UO-Explainer: A Unified Approach to GNN Explanation Harnessing Human-Interpretable Structural Units

***Abstract*—As Graph Neural Networks (GNNs) are increasingly applied across various domains, explainability has become a critical factor for real-world applications. Existing post-hoc explainability methods primarily focus on estimating the importance of edges, nodes, or subgraphs in the input graph to identify substructures crucial for predictions. However, these methods often lack human interpretability and do not provide a unified framework that incorporates both model-level and instance-level explanations. In this context, we propose leveraging a set of graphlets—small, connected, non-isomorphic induced subgraphs widely used in various scientific fields—and their associated orbits as human-interpretable units to decompose GNN predictions. Domain experts can select the most relevant graphlets as interpretable units and request unified explanations based on these units. To address this problem, we introduce UO-Explainer, the Unified and Orbit-based Explainer for GNNs, which utilizes predefined orbits that are generalizable and universal across graph domains as interpretable units. Our model decomposes GNN weights into orbit units to extract class-specific graph patterns (model-level) and to identify important subgraphs within individual data instances for prediction (instance-level). Extensive experimental results demonstrate that UO-Explainer outperforms existing baselines in providing meaningful and interpretable explanations across both synthetic and real-world datasets. Our code and datasets are available at https://anonymous.4open.science/r/uoexplainer-F12C.**

***Index Terms*—Graph neural networks, Explainable artificial intelligence, Orbit-based decomposition, Unified explanation framework**

I. Introduction

GRAPH Neural Networks (GNNs) have achieved state-of-the-art performance in various domains including real-world graph-structured data, such as social networks [1], molecules [2], and knowledge graphs [3]. Despite the remarkable advancements in GNN architectures [4]–[7], they are still perceived as black box models due to their lack of explainability. This deficiency limits trust in GNN predictions, hindering their application in areas such as drug development [8] and education [9]. Therefore, interpreting the prediction of GNNs has become crucial and has led to the emergence of various explanatory approaches.

Explainability methods in graph domains deliver explanations through subgraphs that play a significant role in predictions regarding the input graph. Two primary issues arise regarding explainability: i) **human-interpretability** and ii) **unified framework** encompassing both model and instance levels [10]. Many existing post-hoc approaches are grounded on perturbation-based methods [11]–[15] and gradient-based methods [16], [17] to approximate the importance of edges or nodes within subgraphs. This stochastic optimization of importance is computationally effective in applying any graph-structured data; however, these methods have the potential risk that the output subgraph does not align with prior human assumptions or knowledge. Specifically, consider a scientist studying gene networks who is interested in understanding whether the presence of certain structures, such as triangles or rectangles, plays a crucial role in predicting specific properties. This scientist would want to audit the model's predictions by utilizing an explanation method that can highlight the importance of these structures. With existing methods, it is challenging to obtain explicit insights into whether a triangular or rectangular structure in the network is more important; instead, users often have to rely on conjecture based on the generated explanations. Additionally, these explanations do not always present results in a human-intuitive form, as they may include isolated nodes or disconnected edges, further hindering their interpretability. Therefore, we emphasize the need for explainability that centers on human-interpretability, a perspective that has been largely underexplored.

Last but not least, human-interpretable explanations should be delivered in a unified framework, incorporating both model- and instance-levels in terms of the scope of explanation [18], [19]. Most existing methods primarily specialize in one-level explanation either model or instance-level explanations. Model-level methods (e.g., [20], [21]) reveal patterns that GNNs deem significant for specific classes, eventually offering a broad understanding of the GNNs' general behavior. On the other hand, instance-level methods (e.g., [11], [14], [22], [23]) focus on individual predictions, identifying the subgraphs most relevant to the target node or graph. As each type of explanation complements the other from different perspectives, understanding at both levels enhances the explainability necessary for grasping the decision-making process of GNNs. Recent attempts to unify model- and instance-level explanations include GLGExplainer [24], which extracts logical formulas summarizing model-level decision patterns, and D4Explainer [25], which highlights graph regions through counterfactual analysis. However, neither approach guarantees that the final explanations are decomposed into clear, interpretable substructures that can be directly mapped to prior domain knowledge.

In this paper, we prioritize the two crucial perspectives that explanation should be human-interpretable in the unified framework incorporating both model and instance levels. Our proposed model, UO-Explainer, the **U**nified and **O**rbit-based Explainer for GNNs, allows users to harness their prior knowledge by giving room to select the user-defined explanation

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received Feb. 07, 2025

units in unified views. Considering the uniqueness of the graph domain to define explanation unit, we exploit orbits within graphlets that have been studied as a generalizable and universal pattern in many scientific fields such as protein interaction networks [26], [27], social networks [28], [29], and molecular structure networks [30], while users can also adopt a unique prior perspective to define their unit instead of orbit. To provide unified explanations, we decompose the weights into orbit representation vectors to understand the contribution of each orbit for a specific class or prediction. When we break down the weight into orbits, we acknowledge the contribution of each orbit for model decision-making, which is supposed to be further discussed in the method section in detail. Through extensive experiments, we demonstrate the superior performance of UO-Explainer on eight benchmark datasets. Consequently, UO-Explainer shows the concrete performance in a unified framework leveraging the human prior knowledge as orbit generalizable unit in graph-structured datasets.

In summary, the contributions of our research are as follows:

- We propose UO-Explainer, a unified framework to provide both model- and instance-level explanations for node classification based on human-defined explanation units.
- We demonstrate the human-interpretable explanation based on orbits generalizable and effective on graph-structured datasets.
- We perform rigorous and extensive experiments on 8 datasets to evaluate the quality of our explanations in both model- and instance-level explanations.

II. Related Work

As methodologies for explaining GNNs have been actively studied, explainability has gained significant attention as a critical area of focus. Explainability refers to the capacity to render the predictions of GNNs are transparent and understandable, and two key aspects are emphasized to ensure it: the **human-interpretability** and the **scope of explanation**.

Due to the structural nature of GNNs, where messages are propagated along the edges between connected nodes, human-interpretability often relies on the form of the explanatory subgraphs [31], [32]. GNNExplainer [11] is one of the early methodologies that optimizes edge masks to maximize mutual information with GNN predictions. Subsequently, PG-Explainer [12] utilizes node representation vectors and trains a parameterized mask predictor to optimize edge masks for explanations in inductive settings. TAGE [13] explains edge masks of the GNN embedding models, allowing efficient explanations for multiple downstream tasks. MixupExplainer [14] enhances the robustness of explanations through mixup data augmentation. These methods that employ edge masks to construct explanatory subgraphs may incorporate disconnected or isolated edges, which poses a risk of diminishing human-interpretability. MotifExplainer [33] provides an explanation that does not rely on edge masks. It computes embedding vectors for each motif and extracts explanations by restoring the GNN's prediction value using an attention network. This approach provides explanations that are easier for human users (e.g., scientists and engineers) to interpret, as the motifs are extracted based on user-defined rules. However, it still has limitations from the perspective of the **scope of explanation**.

GNN explanation methods can be categorized into model-level and instance-level, each with its scope of explanation [18]. Model-level methods [20], [21], [24] offer explanations that describe the general behavior underlying GNN predictions regarding to specific class. For example, XGNN [20] aims to generate model-level patterns that maximize the predictive probability of a certain class by training a graph generator through reinforcement learning. Similarly, PAGE [21] employs graph representation vectors to search for human-interpretable prototype graphs iteratively. Moreover, GLGExplainer [24] provides general model-level patterns by aggregating instance-level explanations into logical formulas, utilizing an Entropy-Logic Explainer (E-LEN) [34], [35]. However, these studies primarily focus on graph classification tasks and do not directly apply to node classification tasks. To bridge this gap, D4Explainer [25] introduces a method for providing counterfactual model-level explanations for node classification tasks, alongside instance-level explanations, based on a diffusion model. This approach, however, results in a high training cost for the explainer incurred by iterative diffusion- and deonising steps. Moreover, none of the methods for model-level explanations fully consider the user-centric explanation unit, hindering human-interpretability.

Instance-level methods [11]–[14], [22], [23], [33], [36]–[38] provide explanations in the form of subgraph to elucidate the prediction of a specific instance by unveiling relational structures with high contributions in the input graph. The previously mentioned methods, including GNNExplainer, PGExplainer, TAGE, MixupExplainer, and MotifExplainer, are all categorized as instance-level methods. Model-level and instance-level explanations provide complementary perspectives on explaining GNNs. Therefore, a unified framework capable of offering both perspectives can ensure better overall explainability. Furthermore, the abovementioned methods are limited to providing explanations in the unified framework capable of simultaneously providing both model-level and instance-level explanations under the human-interpretable units of interest.

III. Preliminary

A. Graph Neural Networks (GNNs)

We represents a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \mathbf{A}, \mathbf{X})$ where $\mathcal{E}$ denotes an edge set and $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$ denotes a node set. $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ denotes the adjacent matrix and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{in}}$ denotes the node feature matrix. In this study, we focus on GNNs for node classification tasks as presented in [5], [6]. A GNN model $f(\cdot)$ maps input graph into prediction matrix $f(\mathbf{A}, \mathbf{X}) = \mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{C}|}$, $\mathcal{C} = \{c_1, \cdots, c_{|\mathcal{C}|}\}$ in where $\mathcal{C}$ as the set of classes. GNNs can be expressed as a composite function $f = f_D \circ f_E$ of an embedding-model $f_E(\cdot)$ and a downstream-model $f_D(\cdot)$. The embedding-model embeds an adjacent matrix and node feature matrix into a node representation matrix $\mathbf{H}$, i.e., $f_E(\mathbf{A}, \mathbf{X}) = \mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times d}$. The representation vector of each node v_n is denoted by the $\mathbf{h}_{v_n}$, n-th row vector of matrix $\mathbf{H}$. The downstream-model maps the node representation matrix into the prediction matrix to

