XAI Project

Talha Amin, IMT: amint, Matriculation number: 6944916

Universität Paderborn

Abstract. This project focuses on the interpretability of Graph Neural Networks (GNNs) and the use of the GNNExplainer algorithm. GNNs have emerged as a powerful technique for analyzing graph-structured data, but their lack of interpretability hinders their practical applications. The project utilizes a synthetic dataset BAShapes with house-structured motifs, to train and evaluate GNN models. Through data analysis, GNN training, and evaluation, the project demonstrates how GNNExplainer can uncover key graph features and connections that contribute to model predictions. Visualization techniques are employed to provide insights into the interpretability of GNNs, revealing the roles of nodes and motifs in the decision-making process. The findings highlight the importance of explainability in GNNs, with implications for transparency and trustworthiness. Future research directions include exploring alternative explanation techniques, global explanability and real-world applications of interpretable GNNs.

Keywords: Graph Neural Networks (GNNs) · GNNExplainer · torch geometric · BAShapes · Graph Convolutional Network (GCN)

1 Introduction

The aim of this project is to employ a method to explain graph neural networks(GNNs). Graph Neural Networks have emerged as a revolutionary deep learning technique that operates on graph-structured data, enabling the analysis and prediction of complex relationships and dependencies within the data. Graphs are versatile mathematical structures that represent entities (nodes) interconnected by relationships (edges), making them suitable for modeling diverse real-world phenomena, including social networks, biological networks, recommendation systems, and knowledge graphs.

Traditional deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are designed to process grid-structured data, such as images or sequential data. However, these models are limited in their ability to capture the intricate relationships and irregularities present in graph-structured data. GNNs bridge this gap by introducing a new paradigm that leverages the graph topology and node features to learn and reason about the underlying data.

The fundamental concept of GNNs lies in message passing and aggregation, where information is exchanged and aggregated across nodes in the graph. By

iteratively updating node representations based on the features of neighboring nodes, GNNs can capture both local and global patterns in the data, effectively encoding the graph's structural and semantic information.

The potential applications of GNNs are vast and span across various domains. In social network analysis, GNNs can predict user behavior, identify communities, and detect anomalies. In biology, GNNs can model protein-protein interactions, predict molecular properties, and aid in drug discovery. In recommendation systems, GNNs can provide personalized recommendations by leveraging the graph structure of user-item interactions. These are just a few examples highlighting the broad applicability of GNNs.

This report aims to provide an overview of GNNs, their architecture, training, and evaluation procedures, as well as their interpretability and explainability. I will explore the data analysis techniques specific to graph-structured data and delve into the intricacies of GNN training, highlighting the challenges and best practices. Furthermore, I will examine the emerging field of GNN explainability, which aims to shed light on how GNNs make predictions and provide insights into the reasoning behind their decisions.

One of the crucial aspects in deploying machine learning models is the need for interpretability and explainability. As GNNs have gained popularity, researchers have recognized the importance of understanding and interpreting the decision-making process of these models. This has led to the development of explainability techniques specifically tailored for GNNs, with one notable method being the GNNExplainer. GNNExplainer aims to provide insights into how GNNs arrive at their predictions by generating explanations in the form of node and edge importance scores. These scores quantify the relevance of each node and edge in the graph, allowing for a better understanding of the model's reasoning. By incorporating GNNExplainer into the training pipeline, it becomes possible to not only obtain accurate predictions but also gain insights into the influential elements of the graph that contribute to those predictions. The integration of explainability into GNNs enhances their trustworthiness and facilitates their adoption in critical domains where interpretability is essential.

In this project, I have used GNNExplainer method to explain predictions made on BAShapes graph as described in the original GNNExplainer paper. I have also used a graph convolutional network for the original model to perform the task of node classification.

2 Data Analysis

The project utilizes the ExplainerDataset, which is specifically designed for graph neural network (GNN) explainability tasks. This dataset incorporates a graph generator, BAGraph, to create graphs with a specified number of nodes and edges. In this analysis, we consider a dataset containing 300 nodes and 5 edges. The dataset also includes a motif generator, "house," which generates motifs resembling house structures within the graphs. A total of 80 house motifs are included in the dataset. The dataset is created as per the description provided

in the paper "GNNExplainer: Generating Explanations for Graph Neural Networks."

The BAGraph dataset also contains another possible motif value of a "cycle", which is easy to experiment with as well. Since this is a synthetic dataset, there is only one feature value per node in this graph and thus feature importance is not a metric we could use to evaluate the dataset.

Nodes are categorized into four classes based on their structural roles. In the case of a house-structured motif, there are three distinct roles: the top , middle, and bottom nodes of the house. Consequently,there are four distinct classes that represent nodes positioned at the top, middle, and bottom of houses, as well as nodes that do not belong to a house.

