequately designed, then the generated representative graph would be hardly realistic. Furthermore, the usage of such reinforcement learning frameworks introduces another black-box model for explanation, which is a sub-optimal choice. This motivates us to design a more interpretable and convenient model-level method that does not require domain-specific knowledge and learning modules.

We note that the prototype discovery in Phase 2 of our PAGE method can be viewed as a form of graph matching [22], [23], [24], [25], since graph matching attempts to compare common substructures among multiple graph instances. Although graph

matching methods can be utilized in Phase 2 to find the prototype candidates, they require a substantial amount of nontrivial modifications since they cannot incorporate the instance-level embedding vectors, nor can they guarantee that a graph with a single connected component is returned, which thereby severely reduces the quality of the discovered prototype candidates.

III. Methodology

In this section, we first describe our problem setting with the objective. Then, we present the overview of our PAGE method as a model-level explanation for GNNs.

A. Settings and Basic Assumptions

Let us assume that we are given a set of n graphs $\mathcal{G} = \{G_i\}_{i=1}^{n}$ with node attributes. Each graph is denoted as $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{X}_i)$, where $\mathcal{V}_i = \{v_i^1, \ldots, v_i^{|\mathcal{V}_i|}\}$ is the set of nodes with cardinality $|\mathcal{V}_i|$ and $\mathcal{E}_i$ is the set of edges between pairs of nodes in $\mathcal{V}_i$. We assume each graph G_i to be undirected and unweighted without self-edges or repeated edges. We define $\mathcal{X}_i = \{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^{|\mathcal{V}_i|}\}$ as the set of feature (attribute) vectors of nodes in $\mathcal{V}_i$, where $\mathbf{x}_i^j \in \mathbb{R}^d$ is the feature vector of node v_i^j in G_i and d is the number of features per node.

We associate each graph G_i with a label y_i, where y_i is an element in the set $\mathcal{C} = \{c_1, \ldots, c_m\}$, which means that we address an m-class graph classification problem. A pre-trained GNN model $f_{\text{GNN}} : \mathcal{G} \rightarrow \mathcal{C}$ for graph classification corresponds to the model of interest to be explained. Additionally, the number of ground truth explanations per class is assumed to be available for each dataset. While acquiring such knowledge a priori from each dataset may require additional work, we do not include this task in our study to focus primarily on prototype discovery.

B. GNN Models for Graph Classification

In this subsection, we give a brief review of GNN models for graph classification.

First, we show a general form of message passing in GNNs [26], in which we iteratively update the representation of a node by aggregating representations of its neighbors using two functions. Specifically, given a graph G_i, the underlying GNN model produces the latent representation vector $\mathbf{u}_v^p$ of node $v \in \mathcal{V}_i$ at the p-th GNN layer, where $p \in \{1, \ldots, P\}$.[1] Formally, the p-th layer of the GNN model updates the latent representation $\mathbf{u}_v^p$ of each node v in $\mathcal{V}_i$ by passing through two phases, including the message passing phase and the update phase, which are expressed as $\mathbf{m}_v^{p+1} = \sum_{v' \in \mathcal{N}_{G_i}(v)} M_p(\mathbf{u}_v^p \mathbf{u}_{v'}^p)$ and $\mathbf{u}_v^{p+1} = U_p(\mathbf{u}_v^p, \mathbf{m}_v^{p+1})$, respectively, where $\mathcal{N}_{G_i}(v)$ indicates the set of neighbors of node v in the given graph G_i. Here, the message function $M_p(\cdot, \cdot)$ and the update function $U_p(\cdot, \cdot)$ can be specified by several types of GNN models.[2] After passing through all P layers, we acquire the final representation as the *node-level* embedding vector of node v_i^j, which is denoted as $\mathbf{h}_i^j = \mathbf{u}_{v_i^j}^P \in \mathbb{R}^b$ for the dimension b of each vector.

Second, we describe how to produce the *graph-level* embedding vector $\mathbf{h}_{G_i}$ of G_i. To this end, a readout function $R(\cdot)$ collects all node-level embedding vectors $\mathbf{h}_i^j$ for $v_i^j \in \mathcal{V}_i$ and is formally expressed as $\mathbf{h}_{G_i} = R(\mathcal{H}_{G_i})$, where $\mathcal{H}_{G_i} = \{\mathbf{h}_i^j | v_i^j \in \mathcal{V}_i\}$, indicating the set of embedding vectors of nodes in $\mathcal{V}_i$. In practice, differentiable permutation-invariant functions such as the summation and average [2] are typically used. Then, the set of calculated graph-level embedding vectors over all graphs $\mathcal{G}$ is fed into a classifier to perform graph classification.

C. Problem Formulation

The objective of our study is to provide a post-hoc *model-level* explanation by discovering the most distinctive graph patterns that the underlying GNN model has learned during training for graph classification. Towards this end, we present PAGE, a new model-level explanation method along with human-interpretable *prototype* graphs. For a target class $c \in \mathcal{C}$ to which a given graph belongs and a pre-trained GNN model f_{GNN}, PAGE aims to discover a set of prototype graphs, $\mathcal{G}_{\text{proto}}^{(c)}$, each of which represents the graph pattern such that the model f_{GNN} most likely predicts the target class c. Formally, the model-level explanation function $f_{\text{model}}^X(\cdot, \cdot)$ in PAGE is expressed as:

$$\mathcal{G}_{\text{proto}}^{(c)} = f_{\text{model}}^X(c; f_{\text{GNN}}). \tag{1}$$

Note that the model-level explanation differs from the instance-level explanation, which provides an explanation with respect to a given graph instance $G_i \in \mathcal{G}$ rather than a target class.

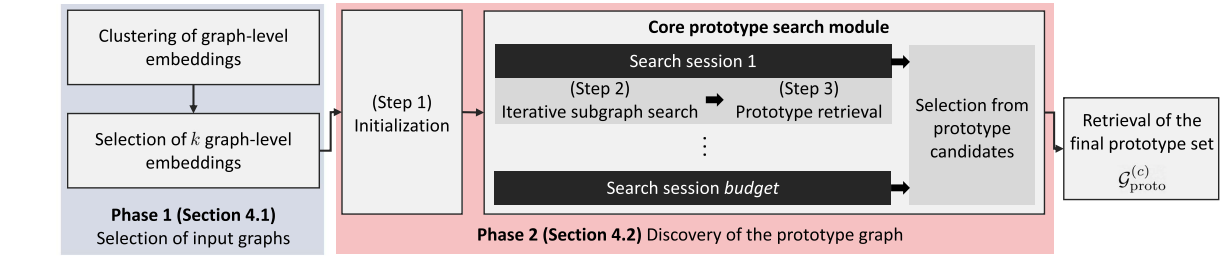
D. Overview of Our PAGE Method

In this subsection, we explain our methodology along with the overview of the proposed PAGE method, which consists of the following two phases: 1) selection of input graphs that precisely represent the given class c through clustering and selection of graph-level embeddings, and 2) discovery of the prototype graph set $\mathcal{G}_{\text{proto}}^{(c)}$.

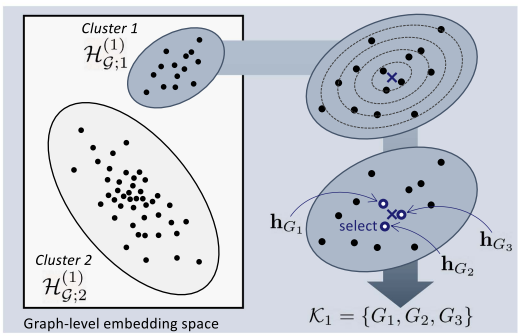
Before describing each phase above, let us briefly state the core difference from XGNN [14], the state-of-the-art model-level explanation method for GNNs. While XGNN was designed to *generate* the prototype graph, PAGE attempts to *discover* the set of prototype graphs, $\mathcal{G}_{\text{proto}}^{(c)}$, using underlying graphs, $\mathcal{G}$, by turning our attention to both the graph- and node-level embedding spaces, where the impacts and benefits of leveraging the two types of embedding spaces are two-fold. First, the behavior of the pre-trained GNN model is encoded to vector representations on low-dimensional embedding spaces, thus facilitating many network analysis tasks. Second, the rich attribute and structural information of $\mathcal{G}$ can be fused efficiently and effectively into vector representations in conducting the prototype discovery. In particular, the set of graph-level embedding vectors can be used to reduce the prototype graph search space within the set $\mathcal{G}$ by

[1] To simplify notations, v_i^j will be written as v unless dropping the superscript j and the subscript i causes any confusion.

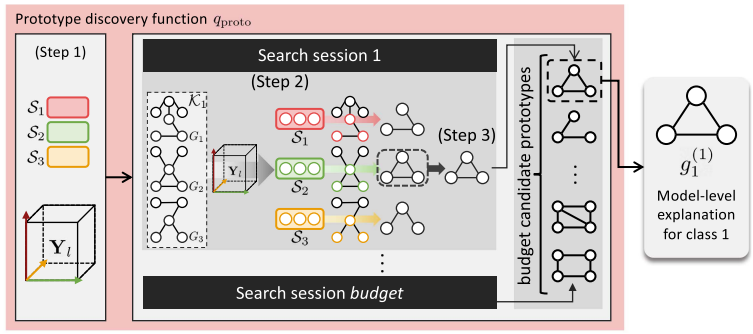
[2] As an example, GCN [27] can be implemented by using $M_p(\mathbf{u}_v^p, \mathbf{u}_w^p) = (\sqrt{\deg(v)\deg(w)})^{-1/2}\mathbf{u}_v^p$ and $U_p = \sigma(W^p \cdot \mathbf{m}_v^{p+1})$, where $\deg(\cdot)$ is the degree of each node and $\sigma(\cdot)$ is the activation function.



(a) A schematic overview of our PAGE method.



(b) Phase 1 of PAGE when $k = 3$.



(c) Phase 2 of PAGE when $k = 3$.

Fig. 2. Schematic overview overview and an illustrative example of our PAGE method.

selecting only a small number of embedding vectors that are most representative.

Now, we describe each phase of our PAGE method as follows. First, we select a small number of graphs from the input that represent the given class c by performing clustering on the graph-level embedding space (see the left part in Fig. 2(a)). This phase is motivated by the observation that graph-level embedding vectors revealing common subgraph patterns tend to be co-located on the graph-level embedding space (which will be empirically shown in Section IV-A2). The clustering phase also provides a practical advantage, where it provides a natural anchor point to which we select graph-level embedding vectors (and their corresponding graphs) as the centroid of each cluster. For a target class c, we focus on $\mathcal{H}_{\mathcal{G}}^{(c)}$, which is defined as a subset of graph-level embedding vectors such that the model f_{GNN} predicts as class c. In this phase, we discover L clusters using a Gaussian mixture model (GMM) [28] for a pre-defined parameter L. Consequently, we are able to map each graph-level embedding vector to one of L clusters. After acquiring L clusters, we select the k-nearest neighbors (kNNs) from each cluster's centroid on the graph-level embedding space for a pre-defined parameter k. Since centroids are typically located near the center of each cluster, graph-level embedding vectors near centroids can be considered good representations of the clusters to which they belong. Subsequently, we acquire $\mathcal{K}_l$, which represents the subset of underlying graphs in $\mathcal{G}$ corresponding to k selected graph-level embedding vectors. That is, we retrieve a subset of graphs from the selected graph-level embedding vectors. Fig. 2(b) illustrates an example of retrieving $\mathcal{K}_1 = \{G_1, G_2, G_3\}$ using the 3-nearest neighbors $\mathbf{h}_{G_1}$, $\mathbf{h}_{G_2}$, and $\mathbf{h}_{G_3}$ from the centroid when $k = 3$.

