GIN-Graph: A Generative Interpretation Network for Model-Level Explanation of Graph Neural Networks

1st Xiao Yue
Department of Computer Science and Engineering
Oakland University
Rochester, MI, USA
xiaoyue@oakland.edu

2nd Guangzhi Qu
Department of Computer Science and Engineering
Oakland University
Rochester, MI, USA
gqu@oakland.edu

***Abstract*—One significant challenge of exploiting Graph neural networks (GNNs) in real-life scenarios is that they are treated as black boxes, therefore leading to the requirement of interpretability. To address this, model-level interpretation methods have been developed to explain what patterns maximize probability of predicting to a certain class. However, existing model-level interpretation methods pose several limitations such as generating invalid explanation graphs and lacking reliability. In this paper, we propose a new Generative Interpretation Network for Model-Level Explanation of Graph Neural Networks (GIN-Graph), to generate reliable and high-quality model-level explanation graphs. The implicit and likelihood-free generative adversarial networks are exploited to construct the explanation graphs which are similar to original graphs, meanwhile maximizing the prediction probability for a certain class by adopting a novel objective function for generator with dynamic loss weight scheme. Experimental results demonstrate that GIN-Graph consistently generates high-quality model-level explanation graphs with high stability and reliability across diverse graph datasets.**

***Index Terms*—Graph neural networks, Graph generative models, Model-Level Interpretation**

I. Introduction

The increasing availability of graph data has led to the deployment of graph neural networks (GNNs) in various domains, including social analysis [1], biology, transportation, and financial systems [2]. Although GNNs have shown strong capability in learning from graph-structured data, one significant challenge in real-world applications is that they are typically treated as black boxes, leading to the requirement for interpretability. In critical domains, models can only be trusted if their predictions can be explained in human-understandable ways. Interpretation methods for GNNs can be categorized into instance-level and model-level approaches, where the latter aims to explain general model behaviors. Model-level interpretations on GNNs are less explored, as input optimization methods [3] cannot be directly applied due to the discrete topological information of graphs such as the adjacency matrix. Yuan et al. first proposed XGNN [4] to interpret GNNs at the model level by training a graph generator to produce explanation graphs that maximize the prediction probability for a certain class. However, it formulates graph generation as a reinforcement learning task with predefined policies, whose manual design demands significant human effort. To address this limitation, GNNInterpreter (GNNI) [5] was developed to build explanation graphs by learning a probabilistic generative graph distribution. However, the two objective properties optimized by GNNI, prediction probability and embedding similarity, do not necessarily lead to meaningful explanation graphs. Moreover, it lacks an effective metric for evaluating explanation quality, and the generated graphs may suffer from limited stability and reliability.

To address these limitations, we propose a new Generative Interpretation Network for Model-Level Explanation of Graph Neural Networks (GIN-Graph), which generates more reliable and high-quality model-level explanation graphs. GIN-Graph exploits an implicit and likelihood-free generative adversarial network (GAN) to construct explanation graphs that are topologically similar to real graphs while maximizing the prediction probability for a target class through a novel objective function with a dynamic loss weighting scheme. The Gumbel-Softmax [6] reparameterization is adopted to handle the discrete adjacency matrix in back-propagation. In addition, a graph pruning strategy is introduced to improve fine-grained explanations when the important patterns learned by a GNN are much smaller than the full graphs. The contributions of this paper are as follows:

- We investigate the properties of model-level explanation graphs and define rules and a score metric for selecting valid explanation graphs.
- We develop GIN-Graph with a novel objective function and dynamic loss weighting scheme to generate high-quality model-level explanation graphs, and introduce a graph pruning preprocessing strategy for fine-grained model-level explanations.

It is important to clarify that the explanation graphs discussed in this paper refer to model-level explanations for a GNN on a specific class, rather than explanations of a class within a graph dataset. A model-level explanation focuses on interpreting the patterns learned by the model itself, and there-

fore its generation must be guided by the model's feedback.

