One Graph is not Enough: Increasing Graph Neural Network Explainability through Multiple Subgraph Explanations

Dylan Kupsh*
dkupsh@ucla.edu
UCLA
Los Angeles, California, USA

Wenhan Yang*
hangeryang18@g.ucla.edu
UCLA
Los Angeles, California, USA

Baiting Zhu*
baitingzbt@g.ucla.edu
UCLA
Los Angeles, California, USA

Zongyang Yue*
zongyangyue@g.ucla.edu
UCLA
Los Angeles, California, USA

ABSTRACT

Machine learning explainability facilitates user trust, and is crucial for furthering the machine learning application space into more sensitive areas, like medical treatment. As graph neural networks are an emerging subfield in machine learning, recent literature on explainability, although expanding, is insufficient. Our project team aims to transfer recent machine learning explainability models from image recognition to graph neural networks, aiming to increase the explainability of machine learning through structured confidence subgraphs. In our project, we generate these structured confidence subgraphs from previous graph explainability papers and evaluate their intuitive effectiveness.

CCS CONCEPTS

• **Machine Learning → Graph Neural Networks**; *Explainability*.

KEYWORDS

graph neural networks, explainability, subgraphs

ACM Reference Format:
Dylan Kupsh, Wenhan Yang, Baiting Zhu, and Zongyang Yue. 2022. One Graph is not Enough: Increasing Graph Neural Network Explainability through Multiple Subgraph Explanations. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Graph is a powerful data structure that can include rich information in a variety of real-world applications. Its ability to encode complex relations in different data like social networks, biological compound structures, and citation networks help it gain popularity in different domains. However, it has always been a challenge to work with the graph data, which requires meaningful encoded representations of node features and edge relationships. Its atypical representation space in the non-Euclidean space also makes it difficult to transfer the success of neural networks directly to graphs. With the arise of Convolutional Neural Networks (CNNs) on images[5], Graph Neural Networks (GNNs)[4] are proposed in the graph domain. Different from their predecessors like DeepWalk[6] and node2vec[3] which can only learn shallow embedding from the data, they are able to recursively learn both the graph structures and the nodes' features via the information passed from every node's neighborhood, thus extracting a more complex meaningful representation from the data.

Despite its success, GNNs lack interpretability due to its deep structure. In a node classification task setting, a general sense of homogeneity is helpful in the explanation, but due to the recursive processes of message passing in the training, the local neighborhood alone is not sufficient to explain the prediction results. In a graph classification task setting, the interpretability of the result becomes even worse: It is difficult to decide which neighborhoods or graphlets are essential to the final prediction.

Increasing model transparency can increase human trust in the GNN models, thus help it to gain more popularity and acceptability in the real-world deployments. It is also beneficial to the development of fairness and privacy algorithms which rely mostly on the models' decision making process. Better understanding the models' predictions can also be inspiring and critical to our understanding of the data as well: For example, when analysing molecular graphs in a medical setting, learning how the models make the decisions may bring new insights for experts to the targeted atoms or atom bonds.

In recent years, GNNs explanability are drawing more and more attentions from the graph learning community and many methods have been proposed. One of the earliest attempts, GNNExplaner [8], utilize perturbations to generate explanations. It first initializes random soft masks for edges and node features in the graph. Then, via recursive training on the original graph and randomly generated subgraphs, the model learn the important components of the graph in the format of node and edge masks. Another line of work makes use of the gradient. The model assumes that the higher gradient values is equivalent to more important features, and thus interpret the prediction result via evaluating the gradient values during the training phase. An example of this is the SA

*All authors contributed equally to this research.

model[1]. A newer trend of GNNs explanability researches focus on model-level explanations instead of the instance-level, which seeks to provide high-level insights to the model itself. Specifically, XGNN[10] is proposed to explain the models via graph generations. Its end goal is not to modify the input data, but to train a graph generator which seeks to optimize the prediction result. All these works and their precursors have achieved tremendous success on generating meaningful explanations, but some problems persist: most of the existing methods seek to display the result in a single graph, with different transparency of the nodes or edges representing the weights assigned to them by the explanation models. This strategy may work in a node classification setting, when there are few components in the graph that contribute to the final prediction result. However, in the graph classification scenario, such a method becomes limited and clumsy. Especially when the targeted graph is large, the methods generally have a limited visualization capacity and can be difficult to interpret. In addition, the lack of logical structure also undermines the explanability of the visualization result. In this paper, inspired by the previous works in the image domain[9], we propose a novel method to visualize the explanation result. By constructing a logical graph, formally defined as Structure Attention Graph, our model present the prediction result in a more intuitive and interpretable way.