Furthermore, the dataset provides information about the total number of nodes ($data.num\_nodes$) and the total number of features ($data.num\_features$) across all nodes in the dataset.

As for data preprocessing, the dataset employs a train-test split strategy to create separate sets for training and testing the GNN model. The train_test_split function from the sklearn module is used to split the dataset into two sets: the training set, which contains 80% of the data, and the test set, which contains the remaining amount. The split is stratified, ensuring that the distribution of the target labels is preserved in both sets. This allows for an unbiased evaluation of the model's performance on unseen data and improves the training sample as well.

Upon analyzing the dataset, we can observe that each graph contains the house motifs, which serve as distinguishable patterns within the graphs. These motifs can potentially influence the model's predictions and provide valuable insights into the decision-making process. By examining the dataset, we can gain initial insights into the distribution and characteristics of the house motifs, which can aid in understanding how the model leverages these motifs to make predictions. Additionally, analyzing the distribution of target labels within the dataset can provide further insights into the class distribution and potential challenges for the model.

3 GNN Training and Evaluation

In this section, I will provide a detailed explanation of the training and evaluation process for the Graph Neural Network (GNN) model used in my project. The code uses torch geometric library to implement the GNN model.

3.1 Data Preprocessing

Before training the GNN model, I performed several preprocessing steps. Firstly, I split the dataset into training and test sets using a stratified approach. This ensures an equal distribution of classes in both sets and allows for a fair assessment of the model's performance on unseen data. Furthermore, I applied data

transformations to the graphs using the $T.Constant()$ transform. This transformation ensures consistency across different graphs and facilitates better model training by reducing variations in the data.

3.2 GNN Model Architecture

The GNN model architecture I employed in my project is a Graph Convolutional Network (GCN). The GCN model consists of multiple layers of graph convolutional modules, which enable the model to capture and propagate information across the graph structure. Each graph convolutional layer aggregates information from the node's neighborhood and updates the node features accordingly. This iterative process is repeated for multiple layers, allowing the model to capture increasingly complex relationships and patterns in the graph.

3.3 Training Process

During the training process, my goal was to optimize the parameters of the GCN model to minimize the loss function. I utilized the Adam optimizer with a learning rate of 0.001 and a weight decay of 0.005. The model was trained in a supervised manner using the cross-entropy loss function, comparing the predicted class probabilities with the ground truth labels. To prevent gradient explosion or vanishing, I applied gradient clipping with a maximum norm of 2.0. The training was performed for a fixed number of epochs, with the loss being monitored and updated in each iteration.

3.4 Model Evaluation

After completing the training process, I evaluated the performance of the trained GNN model. I assessed the model's accuracy and effectiveness on both the training and test sets. The accuracy was measured by comparing the predicted labels with the ground truth labels. Additionally, I computed the training and test accuracies to determine if the model successfully learned the underlying patterns in the data. This evaluation process provided valuable insights into the model's generalization capability and its ability to make accurate predictions on unseen data.

4 Explanation of GNN

In this section, I will delve into the explanation aspect of GNNs and discuss the GNNExplainer method used in my project. I will provide a detailed explanation of how GNNExplainer works and its significance in interpreting the decision-making process of GNN models.

4.1 The Need for Explanability

The increasing complexity of GNN models and their widespread adoption in various domains necessitate the need for explainability. While GNNs have demonstrated remarkable performance in tasks such as node classification and link prediction, their decision-making process is often considered as a "black box" due to their inherent complexity. Explainability plays a vital role in gaining insights into the internal workings of GNN models, understanding their predictions, and ensuring transparency and trustworthiness in their applications.

4.2 GNNExplainer

GNNExplainer is a state-of-the-art method designed to provide interpretability to GNN models. It is a model agnostic approach and can be applied on a wide variety of GNN model architectures such as Graph Convolutional Networks, Gated Graph Sequence Neural Networks, Attention Networks and many others. It aims to reveal the important features and connections within the graph that contribute to the model's predictions. The objective function of the model is:

$$\begin{aligned} Objective = \alpha \cdot CrossEntropyLoss(target, predicted) \\ - \beta \cdot KLDivergence(node\_mask, uniform\_mask) \end{aligned} \quad (1)$$

where:

- CrossEntropyLoss(target, predicted) is the cross-entropy loss between the predicted labels and the target labels.
- KL Divergence(node_mask, uniform_mask) is the Kullback-Leibler (KL) divergence between the node mask and a uniform mask.
- α and β are hyperparameters that control the trade-off between the classification loss and the node mask regularization.