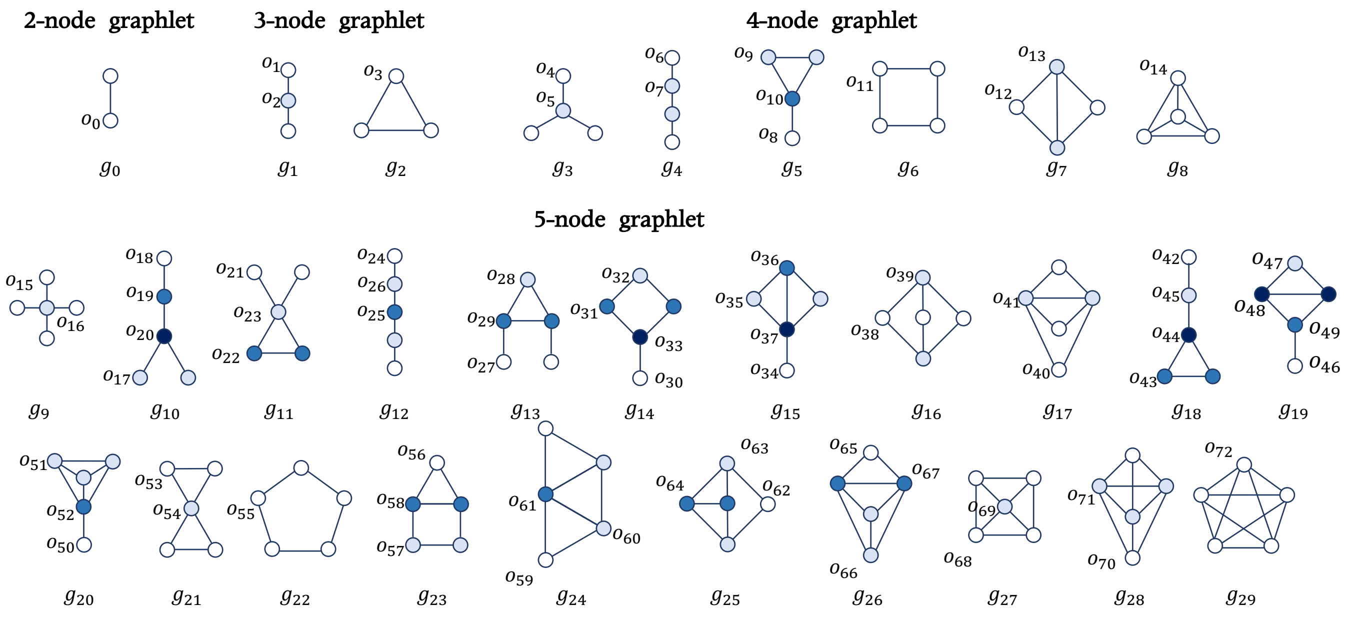
2-node graphlet
3-node graphlet
4-node graphlet
5-node graphlet
g_0 g_1 g_2 g_3 g_4 g_5 g_6 g_7 g_8
g_9 g_{10} g_{11} g_{12} g_{13} g_{14} g_{15} g_{16} g_{17} g_{18} g_{19}
g_{20} g_{21} g_{22} g_{23} g_{24} g_{25} g_{26} g_{27} g_{28} g_{29}

Fig. 1: Graphlets having 2-5 nodes and 0-72 orbits. The same color nodes within each graphlet belong to the same orbit.

classify nodes into each class, i.e., $f_D(\mathbf{H}) = \mathbf{HW} + \mathbf{b} = \mathbf{Z}$ where $\mathbf{W} \in \mathbb{R}^{d\times|\mathcal{C}|}$ denotes weight matrix and $\mathbf{b} \in \mathbb{R}^{|\mathcal{C}|}$ denotes bias vectors. In this operation, only the m-th column vector $\mathbf{w}_{c_m}$ of weight matrix $\mathbf{W}$ and the m-th element b_{c_m} of bias vectors $\mathbf{b}$ are involved in the computation to predict the class c_m, i.e., $\mathbf{z}_{c_m} = \mathbf{H}\mathbf{w}_{c_m} + b_{c_m}$ where $\mathbf{z}_{c_m}$ denotes the m-th row vector of matrix $\mathbf{Z}$. Weight regards a specific class as $\mathbf{w}_{c_m}$ affects only the prediction of the class c_m, so we call this weight vector a class weight. Furthermore, the prediction value z_{v_n,c_m} for the class of each node (the n-th row and m-th column element of $\mathbf{Z}$) can be expressed as $z_{v_n,c_m} = \mathbf{h}_{v_n} \cdot \mathbf{w}_{c_m} + b_{c_m}$, computed by the operation between each node representation vector and class weight.

B. Graphlet and Orbit

Graphlets as predominantly observed patterns are pre-defined subgraphs with a small number of nodes [26]. Figure 1 shows full set of 2-5 node graphlets [26], [27], where g_l denotes the l-th graphlet. All graphlets are non-isomorphic to each other, indicating that each graphlet has a unique structure. Each graphlet contains nodes with identical or distinguishable topological positions known as **orbits**, e.g., the central node of g_1 belongs to o_2 due to its distinguishable position, whereas the remaining nodes belong to o_1. The set of orbits is represented as $\mathcal{O} = \{o_0, \cdots, o_k, \cdots, o_{|\mathcal{O}|}\}$ where o_k denotes the k-th orbit.

Previous studies have highlighted the usefulness of graphlet-based analysis in various graph data domains, including protein interaction networks [26], [27], social networks [28], [29], and molecular structure networks [30]. For example, [27] defined the Graphlet Degree Distribution (GDD) using 2-5 node graphlets and orbits as units to analyze agreement among biological and chemical networks. [39], [40] further compared the empirical similarity of various chemical compound networks using a 2-5 node graphlet kernel. Since these analyses indicate that 2- to 5-node graphlets can serve as simple yet effective units for interpreting graph data, we primarily employ them to provide orbit-based explanations unless otherwise specified by the user.

IV. UO-Explainer

UO-Explainer serves as a unified explainer capable of delivering both model-level and instance-level explanations for node classification tasks. To deliver explanations in a human-interpretable way, we exploit a user-defined set of orbits as the explanatory unit. Upon the interpretable unit, we decompose the weights that directly influence classification concerning two components such as an embedding model and a downstream-model. An overview of UO-Explainer is presented in Figure 2.

A. Orbit Basis Learning

To decompose class weights into orbit units requires orbit bases, which are the representation vectors of each orbit. Orbit bases must necessarily reflect the following two aspects: (1) The distribution of each orbit within the input graph, and (2) The message passing and aggregation behavior of the embedding model. To meet the first requirement, we pre-process the orbit-existences on each node in the input graph. Orbit existence is denoted as y_{v_n,o_k} and determines whether each node v_n belongs to the orbit o_k:

$$y_{v_n,o_k} = \begin{cases} 0 & if\ v_n\ doesn't\ belong\ to\ o_k, \\ 1 & if\ v_n\ belongs\ to\ o_k. \end{cases} \tag{1}$$

We present a toy example of this pre-processing through Figure 3. (a) portrays the input graph. (b) represents the graphlets and orbits employed in the pre-processing. For simplicity, let us assume that only the graphlets g_2, g_3, g_6 are utilized and the orbits used for pre-processing are o_3, o_4, o_5, o_{11}. Different

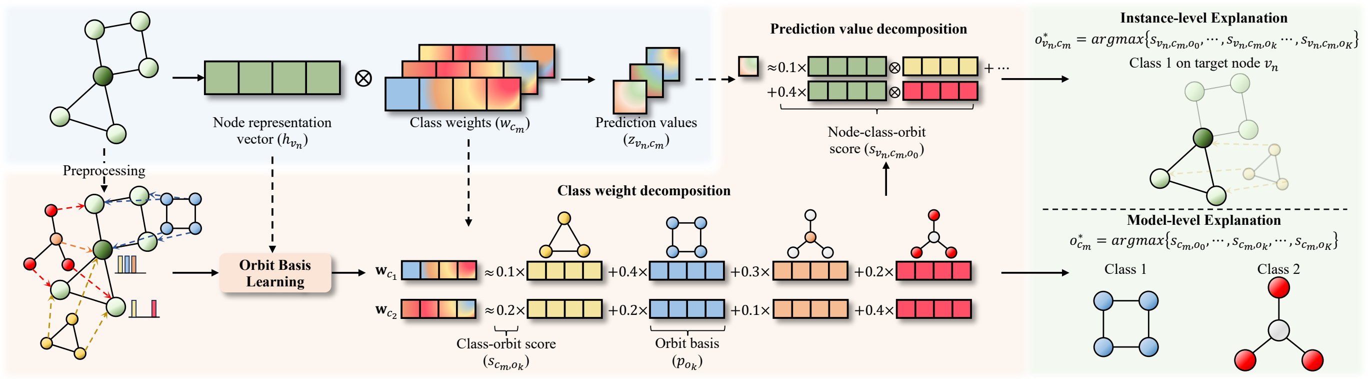
Node representation vector (h_{v_n})
Class weights (w_{c_m})
Prediction values (z_{v_n,c_m})
Preprocessing
Orbit Basis Learning
Prediction value decomposition
≈0.1× ⊗ + ⋯
+0.4× ⊗
Node-class-orbit score (s_{v_n,c_m,o_0})
Class weight decomposition
w_{c_1} ≈0.1× +0.4× +0.3× +0.2×
w_{c_2} ≈0.2× +0.2× +0.1× +0.4×
Class-orbit score (s_{c_m,o_k})
Orbit basis (p_{o_k})
Instance-level Explanation
$o^*_{v_n,c_m} = argmax\{s_{v_n,c_m,o_0}, \cdots, s_{v_n,c_m,o_k} \cdots, s_{v_n,c_m,o_K}\}$
Class 1 on target node v_n
Model-level Explanation
$o^*_{c_m} = argmax\{s_{c_m,o_0}, \cdots, s_{c_m,o_k}, \cdots, s_{c_m,o_K}\}$
Class 1
Class 2

Fig. 2: Overview of UO-Explainer: The propagation process of the GNN is depicted in the blue box, while the orange box represents the pipeline of UO-Explainer. The provided explanations are illustrated in the green box. The dark green node of the input graph denotes the target node. The colors (red, orange, blue, and yellow) correspond to the orbits within the graphlets.

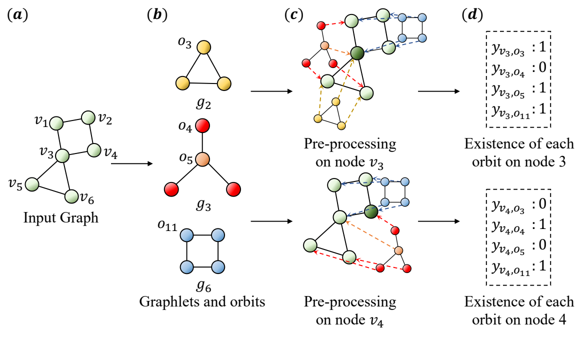

Fig. 3: Detailed example of the pre-processing step. The dark green node represents the target node for explanation. The colors of nodes within the graphlets represent the orbits respectively.

color nodes inside each graphlet refer to nodes belonging to different orbits. (c) is a substantial pre-processing process, which checks whether each node can belong to each orbit and assigns 1 or 0 to the orbit's existence. The dark green node represents the pre-processing node. You can see that node 3 belong to o_3, o_5, o_{11}. Therefore, $y_{v_3,o_3}, y_{v_3,o_4}, y_{v_3,o_5}, y_{v_3,o_{11}}$ are assigned values of 1, 0, 1, 1, respectively. This pre-processing is performed for all orbits in 2-5 node graphlets at all nodes within the input graph.