Second, we turn to discovering the prototype graph (see the right part in Fig. 2(a)). For the ease of explanation, we focus only on the l-th cluster. Given $\mathcal{K}_l = \{G_1, \ldots, G_k\}$ for the l-th cluster, we aim to discover a common subgraph pattern within the subset $\mathcal{K}_l$, which serves as the prototype graph, denoted as $g_l^{(c)}$, in the set $\mathcal{G}_{\text{proto}}^{(c)}$. To this end, we compare nodes across k different graphs in $\mathcal{K}_l$ in order to select the most probable nodes according to the *matching score*. The matching score, which is calculated among k nodes, takes advantage of the structural and attribute information embedded by the model f_{GNN}, which measures how high the matching score for the node-level embedding vectors of nodes $v_1^{i_1}, \ldots, v_k^{i_k}$ is. After we use the matching scores to select the set of nodes to be included in the explanation, we extract k induced subgraphs from $\mathcal{K}_l$ by only taking into account the selected nodes. We can then calculate the output probability p_{GNN} for each subgraph extracted from k graphs in the set $\mathcal{K}_l$ by feeding each into f_{GNN}, where the subgraph with the highest p_{GNN} is selected. We further repeat this procedure with a different search process during multiple search sessions alongside a given *search budget*. Finally, the subgraph exhibiting the highest p_{GNN} among the ones found across all search sessions is retrieved as the resulting prototype graph, which provides an intuitive model-level explanation for GNN's prediction given the target class c. Fig. 2(c) illustrates an example of retrieving the final prototype graph $g_1^{(1)}$ based on the prototype discovery function q_{proto} using $\mathcal{K}_1 = \{G_1, G_2, G_3\}$.

IV. PAGE: Proposed Method

In this section, we elaborate on PAGE, the proposed post-hoc model-level GNN explanation method for graph classification.

We first describe how to cluster and select graph-level embeddings for the subset selection of input graphs in detail. We also provide the theoretical and empirical validation for our clustering. Then, we present how to discover prototype graphs.

A. Clustering and Selection of Graph-Level Embeddings (Phase 1)

1) Methodological Details: We assume that the parameters of a GNN model f_{GNN} are trained by a set of graphs, $\mathcal{G}$. Given a target class $c \in \mathcal{C}$, we obtain a set of graph-level embedding vectors, $\mathcal{H}_{\mathcal{G}}^{(c)}$, through a feed-forward process of GNN.

Then, we estimate the parameters of the Gaussian mixture distribution to fit the embedding vectors $\mathcal{H}_{\mathcal{G}}^{(c)}$. More precisely, we apply an expectation maximization (EM) algorithm [29] to estimate the parameters $\{(\pi_l, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\}_{l=1}^{L}$ with the mean vector $\boldsymbol{\mu}_l$ and the covariance matrix $\boldsymbol{\Sigma}_l$ for the l-th cluster of the Gaussian mixture distribution $p(\mathbf{h}) = \sum_{l=1}^{L} \pi_l \mathcal{N}(\mathbf{h}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$ due to the theoretical guarantees of monotonically increasing likelihood and convergence [30] as well as the robustness to noisy input samples [31].[3] The EM iteration alternates between performing the E-step and M-step as follows. In the E-step, we calculate $\gamma_{nl}^{(t)}$, which represents the posterior probability of the n-th vector in $\mathcal{H}_{\mathcal{G}}^{(c)}$ being assigned to the l-th cluster and is expressed as $\gamma_{nl}^{(t)} = \frac{\pi_l \mathcal{N}(\mathbf{h}_{G_n}|\boldsymbol{\mu}_l^{(t)}, \boldsymbol{\Sigma}_l^{(t)})}{\sum_{i=1}^{L} \pi_i \mathcal{N}(\mathbf{h}_{G_n}|\boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i^{(t)})}$, where the superscript (t) indicates the EM iteration index. In the M-step, we re-estimate the GMM parameters, $\boldsymbol{\mu}_l^{(t+1)} = \frac{1}{N_l} \sum_{n=1}^{|\mathcal{H}_{\mathcal{G}}^{(c)}|} \gamma_{nl}^{(t)} \mathbf{h}_{G_n}$, $\boldsymbol{\Sigma}_l^{(t+1)} = \frac{1}{N_l} \sum_{n=1}^{|\mathcal{H}_{\mathcal{G}}^{(c)}|} \gamma_{nl}^{(t)} (\mathbf{h}_{G_n} - \boldsymbol{\mu}_l^{(t+1)})(\mathbf{h}_{G_n} - \boldsymbol{\mu}_l^{(t+1)})^T$, where $N_l = \sum_{n=1}^{|\mathcal{H}_{\mathcal{G}}^{(c)}|} \gamma(z_{nl})$. Using the estimated parameters, we assign each graph-level embedding vector $\mathbf{h}_{G_i}$ to the cluster with the highest probability $\gamma_{nl}^{(t)}$. In other words, we divide $\mathcal{H}_{\mathcal{G}}^{(c)}$ into L disjoint subsets $\mathcal{H}_{\mathcal{G};1}^{(c)}, \ldots, \mathcal{H}_{\mathcal{G};L}^{(c)}$, where $\mathcal{H}_{\mathcal{G};l}^{(c)}$ contains graph-level embedding vectors assigned to the l-th cluster by the GMM.

Next, we select the kNNs from each cluster's centroid on the graph-level embedding space in order to find the set of k graphs, $\mathcal{K}_l$, for each cluster. Since the covariance matrix is available for each cluster, we are able to utilize the Mahalanobis distance that takes into account both the correlations and the shape of each cluster on the graph-level embedding space. We sort the graph-level embedding vectors $\mathbf{h}_{G_i}$ in descending order with respect to the Mahalanobis distance from the mean vector $\boldsymbol{\mu}_l$ within the l-th cluster [28]: $((\mathbf{h}_{G_i} - \boldsymbol{\mu}_l)^T \boldsymbol{\Sigma}_l^{-1} (\mathbf{h}_{G_i} - \boldsymbol{\mu}_l))^{1/2}$. Then, for each cluster, we select k graph-level embedding vectors that are closest to $\boldsymbol{\mu}_l$ and their corresponding input graphs $\mathcal{K}_l$ in $\mathcal{G}$.

2) Theoretical and Empirical Validation: In Section IV-A2, we aim at justifying the usage of clustering of graph-level embeddings in PAGE via our theoretical and empirical validation. Our study basically presumes that graphs containing different prototypes are highly likely to be mapped into different vector representations on the graph-level embedding space, even though they belong to the same class. To validate this claim, we first provide a theoretical foundation that connects GNNs and the *Weisfeiler-Lehman (WL) kernel* [32]. We then empirically validate the above claim by visualizing graph-level embeddings for a benchmark dataset.

We begin by describing the WL kernel [32]. It initially colors each node of a given graph by the node attribute and iteratively refines the node coloring in rounds by taking into account the colors in the neighbors of each node until stabilization. We formally state this by assuming that a graph G_i is given along with the initial color for each node. A node coloring $c^{(p)}$ is a function that bijectively maps each node $v \in \mathcal{V}_i$ in G_i to a unique color that has not been used in previous iterations, where $p \geq 0$ indicates the iteration index. For the target node v, the WL kernel iteratively updates $c^{(p)}(v)$ by recoloring over the multiset of neighborhood colors, i.e., $\{\{c^{(p-1)}(v')|v' \in \mathcal{N}_{G_i}(v)\}\}$, where $\mathcal{N}_{G_i}(v)$ indicates the set of neighbors of node v in G_i.[4] Now, we are ready to establish the following theorem, which verifies that there exists a GNN model equivalent to the WL kernel as long as node-level embeddings are concerned.

Theorem 1 (Morris et al. 2019): Let $f_{\text{GNN}}^{(p)}$ denote the function that returns node-level vector representations at the p-th GNN layer in f_{GNN}. Then for all $p \geq 0$, there exists a GNN model *equivalent* to a node coloring such that $f_{\text{GNN}}^{(p)}(u) = f_{\text{GNN}}^{(p)}(v)$ if and only if $c^{(p)}(u) = c^{(p)}(v)$ for every $u, v \in \mathcal{V}_i$.

From Theorem 1, the node-level embedding vector $\mathbf{h}_i^j$ of node v_i^j is equivalent to $c^{(P)}(v_i^j)$. Next, we explore the relationship between a GNN model and the WL kernel in discovering *graph-level* embeddings. To this end, we first acquire a graph feature vector of G_i, denoted as $\mathbf{l}_{G_i} = [n_i^1, \ldots, n_i^s]$, where s is the number of unique node colors and n_i^l is the number of nodes for which color l has been assigned for $l = 1, \ldots, s$. The graph feature vector $\mathbf{l}_{G_i}$ is used to compute similarities between pairs of graphs. The following corollary states the equivalence between the GNN model and the WL kernel in terms of graph-level embeddings.

Corollary 2: Suppose that two graph feature vectors $\mathbf{l}_{G_a}$ and $\mathbf{l}_{G_b}$ are obtained via color refinement for two graphs G_a and G_b, respectively. Then, for a GNN model with P layers such that $c^{(P)}$ and $f_{\text{GNN}}^{(P)}$ are equivalent, $\mathbf{l}_{G_a} = \mathbf{l}_{G_b}$ implies $R(\mathcal{H}_{G_a}) = R(\mathcal{H}_{G_b})$, when $R(\cdot)$ is either the summation or average function.

Proof: From Theorem 1, there exists a bijective function $\nu : c^{(P)}(v_i^j) \rightarrow \mathbf{h}_i^j$ for node $v_i^j \in \mathcal{V}_i$, where i is *either* a or b. Moreover, $\mathbf{l}_{G_a} = \mathbf{l}_{G_b}$ indicates that both the number of nodes and the distribution of node colors over each graph are the same for G_a and G_b. Due to the fact that each node-level embedding vector $\mathbf{h}_i^j$ can be regarded as a bijection ν of a node color, the distributions of unique node-level embedding vectors for G_a and G_b are identical. In this context, if the readout function $R(\cdot)$ is

[3] Although the convergence of EM is slow (i.e., linear convergence) in approaching the true parameters, we empirically showed that the runtime of the EM algorithm in Phase 1 is not a bottleneck.

[4] A multiset is a generalized version of a set that allows multiple instances for each of its elements, where the multiplicity of each element in the multiset corresponds to the number of instances.