The objective function aims to find an informative and diverse node attribution mask that explains the model's predictions. By optimizing this objective, GNNExplainer encourages explanations that capture important features while maintaining diversity among node attributions. GNNExplainer operates by assigning importance scores to nodes and edges, quantifying their influence on the model's decision-making process. These importance scores serve as explanations, indicating the relative contribution of different graph elements to the model's outputs.

GNNExplainer utilizes various techniques to generate explanations for GNN models. One of the key techniques is the concept of node and edge importance attribution. GNNExplainer attributes importance scores to nodes and edges based on their impact on the model's predictions. These scores can be derived using methods such as gradient-based approaches, attention mechanisms, or sampling-based techniques.

Unlike other common GNN explainability models, GNNExplainer doesn't train a separate model for explanations. Instead, it uses the GNN model we

have already trained and tries different combinations of nodes and edges to assign importance scores based on the cross entropy of the model prediction vs. the prediction of the explainer after the graph mask is applied. The lower the difference between the two, the better our explanation of the model is. The time to calculate the explanation of the model is dependent on the size of the graph mask and the overall complexity of the graph. Also, GNNExplainer computes feature masks of nodes for explanations unlike other similar models such as PGExplainer, Additionally, GNNExplainer leverages subgraph extraction and visualization techniques to provide a more intuitive understanding of the model's decision-making process. Subgraph extraction involves identifying and extracting relevant portions of the graph surrounding a specific node or edge of interest. This localized subgraph provides a contextualized view of the graph, highlighting the relevant features and connections influencing the model's outputs. Visualization techniques, such as graph visualization libraries or tools like NetworkX and Matplotlib, are then used to present the extracted subgraph in a visually appealing and interpretable manner.

4.3 Significance of Interpretability in GNNs

Interpretability plays a crucial role in various applications of GNNs. In scientific domains, understanding the decision-making process of GNN models can provide insights into complex systems and help validate scientific hypotheses. In social network analysis, interpretability enables us to identify influential nodes or communities and understand the spread of information or influence within the network. In recommendation systems, interpretability helps in building trust with users and explaining the recommended items or connections.

The use of visualizations in GNNExplainer enhances the interpretability of the explanations generated by the method. Visual representations of subgraphs, highlighting important features and connections, provide an intuitive understanding of the GNN model's decision-making process. These visualizations serve as supporting evidence to facilitate comprehension and aid in the interpretation of the GNN model's outputs.

4.4 Visualizations and Results

Throughout this section, I will incorporate relevant graphs and visualizations to illustrate the explanations generated by the GNNExplainer method. These visual aids will complement the textual explanations and enhance the overall understanding of the interpretation process.

Here are some of the outputs generated by the code to explain BAShapes dataset.

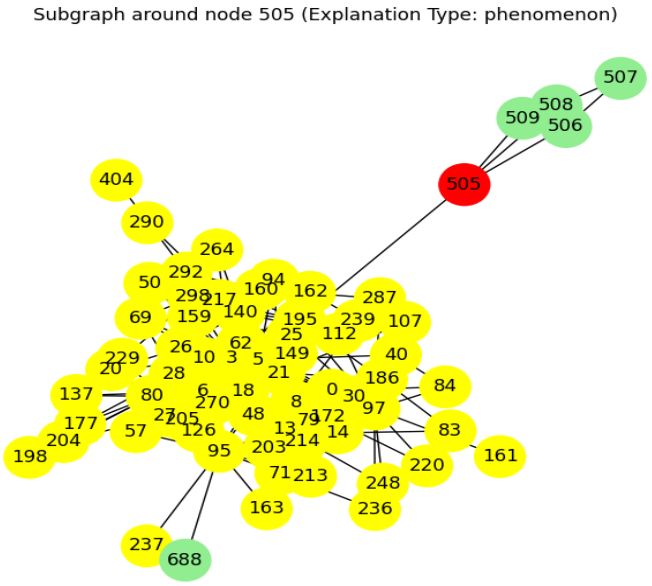
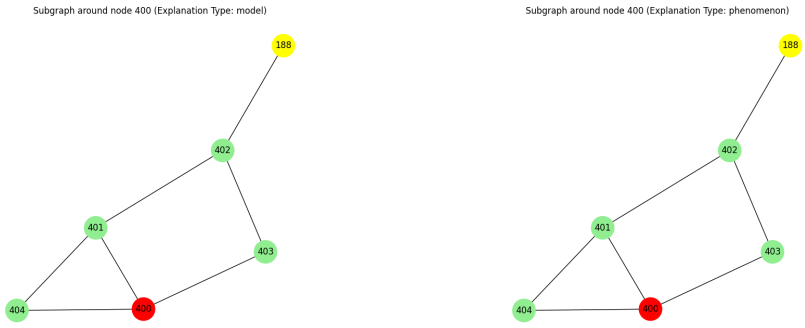