Next, we train a logistic binary classifier to predict the existence of each orbit, initializing each orbit with $\hat{\mathbf{p}}_{o_k}$ vector, and taking node representation as input, described by the following equation:

$$\hat{y}_{v_n,o_k} = sigmoid(\hat{\mathbf{p}}_{o_k} \cdot h_{v_n}). \tag{2}$$

To satisfy the second aspect, we learn the orbit basis by incorporating the node representations from the node embedding model when training. Then, we apply normalization in Equation as $\mathbf{p}_{o_k} = \frac{\hat{\mathbf{p}}_{o_k}}{||\hat{\mathbf{p}}_{o_k}||}$ to ensure that the size of the orbit basis remains constant. Orbit-basis learning is conducted for all orbits.

Algorithm 1 illustrates the detailed procedure for orbit-based learning. This procedure is similar to training a logistic binary classifier with $\hat{\mathbf{p}}_{o_k}$ as the weight. The input consists of node representation vectors $\mathbf{h}_{v_n}$, and the output is the predicted probability $\hat{y}_{v_n,o_k}$ for each orbit on nodes (line 4). The weight $\hat{\mathbf{p}}_{o_k}$ is updated based on the loss between the actual existence of the orbit y_{v_n,o_k} and the probability value $\hat{y}_{v_n,o_k}$ (lines 5-6). Finally, the orbit basis for a specific orbit is computed through vector normalization (line 8) and added to the set L. This procedure is iteratively performed for all orbits used as units of explanation.

Algorithm 1 Orbit Basis Learning

Input: A set of node representation vectors $\{\mathbf{h}_{v_1}, \cdots, \mathbf{h}_{v_n}, \cdots, \mathbf{h}_{v_{|\mathcal{V}|}}\}$, a set of existences of each orbits for each nodes $\{y_{v_1,o_0}, y_{v_2,o_0}, \cdots, y_{v_1,o_{23}}, \cdots, y_{v_{|\mathcal{V}|},o_K}\}$.
Output: A set of orbit bases $P = \{\mathbf{p}_{o_0}, \cdots, \mathbf{p}_{o_k}, \cdots, \mathbf{p}_{o_K}\}$.
Initialize P as an empty set

```
1: for k = 0 to $K$ (the number of orbits used) do
2:     Initialize $\hat{\mathbf{p}}_{o_k}$ as a random vector.
3:     for n=1 to $|\mathcal{V}|$ (the number of nodes) do
4:         $\hat{y}_{v_n,o_k} = sigmoid(\hat{\mathbf{p}}_{o_k} \cdot \mathbf{h}_{v_n})$
5:         $L = BCE(y_{v_n,o_k}, \hat{y}_{v_n,o_k})$
6:         Update $\hat{\mathbf{p}}_{o_k}$
             via $\nabla_{\hat{\mathbf{p}}_{o_k}} L$
7:     end for
8:     $\mathbf{p}_{o_k} = \frac{\hat{\mathbf{p}}_{o_k}}{||\hat{\mathbf{p}}_{o_k}||}$
9:     $P.add(\mathbf{p}_{o_k})$
10: end for
11: Return $P$
```

B. Model-level Explanations

Model-level explanations are provided by decomposing class weights into a linear combination of orbit bases, as the following equation:

$$\begin{aligned} \mathbf{w}_{c_m} &\approx s_{c_m,o_0}\mathbf{p}_{o_0} + \cdots + s_{c_m,o_k}\mathbf{p}_{o_k} + \cdots + s_{c_m,o_K}\mathbf{p}_{o_K} \\ &\approx \sum_{o_k \in \mathcal{O}} s_{c_m,o_k}\mathbf{p}_{o_k}. \end{aligned} \tag{3}$$

Generally, when a vector is expressed as a linear combination of bases, the coefficients of each basis indicate to what extent they contribute to forming the vector. Accordingly, the coefficients of orbit bases are regarded as contributions to the class weights. Furthermore, the bases are learned by considering each orbit distribution, thereby treating the contribution of orbit basis as the contribution of each orbit. We define the contribution of orbit o_k to the class c_m classification as a class-orbit score s_{c_m,o_k}.

The class-orbit scores are trained by the following objective function derived from Equation 3:

$$\min_{s_{c_m,o_k}>0} \left|\left|\mathbf{w}_{c_m} - \sum_{o_k\in\mathcal{O}} s_{c_m,o_k}\mathbf{p}_{o_k}\right|\right|. \tag{4}$$

To consider only the positive impact of the contributing orbits, we limit the class-orbit score to positive. However, directly optimizing the objective function for all orbit bases involves a significant amount of randomness. For example, in the worst case, if all orbit bases are orthogonal, and the dimension of the class weight is $d < |\mathcal{O}|$, an infinite number of combinations of s_{c_m,o_k} can be found to optimize the Equation 4. This randomness hinders the learning of the correct contribution of orbits. Therefore, we modify the objective function using a greedy approach by selecting the orbit that minimizes the difference between the class weight and the linear combination of selected orbits in each iteration, as shown in the following equation:

$$\arg\min_{o_k\in\mathcal{O}} \min_{\mathbf{S}_{c_m}>0} ||\mathbf{w}_{c_m} - [\mathbf{P}_{c_m}|\mathbf{p}_{o_k}]\mathbf{S}_{c_m}||. \tag{5}$$

$\mathbf{P}_{c_m}$ is a matrix consisting of the selected $\mathbf{p}_{o_k}$ as columns, and $\mathbf{S}_{c_m}$ represents a column vector consisting s_{c_m,o_k} of the selected orbits. $[\mathbf{P}_{c_m}|\mathbf{p}_{o_k}]$ denotes concatenation of the $\mathbf{p}_{o_k}$ to $\mathbf{P}_{c_m}$ as a column, e.g., if orbits 1, 3, and 5 are selected, then $\mathbf{P}_{c_m} = [\mathbf{p}_{o_1}|\mathbf{p}_{o_3}|\mathbf{p}_{o_5}]$ and $\mathbf{S}_{c_m}$ is a vector composed of s_{c_m,o_1}, s_{c_m,o_3} and s_{c_m,o_5}. By stopping the selection when the difference between the class weight and the linear combination does not decrease, we reduce the randomness and prevent too many orbits from being included in the explanation for each class. The detailed procedure can be found in Algorithm 2.

UO-Explainer uses the orbit $o^*_{c_m}$ with the highest class-orbit score from Equation 6 as the model-level explanation.

$$o^*_{c_m} = \arg\max_{o_k\in\mathcal{O}} \{s_{c_m,o_0}, \cdots, s_{c_m,o_k}, \cdots, s_{c_m,o_K}\}. \tag{6}$$

The orbit with the highest class-orbit score is always accompanied by its corresponding graphlet. Therefore, the model-level explanation is provided in the form of graph patterns, with the given orbit as the target node and its corresponding graphlet.

Algorithm 2 Class-Orbit Score Learning

```
Input: A set of orbit bases $\{\mathbf{p}_{o_0}, \cdots, \mathbf{p}_{o_k}, \cdots, \mathbf{p}_{o_K}\}$, a set of class weight vectors
    $\{\mathbf{w}_{c_1}, \cdots, \mathbf{w}_{c_m}, \cdots, \mathbf{w}_{c_{|\mathcal{C}|}}\}$.
Output: A set $S$ of selected orbit's class-orbit scores $s_{c_m,o_k}$
Variables: A vector composed of an element as a selected orbit's class-orbit score $\mathbf{S}_{c_m}$.
Initialize $S$ as an empty set
 1: for m=1 to $|\mathcal{C}|$ (the number of classes) do
 2:     Initialize $l_{min} = \infty$
 3:     Initialize $\mathbf{P}_{c_m}$ as an empty matrix
 4:     while do
 5:         $selected\_orbit = \arg\min_{o_k\in\mathcal{O}} \min_{\mathbf{S}_{c_m}>0} ||\mathbf{w}_{c_m} - [\mathbf{P}_{c_m}|\mathbf{p}_{o_k}]\mathbf{S}_{c_m}||$
 6:         $l = ||\mathbf{w}_{c_m} - [\mathbf{P}_{c_m}|\mathbf{p}_{selected\_orbit}]\mathbf{S}_{c_m}||$
 7:         if $l < l_{min}$ then
 8:             $l_{min} = l$
 9:             $\mathbf{P}_{c_m} = [\mathbf{P}_{c_m}|\mathbf{p}_{selected\_orbit}]$
10:         else
11:             $S.add(all\ s_{c_m,o_k}\ in\ the\ \mathbf{S}_{c_m})$
12:             break
13:         end if
14:     end while
15: end for
16: Return $S$
```

C. Instance-level Explanations

Instance-level explanations are provided by decomposing the prediction value of the target node into orbit units. The class weight decomposition of Equation 3 extends to the decomposition of prediction values as follows:

$$\begin{aligned} z_{v_n,c_m} &\approx \mathbf{h}_{v_n}\cdot\mathbf{w}_{c_m} + b_{c_m} \\ &\approx \underbrace{s_{c_m,o_0}\mathbf{h}_{v_n}\cdot\mathbf{p}_{o_0}}_{s_{v_n,c_m,o_0}} + \cdots + \underbrace{s_{c_m,o_k}\mathbf{h}_{v_n}\cdot\mathbf{p}_{o_k}}_{s_{v_n,c_m,o_k}} \\ &\quad + \cdots + \underbrace{s_{c_m,o_K}\mathbf{h}_{v_n}\cdot\mathbf{p}_{o_K}}_{s_{v_n,c_m,o_K}} + b_{c_m}. \end{aligned} \tag{7}$$

The equation above directly decomposes the prediction value of the target node into orbit units. Each term represents the magnitude of the decomposed prediction value, similarly indicating the contribution of orbits, akin to class weight decomposition. Therefore, we define the contribution of orbit o_k to the class c_m prediction of target node v_n as node-class-orbit score, s_{v_n,c_m,o_k}. To provide instance-level explanations, UO-Explainer extracts the orbit $o^*_{v_n,c_m}$ with the highest node-class-score as follows:

$$o^*_{v_n,c_m} = \arg\max_{o_k\in\mathcal{O}} \{s_{v_n,c_m,o_0}, \cdots, s_{v_n,c_m,o_k}, \cdots, s_{v_n,c_m,o_K}\}. \tag{8}$$

Unlike the model-level explanation, which provides the orbit with the highest contribution and its corresponding graphlet as an explanation, instance-level explanations must provide subgraphs around the target node within the input graph. To achieve this, we use the search algorithm based on the Breadth-First Search (BFS). These algorithms initiate the search from the target node and explore neighboring nodes by verifying whether their connectivity matches the highest contributed orbit's corresponding graphlets. A detailed algorithm is shown in Section IV-D. Through this search, UO-Explainer can extract a subgraph within the input graph that matches the highest-contributing orbit for the target node, i.e., the explored subgraph is provided as an instance-level explanation along with the target node.