2 ORGANIZATION

The rest of this paper is organized as follows. In section three we provide our problem definition and formulation of explaining GNN. In section 4, we talk about the related work on CNN explanations and how are they transferrable to explaining GNN, and also a current GNN Explainer. In section 5, we describe the dataset, our definition of minimum sufficient explainer, our methods of pre-training GNN and dividing graphs into sub-graphs in details. In section 6, we show our experimental design and evaluation, and also visualization of our results. In section 7 and 8, we conclude our work and shed light on potential directions for future work.

3 PROBLEM DEFINITION AND FORMULATION

Denote a graph of dimension $m \times n$ as $G = (V, E)$, with V and E represent the node set, $\{v_1, v_2, \ldots, v_m\}$, and the edge set, $V \times V$, respectively. Denote its symmetric Adjancency matrix as A. $A_{ij} = 1$ if and only if $(v_i, v_j) \in E$, and equals to 0 otherwise. Denote the Feature matrix as X, with $x_i \in R^m$ represents the feautre vector of the i_{th} node. D is the Degree matrix of the graph, with $D_{ii} = \sum_j A_{ij}$. Denote the trained classification model as $M(X, A)$. It takes the graph's node features and adjancency matrix as input and the graph label as output. Denote the subgraphs of the original graph as $G_{sub} = (V_{sub}, E_{sub})$, with their subfeature matrix and subadjancency matrix as X_{sub} and A_{sub} respectively. Our goal is to find multiple subgraphs such that $M(X, A) \approx M(X_{sub}, A_{sub})$

4 RELATED WORK

Academic work in graph neural network explainability is rapidly expanding. This section provides a brief explanation of different graph neural network explainability methods attempted by previous researchers.

4.1 Explainability within Image Processing

In the field of Computer Vision, Zhou et al. proposed the model explanation method based on Class Activation Mapping (CAM) [2] in 2015. This method uses the constructs a heat-map using the activation in the last convolution layer. In specific, it replaces the final linear layers to convolution layers with global average pooling. The output would then become a heat-map in the dimension proportional to the input. The heat-map activation value for a class c at point (x, y) is given as

$$M_c(x, y) = \sum_k w_c^k f_k(x, y)$$

.

In 2019, Selvaraju et al. proposed the method of Gradient-weighted Class Activation Mapping (Grad-CAM) [7], which is built upon the normal CAM. Grad-CAM generalizes CAM and doesn't modify the original model architecture. Furthermore, Grad-CAM uses both the final convolution layer and the gradient value that flows in. Then, it calculates the neuron importance weights given by $\alpha_{x,y}^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{i,j}^k}$ for Grad-CAM, or $\alpha_{x,y}^c = \frac{1}{Z} \sum_i \sum_j -\frac{\partial y^c}{\partial A_{i,j}^k}$ fo Guided Grad-CAM and finally the heat-map activation value as

$$L_{Grad-CAM}^c = ReLU(\sum_{x,y} \alpha_{x,y}^c A^k)$$

Another approach proposed by Qi et al. in 2020 is IntegratedGradient Optimized Saliency (I-GOS) [11]. Different from CAM and Grad-CAM, I-GOS's objective is to maximally decrease the classification score after masking parts of the original image. Consider an input I, a mask M as the heat-map, and a random noise n_s, I-GOS computes the integrated gradient as

$$\nabla^{IG} f_c(M) = \frac{1}{S} \sum_{s=1}^{S} \frac{\partial f_c(\Phi(I + n_s, \frac{s}{S}(M))}{\partial M}$$

Then, I-GOS calculate the total gradient (TG) and learning-rate to perform update $M_{k+1} = M_k - \alpha_k \cdot TG(M_k)$. The details for total gradient and learning-rate adaption can be found in the original paper.

Finally, Shitole et al. proposed structured attention graphs (SAGs) in 2021 [9], which aims on finding a hierarchical explanation for each classification, represented by a tree-structure. This approach first masks and down-samples the input, searches for the lowest MSE through Beam Search, and constructs a tree-like hierarchical explanation. In comparison to previous approaches, SAGs directly searches for the explanation and doesn't require training, which makes the model very efficient. In addition, the down-sampling of images is parallel to a sub-graph method. As a result, we want to implement a similar approach for Graph data rather than images.

4.2 Graph Neural Networks

There are existing works focusing on explaining GNN models, such as GNN-Explainer by Ying et al. [8]. GNN-Explainer initializes random soft masks for edges and node features to learn the important components of the graph. Given a graph G, a trained GNN model Φ, and a label distribution Y, GNN-Explainer looks for the subgraph G_S and node features X_S such that the conditional entropy $H(Y|G_S, X_S)$ is minimized. This is similar to maximizing the mutual information (MI) given as

$$\max_{G_S,Y} MI(Y,(G_S,X_S)) = H(Y) - H(Y|G_S,X_S)$$

.