II. Related Works

A. GNNs Interpretations

There is a rising interest in enhancing the interpretations of GNNs [7] due to the growing popularity of real-world graph data. One significant challenge of interpreting graph-related models is the discrete nature of the adjacency matrix that represents graph topology, leading to difficulties in employing existing interpretation methods such as input optimization [8]. Based on the type of information being delivered, interpretations of GNNs can be categorized into instance-level and model-level. Instance-level interpretation methods explain individual predictions by identifying important features, nodes, or subgraphs. For example, GNNExplainer [9] learns a compact subgraph and a subset of node features that are most influential for a specific prediction. In contrast, model-level interpretations aim at explaining the general behaviors of models. XGNN, the first model-level interpretation method for GNNs, exploits a graph generator to construct explanation graphs that maximize the prediction probability for a certain class. However, it formulates graph generation as a reinforcement learning task with predefined policies, whose manual design is labor-intensive. To address this limitation, GNNI learns a probabilistic generative graph distribution to generate discriminative graph patterns by optimizing a novel objective function, making it more flexible and computationally efficient than XGNN. GLGExplainer [10] generates model-level explanations in the form of logic formulas based on local explanations, but its performance heavily depends on the quality of the underlying instance-level interpreter.

B. Graph Generative Models

Deep graph generative models have drawn significant attention due to the successful applications of generative models in other domains. Similar to GNNs, graph generative models face the challenge that machine learning techniques developed primarily for continuous data cannot be directly applied to a discrete adjacency matrix. Existing graph generative models can be broadly categorized into sequential generation and one-shot generation [11]. Sequential generation methods [12], [13] construct nodes and edges step by step, while one-shot generation methods [14], [15] generate all nodes and edges simultaneously based on matrix representations. These models have been widely used in graph-centric applications such as molecule design [16], [17] and protein structure modeling [18].

III. Model-level Explanation Graphs

A. Valid and invalid explanations

Model-level interpretations of GNNs aim to provide explanation graphs that maximize the probability of predicting a specific class. However, relying only on the prediction probability is insufficient for evaluating explanation graphs. Since most GNNs are not trained to directly handle out-of-distribution (OOD) graphs, some graphs with high prediction probabilities may still be structurally unrealistic and fail to provide meaningful insights. GNNI addresses this issue by additionally maximizing the similarity between the embedding of an explanation graph and the average embedding of all graphs of a certain class. Nevertheless, prediction probability and embedding similarity alone may still lead to invalid explanation graphs, as illustrated in Figure 1.

To better evaluate explanation graphs, we further utilize the degree density x of a graph ($x = e/n$, where e is the number of edges and n is the number of nodes). We compute the mean (μ) and standard deviation (σ) of degree densities for all graphs in a certain class, and regard graphs with degree densities outside the range $[\mu - t * \sigma, \mu + t * \sigma]$ as invalid explanation graphs, where $t = 3$. Based on this, we define a validation score v as $v = \sqrt[3]{s * p * d}$, which integrates embedding similarity (s), prediction probability (p), and degree score (d), where $d = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. In this work, the validation score is used to assess the quality of an explanation graph. The score ranges from *0* to *1*, with higher values indicating higher-quality explanations.

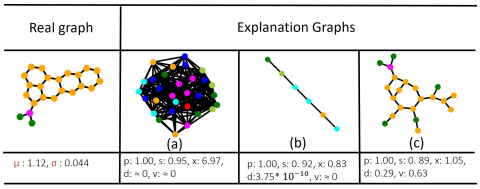

Fig. 1. Illustration of valid and invalid explanation graphs

B. Fine-grained and coarse-grained explanations

Explanation graphs can also vary in their level of detail. A smaller explanation graph tends to highlight fine-grained patterns learned by the model, while a larger explanation graph tends to reflect broader structural patterns. Accordingly, we categorize explanation graphs into two types: *fine-grained* and *coarse-grained*. As shown in Figure 2, fine-grained explanation graphs capture more detailed substructures, whereas coarse-grained explanation graphs are usually more similar to the original graphs and describe higher-level structural patterns. To quantify this property, we define a granularity-related metric k as $k = 1 - min(1, b/a)$, where a is the average number of nodes across all graphs in a certain class and b is the number of nodes in an explanation graph for that class. The metric ranges from *0* to *1*, where a larger value indicates a more fine-grained explanation.