D. Orbit Search Algorithm

We have mentioned the implementation of a graphlet search algorithm based on the Breadth-First Search (BFS) algorithm for extracting the orbit with the highest score and its corresponding graphlet on the target node. These algorithms initiate the search from the target node and explore neighboring nodes by verifying whether their connectivity matches the desired graphlets. We present two algorithmic examples for identifying specific orbits in our study. Algorithm 3 describes the process of identifying the o_{10} with its corresponding graphlet g_5, while Algorithm 4 outlines the steps for identifying the o_{56} with its corresponding graphlet g_{23}. In both algorithms, $N(v)$ denotes

Algorithm 3 Orbit o_{10} Search Algorithm

```
Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, \cdots, v_i, \cdots, v_{|\mathcal{V}|}\}$ denotes a set of node and $\mathcal{E} = \{(v_i, v_j), \cdots, (v_k, v_l)\}$ denotes a set of edges, Target node $v_n$.
Output: A set $L$ which contains edge sets of $g_5$ graphlets that make target node belong to orbit $o_{10}$.
Initialize $L$ as an empty set
1: Neighbor set of the target node $\mathcal{N}_{v_n} = N(v_n)$ where $N$ denotes the neighbor function.
2: A combination set with three elements $\mathcal{C}_1 = C(\mathcal{N}_{v_n}, 3)$ where $C$ denotes Combination function
3: for $v_a, v_b, v_c$ in $\mathcal{C}_1$ do
4:     a set of candidate combinations $\mathcal{C}_1^{'} = \{(v_a^{'}, v_b^{'}, v_c^{'}), (v_b^{'}, v_a^{'}, v_c^{'}), (v_c^{'}, v_a^{'}, v_b^{'})\}$.
5:     for $v_a, v_b, v_c$ in $\mathcal{C}_1^{'}$ do
6:         if $(v_b, v_c) \in \mathcal{E}$ and $\{(v_a, v_b), (v_a, v_c)\} \not\subset \mathcal{E}$ then
7:             $L.add(\{(v_n, v_a), (v_n, v_b), (v_n, v_c), (v_b, v_c)\})$
8:         end if
9:     end for
10: end for
11: Return $L$
```

the neighbor function to construct the set of neighboring nodes around a given node v. Additionally, $C(\cdot, n)$ denotes a combination function to generate the set of combinations consisting of n elements from a given set $\cdot$.

We present a detailed line-by-line explanation of Algorithm 3 as an exemplary demonstration of the orbit search process. This algorithm is designed to find g_5 graphlets in which the target node belongs to orbit o_{10}. In line 1, we generate the neighbor set $\mathcal{N}_{v_n}$ from the target node v_n. Since the number of neighboring nodes within orbit o_{10} in graphlet g_5 is three, line 2 constructs the set $\mathcal{C}_1$ by forming combinations of three nodes from the neighbor set. Within graphlet g_5, there is one orbit, o_8, that distinguishes it from the two neighboring nodes, o_9. Therefore, to ensure that each node combination in $\mathcal{C}_1$ also distinguishes between o_{10} and o_9, line 4 generates the set $\mathcal{C}_1^{'}$, which reflects requirement. Notably, the first nodes of each tuple in the set are distinct from the remaining nodes. By observing the connectivity of the nodes in lines 5-6, if it matches the connectivity of the orbits within g_5, the edge set of g_5 can be obtained in line 7. It is added to the set L. This algorithm iterates by considering all possible combinations of neighboring nodes surrounding the target node v_n. Therefore, all existing graphlet g_5 that make the target node belong to orbit o_{10} are in the output set L. This format can be applied to find entire 2-5 node graphlets. However, it is computationally expensive. To alleviate this computational cost, by introducing randomness during the generation of the neighbor set (line 1) or combination set in the algorithm (line 2 and line 4), it becomes possible to sample the desired number of graphlets.

V. Experiments

We evaluate explanations provided by UO-Explainer at the model-level and instance-level on synthetic and real-world datasets. These extensive experiments include quantitative and qualitative analysis of UO-Explainer's performance compared to recent baselines.

A. Datasets

We conduct experiments using five synthetic datasets and three real-world data sets. Synthetic datasets [11], [41] such as Random Graph, BA-Shapes, BA-Community, Tree-Cycle,

Algorithm 4 Orbit o_{56} Search Algorithm

```
Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, \cdots, v_i, \cdots, v_{|\mathcal{V}|}\}$ denotes a set of nodes and $\mathcal{E} = \{(v_i, v_j), \cdots, (v_k, v_l)\}$ denotes a set of edges, Target node $v_n$.
Output: A set $L$ which contains edge sets of $g_{23}$ graphlets that make target node belong to orbit $o_{56}$.
Initialize $L$ as an empty set
1: Neighbor set of the target node $\mathcal{N}_{v_n} = N(v_n)$ where $N$ denotes the neighbor function.
2: A combination set with three elements $\mathcal{C}_1 = C(\mathcal{N}_{v_n}, 2)$ where $C$ denotes the Combination function.
3: for $v_a, v_b$ in $\mathcal{C}_1$ do
4:     if $v_b, v_c \in \mathcal{E}$ then
5:         Neighbor set of $v_a$, $\mathcal{N}_{v_a} = N(v_a)$
6:         Neighbor set of $v_b$, $\mathcal{N}_{v_b} = N(v_b)$
7:         for $v_c$ in $\mathcal{N}_{v_a}$ do
8:             for $v_d$ in $\mathcal{N}_{v_b}$ do
9:                 if $(v_c, v_d) \in \mathcal{E}$ and $\{(v_n, v_c), (v_n, v_d), (v_b, v_c), (v_a, v_d)\} \not\subset \mathcal{E}$ then
10:                    $L.add(\{(v_n, v_a), (v_n, v_b), (v_a, v_b), (v_a, v_c), (v_b, v_d), (v_c, v_d)\})$
11:                end if
12:            end for
13:        end for
14:    end if
15: end for
16: Return $L$
```

and Tree-Grid are used to evaluate GNN explanation methods to compare the generated explanation based on pre-defined ground truths of each dataset. Also, real-world datasets such as Protein-Protein Interaction (PPI) [42], LastFM-Asia [43], and Gene [42] are used for node classification tasks. Additionally, the detailed information on each dataset is described below.

Random Graph [41]: This dataset does not have node feature values, so we perform node classification tasks using only the structural information by setting all node feature values to the same constant value. We set the class of each node based on the existence of its orbit same as each task number. That is, each node has two classes, c_0 (orbit doesn't exist) and c_1 (orbit exists) for a total of 73 independent tasks, one for each orbit. The ground truth for the class c_1 is the target orbit o_k for each task.

BA-Shapes [11]: The node classification dataset consists of a BA graph with 300 nodes as the base and 80 "house-like motifs" randomly attached, each consisting of 5 nodes. The house-like motif precisely matches graphlet g_{23}. The class is determined by the positions of the nodes within the house-like motif; these node positions precisely match the orbits o_{58}, o_{57}, o_{56} in g_{23}. Therefore, the ground truth for each class is set to the orbits o_{58}, o_{57}, o_{56}.

BA-Community [11]: The node classification dataset is generated by randomly combining two house-like motifs through edges. The first motif is assigned classes c_0, c_1, c_2, c_3 based on the positions of the nodes, and the second motif is assigned classes c_4, c_5, c_6, c_7, resulting in a total of eight class labels. The ground truth of each motif is the same as that of the BA-Shapes dataset.

Tree-Cycle [11]: The dataset consists of a balanced binary tree as the base and 80 "six-node cycle motifs" randomly attached. The task is to classify nodes as either not belonging (c_0) to or belonging to a circle (c_1). The ground truth for c_1 class corresponds to the circle motif. However, 2-5 node graphlets do not encompass the six-node circle motif. Thus, we conduct experiments by incorporating the circle motif and the orbit within the circle into candidate graphlets and orbits.

Tree-Grid [11]: The base is identical to that of the Tree-

TABLE I: Statistics of the dataset used for the experiment. # symbols mean the number.

Dataset	Random Graph	BA-Shapes	BA-Community	PPI	LastFM-Asia	Gene	Tree-Cycle	Tree-Grid
Avg # of nodes	340	700	1,400	2,373	7,624	857	871	1,231
Avg # of edges	2,690	4,110	8,920	66,136	55,612	13,992	1,950	3,410
# of classes	2	4	8	2	18	2	2	4
# of tasks	73	1	1	121	1	1	1	1

TABLE II: Accuracy and used hyper-parameters of pre-trained GNNs.

Dataset	BA-Sahpes	BA-Community	PPI 0	PPI 1	PPI 2	PPI 3	PPI 4	PPI 5	LastFM Asia	Gene	Tree-Cycle	Tree-Grid
learning Rate	0.001	0.001	0.003						0.001	0.001	0.001	0.001
Epoch	2,000	5,000	300						600	100	600	3,000
Hidden Dimension	10	30	200						30	100	30	30
Train:Val:Test	8:1:1	8:1:1	10:1:1						8:1:1	8:1:1	8:1:1	10:1:1
Train Accuracy	0.982	0.998	9.998	0.998	0.997	0.998	0.994	0.998	0.997	0.999	0.993	0.987
Val Accuracy	0.943	0.723	0.953	0.959	0.970	0.962	0.976	0.981	0.814	0.744	0.966	0.878
Test Accuracy	0.971	0.800	0.971	0.967	0.973	0.957	0.980	0.936	0.841	0.686	0.966	0.863

TABLE III: Hyperparameter settings for UO-Explainer. LR means learning rate.