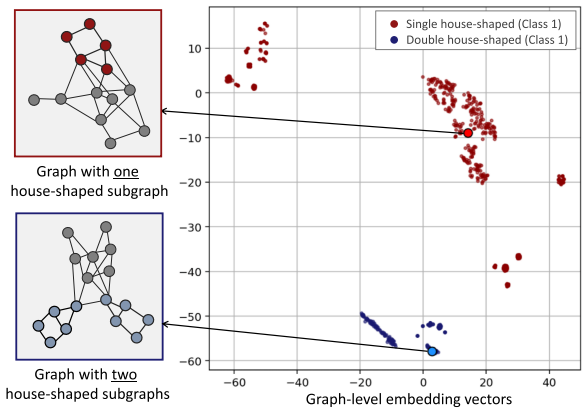


Fig. 3. Visualization of graph-level embeddings for the BA-house dataset.

the summation, then we have

$$\sum_{j=1}^{|\mathcal{V}_a|} \mathbf{h}_a^j = \sum_{j'=1}^{|\mathcal{V}_b|} \mathbf{h}_b^{j'}, \tag{2}$$

thus resulting in $R(\mathcal{H}_{G_a}) = R(\mathcal{H}_{G_b})$, since the summation is permutation-invariant. When $R(\cdot)$ is given by the average function, this corollary can also be proven by dividing each side of (2) by the number of nodes. This completes the proof of this corollary. □

From Corollary 2, one can expect that GNNs will behave as WL kernels in the sense of differentiating graphs (i.e., distinguishing non-isomorphic graphs).

Next, we empirically validate our claim that GNNs can differentiate graphs containing different prototypes.

Example 1: Fig. 3 empirically demonstrates such a tendency by visualizing the graph-level embedding vectors for the BA-house dataset via t-SNE [33] in which there are multiple graphs with and without the house-shaped subgraph(s) (refer to Section V-A for more description of BA-house). Here, we train a GNN model to perform a graph classification task that classifies whether an input graph contains the house-shaped subgraph(s) (Class 1) or not (Class 0). However, as each graph in Class 1 may have either one or two house-shaped subgraphs, we are interested in further investigating whether graphs with different numbers of house-shaped subgraphs, belonging to Class 1, are separately embedded into the graph-level embedding space. Although the GNN model is trained while being unaware of the existence of such prototypes, we observe that the GNN model embeds input graphs with different prototypes to distant clusters in the graph-level embedding space (see red and blue points in Fig. 3).

Through this empirical validation, we conclude that GNNs are indeed capable of differentiating graphs with different prototypes as in the WL kernel.

B. Prototype Discovery (Phase 2)

In this section, we turn our attention to describing our prototype discovery algorithm q_{proto}, which calculates the matching score by comparing k nodes across k different graphs in the set $\mathcal{K}_l = \{G_1, \ldots, G_k\}$. Since we iteratively discover a prototype graph $g_l^{(c)}$ for each cluster $l \in \{1, \ldots, L\}$, we concentrate only on obtaining $g_l^{(c)}$ for the l-th cluster in this subsection. The prototype discovery phase consists of the initialization step and the core prototype search module that is made up of two steps. The mechanism of the core prototype search module for a certain search session is illustrated in Fig. 4 where the parameter k in kNN is given by 3.

Remark 1: It is possible to employ alternatives (e.g., using graph matching) to achieve our goal of discovering prototype graphs. Nonetheless, when we adopted the graph matching method in [24], [25] to discover prototype graph candidates, we empirically found that the resulting candidates are less likely to include the ground truth explanation, if not unrecognizable at all. This implies that prototype discovery based on alternative approaches may require major modifications for satisfactory performance.

1) Initialization (Step 1): The initialization step involves the creation of an order-k matching tensor $\mathbf{Y}_l$ and the creation of empty node lists. We first describe how to create $\mathbf{Y}_l$. From the set of node-level embedding vectors, $\mathcal{H}_{G_i} \in \mathbb{R}^{|\mathcal{V}_i| \times b}$ for each $G_i \in \mathcal{K}_l$, we are interested in calculating the order-k tensor $\mathbf{Y}_l$, where each element of $\mathbf{Y}_l$ is the matching score among k nodes. To consider the node-level embedding vectors from f_{GNN} to efficiently calculate the matching score, we formally define a new scoring function:

Definition 1 (Prototype Scoring Function): Given a set of b-dimensional vectors $\{\mathbf{v}_i\}_{i=1}^k$, we define the *prototype scoring* function $s : \mathbb{R}^b \times \cdots \times \mathbb{R}^b \to \mathbb{R}$ as follows:

$$s(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \mathbf{1}^T (\mathbf{v}_1 \odot \cdots \odot \mathbf{v}_k), \tag{3}$$

where $\odot$ indicates the element-wise product and $\mathbf{1} \in \mathbb{R}^b$ is the all-ones vector.

We now focus on calculating each element of $\mathbf{Y}_l$ by using the node-level embeddings as input of the prototype scoring function s. To this end, we first denote $\mathbf{X}_l \in \mathbb{R}^{|\mathcal{V}_1| \times \cdots \times |\mathcal{V}_k|}$ as another order-k tensor such that $\mathbf{X}_l[i_1, \ldots, i_k] \triangleq s(\mathbf{h}_{G_1}^{i_1}, \ldots, \mathbf{h}_{G_k}^{i_k})$. Each element of $\mathbf{X}_l$ represents a matching score that encodes the likelihood that k nodes, each of which comes from the corresponding k graphs in the set $\mathcal{K}_l$, match well on the node-level embedding space according to the GNN model f_{GNN}. In order to normalize $\mathbf{X}_l$ and encode additional information, we then obtain $\mathbf{Y}_l$ by going through two operations $u_{k'}^1(\cdot)$ and $u_{k'}^2(\cdot)$ on $\mathbf{X}_l$ for $k' \in \{1, \ldots, k\}$ that modify the values of the input tensor while not changing the dimensions: 1) $u_{k'}^1(\mathbf{X}_l)$ applies the softmax normalization along the k'-th dimension; and 2) $u_{k'}^2(\mathbf{X}_l)$ returns the average matching score along the k'-th dimension and is expressed as $\sigma(\sum_{i_{k'}=1}^{|\mathcal{V}_{k'}|} \mathbf{X}_l[\cdots, i_{k'}, \cdots])/|\mathcal{V}_{k'}|$, where $\sigma(\cdot)$ is the sigmoid function. Finally, we are capable of calculating $\mathbf{Y}_l$ by taking the average of the element-wise products of $u_{k'}^1(\mathbf{X}_l)$ and $u_{k'}^2(\mathbf{X}_l)$ over all dimensions:

$$\mathbf{Y}_l = \frac{1}{k} \left(\sum_{k'=1}^{k} u_{k'}^1(\mathbf{X}_l) \odot u_{k'}^2(\mathbf{X}_l) \right). \tag{4}$$

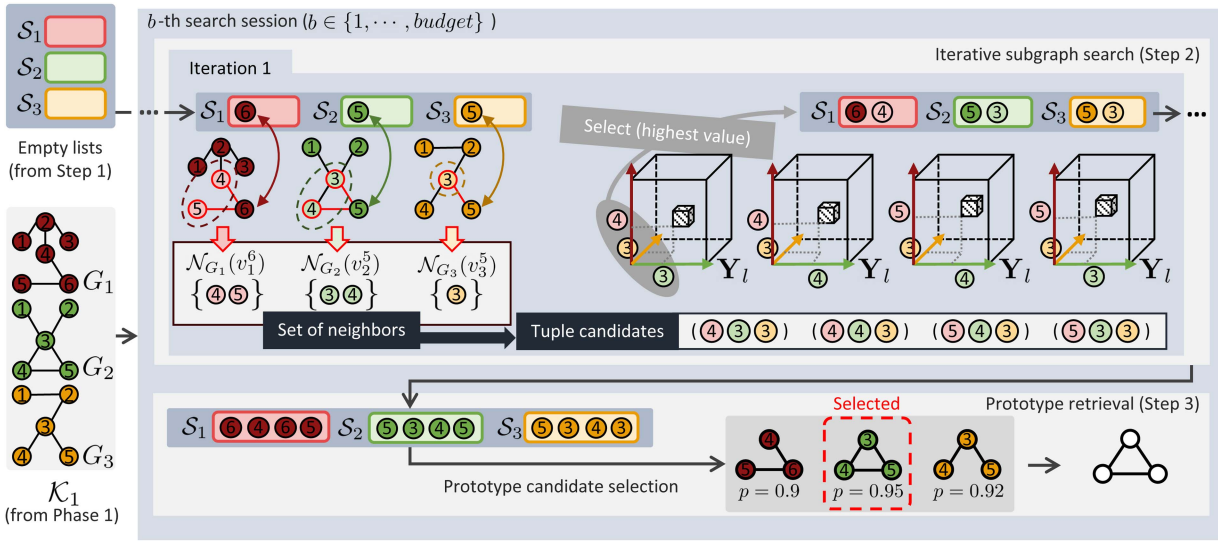


Fig. 4. Example that illustrates the mechanism of the core prototype search module for a certain search session when $k = 3$.

For our matching score calculation, we address the following computational complexity issues.

Remark 2: Although the inner product is a natural way of encoding similarities between two node-level embedding vectors, its extension to the case of $k \geq 3$ is not straightforward. Alternative designs of the prototype scoring function include, but are not limited to, the approach for taking the average of the inner products between each vector and the arithmetic mean vector. This approach requires the calculation of all possible inner products between pairs of k embedding vectors, which is inefficient because such pairwise comparisons take $\mathcal{O}(k^2)$. On the other hand, it is possible to calculate the matching score among k nodes with a linear scaling of k (i.e., $\mathcal{O}(k)$) by using the prototype scoring function s in (3), which is much more efficient. Additionally, the prototype scoring function based on such an alternative approach does not naturally boil down to the inner product when $k = 2$. Moreover, we empirically found that the output of PAGE using such an alternative is shown to be less stable for prototype discovery.

Remark 3: To reduce the search space, we can exclude node tuples in the matching tensor $\mathbf{Y}_l$ such that the feature vectors of the corresponding nodes do not coincide. To achieve this, we post-process $\mathbf{Y}_l$ by assigning zeros to the elements of $\mathbf{Y}_l$, where the feature vectors of nodes are not identical to each other. For example, suppose that $k = 3$ along with the order-3 matching tensor $\mathbf{Y}_l$. For each element $\mathbf{Y}_l[i, j, m]$, we examine the feature vectors of nodes, i.e., $\mathbf{x}_1^i$, $\mathbf{x}_2^j$, and $\mathbf{x}_3^m$. We do not modify the value of $\mathbf{Y}_l[i, j, m]$ if $\mathbf{x}_1^i = \mathbf{x}_2^j = \mathbf{x}_3^m$; we assign zeros to $\mathbf{Y}_l[i, j, m]$ otherwise. Thus, it is possible to avoid searching for such zero-injected node tuples in $\mathbf{Y}_l$. This post-processing does not deteriorate the performance of PAGE while substantially reducing the computational complexity of prototype search.