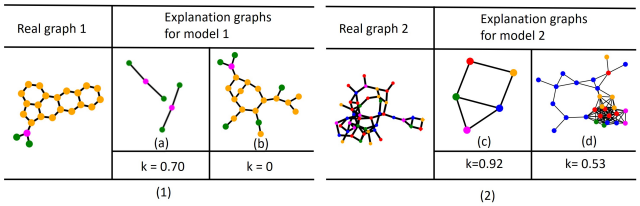

Fig. 2. Illustration of fine-grained and coarse-grained explanation graphs

IV. Approaches

We present the details of GIN-Graph in this section. The overview of GIN-Graph is shown in Figure 3. It exploits a generative adversarial network to construct explanation graphs using the feedback from the GNN to be explained.

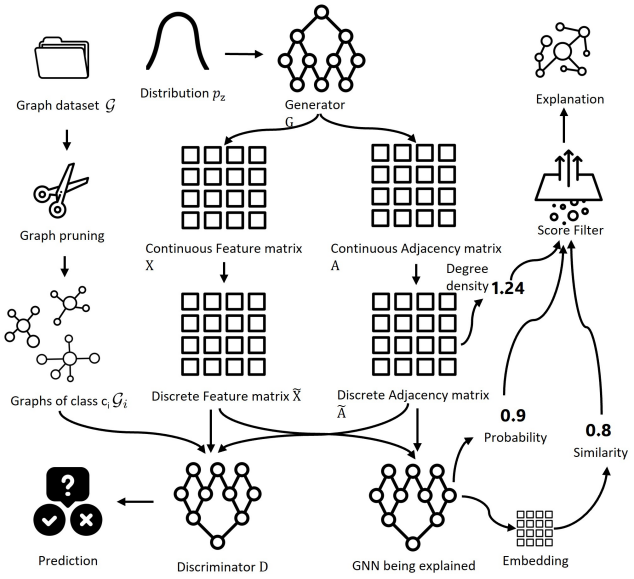

Fig. 3. Overview of the Architecture of GIN-Graph

A. Optimization goal

Let $f(\cdot)$ denote a GNN model to be interpreted, which is trained on a graph dataset $\mathcal{G}$ with classes $\{c_1, ..., c_l\}$. For a target class c_i, the model-level interpretation of $f(\cdot)$ can be formulated as deriving an explanation graph g^* such that $g^* = \arg\max_g P(f(g) = c_i)$. Since prediction probability alone is insufficient for assessing the quality of an explanation graph, we introduce a score filter $W(\cdot)$ for post hoc selection, which determines whether a graph satisfies preset rules, such as having a validation score (v) above a threshold. Accordingly, the objective is to derive an explanation graph g^* such that $g^* = \arg\max_g P(f(g) = c_i | W(g) = True)$.

B. Generative Adversarial Network

1) Generator and Discriminator: We denote the set of all graphs in class c_i as $\mathcal{G}_i$. A generative adversarial network consisting of a generator $G(\cdot)$ and a discriminator $D(\cdot)$ is exploited to generate explanation graphs, with parameters denoted as θ_G and θ_D, respectively. The generator $G(\cdot)$ is implemented as a multi-layer perceptron, which takes a sampled vector z from a distribution p_z as input and generates graphs, denoted as $G(z) = \hat{g}$. In this case, two matrices with continuous values, an adjacency matrix and a node feature matrix, represent the graph for downstream tasks. The parameters of the generator θ_G are optimized using the objective function defined in Equation 1, where $L_{GAN} = -\mathbf{E}_{z\sim p_z}[D(G(z))]$ and L_{GNN} is the cross-entropy loss or logits loss derived from the GNN.