Dataset	Random Graph	BA-Shapes	BA-Community	PPI	LastFM-Asia	Gene	Tree-Cycle	Tree-Grid
Epochs (Orbit basis learning)	3,000	3,000	3,000	3,000	1,000	1,000	1,000	1,000
LR (Orbit basis learning)	0.005	0.005	0.005	0.001	0.003	0.005	0.005	0.005
Batch size (Orbit basis learning)	256	256	256	256	2048	256	256	256
Epochs (Class-orbit score learning)	2,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
LR (Class-orbit score learning)	0.003	0.005	0.005	0.005	0.005	0.005	0.005	0.005

Cycle, with a modification involving the attachment of a "nine-node grid" in place of the circle. The task has been further augmented to classify each node as belonging to the center (c_1), periphery (c_2), or cross-lines (c_3) based on their positions within the grid. Similar to the case of Tree-Cycle, 2-5 node graphlets do not contain the nine-node grid. Therefore, we include the grid motif and the orbits within the grid in the candidate set for experimentation.

Protein-Protein Interaction (PPI) [42]: This is a node classification dataset that represents protein-protein interactions in various human tissues using 24 different graphs. Each node represents a protein and features such as positional, motif gene, and immunological signature are used with a size of 50 dimensions. Protein-to-protein interactions are represented as edges. Each node is labeled with 121 dimensions of gene ontology sets, allowing for binary classification of 121 categories, similar to a random graph. We pre-train a GNN on 20 of these graphs and extract explanations for randomly selected graphs to conduct experiments.

LastFM-Asia [43]: This is a social network dataset of LastFM users in Asia. Each node represents a LastFM user in Asian countries, and edges represent the following relationships between them. Node features are composed of artists that users like, which we convert to one-hot encoding for use in experiments. We perform a node classification task to predict the country of each user across 18 countries.

Gene: For qualitative experiments, we create a straightforward dataset using the gene network of natural killer cells in humans, as provided by [42]. Each node of the dataset represents genes, while the features include positional genes, motif genes, and immunological signature information of each gene, with each feature consisting of 50 dimensions. The Molecular Signatures Database [44] was used to collect the features for each gene. The labels in the dataset indicate whether a given gene belongs to the cell surface receptor signaling pathway, with 1 indicating inclusion and 0 indicating exclusion in the gene ontology set.

B. Baselines

Among the existing methods, D4Explainer [25] and GLGExplainer [24] provide model-level explanations for node classification tasks. For instance-level explanations, we set baselines based on common methods that learn edge masks, such as GNNExplainer [11], PGExplainer [12], TAGE [13], MixupExplainer [14], SAME [22], and EIG [23]. Additionally, MotifExplainer [33], which provides explanations based on motifs similar to our model, was also set as a baseline.

We primarily utilize the PyTorch [45] and PyTorch Geometric [46] frameworks in our code implementation. Additionally, for the GNNExplainer and PGExplainer, we employ implementations from the PyTorch Geometric framework. For TAGE, we employ implementations from the Dive into Graph (DIG) [47] framework. For MotifExplainer, we utilize the code provided in the supplementary material on OpenReview, which we modified following the advice of the authors. This revised implementation is incorporated within our publicly accessible code.

a) Hyperparameter Tuning: To ensure fairness in our comparisons, we thoroughly explore the hyperparameter space for each baseline, including the default parameters recommended in their original publications. For **GNNExplainer**, we tune the learning rate, number of epochs, edge size, node feature size, edge entropy, and node feature entropy. Specifically, we search over the candidate values $\{100, 300, 600\}$ for epochs, $\{0.01, 0.05, 0.1\}$ for learning rate, $\{0.005, 0.01, 0.1\}$ for edge size, $\{0.05, 0.1\}$ for node feature size, $\{0.5, 1.0\}$ for edge entropy, and $\{0.5, 1.0\}$ for node feature entropy. For **PG-Explainer**, we vary the learning rate $\{0.001, 0.003, 0.01, 0.5\}$, number of epochs $\{0.01, 0.05, 0.1\}$, edge size $\{0.05, 0.1\}$, edge entropy, and the MLP dimension $\{64, 128\}$. For **TAGE**,

TABLE IV: Model-level explanations on random graph datasets: (I) with 2 or 3-layer GCN, and (II) with 2 or 3-layer GIN. Each task is to classify whether the node belongs to the orbit corresponding to the task number. The evaluation metric is the *Sub-recall*. We conduct experiments five times and then reported the average and standard deviations. The best performances are shown in **bold**.

(I) 2 or 3-layer GCN

(a) Tasks 8, 11, 16, 21, 27

Task number	8	11	16	21	27
Ground-truth orbit	o_8	o_{11}	o_{16}	o_{21}	o_{27}
D4Explainer	0.2 ± 0.4	0.8 ± 0.4	0.4 ± 0.5	0.4 ± 0.5	0.2 ± 0.4
GLGExplainer	0.8 ± 0.4	0.6 ± 0.5	0.8 ± 0.4	0.6 ± 0.5	0.8 ± 0.4
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(b) Tasks 31, 32, 33, 35, 39

Task number	31	32	33	35	39
Ground-truth orbit	o_{31}	o_{32}	o_{33}	o_{35}	o_{39}
D4Explainer	0.4 ± 0.5	0.4 ± 0.5	0.4 ± 0.5	$\mathbf{1.0 \pm 0.0}$	0.8 ± 0.4
GLGExplainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.8 \pm 0.4}$	0.8 ± 0.4	0.8 ± 0.4
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	0 ± 0.0	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(c) Tasks 45, 47, 49, 57, 59

Task number	45	47	49	57	59
Ground-truth orbit	o_{45}	o_{47}	o_{49}	o_{57}	o_{59}
D4Explainer	0.2 ± 0.4	0.4 ± 0.5	0.4 ± 0.5	0.8 ± 0.4	0.2 ± 0.4
GLGExplainer	0.8 ± 0.4	0.6 ± 0.5	0.6 ± 0.5	$\mathbf{1.0 \pm 0.0}$	0.6 ± 0.5
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(d) Tasks 60, 61, 62, 64

Task number	60	61	62	64
Ground-truth orbit	o_{60}	o_{61}	o_{62}	o_{64}
D4Explainer	0.0 ± 0.0	0.6 ± 0.5	$\mathbf{0.6 \pm 0.5}$	0.2 ± 0.4
GLGExplainer	$\mathbf{0.6 \pm 0.5}$	0.8 ± 0.4	0.0 ± 0.0	$\mathbf{1.0 \pm 0.0}$
UO-Explainer	0.0 ± 0.0	$\mathbf{1.0 \pm 0.0}$	0 ± 0.0	$\mathbf{1.0 \pm 0.0}$

(II) 2 or 3-layer GIN

(a) Tasks 8, 11, 16, 21, 27

Task number	8	11	16	21	27
Ground-truth orbit	o_8	o_{11}	o_{16}	o_{21}	o_{27}
D4Explainer	0.8 ± 0.4	0.6 ± 0.5	0.6 ± 0.5	$\mathbf{1.0 \pm 0.0}$	0.6 ± 0.5
GLGExplainer	0.8 ± 0.4	0.8 ± 0.4	$\mathbf{1.0 \pm 0.0}$	0.4 ± 0.5	$\mathbf{1.0 \pm 0.0}$
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(b) Tasks 31, 32, 33, 35, 39

Task number	31	32	33	35	39
Ground-truth orbit	o_{31}	o_{32}	o_{33}	o_{35}	o_{39}
D4Explainer	0.8 ± 0.4	0.8 ± 0.4	0.6 ± 0.5	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$
GLGExplainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	0.8 ± 0.4	$\mathbf{1.0 \pm 0.0}$
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(c) Tasks 45, 47, 49, 57, 59

Task number	45	47	49	57	59
Ground-truth orbit	o_{45}	o_{47}	o_{49}	o_{57}	o_{59}
D4Explainer	0.6 ± 0.5	0.4 ± 0.5	0.8 ± 0.4	$\mathbf{1.0 \pm 0.0}$	0.8 ± 0.4
GLGExplainer	$\mathbf{1.0 \pm 0.0}$	0.6 ± 0.5	0.8 ± 0.4	$\mathbf{1.0 \pm 0.0}$	0.6 ± 0.5
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(d) Tasks 60, 61, 62, 64

Task number	60	61	62	64
Ground-truth orbit	o_{60}	o_{61}	o_{62}	o_{64}
D4Explainer	0.8 ± 0.4	0.2 ± 0.4	0.4 ± 0.5	0.6 ± 0.5
GLGExplainer	0.8 ± 0.4	$\mathbf{1.0 \pm 0.0}$	0.0 ± 0.0	$\mathbf{1.0 \pm 0.0}$
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

we consider the learning rate $\{5\times10^{-6}, 5\times10^{-5}, 5\times10^{-4}, 5\times10^{-3}, 5\times10^{-2}, 0.1\}$, number of epochs $\{1, 3, 5, 10, 20\}$, batch size $\{4, 16, 64, 128\}$, coefficient size $\{0.01, 0.05\}$, coefficient entropy $\{0.0005, 0.005\}$, and two loss types $\{\mathrm{NCE}, \mathrm{JSE}\}$. Finally, for **MotifExplainer**, we tune the learning rate $\{0.001, 0.005, 0.01\}$, number of epochs $\{30, 50, 100\}$, and the embedding dimension of the attention module. In all cases, we select the best configurations based on their performance on a validation split.

b) Motif extraction rule for MotifExplainer: For motif extraction for MotifExplainer, we use three rules: (1) identifying cycles within the graph, (2) identifying edges that do not form cycles, and (3) extracting motifs that combine two or more identical nodes involved in cycles. When fewer than 5,000 motifs were detected, all three rules were applied in full. Otherwise, to control computational overhead, a random subset of 5,000 motifs was used.

C. Detailed Experimental Settings

In this section, we cover the details of experiments that were not discussed in the main text, such as the structure of the pre-trained GNN and the hyperparameter settings of UO-Explainer.

We provide detailed information about pre-trained GNNs for datasets. For each dataset, we utilize a pre-trained GNN architecture consisting of a 3-layer GCN as the embedding model and a 1-layer MLP as the downstream model. The hyperparameters and split ratio are shown in Table II.

UO-Explainer has a total of five hyper-parameters including batch size, epochs, learning rate (LR) for orbit basis learning, and additional epochs, learning rate (LR) for class-orbit score learning. Hyper-parameters used in the experiments are reported in Table III.

D. Evaluation Metric

We evaluate the quality of explanations using Sparsity, Fidelity, Edge-recall, and Sub-recall as evaluation metrics.