The advantages of GNN-Explainer are: it works nicely using graph data as an input and doesn't require a training of a deep-learning model. However, the results of GNN-Explainer doesn't look into sub-graphs in details and cannot be easily represented as a hierarchical structure.

5 METHODS

We accommodated the idea of minimum sufficient explanation and beam search from structured attention graphs and try to apply it on explaining Graph Neural Networks.

Instead of dividing the full image into 7x7 sub-images in the image/CNN setting, we divided the original full graphs into sub-graphs. We obtained sub-graphs by randomly choosing a node and getting all its neighbors, where we define neighbors in several fashions, e.g.directed connected, indirectly connected within two hops, etc. Then, we applied pre-trained GNNs on the sub-graphs. If the prediction of a sub-graph is the same with the full graph it divided from with a confidence score higher than a threshold, we recognized it as a minimal sufficient explanation for the full graph. In this way, for every full graph we can get a list of its minimal sufficient explanation sub-graphs which can collectively serve as the root node for the structured attention graph.

5.1 Pre-training GNN

We pre-trained GNN using only the full graphs as the training data and excluding the sub-graphs, because only the full graphs are correctly labelled and standardized.

We used a simple GCN network as the pre-trained GNN. It only contains three graph convolution layers, each followed by a ReLU, and after those three layers we append a readout layer (global mean pooling), a dropout layer, and a fully connected layer.

Formally, a graph convolution layer is defined as follows. Let $h_v^{(0)}$ denote the initial vector embeddings of the node features, i.e. $h_v^{(0)} = x_v$ for all $v \in V$ where v is the set of nodes in the graph, and x_v means the original features of the nodes. Let K denote the total number of layers in GCN, in our case K = 3. Then for k = 1, 2, ..., up to k,

$h_v^{(k)} = f^{(k)}(W^{(k)} \cdot \frac{\sum_{u \in N(v)} h_u^{(k-1)}}{|N(v)|} + B^{(k)} \cdot h_v^{(k-1)})$ for all $v \in V$, where $N(v)$ is the set of v's neighbors. Here we are essentially taking the average of v's neighbor's embeddings at the k-1 th step, matrix-multiplying it with a weight matrix, and adding to it a bias term, and then applying a function f at the end at each layer. In our case, the f in the formula used is a non-linear activation function ReLU.

The global mean pooling layer, or readout layer, is getting the graph embedding by averaging all node embeddings of nodes in the graph. Formally, suppose h_G is the graph embedding, then

$h_G = \sum_{n=1}^{N} h_n^{(k)}$, where N is the number of nodes in this graph. Here we average the output of the k-th layer (final layer) node embeddings to get the graph embedding.

5.2 Dataset

We tested our design on two published datasets in the standard PyG/torch geometric library so that it is easy to reproduce and does not require pre-processing the data.

The first dataset we used is MUTAG, a collection of nitroaromatic compounds represented by each graph. It contains 188 graphs classified into two classes. Nodes in the graphs are atoms labelled by atom types. Edges are chemical bounds between corresponding atoms.

The second dataset is ENZYMES, also about compounds. It collects information about protein tertiary structures obtained from the BRENDA enzyme database, consisting of 600 graphs classified into 6 classes. Nodes in graphs represent secondary structure elements, and an edge connects two nodes if they are neighbors along the amino acid sequence or one of three nearest neighbors in space.

5.3 Minimal Sufficient Explanations

We adopt the idea of Minimal Sufficient Explanation from the image setting and adapt it to the graph setting. Here we define a subgraph to be a minimal sufficient explanation if it is sufficient to lead to the same model prediction with the full graph with a high confidence.