$$\theta_G = \arg\min_{\theta}((1-\lambda)L_{GAN} + \lambda L_{GNN}(f(\hat{g}), c_i)) \quad (1)$$

As both generated and real graphs are represented by an adjacency matrix and a node feature matrix, we exploit a GNN as the discriminator. The discriminator $D(\cdot)$ is utilized to differentiate real graphs from generated ones. We denote the distribution of existing graphs as p_x. The parameters of the discriminator are optimized by the Wasserstein GAN (WGAN) loss with gradient penalty shown in Equation 2, where $\hat{x} = \epsilon x + (1-\epsilon)G(z)$, $\epsilon \sim \mathcal{U}(0,1)$, and α is a hyperparameter of gradient penalty.

$$\theta_D = \arg\min_{\theta} \mathbf{E}_{z\sim p_z}[D(G(z))] - \mathbf{E}_{x\sim p_x}[D(x)] + \alpha(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2 \quad (2)$$

2) Adjacency matrix and node feature matrix: The generator $G(\cdot)$ outputs two matrices with continuous values, a continuous adjacency matrix A and a continuous node feature matrix X. GIN-Graph uses fixed-size graph matrices to handle variable graph sizes. Larger graphs are reduced via preprocessing by removing nodes and edges, which is adequate for capturing model-level discriminative patterns. If there is no edge attribute in a graph, each element in the discrete adjacency $\widetilde{A}$ should be either *0* or *1*, indicating the absence or presence of an edge, respectively. The discrete adjacency matrix can then be derived through binary categorical sampling from A. If edges have categorical features with *k* classes, these features are encoded into a (*k+1*)-dimensional vector by one-hot encoding, where the extra dimension indicates the absence of an edge. In this case, each element in the adjacency matrix is treated as a (*k+1*)-dimensional vector instead of a single value. The node feature matrix X can be directly fed into the discriminator if node features are numerical. If node features are categorical, we exploit the same categorical sampling method used for the adjacency matrix to derive a discrete node feature matrix $\widetilde{X}$. Since categorical sampling is not differentiable, we address this obstacle by exploiting the categorical reparameterization with Gumbel-Softmax [6].

3) GNN being explained: Similar to the discriminator $D(\cdot)$, the GNN being explained should also take graph data as input. The GNN outputs the embedding and prediction probability of a generated explanation graph for evaluation. Note that the GNN being explained does not have to share the same model structure as the discriminator, as long as it can accept an adjacency matrix and a node feature matrix as inputs, or any other data formats that can be derived through differentiable transformations from these two matrices.

C. Dynamic loss weight scheme

1) Pitfalls of gradients from GNNs: One might envision an ideal model-level GNN interpreter as a model that takes a pre-trained GNN as input and generates explanation graphs that maximize the prediction probability of a certain class without relying on graphs from the original training dataset.

However, as most GNNs are not trained to directly handle out-of-distribution graphs, the gradients derived from such graphs do not necessarily guide the generator toward valid explanation graphs. As a result, the generator tends to converge to invalid explanation graphs with maximum prediction probability. Our experiments showed that a generator trained solely by optimizing the prediction probability from a GNN quickly converged to invalid explanation graphs. This highlights the necessity of using the original training dataset for interpretation.