Sparsity [48] refers to the ratio of edges in the explanation compared to the total number of edges in the computation graph of the target node. Sparsity [48] means the proportion of the presented explanation in the computation graph for the target node as follows:

$$Sparsity = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|} 1 - \frac{|\mathcal{G}_{v_i}^{ex}|}{|\mathcal{G}_{v_i}|}, \tag{9}$$

where $\mathcal{G}_{v_i}^{ex}$ denotes the subgraph presented as the explanation for the node v_i, $\mathcal{G}_{v_i}$ denotes to the computation graph for the node v_i, and $|\mathcal{G}|$ means the number of edges in the graph $\mathcal{G}$. High sparsity implies that the proposed explanation has a small number of edges.

Fidelity [48] is calculated by taking the difference in the probability values of the input graph and the probability values when the explanation is excluded from the computation graph based on the target node, as follows:

TABLE V: Model-level explanation results on synthetic datasets. The evaluation metric is the *Sub-recall*. The best performances are shown in **bold**. We conduct 5 experiments for each setting and report consistent results.

(a) BA-Shapes

	Class 1	Class 2	Class 3
Ground-truth orbit	o_{58}	o_{57}	o_{56}
D4Explainer	0.4 ± 0.5	0.6 ± 0.5	0.4 ± 0.5
GLGExplainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(b) Tree-Grid

	Class 1	Class 2	Class 3
Ground-truth orbit	o_{73}	o_{74}	o_{75}
D4Explainer	0.6 ± 0.4	0.0 ± 0.0	0.2 ± 0.4
GLGExplainer	0.0 ± 0.0	0.8 ± 0.4	0.2 ± 0.4
UO-Explainer	$\mathbf{0.8 \pm 0.4}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(c) BA-Community

	Class 1	Class 2	Class 3	Class 5	Class 6	Class 7
Ground-truth Orbit	o_{58}	o_{57}	o_{56}	o_{58}	o_{57}	o_{56}
D4Explainer	0.8 ± 0.4	0.2 ± 0.4	0.8 ± 0.4	0.0 ± 0.0	0.6 ± 0.5	0.4 ± 0.5
GLGExplainer	$\mathbf{1.0 \pm 0.0}$	0.8 ± 0.4	0.2 ± 0.4	$\mathbf{0.8 \pm 0.4}$	$\mathbf{1.0 \pm 0.0}$	0.8 ± 0.4
UO-Explainer	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.8 \pm 0.4}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

(d) Tree-Cycle

	Class 1
Ground-truth Orbit	o_{76}
D4Explainer	0.8 ± 0.4
GLGExplainer	$\mathbf{1.0 \pm 0.0}$
UO-Explainer	$\mathbf{1.0 \pm 0.0}$

TABLE VI: Instance-level explanation results on synthetic datasets. The best performances on each dataset are shown in **bold**. we conduct five experiments for each method by recording the mean and standard deviation.

(a) BA-Shapes

Method	**Sub-recall**	**Edge-recall**	**Fidelity**
GNNExplainer	0.004 ± 0.008	0.616 ± 0.010	0.580 ± 0.031
PGExplainer	0.760 ± 0.025	0.915 ± 0.003	0.574 ± 0.020
TAGE	0.682 ± 0.045	0.900 ± 0.008	0.601 ± 0.016
MixupExplainer	0.696 ± 0.069	0.906 ± 0.012	0.612 ± 0.006
SAME	0.343 ± 0.003	0.720 ± 0.009	0.547 ± 0.003
EiG-Search	0.878 ± 0.000	0.520 ± 0.000	0.605 ± 0.000
MotifExplainer	0.873 ± 0.069	0.890 ± 0.069	0.548 ± 0.072
UO-Explainer	$\mathbf{0.948 \pm 0.016}$	$\mathbf{0.984 \pm 0.005}$	$\mathbf{0.623 \pm 0.000}$

(b) Tree-Grid

Method	**Sub-recall**	**Edge-recall**	**Fidelity**
GNNExplainer	0.000 ± 0.000	0.629 ± 0.002	0.872 ± 0.006
PGExplainer	0.000 ± 0.000	0.647 ± 0.043	0.876 ± 0.002
TAGE	0.003 ± 0.000	0.693 ± 0.013	0.874 ± 0.004
MixupExplainer	0.047 ± 0.001	0.712 ± 0.001	0.877 ± 0.010
SAME	0.000 ± 0.000	0.238 ± 0.013	0.846 ± 0.010
EiG-Search	0.004 ± 0.000	0.723 ± 0.000	0.885 ± 0.000
MotifExplainer	0.793 ± 0.046	0.857 ± 0.051	0.879 ± 0.010
UO-Explainer	$\mathbf{0.859 \pm 0.032}$	$\mathbf{0.900 \pm 0.023}$	$\mathbf{0.888 \pm 0.024}$

(c) BA-Community

Method	**Sub-recall**	**Edge-recall**	**Fidelity**
GNNExplainer	0.006 ± 0.008	0.491 ± 0.010	0.653 ± 0.012
PGExplainer	0.238 ± 0.015	0.667 ± 0.011	0.652 ± 0.017
TAGE	0.352 ± 0.012	0.754 ± 0.008	0.672 ± 0.010
MixupExplainer	0.496 ± 0.029	0.857 ± 0.019	0.693 ± 0.004
SAME	0.132 ± 0.002	0.680 ± 0.010	0.642 ± 0.006
EiG-Search	0.078 ± 0.000	0.681 ± 0.000	0.695 ± 0.000
MotifExplainer	0.423 ± 0.045	0.714 ± 0.042	0.683 ± 0.017
UO-Explainer	$\mathbf{0.921 \pm 0.083}$	$\mathbf{0.970 \pm 0.030}$	$\mathbf{0.716 \pm 0.004}$

(d) Tree-Cycle

Method	**Sub-recall**	**Edge-recall**	**Fidelity**
GNNExplainer	0.119 ± 0.000	0.699 ± 0.005	0.724 ± 0.009
PGExplainer	0.926 ± 0.051	0.992 ± 0.005	0.732 ± 0.005
TAGE	0.963 ± 0.011	0.994 ± 0.002	0.734 ± 0.004
MixupExplainer	0.930 ± 0.012	0.994 ± 0.013	0.734 ± 0.002
SAME	0.101 ± 0.017	0.635 ± 0.023	0.692 ± 0.019
EiG-Search	0.083 ± 0.000	0.812 ± 0.000	0.703 ± 0.000
MotifExplainer	0.991 ± 0.000	0.993 ± 0.000	0.736 ± 0.001
UO-Explainer	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.737 \pm 0.000}$

$$Fidelity = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} f_{prob}(\mathcal{G}_{v_i}) - f_{prob}(\mathcal{G}_{v_i} - \mathcal{G}_{v_i}^{ex}), \quad (10)$$

where $f_{prob}(\mathcal{G}_{v_i})$ refers to the probability values of each node v_i from the trained GNN $f(G)$ with respect to the correct class.

Edge-recall indicates how many edges in the explanations match the edges in the ground truths. *Sub-Recall* indicates the proportion of correct answers that the entire presented explanations match with ground truths.

E. Results: Model-level Explanations

We first validate whether UO-Explainer can identify the correct orbits for model-level explanations. We pre-train 2 or 3-layer GCN models [5] and 2 or 3-layer GIN models [6] on Random Graph datasets. Each task is to classify whether the node belongs to the orbit corresponding to the task number. Consequently, explanation methods are expected to provide the ground-truth pattern in the form of the orbit (target node) with its corresponding graphlet (pattern) for each task. Tasks with accuracy below 0.8 are excluded as they are unlikely to yield accurate explanations. We use the *Sub-recall* metric to evaluate whether the provided explanation matches the ground-truth pattern.

In Table IV, UO-Explainer shows superior performance compared to other baselines. In particular, UO-Explainer constantly provides model-level explanations matching to ground truths for all tasks except 33, 60, and 62 in the GCN model as shown in Table IV-I. This limitation may arise from the GNN's expressiveness, failing to learn intended orbits during the pre-training. To address this, we conduct experiments in the same manner using GIN, which is known for its better expressiveness. The results are shown in Table IV-II that UO-Explainer accurately provides the explanations matching the correct ground-truth for all tasks. These findings confirm that UO-Explainer is capable of detecting various orbits and providing correct explanations while UO-Explainer's explanations mirror the expressiveness of the GNN's embedding model. Notably, comparing the performance of original GNNs to the decomposed class weights model, the performance degradation is less than 5%. On the other hand, D4Explainer and GLGExplainer show slightly improved or even decreased performance in the GIN setting and fail to provide consistent explanations matched with the ground truth.

As observed in Table V, UO-Explainer also outperforms other baselines on the BA-Shapes and BA-Community dataset. Using 3-layer GCNs for this experiment, UO-Explainer successfully provides accurate model-level explanations for each

TABLE VII: Instance-level explanation results on PPI datasets. The best fidelity on each dataset is shown in **bold**. $*$ notation indicates the lower sparsity setting. We conduct five experiments for each method by recording the mean and standard deviation.

(a) PPI – Task 0

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.100 ± 0.016	0.973
PGExplainer*	0.023 ± 0.000	0.973
TAGE*	0.031 ± 0.002	0.973
MixupExplainer	0.005 ± 0.000	0.973
EiG-Search	0.180 ± 0.072	0.973
SAME	0.022 ± 0.002	0.973
MotifExplainer	0.070 ± 0.008	0.992
UO-Explainer	$\mathbf{0.423 \pm 0.007}$	0.999

(b) PPI – Task 1

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.325 ± 0.004	0.999
PGExplainer*	0.155 ± 0.008	0.999
TAGE*	0.109 ± 0.003	0.999
MixupExplainer	0.002 ± 0.001	0.999
EiG-Search	0.269 ± 0.032	0.999
SAME	0.189 ± 0.003	0.999
MotifExplainer	0.074 ± 0.003	0.999
UO-Explainer	$\mathbf{0.358 \pm 0.022}$	0.999

(c) PPI – Task 2

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.417 ± 0.067	0.999
PGExplainer*	0.223 ± 0.077	0.999
TAGE*	0.233 ± 0.052	0.999
MixupExplainer	0.257 ± 0.002	0.999
EiG-Search	0.180 ± 0.053	0.999
SAME	0.194 ± 0.001	0.999
MotifExplainer	0.012 ± 0.028	0.999
UO-Explainer	$\mathbf{0.425 \pm 0.000}$	0.999

(d) PPI – Task 3

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.480 ± 0.012	0.999
PGExplainer*	0.101 ± 0.067	0.999
TAGE*	0.138 ± 0.034	0.999
MixupExplainer	0.246 ± 0.003	0.999
EiG-Search	0.379 ± 0.073	0.999
SAME	0.189 ± 0.004	0.999
MotifExplainer	0.129 ± 0.034	0.999
UO-Explainer	$\mathbf{0.510 \pm 0.057}$	0.999

(e) PPI – Task 4

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.250 ± 0.001	0.999
PGExplainer*	0.220 ± 0.001	0.999
TAGE*	0.214 ± 0.019	0.999
MixupExplainer	0.246 ± 0.003	0.999
EiG-Search	0.100 ± 0.079	0.999
SAME	0.034 ± 0.005	0.999
MotifExplainer	0.097 ± 0.005	0.999
UO-Explainer	$\mathbf{0.623 \pm 0.029}$	0.999

(f) PPI – Task 5

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.331 ± 0.001	0.999
PGExplainer*	0.005 ± 0.000	0.999
TAGE*	0.101 ± 0.020	0.999
MixupExplainer	0.129 ± 0.001	0.999
EiG-Search	0.180 ± 0.072	0.999
SAME	0.129 ± 0.033	0.999
MotifExplainer	0.050 ± 0.021	0.999
UO-Explainer	$\mathbf{0.413 \pm 0.027}$	0.999

TABLE VIII: Instance-level explanation results on LastFM dataset. The best fidelity is shown in **bold**. $*$ notation indicates the lower sparsity setting. We conduct five experiments for each method by recording the mean and standard deviation.