Next, we create k empty node lists $\mathcal{S}_1, \ldots, \mathcal{S}_k$ in which node indices will be included during the iterative search. As illustrated in the top-left of Fig. 4, for $k = 3$, three empty lists $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ are being initialized. The node indices in each list after the final iteration correspond to the nodes in the prototype graph $g_l^{(c)}$.

2) Core Prototype Search Module: For the remaining part of our prototype discovery phase, we would like to materialize the prototype discovery function q_{proto}, which is designed to extract common subgraph patterns in a reliable manner. In a nutshell, as depicted in Fig. 2(a), we run multiple search sessions, each producing one prototype graph candidate. After running all search sessions, we feed each prototype candidate to the underlying model f_{GNN}, and acquire the one with the highest output probability p_{GNN} as the final prototype graph. This ensures that we leverage various subgraphs as candidates for the final prototype graph, thereby increasing the chances of retrieving a trustworthy prototype graph that conforms to the underlying GNN model.

More concretely, we introduce a search budget, denoted as the variable $budget$, which determines the number of search sessions that we run. Each search session is composed of iterative subgraph search (Step 2) and prototype retrieval (Step 3). In other words, for a given *budget*, we run up to *budget* sessions for prototype search. When the prototype search is over for all sessions, we select the one having the highest output probability p_{GNN} among the $budget$ discovered subgraphs as the final prototype graph $g_l^{(c)}$ for the target class c.

3) Iterative Subgraph Search (Step 2): We now describe the iterative subgraph search step in each search session. In this step, we iteratively find k-tuples of node indices to be inserted into the created k node lists. Each iteration starts by collecting all possible k-tuples to be selected. In Iteration 0 (i.e., initial search), all combinations of k nodes across k different graphs in the set $\mathcal{K}_l$ are valid candidates for selection. For the rest of the iterative search process, we construct node sequences by traversing for each graph; that is, combinations only from neighbors for each of the k selected nodes in the previous iteration

are valid candidates for selection (See the block of Iteration 1 in Fig. 4 for $k = 3$). Selection among the valid candidates for the k-tuples is determined based on the corresponding value in $\mathbf{Y}_l$. Each k-tuple with the highest matching score is chosen out of possible combinations from the sets of neighbors of already selected nodes during the iteration process, except the initial search. For the initial search, the k-tuple with the b-th highest matching score is selected, where $b \in \{1, \ldots, budget\}$ indicates the index of the search session described in Section IV-B2. We describe this step more concretely when $k = 3$ as follows.

Example 2. As illustrated in Fig. 4, in our example, suppose that $k = 3$ and node indices 6, 5, and 5 are in the node lists $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, respectively, in Iteration 0 (initialization). In Iteration 1, we focus only on the set of neighbors for each node, which are $\mathcal{N}_{G_1}(v_1^6) = \{4, 5\}$, $\mathcal{N}_{G_2}(v_2^5) = \{3, 4\}$, and $\mathcal{N}_{G_3}(v_3^5) = \{3\}$, where $\mathcal{N}_{G_i}(v)$ is the set of neighbors of node v in G_i. Then, all possible candidates are the Cartesian product of the three sets of neighbors: $\mathcal{N}_{G_1}(v_1^6) \times \mathcal{N}_{G_2}(v_2^5) \times \mathcal{N}_{G_3}(v_3^5) = \{4, 5\} \times \{3, 4\} \times \{3\}$. In consequence, the possible node candidates are given by (4,3,3), (4,4,3), (5,3,3), and (5,4,3) as depicted in Iteration 1.

Each node index of the selected k-tuple is added to each node list $\mathcal{S}_1, \ldots, \mathcal{S}_k$. After selection, we update $\mathbf{Y}_l$ by multiplying a certain decay factor along each dimension from the selected k-tuple.[5] The whole subgraph search iteration terminates when either the allowed maximum number of iterations is reached or the size of the resultant subgraph exceeds a pre-defined value.

4) Prototype Retrieval (Step 3): As the next step, we are capable of discovering k subgraphs, each of which is retrieved from the node list $\mathcal{S}_i$ for G_i. In Step 3 of prototype discovery for each search session, we describe how to retrieve a candidate of the final prototype graph by evaluating the output probabilities p_{GNN} for each discovered subgraph from the list $\mathcal{S}_i$. Specifically, we feed each subgraph as input to the model f_{GNN} to get p_{GNN} and choose the subgraph with the highest p_{GNN} among k subgraphs (see the bottom-right in Fig. 4 for $k = 3$).

As we run *budget* search sessions, we acquire *budget* prototype candidates. We retrieve the final prototype graph $g_l^{(c)}$ by selecting the prototype candidate with the highest output probability p_{GNN} among *budget* candidates.

V. Experimental Evaluation

In this section, we first describe synthetic and real-world datasets used in the evaluation. We also present the benchmark GNN explanation method for comparison. After describing our experimental settings, we comprehensively evaluate the performance of PAGE and benchmark methods. The source code for PAGE is made publicly available online.[6]

A. Datasets

In our study, two synthetic datasets, including the BA-house and BA-grid datasets, and four real-world datasets, including the Solubility, MUTAG, Benzene, and MNIST-sp datasets, are used for the evaluation of our proposed PAGE method. The ground truth explanations (i.e., ground truth prototypes) as well as the ground truth labels for graph classification are available for all datasets. The main statistics of each dataset are summarized in Table I. We describe important characteristics of the datasets.

TABLE I
Statistics of the Six Datasets and Test Performance of the GNN Model Trained for Each Dataset, Where n, $\sum_i \mathcal{V}_i$, $\sum_i \mathcal{E}_i$, and TP Denote the Number of Graphs, the Total Number of Nodes, the Total Number of Edges, and the Test Performance of the GNN Model, Respectively

Dataset	n	$\sum_i \mathcal{V}_i$ (Avg.)	$\sum_i \mathcal{E}_i$ (Avg.)	TP
BA-house	2,000	21,029 (10.51)	62,870 (31.44)	1.000
BA-grid	2,000	29,115 (14.56)	91,224 (45.61)	0.9583
Benzene	12,000	246,993 (20.58)	523,842 (43.65)	0.9444
MUTAG	4,337	131,488 (30.32)	266,894 (61.54)	0.7247
Solubility	708	9,445 (13.34)	9,735 (13.75)	0.8717
MNIST-sp	70,000	5,250,000 (75)	41,798,306 (696.63)	0.7595

We also report the average number of nodes and edges per graph for each dataset in the parenthesis.

BA-house and *BA-grid*. Inspired by [5], we generate new synthetic datasets for graph classification. In both datasets, we use the Barabási-Albert (BA) model as a backbone graph, and complete each graph by attaching certain subgraph patterns (or motifs), which serve as the ground truth explanations. Both datasets contain two classes, where one class includes the ground truth explanation(s) in the underlying graph, while another class has an incomplete subgraph pattern. To generate the dataset, we first assign the class label 0 or 1 to a graph to be generated. We then generate a backbone graph using the BA model consisting of 5 to 10 nodes. If the assigned class label is 0, then we connect additional subgraph patterns to the backbone graph. Otherwise, we connect an incomplete subgraph pattern to the backbone graph. For BA-house, the subgraph pattern is a house-shaped subgraph, while, for BA-grid, a grid-like subgraph is used. In the case of BA-house, either single *or* double house-shaped subgraphs are randomly added within Class 0. Each node has a one-hot encoded feature vector indicating whether the node belongs to the backbone graph, the head of the motif (i.e., the ground truth explanation), or the body of the motif, depending on the node's position in the given graph.

Benzene [15]: The Benzene dataset contains molecules, where nodes and edges represent atoms and chemical bonds, respectively. The graphs are partitioned (labeled) into two different classes with respect to the existence of the benzene structure within the molecule (i.e., the structure having a six-carbon ring as the dataset neglects the Hydrogen atom). Each node has the corresponding one-hot encoded feature vector representing one possible atom type, which can be one of Carbon, Nitrogen, Oxygen, Fluorine, Iodine, Chlorine, and Bromine.

MUTAG [34]: The MUTAG dataset contains graphs representing molecules, where nodes represent different atoms, similarly as in the Benzene dataset. The graphs are partitioned into two different classes according to their mutagenic effect on a bacterium [5]. The node features are one-hot encoded vectors

[5]We have empirically found that such an update of $\mathbf{Y}_l$ with a decay factor enables us to avoid searching for the previously selected k-tuples, resulting in a more stable subgraph search process.

[6]github.com/jordan7186/PAGE.

that represent types of atoms, including Carbon, Nitrogen, Oxygen, Fluorine, Iodine, Chlorine, Bromine, and Hydrogen. In our experiments, the edge labels are ignored for simplicity.

Solubility [4]: The Solubility dataset is composed of real-world molecules with different levels of solubility, where nodes and edges represent atoms and their chemical bonds, respectively. In our experiments, we ignore the edge connection types. Although this dataset was originally intended to be used as a regression task, it is partitioned into two classes for the graph classification task. We label molecules with log solubility values lower than -4 as 0, and those with values higher than -2 as 1. Rigid carbon structures are considered as the ground truth explanation for insolubility, while $R\text{-}OH$ chemical groups are treated as the ground truth explanation for solubility. The one-hot encoded feature vector of each node represents types of atoms, which can be one of Carbon, Nitrogen, Oxygen, Fluorine, Iodine, Chlorine, Phosphorus, Sulfur, Hydrogen, and Bromine.

MNIST-sp [35]: MNIST-sp is a dataset for graph classification originated from the MNIST dataset used for image classification. Since local pixels are processed to form a node (superpixel) for each image, the image classification is transformed into a graph classification problem. Each graph contains 75 nodes, and edges are connected depending on whether the superpixels are neighbors in the original image. The node features include the center positions of the superpixel and the continuous values of the pixels. Therefore, we categorize the continuous values to discretize them. The pixel values are categorized into three groups according to their original values: 0 (completely black), $(0, 0.5]$ (dark), $(0.5, 1]$ (bright). The position values are categorized to an evenly divided 14×14 grid of the image via their association.

Note that model-level explanation methods aim to identify explanations based on the classes of each dataset. Thus, in all the experiments, model-level explanation methods use the class including a precise subgraph pattern as input and then attempt to retrieve the class-distinctive subgraph as the prototype graph.

B. Benchmark Method

In this subsection, we present a state-of-the-art method for comparison. As the representative model-level explanation, we use XGNN [14] as our benchmark method, which trains a graph generator that generates a subgraph pattern (i.e., a prototype graph) via reinforcement learning in the sense of maximizing a certain prediction of the underlying GNN model. More specifically, the graph generator is trained to add edges in an iterative fashion. At step t of the generation, the generated graph G_{t+1} is fed into the underlying GNN to calculate the reward function R_t:

$$R_t = R_{t,f_{\text{GNN}}}(G_{t+1}) + \lambda_1 \frac{\sum_{i=1}^{m} \text{Rollout}(G_{t+1})}{m} + \lambda_2 R_{t,r}, \tag{5}$$

where λ_1 and λ_2 are hyperparameters; $R_{t,f_{\text{GNN}}}(G_{t+1})$ is the predicted class probability of the underlying GNN f_{GNN}; $\text{Rollout}(\cdot)$ is the rollout operation [36]; m is the number of times rollout is performed, and $R_{t,r}$ is the additional term that provides domain-specific rewards based on various graph rules.