2) Time-dependent weighting parameter: Compared to the large and unstable gradients derived from the GNN, the GAN loss (L_{GAN}) from the discriminator typically produces smaller and more stable gradients for the generator due to the adversarial training strategy. Training the GAN solely with L_{GAN} makes the generator tend to produce graphs similar to those in the training dataset, rather than graphs that maximize the prediction probability of a certain class. However, directly combining L_{GAN} with the loss derived from the GNN being explained (L_{GNN}) introduces a risk that L_{GNN} may dominate the training process, leading the generator to converge to invalid explanation graphs. To mitigate this risk, we propose a dynamic weighting scheme by developing a time-dependent weighting function for λ in Equation 1. The weighting function is shown in Equation 3, where $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum and maximum values of the weight parameter, t is the current training iteration, T is the total number of iterations, p controls the ratio of the training period used to increase λ, and k is a temperature parameter controlling the sharpness of the transition. Function σ represents a weight adjustment function such as a *sigmoid* function. This dynamic weighting scheme ensures that L_{GAN} dominates the early stages of training to first anchor the generator in-distribution, while L_{GNN} gradually takes effect in later stages to fine-tune explanation graphs for maximum prediction probability.

$$\lambda(t) = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min})\,\sigma\left(2k\left(\frac{t/T - p}{1-p}\right) - 1\right) \quad (3)$$

D. Graph pruning

Most message-passing-based GNNs rely on local structural patterns (e.g., subgraphs) for prediction [9], suggesting that many nodes and edges in a graph may be redundant. To better capture these informative patterns, especially when they are much smaller than the full graph, we introduce a graph pruning strategy as a preprocessing step. The key idea is to iteratively remove nodes and edges that have limited impact on the prediction of the GNN being explained. Specifically, given a graph, we progressively prune a subset of nodes and evaluate the prediction probability of the pruned graph. If the prediction confidence for the target class is preserved, the pruning step is accepted; otherwise, it is discarded. This process is repeated until a stopping condition is met. By filtering out irrelevant structures while preserving predictive patterns, the resulting pruned graphs are more compact and better aligned with the underlying decision logic of the GNN. Consequently, GIN-Graph is able to generate more fine-grained and interpretable explanation graphs based on the pruned graph distribution.

V. Experiments

A. Dataset and experiment setting

GIN-Graph was evaluated on four GNN models trained on four datasets: one real-world dataset *MUTAG* and three synthetic datasets *Cyclicity*, *Shape* and *Motif*. The *MUTAG* [19] dataset is a widely recognized benchmark dataset in graph learning, where each compound sample is labeled according to its mutagenic effect on a bacterium. The datasets *Cyclicity*, *Shape* and *Motif* are introduced in [5]. GIN-Graph is implemented using PyTorch with python version 3.10. To compare GIN-Graph with GNNI, we evaluated both interpreters on the same GNNs. Since GNNI has already been shown to outperform XGNN and GLGExplainer provides results that are not fully model-level and cannot be evaluated using the same metrics, we focus on comparing GIN-Graph to GNNI. We applied graph pruning to the *Cyclicity* and *Motif* datasets, where the important patterns are much smaller than the full graphs. Since graph pruning is part of GIN-Graph and GNNI does not take graphs as inputs, it cannot be applied to GNNI. We trained both GNNI and GIN-Graph 100 times per class and selected the best explanation graphs using the same qualitative criteria based on dataset knowledge. Since model-level explanation of GNNs lacks widely accepted quantitative evaluation metrics, we adopt the validation score as a validity constraint rather than a definitive performance metric.

B. Experimental results

The experimental results for GNNs trained on *MUTAG*, *Motif*, *Shape* and *Cyclicity* are shown in Figure 4. The best explanation graphs of each class generated by GNNI are presented in the column **GNNI (validation)**. Since we were unable to reproduce all results in the GNNI paper due to the loss of original data and the lack of detailed hyperparameter guidance, we also present the explanation graphs from their paper [5] in column **GNNI (paper)** for reference. Experimental results demonstrate that GIN-Graph consistently outperforms GNNI across all evaluated datasets. For the GNN trained on the *Motif* dataset, GIN-Graph accurately generated all motifs learned by the GNN, while GNNI failed for classes *house*, *house-X* and *comp4*. For the GNN trained on the *MUTAG* dataset, neither interpreter could show that both NO_2 and NH_2 substructures are strong evidence for molecule mutagenicity, but GIN-Graph was able to capture the carbon-based chemical structures of molecules. For the GNNs trained on the *Shape* and *Cyclicity* datasets, both interpreters produced valid explanation graphs for all classes.