Method	**Fidelity**	**Sparsity**
GNNExplainer	0.114 ± 0.005	0.974
PGExplainer*	0.011 ± 0.002	0.974
TAGE*	0.086 ± 0.008	0.974
MixupExplainer	0.100 ± 0.001	0.974
EiG-Search	0.095 ± 0.021	0.974
SAME	0.049 ± 0.012	0.974
MotifExplainer	0.085 ± 0.004	0.994
UO-Explainer	$\mathbf{0.115 \pm 0.009}$	0.993

class on both datasets. Specifically on the BA-Shapes dataset, we observe the explanations that the house-like motif plays a crucial role in node classification by detecting the orbit such as o_{56}, o_{57}, and o_{58} in the graphlet g_{23} as explanations. This explanation sheds light on the overall behavior of the GNN beyond individual nodes, enabling a broader interpretation of GNNs. Moreover, our approach provides orbit-corresponding graphlets, allowing us to determine the topological positions of nodes within the motif for each class. Thus, UO-Explainer shows that the GNN recognizes the house-like motif as an important pattern for node classification, assigning nodes on the roof, floor, and top of the roof of the house-like motif to classes 1, 2, and 3, respectively. On the BA-Community dataset, UO-Explainer also finds the ground-truth pattern as the model-level explanation by detecting the orbit such as o_{56}, o_{57}, and o_{58} in the graphlet g_{23} as explanations, since the dataset is a union of two BA-SHAPES graphs. In conclusion, the experimental result demonstrates that orbits as pre-defined explanation units of UO-Explainer serve crucial patterns of the specific classes for prediction, showing accurate and consistent explanations.

F. Results: Instance-level Explanations

Except for MotifExplainer, all baselines provide explanations in the form of subgraphs obtained by extracting edges exceeding specific threshold values or ranking the top-k edge considered important. For fair experiments, explanations composed of top-k edges were extracted considering a sparsity level similar to that of UO-Explainer. In the case of MotifExplainer, we extract one motif as an instance-level explanation. UO-Explainer use a subgraph within the input graph that matches the highest-contributing orbit for the target node for the instance-level explanation.

The experimental results on synthetic datasets are shown in Table VI. UO-Explainer outperforms the baseline methods across all evaluation metrics while maintaining a comparable sparsity level. Notably, UO-Explainer achieves higher sub-recall, indicating accurate detection of ground-truth subgraphs as explanations. In cases such as Tree-grid and Tree-cycle, where grid- and cycle-shaped graphlets do not exist within the 2- to 5-node graphlets, we employ grid and cycle graphlets

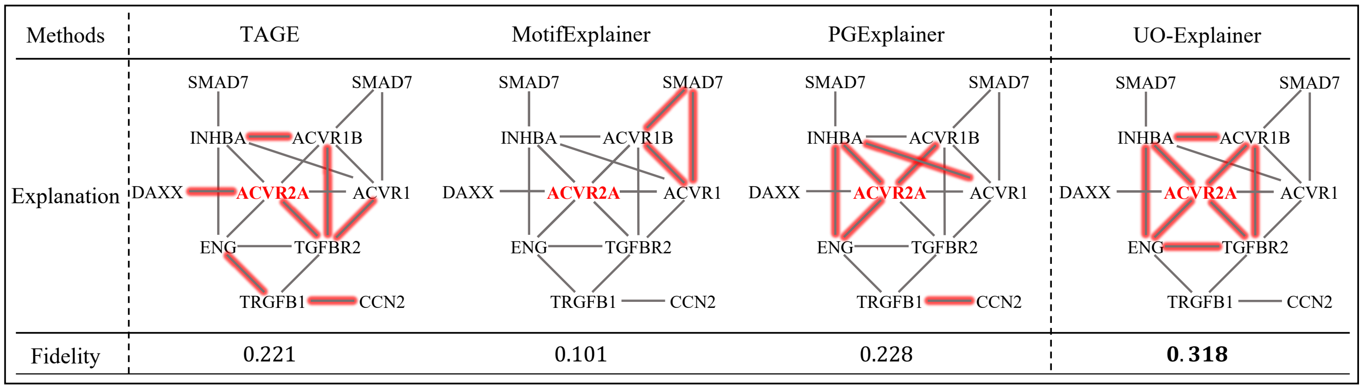

Fig. 4: Visualization of the explanations provided in the Gene dataset. Each node represents the ID of a gene, and the red nodes correspond to the target gene mentioned in the explanation. The red edges denote the edges included in the subgraph provided as part of the explanation.

along with their corresponding orbits and include them as units of explanation. We note that by defining custom graphlets based on background knowledge or extracted rules tailored to specific problem settings, the proposed method can be extended beyond the pre-defined 2- to 5-node graphlets and orbits.

Table VII shows the performance on PPI datasets. UO-Explainer outperforms other baselines in the fidelity metric. MotifExplainer struggles due to extracting too many motifs, leading to less meaningful explanations. PGExplainer and TAGE also show poor performance, often identifying the same or irrelevant edges across nodes, regardless of experimental changes. GNNExplainer, which is trained iteratively for node-specific explanations, extracts more relevant edges, especially on larger datasets. The $*$ notation in Table VII highlights that lower sparsity is needed for competing methods to match UO-Explainer, yet they still include noise edges or less important subgraphs. In contrast, UO-Explainer performs consistently well even under high sparsity, using just one orbit-based subgraph for explanations, demonstrating its ability to provide high-quality instance-level explanations. Similarly, Table VIII shows that our model achieves the best performance on the LastFM Asia dataset. Although GNNExplainer attains comparable results, our model exhibits high sparsity, yielding more concise explanations while maintaining high fidelity.

G. Case Study on Gene dataset

As the qualitative analysis, we visualize the explanatory subgraphs described in our method and the baselines on the gene dataset. The visualization results are presented in Figure 4. The experiment results demonstrate that PGExplainer and TAGE provide scattered subgraphs with discontinuous edges while the sparsity remains at $\mathbf{0.900}$ for fair comparison. In contrast, UO-Explainer offers a connected subgraph that is more intuitive while maintaining relatively high fidelity. MotifExplainer also presents a connected subgraph as an explanation but with relatively lower fidelity. Additionally, several studies provide evidence that the genes (TGFBR2 [49], ENG [50], INHBA, and ACVR2B [51]) identified by UO-Explainer in the explanations have an impact on the surface receptor signaling pathway, which is the label of the dataset. For example, in [52], it was mentioned that TGFBR2 is one of the TGF-beta receptors that transmit signals within natural killer cells, exerting a significant influence on cell development and the function of natural killer cells. These results imply that UO-Explainer provides human-interpretable explanations compared to other baselines regarding the perspective of interest.

H. Time Complexity Analysis

When UO-Explainer provides instance-level explanations, there are four time-consuming processes: 1) Pre-processing for finding the existence of the orbit, 2) Orbit basis learning, 3) Class-orbit score learning (Model-level explanation generation), and 4) Generating instance-level explanations.

1) Pre-processing for finding the existence of each orbit. As mentioned in Section IV-A, to learn the orbit basis, we must find the existence of each orbit for every node. This pre-processing can be conducted to search 2-5 node graphlets for each node. The time complexity of finding each graphlet that includes specific orbits, as can be inferred from Algorithm 3 and Algorithm 4, is $O(d^{k-1})$ [29]. Here, d represents the maximum node degree of the graph data, and k denotes the number of nodes included in each graphlet and is less than 5. Therefore, the time complexity of pre-processing for finding all 2-5 node graphlets that include a total 0-72 orbit for an entire node of graph data is $O(|\mathcal{O}|\,|\mathcal{V}|\,d^{k-1})$ where $|\mathcal{O}|$ denotes the number of orbits and $|\mathcal{V}|$ represents the number of nodes.

2) Orbit basis learning. The time complexity of orbit basis learning is $O(|\mathcal{O}|\,|\mathcal{V}|)$ since training is performed for each orbit basis over all nodes as shown in Algorithm 1.

3) Class-orbit score learning. Class-orbit score is trained based on the number of orbits selected by the greedy search for each class weight. In our experiments, the number of selected orbits was less than or equal to 5, so this can be sufficiently neglected. Therefore, the time complexity of class-orbit score learning considers only the number or class $\mathcal{C}$; $O(|\mathcal{C}|)$ as shown in Algorithm 2. The training time required for UO-Explainer and the baselines can be found in Table IX. Model-level explanations can be provided immediately after

TABLE IX: The training time required by the baselines and UO-Explainer to generate explanations on the Gene dataset. The training time of the explainer is significantly influenced by the epochs. Therefore, we set the epochs for each method through a hyperparameter search to achieve the highest fidelity.

	Instance and Model-level	Instance-level				Model-level
	UO-Explainer	GNNExplainer	MotifExplainer	PGExplainer	TAGE	D4Explainer
Training Time(s)	268 (orbit basis learning)+21 (class-orbit score learning)	$1,681$	$6,074$	98	204	502

TABLE X: The time required and fidelity based on the number of sampled graphlets of UO-Explainerin the gene dataset. We conduct five experiments for each method and reported the mean and standard deviation values. The "Entire" refers to the extraction of all graphlets surrounding the target node without sampling.