C. Experimental Settings

We first describe the settings of the GNN model. We adopt GCN [27] as one of the widely used GNN architectures. The summation function is used as a readout function. We use 2 GNN layers for the BA-house and BA-grid datasets and 3 GNN layers for other three datasets unless otherwise stated. We set the dimension of each hidden latent space to 32 for all the datasets. We train the GNN model with Adam optimizer [37] with a learning rate of 0.001 and a batch size of 16. For all five datasets excluding MNIST-sp, we split each dataset into training/validation/test sets with a ratio of 90/5/5%. For the MNIST-sp dataset, we follow the given train/test split and further split the training set into train/val sets with a ratio of 5:1. The training set is used to learn the model parameters with the cross-entropy loss; and the validation set is used for early stopping with a patience of 5 epochs. Unless otherwise specified, we use all the graphs in the training set to run PAGE. We report the graph classification performance on the test set along with the datasets in Table I.

Next, we turn to describing the settings of explanation methods including PAGE and XGNN. In PAGE, the implementations of clustering with the GMM are those from the scikit-learn Python package [38]. We set $L = 2$ and $k = 3$ since we have found that such a setting produces stable results across different datasets. We also set $budget$, indicating the number of search sessions, to 5. Additionally, we set the decay rate and the maximum number of iterations used in the iterative subgraph search (Step 2 of Phase 2) as 10 and 1,000, respectively. For the implementations of XGNN, we tuned the hyperparameters by essentially following the same settings as those in the original paper [14].

The GNN model and explanation methods used in our experiments were implemented with Python 3.8.12, PyTorch 1.11.0, PyTorch Geometric 2.0.4 [39], and Captum 0.5.0 [40], and were run on a machine with an Intel Core i7-9700 K 3.60 GHz CPU with 32 GB RAM and a single NVIDIA GeForce RTX 3080 GPU.

D. Experimental Results

Our experiments are designed to answer the following eight key research questions (RQs).

- *RQ1:* How are the model-level explanations of PAGE *qualitatively* evaluated in comparison with the benchmark method?
- *RQ2:* How are the model-level explanations of PAGE *quantitatively* evaluated in comparison with the benchmark method?
- *RQ3:* When does PAGE return the final prototype graph during multiple search sessions?
- *RQ4:* How do the explanations of PAGE relate to instance-level explanation methods?
- *RQ5:* How many input graphs does PAGE require for model-level explanations?
- *RQ6:* How effective is the prototype scoring function in PAGE in comparison with naïve alternatives?
- *RQ7:* How expensive is the computational complexity of PAGE in comparison with the benchmark method?

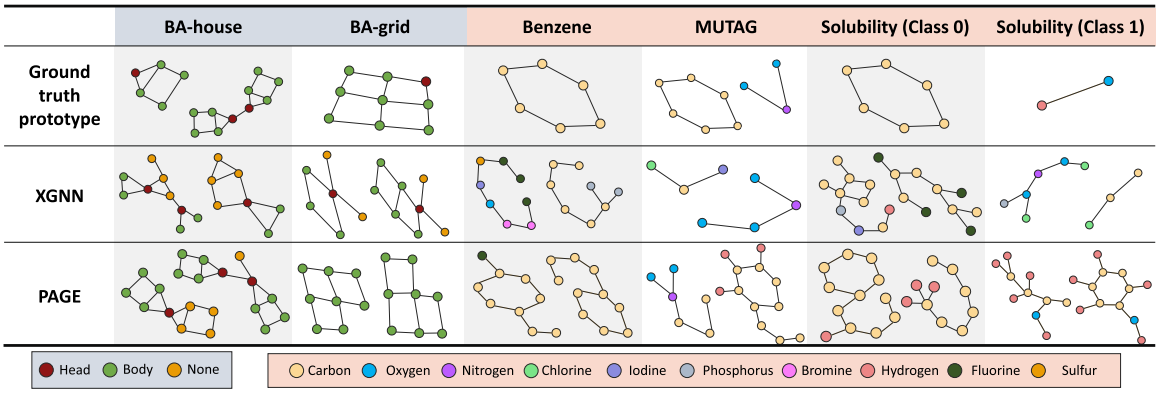
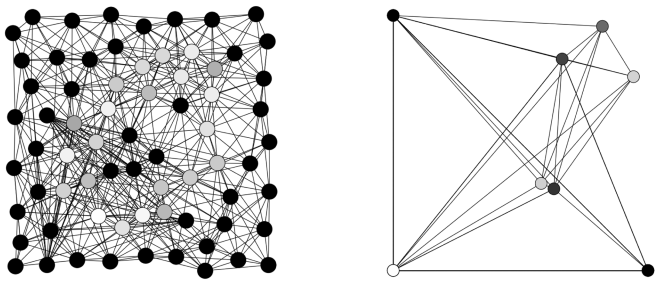


Fig. 5. Qualitative comparison of PAGE and XGNN for five datasets (excluding MNIST-sp, which is separately presented in Fig. 6), where each node is colored differently according to the types of nodes.



(a) PAGE. (b) XGNN.

Fig. 6. Qualitative comparison of PAGE and XGNN for Class 0 on the MNIST-sp dataset.

- *RQ8:* How does PAGE perform for datasets without explicit ground truth explanations?

1) Qualitative Analysis of Model-Level Explanation Methods (RQ1): We perform a qualitative evaluation of both PAGE and XGNN by visualizing their explanations (i.e., prototype graphs) along with the ground truth explanations for each dataset. As illustrated in Fig. 5, we observe that the explanations discovered by PAGE and the ground truth explanations exhibit a high similarity in terms of the graph structure and node features (i.e., types of nodes), while XGNN mostly fails to produce prototypes that contain the entire ground truth explanations. From Figs. 5 and 6, our findings for each dataset are as follows:

- For the BA-house dataset, PAGE successfully returns the class-discriminative subgraphs that we add to the backbone graph, that is, the house-shaped subgraphs on BA-house. However, the prototypes generated by XGNN contain only parts of the house-shaped subgraphs. For instance, the outcome from XGNN contains only three out of five nodes of the house-shaped subgraph, thus making the explanation incomplete. On the other hand, PAGE can produce much more precise explanations by returning two types of prototype graphs containing either a single or double house-shaped subgraph. Such preciseness of PAGE is possible primarily due to the usage of clustering of graph-level embeddings.
- The results from the BA-grid dataset show a tendency similar to those on BA-house. PAGE returns most of grid-like subgraphs, while XGNN recovers smaller portions of them.
- For the case of the Benzene dataset, PAGE correctly identifies the ring structure with 6 carbon atoms as a part of the prototype graph, while XGNN does not generate any carbon ring. Although XGNN produces six carbon atoms, it fails to connect the atoms to a closed ring structure.
- For the MUTAG dataset having two types of prototype graphs, PAGE identifies both carbon rings and NO_2 chemical groups. However, XGNN only recovers NO_2 while generating small molecules.
- We analyze model-level explanations for two classes on Solubility. For Class 0 (i.e., the case classified as insoluble molecules), PAGE tends to return carbon atoms and occasionally adds hydrogen atoms; XGNN also produces prototypes with mostly carbon atoms but contains a large portion of non-carbon atoms. Unlike the case of PAGE, XGNN fails to produce the carbon ring structure. For Class 1 (i.e., the case classified as soluble molecules), PAGE correctly identifies the R-OH chemical group for all prototypes; in contrast, XGNN fails to include R-OH for its explanations.
- We finally analyze the results for Class 0 on MNIST-sp. As depicted in Fig. 6, the number 0 corresponding to the class of interest is visible in the explanation for PAGE, whereas XGNN struggles to generate a sufficient number of new nodes to provide a comprehensive explanation for Class 0. This indicates the benefit of PAGE that discovers the prototype graph in the dataset over its counterpart generating it.

2) Quantitative Analysis of Model-Level Explanation Methods (RQ2): For the quantitative analysis, we adopt four performance metrics: *accuracy*, *density*, *consistency* and *faithfulness*.

- *Accuracy* compares the prototype graph g with the ground truth explanation g_T. To this end, we assume that we can acquire node/edge correspondences connecting g_T and g (i.e., node matching between the two graphs) while counting the number of edges in g that belong to a part of g_T, which would result in the best explanation performance. In our evaluation, we define the accuracy (Acc.) of model-level

TABLE II
QUANTITATIVE ASSESSMENT RESULTS FOR PAGE AND XGNN FOR SIX DATASETS

Method	BA-house	BA-grid	Solubility	MUTAG	Benzene	MNIST-sp
PAGE	0.5238	0.8571	0.3290	0.9090	0.6667	N/A
XGNN	0.2500	0.3200	0.2341	0.6875	0.2500	N/A

(a) Accuracy (↑).

Method	BA-house	BA-grid	Solubility	MUTAG	Benzene	MNIST-sp
PAGE	0.1481	0.1563	0.0462	0.1389	0.1111	0.2195
XGNN	0.1235	0.1667	0.1195	0.1875	0.1094	0.3593

(b) Density (↓).

Method	BA-house	BA-grid	Solubility	MUTAG	Benzene	MNIST-sp
PAGE	0.0308	0.0615	0.0846	0.1216	0.0639	0.1025
XGNN	0.2152	0.2705	0.3213	0.1269	0.2227	0.0242

(c) Consistency (↓).

Method	BA-house	BA-grid	Solubility	MUTAG	Benzene	MNIST-sp
PAGE	0.7340	0.5636	0.2164	0.4430	0.2364	0.8182
XGNN	-0.4037	-0.1636	0.0983	0.2504	-0.3091	0.1273

(d) Faithfulness (↑).