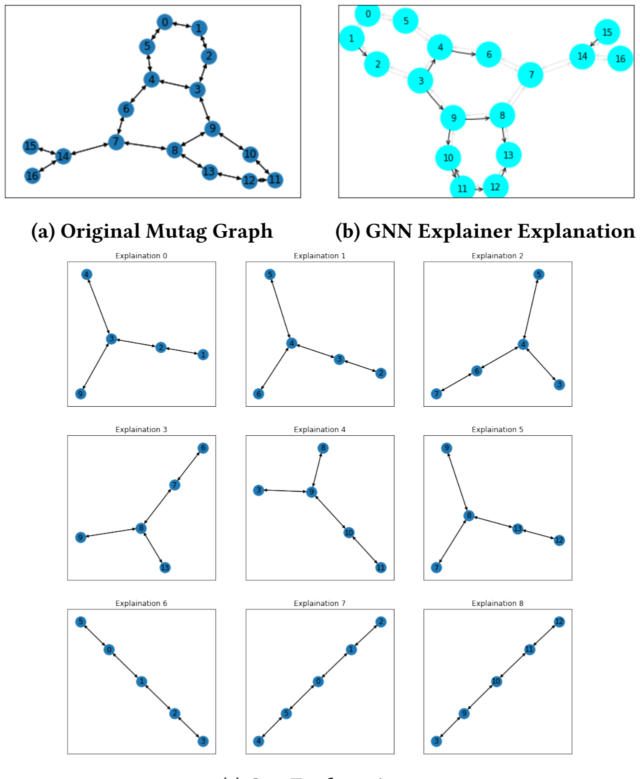
We define high confidence with a threshold. If the full graph goes through the pre-trained GNN and gets a prediction with class C with confidence x, and one of its subgraph goes through the same pre-trained GNN and gets a prediction with also class C with confidence greater than or equal to $x * thresholdvalue$, then we deem the subgraph as a minimal sufficient explanation of the full graph. Typically a threshold value of 0.8 or 0.9 is used.

We measure the confidence score in the above setting by the value of softmax result of the corresponding class. For example, if we have 2 classes and the output of the final softmax layer of the GNN is [0.2, 0.8], then the graph is predicted into the second class with a 0.8 confidence.

Formally, we assume a black-box classifier $f : X- > [0,1]^C$ where X is the set of graphs and C is the set of class labels. Let $x \in X$ be a graph and let $f_c(x)$ be the output of softmax layer after the final fully connected layer in our GCN architecture. For each $x \in X$, we denote $f * (x) = argmax_c f_c(x)$ to be the target of x. A minimal sufficient explanation of a full graph x is a subgraph x_i if $f_c(x_i) > P_h f_c(x)$, and $f * (x_i) = f * (x)$ where P_h is the threshold value.

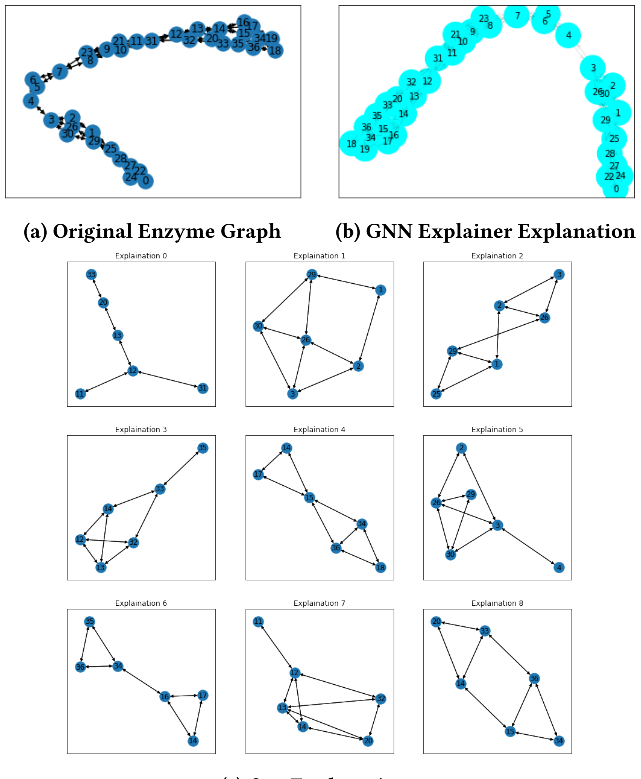
5.4 Method to Divide Graphs

We used a relatively simple methods to get subgraphs. Firstly, we choose a node, then we do a BFS search with a maximum depth on that node to get a subgraph which is expanding from that chosen node. We repeated this process for all nodes in one graph each time with several different maximum depths to get several subgraphs. And we repeated this process for all graphs in the dataset. We kept track of which subgraphs are originated from the same full graph so that it is easy to calculate if they are minimal sufficient explanations. At the end the number of subgraphs we get from the whole dataset is significant.


(a) Original Mutag Graph (b) GNN Explainer Explanation

(c) Our Explanations

Figure 1: The resulting explanations for a Mutag graph


(a) Original Enzyme Graph (b) GNN Explainer Explanation

(c) Our Explanations

Figure 2: The resulting explanations for an Enzyme graph

5.5 Structured Attention Graphs

From the methods we described in section 5.3, we can get a set of minimum sufficient explanations for each graph. That is not enough for human understanding.