# of sampled graphlets	1	10	30	50	70	100	Entire
Time(s)	14.572 ± 0.208	136.342 ± 2.305	379.342 ± 22.438	597.283 ± 3.865	838.991 ± 3.587	1183.459 ± 5.913	6321.783 ± 8.462
Fidelity	0.157 ± 0.018	0.256 ± 0.009	0.297 ± 0.011	0.309 ± 0.014	0.326 ± 0.008	0.324 ± 0.007	0.3543 ± 0

class-orbit score learning. Therefore, the training time of UO-Explainer listed in Table X equals the time required to provide model-level explanations.

4) Generating instance-level explanations. To provide instance-level explanations, it is necessary to search graphlets that include the given orbits as explanations for each node and select just one graphlet having the highest fidelity. Therefore, the time complexity is $O(|\mathcal{V}|\,d^{k-1})$ since graphlets need to be directly found for each node. However, instead of finding all graphlets around each node, we sample a predetermined number of graphlets. Thus, the actual computational cost can be significantly reduced compared to $O(|\mathcal{V}|\,d^{k-1})$, while preserving high explanation performance. The required time and fidelity based on the number of sampled graphlets are shown in Table VI. Even with a time difference of more than 7 times, as observed when comparing the cases of not sampling and sampling 70 graphlets for explanation extraction, it can be observed that the fidelity values do not significantly decrease.

The time complexities of the baseline methods are as follows: D4Explainer has a time complexity of $O(|\mathcal{V}|^3)$, GNNExplainer is $O(|\mathcal{V}|\,|\mathcal{E}|)$ where $|\mathcal{E}|$ denotes the number of edges, PGExplainer and TAGE have a time complexity of $O(|\mathcal{E}|)$, and MotifExplainer operates with a complexity of $O(|\mathcal{V}|\,|\mathcal{M}|)$, where $|\mathcal{M}|$ represents the number of motifs used in explanations. Therefore, in terms of time complexity, UO-Explainer is less demanding compared to D4Explainer and GNNExplainer. Alongside such analysis, our unified model is capable of providing both model-level and instance-level explanations simultaneously, thus demonstrating its competitiveness in terms of time efficiency.

VI. Discussion and Conclusion

We introduce UO-Explainer, a human-interpretable explanation method that leverages pre-defined units, as requested by users, in a unified framework for node classification models. By utilizing orbits as explanatory units, UO-Explainer decomposes model weights into orbit components, which serve as essential, human-interpretable units within graph domains. Experimental results on both synthetic and real-world datasets demonstrate the effectiveness of UO-Explainer, outperforming baseline methods and delivering higher-quality explanations. By using pre-defined explanation units, users can uncover meaningful patterns and gain deeper insights through UO-Explainer, particularly valuable in scientific applications, such as drug development and education, where domain knowledge is critical. While our extensive experiments show promising results, the motifs we currently employ are limited to 2–5 node graphlets. Consequently, we recognize that this constraint may occasionally restrict the capture of key motifs or patterns when critical patterns are extremely large in highly complex graphs. While many existing approaches can approximate various subgraph shapes, they often overlook the user-centric perspective. In contrast, our method focuses on scenarios where prior assumptions or knowledge based on predefined units are crucial for human-interpretable explanations. In conclusion, we believe this work contributes to the understanding of GNN's behavior in a unified framework, leveraging the human prior knowledge as orbits for future research in human-centered explainable AI.

References

[1] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *Proceedings of the Web Conference*, 2019.

[2] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[3] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier *et al.*, "Knowledge graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, 2021.

[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017.

[6] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proceedings of the International Conference on Learning Representations*, 2019.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, 2018.

[8] T. Gaudelet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. Hayter, R. Vickers, C. Roberts, J. Tang *et al.*, "Utilizing graph machine learning within drug discovery and development," *Briefings in bioinformatics*, vol. 22, no. 6, p. bbab159, 2021.

[9] H. Nakagawa, Y. Iwasawa, and Y. Matsuo, "Graph-based knowledge tracing: modeling student proficiency using graph neural network," in *Proceeding of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2019.

[10] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods, and trends," *Proceedings of the IEEE*, vol. 112, no. 2, p. 97–139, 2024.

[11] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[12] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *Advances in Neural Information Processing Systems*, 2020.

[13] Y. Xie, S. Katariya, X. Tang, E. Huang, N. Rao, K. Subbian, and S. Ji, "Task-agnostic graph explanations," *Advances in Neural Information Processing Systems*, 2022.

[14] J. Zhang, D. Luo, and H. Wei, "Mixupexplainer: Generalizing explanations for graph neural networks with data augmentation," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3286–3296.

[15] M. S. Schlichtkrull, N. D. Cao, and I. Titov, "Interpreting graph neural networks for nlp with differentiable edge masking," *arXiv preprint:2010.00577*, 2022.

[16] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," *arXiv preprint:1905.13686*, 2019.

[17] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 772–10 781.

[18] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *arxiv preprint:2012.15445*, 2022.

[19] M. A. Prado-Romero, B. Prenkaj, G. Stilo, and F. Giannotti, "A survey on graph counterfactual explanations: Definitions, methods, evaluation, and research challenges," *ACM Computing Surveys*, vol. 56, no. 7, p. 1–37, Apr. 2024.

[20] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 430–438.

[21] Y.-M. Shin, S.-W. Kim, E.-B. Yoon, and W.-Y. Shin, "Prototype-based explanations for graph neural networks (student abstract)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[22] Z. Ye, R. Huang, Q. Wu, and Q. Liu, "Same: Uncovering gnn black box with structure-aware shapley-based multipiece explanations," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[23] S. Lu, B. Liu, K. G. Mills, J. He, and D. Niu, "Eig-search: Generating edge-induced subgraphs for gnn explanation in linear time," in *Proceedings of the International Conference on Machine Learning*, 2024.

[24] S. Azzolin, A. Longa, P. Barbiero, P. Liò, and A. Passerini, "Global explainability of gnns via logic combination of learned concepts," in *Proceedings of the International Conference on Learning Representations*, 2023.

[25] J. Chen, S. Wu, A. Gupta, and R. Ying, "D4explainer: In-distribution gnn explanations via discrete denoising diffusion," *Advances in Neural Information Processing Systems*, 2023.

[26] N. Pržulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?" *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, 2004.

[27] N. Pržulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, 2007.

[28] X. Chen and J. C. Lui, "Mining graphlet counts in online social networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 4, pp. 1–38, 2018.

[29] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *Proceeding of the International Conference on Data Mining*. IEEE, 2015, pp. 1–10.

[30] R. Kondor, N. Shervashidze, and K. M. Borgwardt, "The graphlet spectrum," in *Proceedings of the International Conference on Machine Learning*, 2009.

[31] E. Dai, T. Zhao, H. Zhu, J. Xu, Z. Guo, H. Liu, J. Tang, and S. Wang, "A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability," *arxiv preprint:2204.08570*, 2023.

[32] B. Wu, J. Li, J. Yu, Y. Bian, H. Zhang, C. Chen, C. Hou, G. Fu, L. Chen, T. Xu, Y. Rong, X. Zheng, J. Huang, R. He, B. Wu, G. Sun, P. Cui, Z. Zheng, Z. Liu, and P. Zhao, "A survey of trustworthy graph learning: Reliability, explainability, and privacy protection," *arxiv preprint:2205.10014*, 2022.

[33] Z. Yu and H. Gao, "Motifexplainer: a motif-based graph neural network explainer," *arXiv preprint arXiv:2202.00519*, 2022.

[34] P. Barbiero, G. Ciravegna, F. Giannini, P. Lió, M. Gori, and S. Melacci, "Entropy-based logic explanations of neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6046–6054.

[35] G. Ciravegna, P. Barbiero, F. Giannini, M. Gori, P. Lió, M. Maggini, and S. Melacci, "Logic explained networks," *Artificial Intelligence*, vol. 314, p. 103822, 2023.

[36] M. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," *Advances in Neural Information Processing Systems*, 2020.

[37] J. Wang, M. Luo, J. Li, Y. Lin, Y. Dong, J. S. Dong, and Q. Zheng, "Empower post-hoc graph explanations with information bottleneck: A pre-training and fine-tuning perspective," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2349–2360.

[38] P. Xiong, T. Schnake, M. Gastegger, G. Montavon, K. R. Muller, and S. Nakajima, "Relevant walk search for explaining graph neural networks," in *Proceedings of the International Conference on Machine Learning*, 2023, pp. 38 301–38 324.

[39] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009, pp. 488–495.

[40] R. Espejo, G. Mestre, F. Postigo, S. Lumbreras, A. Ramos, T. Huang, and E. Bompard, "Exploiting graphlet decomposition to explain the structure of complex networks: the ghust framework," *Scientific Reports*, vol. 10, no. 1, p. 12884, 2020.

[41] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical Review E*, vol. 65, no. 2, p. 026107, 2002.

[42] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[43] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," in *Proceedings of the International Conference on Information & Knowledge Management*, 2020.

[44] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov, "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences*, vol. 102, no. 43, pp. 15 545–15 550, 2005.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[46] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[47] M. Liu, Y. Luo, L. Wang, Y. Xie, H. Yuan, S. Gui, H. Yu, Z. Xu, J. Zhang, Y. Liu *et al.*, "Dig: A turnkey library for diving into graph deep learning research," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 10 873–10 881, 2021.

[48] Y. Li, J. Zhou, S. Verma, and F. Chen, "A survey of explainable graph neural networks: Taxonomy and evaluation metrics," *arXiv preprint arXiv:2207.12599*, 2022.

[49] J. Massagué and R. R. Gomis, "The logic of tgfβ signaling," *FEBS letters*, vol. 580, no. 12, pp. 2811–2820, 2006.

[50] M. J. Breen, D. M. Moran, W. Liu, X. Huang, C. P. Vary, and R. C. Bergan, "Endoglin-mediated suppression of prostate cancer invasion is regulated by activin and bone morphogenetic protein type ii receptors," *PLoS One*, vol. 8, no. 8, p. e72407, 2013.

[51] L. Attisano and J. L. Wrana, "Signal integration in tgf-β, wnt, and hippo pathways," *F1000prime reports*, vol. 5, 2013.

[52] C. Bottino, T. Walzer, A. Santoni, and R. Castriconi, "Tgf-β as a key regulator of nk and ilcs development and functions," p. 631712, 2021.