Here, the accuracy (the higher the better) compares the edge correspondences to the ground truth explanations; the density (the lower the better) measures the ratio of edges over nodes; the consistency (the lower the better) measures the variability of the output probabilities for the explanations over different settings of GNN models; and the faithfulness (the higher the better) measures the correlation between the output probabilities of explanations and the GNN's test accuracies.

explanations as follows:

$$\text{Acc.} = \frac{\text{TP}(g, g_T)}{\text{TP}(g, g_T) + \text{FP}(g, g_T) + \text{FN}(g, g_T)}, \quad (6)$$

where $\text{TP}(g, g_T)$, $\text{FP}(g, g_T)$, and $\text{FN}(g, g_T)$ represent the true positive(s), false positive(s), and false negative(s), respectively, in terms of counting the numbers of relevant nodes and edges between g and g_T.[7]

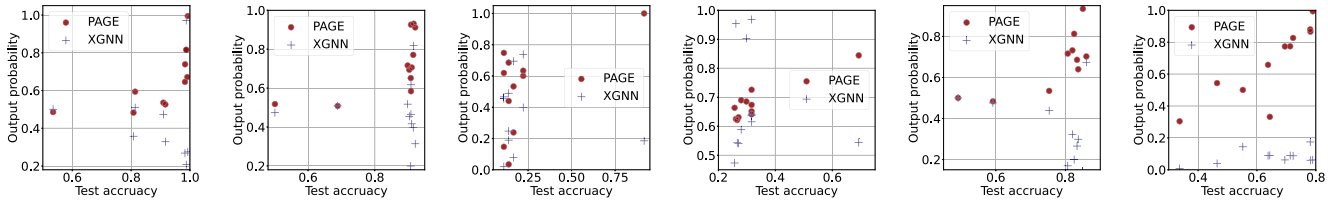
- *Density* measures the ratio of the number of edges relative to the square of the number of nodes in the prototype graph g.
- *Consistency* [15] measures the robustness of explanations for different settings of GNN models. Intuitively, reliable explanation methods should produce *stable* explanations despite different model configurations for the same task. In our experiments, we calculate the *standard deviation* of output probabilities for the explanations (e.g., the prototype graphs retrieved by PAGE) across different GNN hyperparameter settings. To this end, we generate multiple GNN models with various combinations of the hidden layer dimension. For the BA-house and BA-grid datasets, we produce 25 models with $\{4, 8, 16, 32, 64\} \times \{4, 8, 16, 32, 64\}$ hidden dimension configurations. For the Benzene, Solubility, MUTAG , and MNIST-sp datasets, we produce 64 models with $\{4, 8, 16, 32\} \times \{4, 8, 16, 32\} \times \{4, 8, 16, 32\}$ hidden dimension configurations. The lower the value of consistency, the better the performance.
- *Faithfulness* [15] measures how closely the explanation method reflects the GNN model's behavior. Following [15], we regard the test accuracy of the model as the behavior of interest, and empirically observe the relationship between the output probabilities of explanations and the GNN's test accuracies. In our experiments, faithfulness is measured by the Kendall's tau coefficient [41] between the output probability p_{GNN} for the explanation and the GNN's test accuracy (i.e., the performance on graph classification), which represents the degree to which the explanations reflect the GNN's test accuracies. To this end, we acquire a set of GNN models leading to varying accuracies by corrupting portions of labels in the training set from 0% to 50% in increment of 5%. The higher the value of faithfulness, the better the performance.

Table II summarizes quantitative evaluations for PAGE and XGNN with respect to the accuracy, density, consistency, and faithfulness when six datasets are used. Table II(a) shows that PAGE consistently provides more accurate explanations when compared to ground truth explanations regardless of datasets. Table II(b) shows that the explanations produced by PAGE tend to exhibit a lower density in most cases, which implies that the explanations are relatively less complex. In Table II(c), we observe that PAGE provides more stable explanations across different hidden dimension configurations than those of XGNN for all datasets except MNIST-sp (note that such an exception is mainly due to output probabilities of XGNN being near zero for almost all the cases, resulting in a lower standard deviation). From Table II(d), we observe that PAGE apparently exhibits a positive correlation between the GNN's test accuracies and the output probabilities compared to XGNN, which implies that PAGE is able to successfully account for what the GNN model has learned about its explanation outcome.

To further analyze the faithfulness for PAGE and XGNN, we visualize scatter plots of the output probabilities of explanations versus the GNN's test accuracies in Fig. 7. The intriguing observations are made from Table II(d) and Fig. 7:

- The different tendency between PAGE and XGNN is apparent on BA-house, BA-grid, Benzene, and MNIST-sp. For Solubility and MUTAG, the GNN's test accuracy tends to be quite low when labels are corrupted, which thus does not exhibit a clear relationship between the output probability and the test accuracy. This implies that the two datasets are more challenging to be learned in such noisy settings.
- Overall, the performance difference PAGE and XGNN in terms of the faithfulness becomes more prominent for the synthetic datasets. This is because the ground truth explanation can establish a relationship with the corresponding class more clearly due to less noise of the synthetic datasets.
- XGNN reveals a negative Kendall's tau coefficient for some datasets.
- This is because, when the GNN model is trained with high label corruption ratios, it basically passes through as a random classifier and thus cannot decide whether the explanation generated from XGNN is valid or not. When

[7]Note that the accuracy cannot be measured for the MNIST-sp dataset as the dataset does not contain a fixed ground truth explanation g_T.



(a) BA-house. (b) BA-grid. (c) Solubility. (d) MUTAG. (e) Benzene. (f) MNIST-sp.

Fig. 7. Visualization of the output probabilities of explanations versus the GNN's test accuracies for measuring the faithfulness when PAGE and XGNN are used as model-level explanation methods.

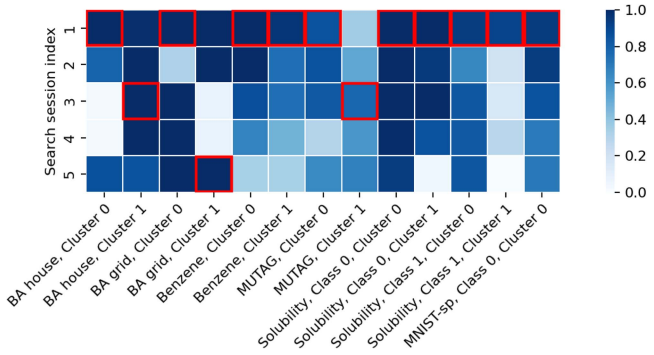


Fig. 8. Heatmap visualization of the output probability for the discovered subgraph in each search session when PAGE is used for all datasets. The subgraph returned as the final prototype graph is marked with a red box.

the GNN model is trained with fewer corrupted data, it now learns the characteristics of the given dataset and will produce low output probabilities if the outcome of XGNN does not contain the complete ground truth prototype. Eventually, this results in a negative correlation between the output probability versus the GNN's test accuracy.

3) Impact of Multiple Search Sessions in PAGE *(RQ3):* Aside from the final prototype graph of PAGE, we are also interested in analyzing the output probability p_{GNN} calculated for each search session in order to see when the prototype candidate having the highest p_{GNN} is selected over *budget* search sessions. Fig. 8 visualizes a heatmap of the output probability p_{GNN} for the discovered subgraph (i.e., the prototype candidate) in each session for all datasets, being partitioned into two clusters for a certain class, when *budget* is given by 5. The prototype candidate chosen as the final prototype graph $g_l^{(c)}$ is highlighted with a red box for each case. We would like to make the following observations:

- In most cases, each search session tends to return a prototype candidate with various values of p_{GNN}. For example, in Cluster 1 of MUTAG, the resultant p_{GNN}'s of the discovered prototype candidates range from the values near 0.4 to the ones near 0.8, which faithfully reflects our intention for the design of the core prototype search module (see Section IV-B2).
- The first search session produces the final explanation (i.e., the final prototype graph $g_l^{(c)}$) for 7 out of 10 cases. This tendency is expected since the first search session starts the iterative subgraph search (Step 2 of Phase 2 in Section IV-B3) alongside the highest matching score in $\mathbf{Y}_l$.
- There are a few cases in which the prototype candidate returned by a search session other than the first one is chosen as the final prototype graph, thereby justifying the necessity of multiple search sessions.
- From these observations, one can see that the number of search sessions, namely *budget*, will not be a critical problem in determining the performance of PAGE.

4) Relation Between PAGE and Instance-Level Explanations (RQ4): Since most studies on GNN explanations focus on instance-level explanations, we perform an additional analysis by investigating the relationship between our proposed PAGE method and instance-level explanation methods. Specifically, we are interested in evaluating the attribution map of the prototype graph, which indicates its significance towards making the model decisions. To this end, after discovering the prototype graph $g_l^{(c)}$ along with PAGE, we use $g_l^{(c)}$ as the input to an instance-level explanation method to acquire the attribution map (also known as the saliency map). This evaluation can be interpreted as measuring the agreement between PAGE and instance-level explanation methods.

For a graph $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{X}_i)$, an instance-level explanation method takes a node v in a graph as input and returns its attribution score, denoted as $\Lambda(v)$, which is normalized across all nodes in the graph. We present the following two quantities that we define in our experiments.

Definition 2 (Concentration Score): Given a prototype graph $g_l^{(c)}$ and the ground truth explanation $g_T^{(c)}$, we define the *concentration score* α as follows:

$$\alpha = \sum_{v \in \mathcal{V}_{g_T^{(c)}} \cap \mathcal{V}_{g_l^{(c)}}} \Lambda(v), \tag{7}$$

where $\mathcal{V}_{g_T^{(c)}}$ and $\mathcal{V}_{g_l^{(c)}}$ are the sets of nodes belonging to subgraphs $g_T^{(c)}$ and $g_l^{(c)}$, respectively. The concentration score measures how much the instance-level explanation concentrates on the nodes belonging to the ground truth explanation $g_T^{(c)}$. If none of the nodes in the prototype $g_l^{(c)}$ belongs to $g_T^{(c)}$, then α becomes

TABLE III
QUANTITATIVE ASSESSMENT RESULTS OF MEASURING THE AGREEMENT BETWEEN PAGE AND TWO INSTANCE-LEVEL EXPLANATION METHODS FOR THE BA-HOUSE AND BENZENE DATASETS, ALONG WITH THE ACCURACY [15] OF THE INSTANCE-LEVEL EXPLANATION METHOD METHODS MEASURED IN TERMS OF THE AUROC

Method	α	β	AUROC
Input × Gradient	0.9128	1.5919	1.0000
GNNExplainer	0.8458	0.0477	1.0000

(a) BA-house.

Method	α	β	AUROC
Input × Gradient	0.4193	-0.0592	0.0000
GNNExplainer	0.7028	0.0170	0.8889

(b) Benzene.

zero. The higher the value of α lying between 0 and 1, the better the performance.

Definition 3 (Relative Training Gain): Given the concentration score α for a prototype graph $g_l^{(c)}$ and the ground truth explanation $g_T^{(c)}$, the *relative training gain* β is defined as

$$\beta = \frac{\alpha}{\alpha_0} - 1, \tag{8}$$

where α_0 denotes the concentration score calculated using an initialized GNN model before training for the same prototype graph $g_l^{(c)}$.

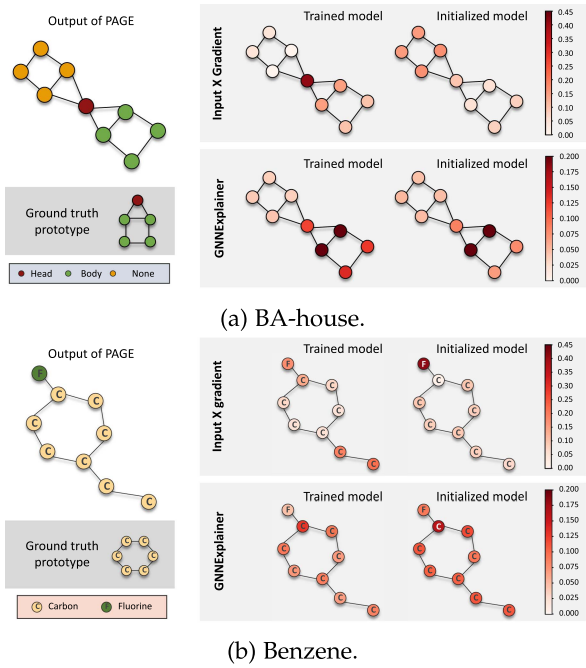
The relative training gain measures the relative gain of the concentration score after the GNN model has been trained. The higher the value of β, the better the performance. Note that β differs from the faithfulness in that it focuses on the *relationship* between model-level and instance-level explanation methods.