We adapt the idea of structured attention graphs(SAG) from image settings to compactly represent a set of minimal suffcient explanation subgraphs for a full graph by visualizing how different combinations of subgraphs impact the confidence of a classifier. A SAG is a directed acyclic graph whose nodes correspond to a subgraph that is a minimal sufficient explanation and edges represent subset relationships between nodes by removing a single node from the subgraph.

Due to the limit of time, we only constructed minimal sufficient explanations(MSE) at this moment, and leave the construction of SAGs from these MSEs to future works.

6 EXPERIMENTAL DESIGN AND EVALUATION

To demonstrate our explanation methodology, we trained a basic graph convolutional network to both the Mutag and Enzyme dataset, and generated explanations, based upon the method described in section 5, for results on both datasets. For the results, we showcased the nine best subgraph explanations, ranked according to their confidence interval. To evaluate our method, we also generated explanations using GNN Explainer [8], the state-of-the-art method for generating explanations. The results are showcased in Figure 1 and Figure 2.

Our produced explanations are also shown within the GNN Explainer explanation set. However, compared to the GNN Explainer explanation, our explanations are much smaller with only a few nodes being selected. As GNN Explainer attempts to interpret the entire prediction within one subgraph, there are multiple disjoint resulting sets of important nodes on the same subgraph, hindering a unified explanation. This is demonstrated within figure 1b, with different sets of strongly connected nodes. In comparison, each of our explanation is a connected subgraph, as shown in Figure 1c and Figure 2c. Thus, each of our explanations represent some succinct graph structure, with users able to infer different properties by combining different explanations together.

Our explanations are also compact, containing only a few nodes instead of the entire graph. This smaller node size allows greater readability, at the expense of graph information. This compactness advantage is seen in Figure 2, with our explanations easier to visualize than the large GNN Explainer.

7 FUTURE WORK

In the future, we believe that extending this method to structured subgraph trees of decreasing size would add additional explainability. We believe this method could easily scale, generating explanations for thousand-node graphs, and would provide users with better understanding of the graph context. Additionally, this method would better translate the method proposed from the structured attention graph methods [9].

We also believe our method would benefit from a user study, with different user's commenting on the effectiveness of the proposed study. Previous work has used user studies to demonstrate their explanation's effectiveness compared to other work [9]. A user study could generate additional insight to improve explanation effectiveness.

Another improvement to this work would be improving the subgraph explanation method. Our current method is relatively basic, using a breadth first search to generate graphs. We believe a method like deep-walk, or something likewise, would generate better, and more diverse, subgraph explanations compared to our current method.

8 CONCLUSION

Our work demonstrates the effectiveness of multiple subgraphs in generating explanations of graph neural network predictions. Our methodology produces multiple compact subgraphs, each providing distinct intuition surrounding the prediction. We believe that future graph neural network explanation models should incorporate multiple possible explanations, providing users with additional context to formulate a full understanding of graph neural network prediction.

9 TASK DISTRIBUTION FORM

Task	People
Literature Review	All members
Related Work Exploration	Baiting
Implement classification models	Zongyang
Train GNN and Make Predictions	Zongyang
Implement proposed explanation models	Wenhan
Implement graph split methods	Baiting, Wenhan
Evaluating the results, Visualization	Dylan
Implement baseline explanation models	Wenhan
Writing report and slide	All members

REFERENCES

[1] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).
[2] Agata Lapedriza Aude Oliva Antonio Torralba Bolei Zhou, Aditya Khosla. 2016. Learning Deep Features for Discriminative Localization. *CVPR* (2016). https://doi.org/10.48550/arXiv.1512.04150
[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.
[4] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).
[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.
[7] Michael Cogswell Abhishek Das Ramakrishna Vedantam Devi Parikh Dhruv Batra Ramprasaath R, Selvaraju. 2019. Learning Deep Features for Discriminative Localization. *CVPR* (2019). https://doi.org/10.1007/s11263-019-01228-7
[8] Jiaxuan You Marinka Zitnik Jure Leskovec Rex Ying, Dylan Bourgeois. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. *NeurIPS* (2019).
[9] Minsuk Kahng Prasad Tadepalli Alan Fern Vivswan Shitole, Li Fuxin. 2021. One Explanation is Not Enough: Structured Attention Graphs for Image Classification. *NeurIPS* (2021). https://doi.org/10.48550/arXiv.2011.06733
[10] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 430–438.
[11] Li Fuxin Zhongang Qi, Saeed Khorram1. 2020. Visualizing Deep Networks by Optimizing with Integrated Gradients. *AAAI* (2020). https://doi.org/10.1609/aaai.v34i07.6863