Evaluating a model only based on the best explanation graph among hundreds of graphs is insufficient. Therefore, we further trained each model with the selected best parameter settings for 10 runs on each class. For each run, we selected the top 10 explanation graphs based on validation scores, resulting in 100 graphs for each class per model. To ensure a fair comparison, we disabled the filter mechanism in GIN-Graph so that low-scoring graphs were not excluded. We computed the average validation scores of the selected explanation graphs

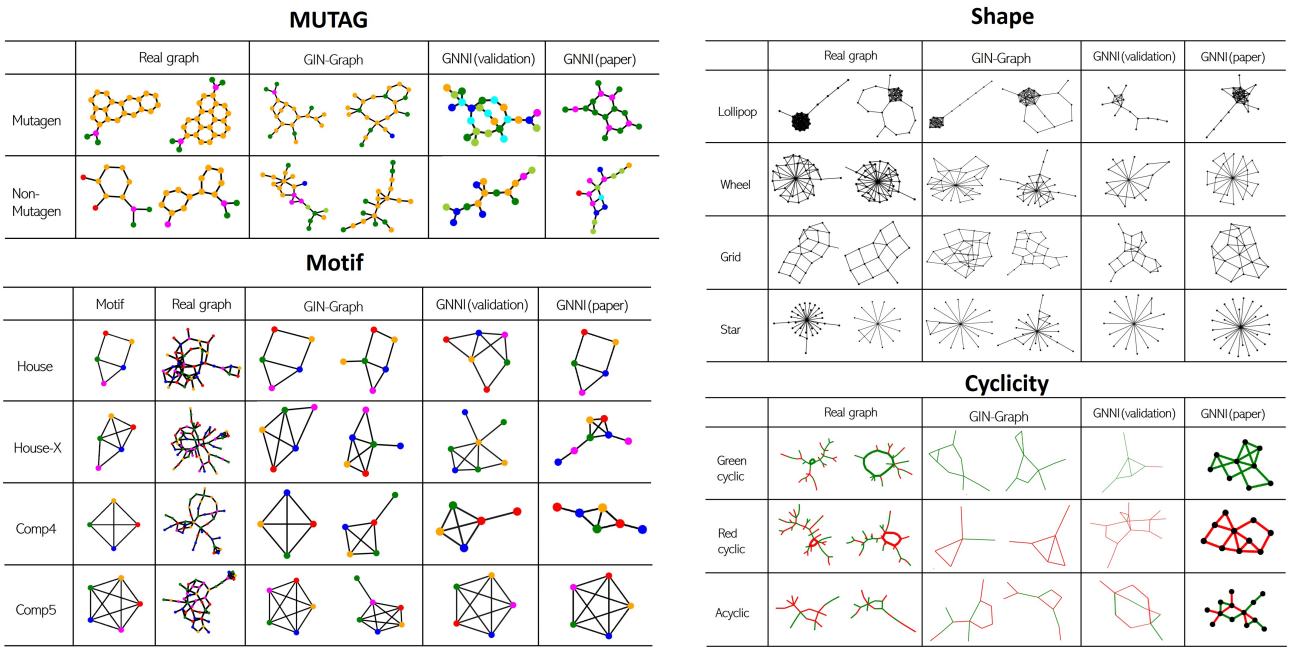

Fig. 4. Experimental results on GNNs trained on four datasets

and present the results in Table I. Note that the objective function of GNNI optimizes both prediction probability and embedding similarity, while prediction probability is the only measurement optimized in GIN-Graph. Therefore, the validation score inherently favors GNNI. Despite this disadvantage, GIN-Graph consistently achieved higher average validation scores than GNNI on all classes.