In our experiments, we adopt the following two instance-level explanation methods:

- *Input × Gradient [42]:* This method takes the gradients of the output with respect to the input and multiplies by the input in order to produce attribution maps.
- *GNNExplainer [5]:* Given an input instance, this method identifies a subgraph structure and a small subset of node features that are most influential to GNN's prediction. GNNExplainer performs an optimization task in the sense of maximizing the mutual information between the prediction and the distribution of possible subgraph structures.

Table III summarizes quantitative evaluations for measuring the agreement between PAGE and two instance-level explanation methods with respect to the concentration score α and the relative training gain β when the BA-house and Benzene datasets are used, along with the performance of each instance-level explanation method by measuring the area under the receiver operating characteristic (AUROC) against the ground truth prototype. To further analyze the relation between PAGE and two instance-level explanation methods, we visualize the attribution score as a heatmap where the two instance-level explanation methods are adopted to the output prototype graphs discovered by PAGE for the two datasets in Fig. 9.

We first analyze the quantitative assessment results with respect to α in Table III and Fig. 9.



(a) BA-house.

(b) Benzene.

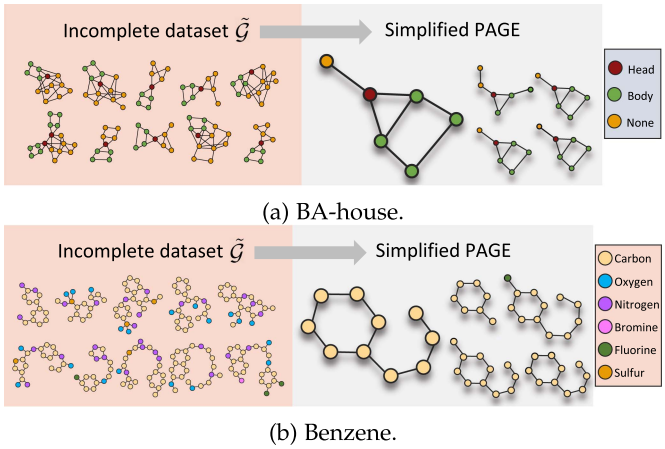
Fig. 9. Visualization of the attribution score as a heatmap where Input × Gradient and GNNExplainer are adopted to the output prototype graphs from PAGE for the BA-house and Benzene datasets.

- For the BA-house dataset, the values of α achieved by both Input × Gradient and GNNExplainer are sufficiently high, which indicates that a vast majority of attribution scores are assigned to the nodes belonging to the house-shaped subgraphs. In other words, the performance of PAGE and two instance-level explanation methods is shown to behave closely and consistently with each other.
- For the Benzene dataset, although the above tendency is observed similarly for GNNExplainer, the value of α tends to diminish for Input × Gradient. It is known that gradient-based attribution methods may often fail to capture the GNN model's behavior since activation functions such as Rectified Linear Units (ReLUs) have a gradient of zero when they are not activated during the feed-forward process of GNN [42].

We turn to analyzing the quantitative assessment results with respect to β in Table III.

- One can expect to see a positive value of β if the performance of both PAGE and instance-level explanation methods behaves closely and consistently.
- However, a negative value of β is observed for the case of Input × Gradient on Benzene, which is in contrast to the case of GNNExplainer producing positive values of β for the two datasets.
- The higher agreement between PAGE and GNNExplainer on Benzene is due to the fact that GNNExplainer was designed for generating explanations on graphs and GNN models, thus reliably producing the resulting attributions.

The above tendency is also found with respect to the AUROC in Table III, where Input × Gradient returns zero as AUROC



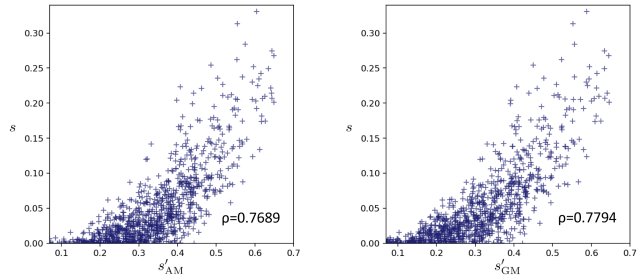
(a) BA-house.

(b) Benzene.

Fig. 10. Qualitative analysis of PAGE in an incomplete dataset setting, where each node is colored differently according to the types of nodes.

TABLE IV
COMPARISON OF DIFFERENT PROTOTYPE SCORING FUNCTIONS IN TERMS OF THE RUNTIME COMPLEXITY IN MICROSECONDS (MEAN ± STANDARD DEVIATION)

Prototype scoring function	s	s'_{AM}	s'_{GM}
Time (μs)	14.42 ± 12.65	38.34 ± 17.81	31.39 ± 17.81



(a) s'_{AM} versus s (b) s'_{GM} versus s

Fig. 11. Visualization of the results of different prototype scoring functions along with the correlation coefficient ρ.

compared to the case of GNNExplainer returning the AUROC of 0.8889. Note that the results from other three datasets showed a tendency similar to those reported in Section V-D4.

5) Robustness to the Number of Available Input Graphs (RQ5): We now perform a qualitative evaluation of PAGE by visualizing the resultant prototype graphs in a more difficult setting where each dataset is composed of only a few available graphs $\tilde{\mathcal{G}}$ from the training set as input of PAGE. This often occurs in real environments since users and organizations aiming at designing explanation methods may have limited access to the input data. To emulate such a scenario, we create the subset of cardinality 10 (i.e., $|\tilde{\mathcal{G}}| = 10$) via random sampling from the training set. In other words, the subset $\tilde{\mathcal{G}}$ is used for running PAGE while the GNN model has been trained using the whole training set from $\mathcal{G}$. Due to the fact that clustering of graph-level embeddings in Phase 2 of PAGE is not possible alongside only a few graphs in $\tilde{\mathcal{G}}$, we simplify PAGE and perform Phase 2 only after selecting k graphs from $\tilde{\mathcal{G}}$ instead of the kNNs.

Fig. 10 visualizes the prototype graphs discovered by simplified PAGE for five independent trials when the subset $\tilde{\mathcal{G}}$ of the BA-house and Benzene datasets is used for $k = 3$. In the case of BA-house in Fig. 10(a), we observe that the prototype contains an entire single house-shaped subgraph for four out of five trials, indicating that the core prototype search module is capable of producing ground truth explanations even in the incomplete dataset setting. However, due to the absence of clustering, the prototype graph containing double house-shaped subgraphs is not retrieved. Therefore, the simplified PAGE does not produce precise explanations but resorts to simpler explanations. In the case of Benzene in Fig. 10(b), we observe that the carbon ring structure is included for all five trials; the number of atoms other than the ground truth explanation varies over different trials due to the random selection of graph subsets $\tilde{\mathcal{G}}$. Note that the results from other three datasets also follow similar trends although not shown in the manuscript.

6) Effectiveness of the Prototype Scoring Function (RQ6): We validate the effectiveness of our prototype scoring function s in Definition 1 in terms of the computation efficiency in comparison to a naïve alternative. Instead of acquiring a set of node-level embedding vectors through a certain GNN model, we synthetically generate and feed them into the function s, which is sufficient to analyze the effectiveness of s. Specifically, we first generate each Gaussian random vector $\mathbf{w}_i$ with zero mean and covariance matrix $\mathbf{I}_b$, i.e., $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}_b, \mathbf{I}_b)$, where $\mathbf{0}_b$ and $\mathbf{I}_b$ denote the zero vector and the identity matrix, respectively, of size b. Then, the vectors $\mathbf{w}_i$'s pass through the ReLU activation function $\mathbf{v}_i = \text{ReLU}(\mathbf{w}_i)$, which corresponds to a synthetic node-level embedding vector.

In our experiments, we generate 10^3 3-tuples of vectors, which are used as the input of $s(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$. We compare s with the following alternative prototype scoring functions: $s_{AM}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) = (\mathbf{v}_1^T\mathbf{v}_2 + \mathbf{v}_2^T\mathbf{v}_3 + \mathbf{v}_3^T\mathbf{v}_1)/3$ and $s_{GM}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) = (\mathbf{v}_1^T\mathbf{v}_2 \times \mathbf{v}_2^T\mathbf{v}_3 \times \mathbf{v}_3^T\mathbf{v}_1)^{1/3}$, which return the arithmetic mean and the geometric mean, respectively, of all possible inner products between pairs of embedding vectors.

We empirically show the average runtime complexities of s, s_{AM}, and s_{GM} in Table IV. This experiment reveals that s_{AM} and s_{GM} are more computationally expensive than s, which is consistent with our analysis in Remark 2. Additionally, Fig. 11 visualizes scatter plots of the proposed s versus each of the naïve alternative, i.e., either s_{AM} or s_{GM}. From this figure, we observe a high correlation between two, where the Pearson correlation coefficient ρ is more than 0.75 for both cases. This demonstrates that our prototype scoring function s is a good approximation of the calculation of all possible inner products between embedding pairs.

7) Efficiency of PAGE (RQ7): We validate the superiority of PAGE in terms of the efficiency. To this end, we first theoretically analyze its computational complexity. In Phase 1, the complexity of GMM is $\mathcal{O}(nkb^3)$ [43] and the kNN selection involves sorting for each cluster, which results in $\mathcal{O}(nk(b^3 + \log n))$, where b is the dimension of node-level

TABLE V
COMPARISON OF RUNTIME COMPLEXITIES BETWEEN PAGE AND XGNN ON THE BA-HOUSE AND BENZENE DATASETS

Dataset	PAGE	XGNN
BA-house	$1.15 \pm 0.01\ s$	$3.14 \pm 0.02\ s$
Benzene	$12.10 \pm 0.02\ s$	$13.50 \pm 0.14\ s$

embedding vectors. In Phase 2, since computing $\mathbf{Y}_l$ and running each search session are highly parallelizable, the computation depends basically on the pre-defined maximum allowed number of iterations during the iterative subgraph search, $N_{\max}$, and the number of non-zero elements in $\mathbf{Y}_l$, N_{nnz}. Thus, the resulting complexity for Phase 2 of PAGE is $\mathcal{O}(N_{\max} N_{\text{nnz}} \log N_{\text{nnz}})$.[6] Therefore, the total computational complexity of PAGE is given by $\mathcal{O}(nk(b^3 + \log n) + N_{\max} N_{\text{nnz}} \log N_{\text{nnz}})$.

Additionally, we conduct experiments using two datasets, BA-house and Benzene, to measure the runtime complexities of PAGE and XGNN. Table V the runtime complexity (mean $\pm$ standard deviation) of each model-level GNN explanation method for 10 independent trials. The results demonstrate the efficiency of PAGE over XGNN regardless of the datasets.