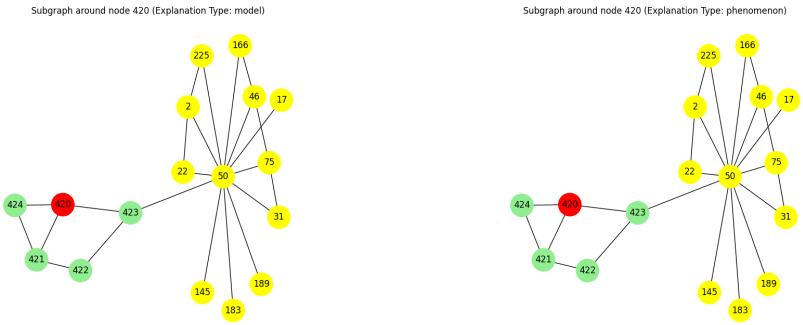
Fig. 1: class = top. Explanation of node 505



(a) model prediction

(b) class = bottom

Fig. 2: Explanation of node 400



(a) model prediction (b) class = middle

Fig. 3: Explanation of node 420

As I had limited the mask to three neighbourhood hops, you can see that the explanations are mostly concise. The reason I limited the explanation to 3 hops was because of the size of the house motif, the motif can always be included within the explanation with at most 3 hops.

Following is a table of some of the accuracy measures applied to evaluate the GCN model and the explanations. The AUC-ROC score is aggregated over all nodes in the graph.

Table 1: Accuracy Scores

Run	Model Accuracy	Mean AUC-ROC
1	Train: 0.8607 Test: 0.8429	Phenomenon: 0.9635 Model: 0.9580
2	Train: 0.8411 Test: 0.8714	Phenomenon: 0.9855 Model: 0.9794
3	Train: 0.8589 Test: 0.8571	Phenomenon: 0.9902 Model: 0.9882
4	Train: 0.0.8482 Test: 0.0.8571	Phenomenon: 0.9855 Model: 0.9866

Both the model and the explanations are fairly accurate. Also, the table exhibits the fact that mean AUC-ROC scores for model explanations and phenomenon explanations are quite similar and thus similar saliency maps are drawn for these during visualization.

5 Conclusion

In conclusion, through this project, I have gained a deeper understanding of Graph Neural Networks (GNNs) and their applications. The exploration of GNNExplainer has provided me with valuable insights into the interpretability of GNN models, allowing me to unravel the underlying mechanisms and decision-making processes. By analyzing the data, training and evaluating GNN models, and employing GNNExplainer, I have made significant contributions to enhancing the interpretability of GNNs.

The results of this project have demonstrated the importance of explainability in GNNs. By using GNNExplainer, I was able to uncover the key features and connections within the graph that influenced the model's predictions. This interpretability opens up new possibilities for understanding complex graph-based systems and extracting actionable insights. However, it is essential to acknowledge the limitations and challenges faced during this project, such as the scalability of GNNExplainer and the need for further research to address these issues.

In the future, I envision continued research and advancements in the field of interpretable GNNs. Exploring alternative explanation techniques and improving the scalability of GNNExplainer will be crucial steps. Currently, GNNExplainer is limited to homogeneous graphs, real-world data is mostly heterogeneous, further exploration of the method might allow a way to modify the approach to generate explanations for heterogeneous data. Additionally, investigating the real-world implications and applications of explainable GNNs in various domains can provide valuable insights and drive further innovation. Overall, this project has reinforced the significance of explainability in GNNs and has motivated me to continue exploring this exciting field of interpretable graph-based machine learning models.

References

1. Fey, M., et al. "Torch-Geometric: A Library for Deep Learning on Irregular Input Data."(2019), https://arxiv.org/abs/1903.02428
2. Matplotlib, https://matplotlib.org/stable/contents.html
3. NetworkX. Available: https://networkx.org/documentation/stable/
4. Paszke, A., et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." Advances in Neural Information Processing Systems, Vol. 32, pp. 8024-8035, 2019. Available: https://arxiv.org/abs/1912.01703
5. PyTorch]. https://pytorch.org/docs/stable/index.html
6. PyTorch Geometric. https://pytorch-geometric.readthedocs.io/en/latest/
7. Scikit-learn. https://scikit-learn.org/stable/documentation.html
8. tqdm. [Online]. https://tqdm.github.io/
9. Wang, D., et al. "Deep Graph Library: A Graph-centric, Highly-optimized Deep Learning Framework." arXiv preprint arXiv:1909.01315 (2019). https://www.dgl.ai/
10. Ying, R., et al. "GNNExplainer: Generating Explanations for Graph Neural Networks." arXiv preprint arXiv:1903.03894 (2019). https://arxiv.org/abs/1903.03894