TABLE I
EXPERIMENTAL RESULTS ON VALIDATION SCORES OF MODELS TRAINED ON FOUR DATASETS

Dataset	Class	GIN-Graph	GNNI
MUTAG	Mutagen	0.972	0.904
	Non-Mutagen	0.981	0.935
Shape	Lollipop	0.994	0.781
	Wheel	0.985	0.923
	Grid	0.994	0.916
	Star	0.999	0.935
Motif	House	0.927	0.157
	House-X	0.910	0.106
	Comp4	0.955	0.187
	Comp5	0.846	0.578
Cyclicity	Red Cyclic	0.989	0.954
	Green Cyclic	0.999	0.847
	Acyclic	0.965	0.886

C. Experiments on convergence states and stability

During our experiments, we observed that valid explanation graphs from GNNI were typically generated at the early stages of training, usually within the first 100 iterations. If no valid explanation graphs emerged early, continued training did not lead to their emergence. In contrast, GIN-Graph consistently generated valid explanation graphs with sufficient training time. We selected the GNN trained on the *Shape* dataset for this analysis, since each class exhibits distinct topological features. Both models were trained with extended training time, and convergence was defined as the state where generated graphs remained stable over a long period. Each model was trained 10 times per class and the results were consistent across runs, as shown in Figure 5. Experimental results indicate that the convergence states of GNNI do not consistently correspond to valid explanation graphs, whereas GIN-Graph consistently converged to states where high-quality valid explanations were generated. In particular, GIN-Graph successfully generated valid explanation graphs in all *10* runs for every class, while GNNI only succeeded in *2*, *9*, *4*, and *10* out of 10 runs for the *Lollipop*, *Wheel*, *Grid*, and *Star*, respectively. These results reveal the limited stability and reliability of GNNI.

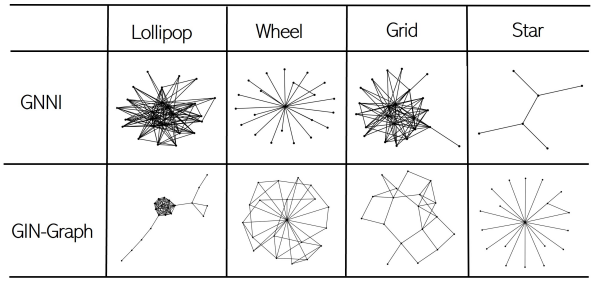

Fig. 5. Generated explanation graphs after convergence

D. Experiments on effectiveness of graph pruning

We evaluated the effectiveness of graph pruning on the *Motif* dataset and the GNN trained on it. Each graph in this dataset is constructed by connecting a random Rome graph to a specific

motif, and the motif type is the only distinguishing factor between classes. We first evaluated the prediction accuracies of the GNN on original graphs, and then on graphs with all motifs removed. For graphs with motifs, the prediction accuracies of *house*, *house-X*, *comp4*, and *comp5* were 99.21%, 99.80%, 99.13%, and 100.00%, respectively. For graphs without motifs, the prediction accuracies were 0.17%, 0.00%, 0.00%, and 0.00%, respectively. This indicates that motifs are the critical patterns for accurate prediction. Since a motif contains only 4 or 5 nodes while each full graph contains more than 50 nodes, graph pruning was applied to remove redundant nodes and edges, and all motifs remained intact regardless of the pruning intensity.

To further analyze the effect of pruning, we projected the embeddings of graphs and motifs into two-dimensional space. As shown in Figure 6(a), motifs lie outside the clusters of original graph embeddings, suggesting that generating fine-grained explanation graphs exactly matching the motifs is difficult when relying on embeddings alone. This observation is consistent with the failure of GNNI to recover the motifs for classes *house*, *house-X* and *comp4*. After pruning, the graph embeddings became more tightly clustered and the motif embeddings moved closer to their corresponding clusters, as shown in Figure 6(b), indicating that the pruned graphs are more similar to the real motifs. However, graph pruning is mainly suitable for datasets in which the important patterns are much smaller than the full graphs. On the *MUTAG* dataset, pruning tends to retain the NO_2 structure, but may break carbon-based structures such as carbon rings, resulting in explanation graphs that no longer resemble valid chemical compounds. Therefore, graph pruning is effective for deriving smaller graphs while retaining important patterns, but its applicability depends on the structural characteristics of the dataset.