8) Performance of PAGE *on the Graph-SST2 Dataset (RQ8):* We evaluate the performance of PAGE on Graph-SST2 [13], the dataset without explicit ground truth explanations. Graph-SST2 is a binary graph classification dataset processed from the SST2 dataset [44]. Sentences from the SST2 dataset are transformed into graphs by representing words as nodes and constructing edges according to the relationships between words. 786-dimensional word embedding vectors are used as node features.

Since Graph-SST2 is a sentiment classification dataset, it does not have a clear *ground truth* explanation that commonly expresses itself across instances, posing a critical challenge to run PAGE. In other words, it is technically difficult to discover ground truth explanations (prototypes) representing the whole class. The fact that the edges represent the lexical collocations makes the task more problematic, since the lexical combinations do not have an explicit connection to the classes.

Despite such a critical limitation, we run a part of PAGE on Graph-SST2 to see what input graphs are selected as the most representative ones. More specifically, we perform Phase 1 on Graph-SST2 as a simplified version of PAGE. Fig. 12 visualizes the results of PAGE by selecting three graphs for each class/cluster. From the figure, we would like to make the following observations.

- As discussed earlier, extracting a common subgraph pattern is not possible. This is because 1) there are cases where node edge exists and 2) additional criteria are necessary to determine whether a node matches another node to form a tuple.
- The selected input sentences tend to be very short in length. This can be naturally interpreted as a result of Phase 1 in PAGE that attempts to discover a common pattern among examples in a cluster. When a sentence becomes longer, it will be more uniquely positioned in the belonging cluster as it is likely to include rarely used words, thus yielding a unique graph structure. Although PAGE cannot provide explicit prototype graphs in contrast to other datasets classified as chemical/molecular datasets, it can still offer a low-resolution concept of each class representing the sentiment of phrases, which is another merit of PAGE behind.

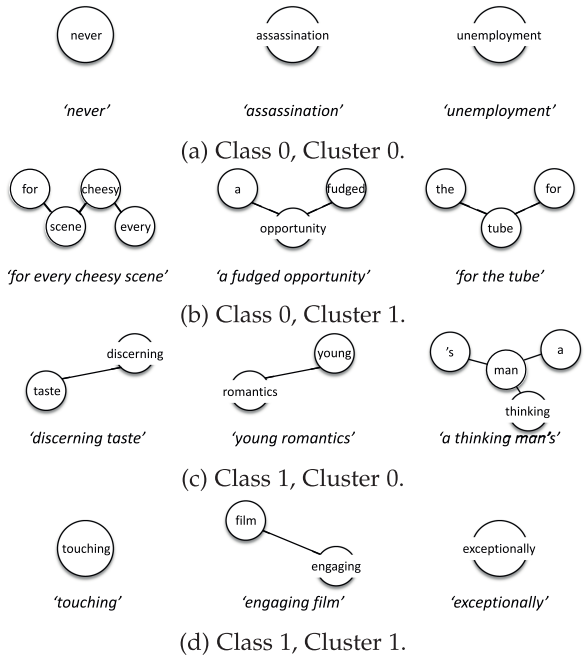


Fig. 12. Explanation results of PAGE on the Graph-SST2 dataset.

VI. CONCLUSIONS AND OUTLOOK

We investigated a largely underexplored yet important problem of explaining the decisions of GNNs at the model level. To tackle this practical challenge, we introduced a novel explanation method that offers model-level explanations of GNNs for the graph classification task by discovering prototypes from both node-level and graph-level embeddings. Specifically, we designed PAGE, an effective two-phase model-level GNN explanation method; after performing clustering and selection of graph-level embeddings in Phase 1, we perform prototype discovery in Phase 2, which is partitioned into the initialization, iterative subgraph search, and prototype retrieval steps. Furthermore, we theoretically validated the usage of clustering of graph-level embeddings in PAGE. Using two synthetic and three real-world datasets, we comprehensively demonstrated that PAGE is superior to the state-of-the-art model-level GNN explanation method both qualitatively and quantitatively. Additionally, through systematic evaluations, we proved the effectiveness of PAGE in terms of 1) the impact of multiple search sessions in the core prototype search module, 2) the relation to instance-level explanation methods via quantitative assessment, 3) the robustness to a difficult and challenging situation where only a few graphs are available as input of PAGE, and 4) the computational efficiency of the proposed prototype scoring function.

The main limitation of our work mainly lies in the assumption that the number of ground truth explanations per class is available, under which we run Phase 1 in PAGE to discover multiple clusters on the graph-level embedding space. In Phase 1, one can use various performance metrics used for the quantitative analysis to find the optimal number of clusters. As an outlook on future research, estimating the number of ground truth explanations per class before running explanation methods is an intriguing and important research topic from the fact that Phase 1 of PAGE needs the number of ground truth explanations per class for clustering, which is however not known a priori. Additional avenues of future research include the design of model-level GNN explanation methods for the node-level or edge-level classification task.

ACKNOWLEDGMENTS

The material in this paper was presented in part at the AAAI Conference on Artificial Intelligence, Virtual Event, February/March 2022 [45].

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, USA, 2019, pp. 1–17.

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, 2016, pp. 1135–1144.

[4] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," in *Proc. ICML Workshop Learn. Reasoning Graph-Structured Data*, Long Beach, CA, USA, 2019, pp. 1–21.

[5] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Vancouver, Canada, 2019, pp. 9240–9251.

[6] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 12241–12252.

[7] D. Luo et al., "Parameterized explainer for graph neural network," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1646.

[8] M. S. Schlichtkrull, N. D. Cao, and I. Titov, "Interpreting graph neural networks for NLP with differentiable edge masking," in *Proc. 9th Int. Conf. Learn. Representations*, 2021, pp. 1–21.

[9] T. Schnake et al., "Higher-order explanations of graph neural networks via relevant walks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7581–7596, Nov. 2022.

[10] M. N. Vu and M. T. Thai, "PGM-Explainer: Probabilistic graphical model explanations for graph neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1025.

[11] A. Lucic, M. A. ter Hoeve, G. Tolomei, M. de Rijke, and F. Silvestri, "CF-GNNExplainer: Counterfactual explanations for graph neural networks," in *Proc. 25th Int. Conf. Artif. Intell. Statist.*, 2022, pp. 4499–4511.

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

[13] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5782–5799, May 2023.

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

[15] B. Sanchez-Lengeling et al., "Evaluating attribution for graph neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 495.

[16] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One*, vol. 10, no. 7, 2015, Art. no. e0130140.

[17] R. R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, 2017, pp. 618–626.

[18] A. M. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 3387–3395.

[19] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2017, pp. 3510–3520.

[20] X. Wang, Y. Wu, A. Zhang, X. He, and T. Chua, "Towards multi-grained explainability for graph neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 18446–18458.

[21] J. Fang, X. Wang, A. Zhang, Z. Liu, X. He, and T. Chua, "Cooperative explanations of graph neural networks," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Singapore, 2023, pp. 616–624.

[22] J. Yan, X. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching," in *Proc. ACM Int. Conf. Multimedia Retrieval*, New York, NY, USA, 2016, pp. 167–174.

[23] M. Fey, J. E. Lenssen, C. Morris, J. Masci, and N. M. Kriege, "Deep graph matching consensus," in *Proc. 8th Int. Conf. Learn. Representations*, Addis Ababa, Ethiopia, 2020, pp. 1–23.

[24] A. Solé-Ribalta and F. Serratosa, "Graduated assignment algorithm for multiple graph matching based on a common labeling," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 27, no. 1, 2013, Art. no. 1350001.

[25] R. Wang, J. Yan, and X. Yang, "Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1671.

[26] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, Australia, 2017, pp. 1263–1272.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, Toulon, France, 2017, pp. 1–14.

[28] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 2nd ed. New York, NY, USA: Springer, 2006.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. Ser. Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.

[30] C. Daskalakis, C. Tzamos, and M. Zampetakis, "Ten steps of EM suffice for mixtures of two Gaussians," in *Proc. Conf. Learn. Theory*, Amsterdam, The Netherlands, 2017, pp. 704–710.

[31] N. Sammaknejad, Y. Zhao, and B. Huang, "A review of the expectation maximization algorithm in data-driven process identification," *J. Process Control*, vol. 73, pp. 123–136, 2019.

[32] C. Morris et al., "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 4602–4609.

[33] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.

[34] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity," *J. Med. Chem.*, vol. 34, no. 2, pp. 786–797, Feb. 1991.

[35] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3510–3520.

[36] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 2852–2858.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, 2015, pp. 1–15.

[38] F. Pedregosa et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[39] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop Representation Learn. Graphs Manifolds*, New Orleans, LA, USA, 2019, pp. 1–9.
[40] N. Kokhlikyan et al., "Captum: A unified and generic model interpretability library for PyTorch," in *Proc. ICLR Workshop Responsible AI*, 2021, pp. 1–11.
[41] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.
[42] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," 2016, *arXiv:1605.01713*.
[43] R. C. Pinto and P. M. Engel, "A fast incremental Gaussian mixture model," *PLoS One*, vol. 10, no. 10, 2015, Art. no. e0139931.
[44] R. Socher et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Seattle, WA, USA, 2013, pp. 1631–1642.
[45] Y. Shin, S. Kim, E. Yoon, and W. Shin, "Prototype-based explanations for graph neural networks (student abstract)," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 13047–13048.

Yong-Min Shin (Student Member, IEEE) received the BS degree in physics from Yonsei University, Seoul, South Korea, in 2019. He is currently working toward the integrated MS/PhD degrees with Yonsei University, Seoul, South Korea. Since March 2019, he has been with the School of Mathematics and Computing (Computational Science and Engineering), Yonsei University, Seoul, South Korea. His research interests include data mining, graph neural networks, and machine learning.

Sun-Woo Kim received the BS degree in applied statistics from Yonsei University, Seoul, South Korea, in 2021. He is currently working toward the MS degree with the Korea Advanced Institute of Science and Technology (KAIST), Seoul, South Korea. He was also with the Machine Intelligence and Data Science Laboratory, Yonsei University, Seoul, South Korea, in 2021. Since 2022, he has been with the Kim Jaechul Graduate School of AI, Korea Advanced Institute of Science and Technology (KAIST), Seoul, South Korea. His research interests include data mining and machine learning.

Won-Yong Shin (Senior Member, IEEE) received the BS degree in electrical engineering from Yonsei University, Seoul, South Korea, in 2002, and the MS and PhD degrees in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2004 and 2008, respectively. In 2009, he joined the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA USA, as a postdoctoral fellow and was promoted to a research associate in 2011. From 2012 to February 2019, he was a faculty member (with *tenure*) with the Department of Computer Science and Engineering, Dankook University, Yongin, South Korea. Since 2019, he has been with the School of Mathematics and Computing (Computational Science and Engineering), Yonsei University, Seoul, South Korea, where he is currently a professor. His research interests include the areas of information theory, mobile computing, data mining, and machine learning.