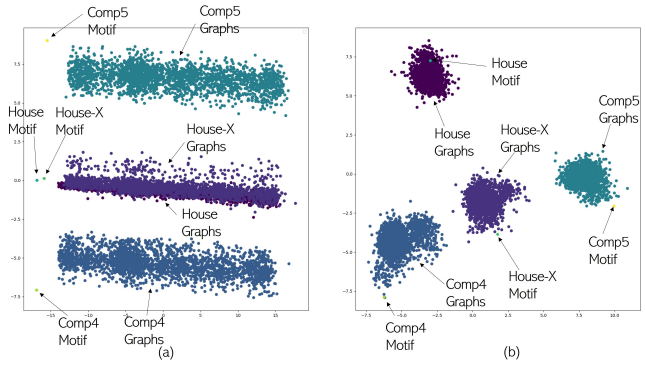

Fig. 6. Embedding distributions of graphs before (a) and after (b) graph pruning

VI. Conclusion

In this paper, we systematically study model-level explanation graphs, and define a validation score and a granularity related explanation metric to evaluate explanation graphs. We proposed the GIN-Graph to generate high-quality model-level explanation graphs that are similar to real graphs, meanwhile maximizing the prediction probability for a certain class. Experimental results demonstrate that GIN-Graph presents superior performance to GNNI for generating valid explanation graphs with better stability and reliability, highlighting its generality in generating model-level explanation graphs for GNNs trained on a variety of graph datasets. In the future, given the considerable diversity within models learned on different datasets, we aim to tackle the challenge of creating a universal metric for accurately evaluating explanation graphs across all models.

References

[1] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 635–644.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[3] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[4] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 430–438.

[5] X. Wang and H.-W. Shen, "Gnninterpreter: A probabilistic generative model-level explanation for graph neural networks," *arXiv preprint arXiv:2209.07924*, 2022.

[6] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[7] A. Longa, S. Azzolin, G. Santin, G. Cencetti, P. Lio, B. Lepri, and A. Passerini, "Explaining the explainers in graph neural networks: a comparative study," *ACM Comput. Surv.*, Sep. 2024, just Accepted. [Online]. Available: https://doi.org/10.1145/3696444

[8] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[9] 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.

[10] S. Azzolin, A. Longa, P. Barbiero, P. Liò, and A. Passerini, "Global explainability of gnns via logic combination of learned concepts," *arXiv preprint arXiv:2210.07147*, 2022.

[11] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5370–5390, 2022.

[12] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International conference on machine learning*. PMLR, 2018, pp. 5708–5717.

[13] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-vae: A variational autoencoder for directed acyclic graphs," *Advances in neural information processing systems*, vol. 32, 2019.

[14] D. Flam-Shepherd, T. Wu, and A. Aspuru-Guzik, "Graph deconvolutional generation," *arXiv preprint arXiv:2002.07087*, 2020.

[15] X. Bresson and T. Laurent, "A two-step graph convolutional decoder for molecule generation," *arXiv preprint arXiv:1906.03412*, 2019.

[16] N. De Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973*, 2018.

[17] M. Popova, M. Shvets, J. Oliva, and O. Isayev, "Molecularrnn: Generating realistic molecular graphs with optimized properties," *arXiv preprint arXiv:1905.13372*, 2019.

[18] N. Anand and P. Huang, "Generative modeling for protein structures," *Advances in neural information processing systems*, vol. 31, 2018.

[19] 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," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.