A Demonstration of Interpretability Methods for Graph Neural Networks

Ehsan B. Mobaraki
ebmo@cs.aau.dk
Aalborg University
Aalborg, Denmark

Arijit Khan
arijitk@cs.aau.dk
Aalborg University
Aalborg, Denmark

ABSTRACT

Graph neural networks (GNNs) are widely used in many downstream applications, such as graphs and nodes classification, entity resolution, link prediction, and question answering. Several interpretability methods for GNNs have been proposed recently. However, since they have not been thoroughly compared with each other, their trade-offs and efficiency in the context of underlying GNNs and downstream applications are unclear. To support more research in this domain, we develop an end-to-end interactive tool, named gInterpreter, by re-implementing 15 recent GNN interpretability methods in a common environment on top of a number of state-of-the-art GNNs employed for different downstream tasks. This paper demonstrates gInterpreter with an interactive performance profiling of 15 recent GNN interpretability methods, aiming to explain the complex deep learning pipelines over graph-structured data.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; • **Information systems** → **Network data models**.

KEYWORDS

Graph neural network, interpretability, explainable AI

ACM Reference Format:
Ehsan B. Mobaraki and Arijit Khan. 2023. A Demonstration of Interpretability Methods for Graph Neural Networks. In *Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) (GRADES & NDA '23), June 18, 2023, Seattle, WA, USA*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3594778.3594880

1 INTRODUCTION

Modern deep learning models are data-hungry and considered as black-box. In black-box models, complex data move through various processes involved in machine learning (ML) to generate the final predictive output, creating a data pipeline [12]. While the data management community extensively worked on early and middle-stages of such data pipelines, e.g., data integration, cleaning, validation, enrichment, models management, query optimization, AutoML, etc., the late stages of the pipeline, such as explaining the results of black-box deep learning models in regards to downstream applications received relatively less attention by this community [3, 6, 16]. Explainability would improve the model's transparency related to fairness, privacy, and other safety challenges, thus enhancing the trust in decision-critical applications, democratizing deep learning techniques, and easing their adoption in the real-world [19].

In this work, we focus on interpretability of deep learning methods over graph-structured data. Graph neural networks (GNNs) [22] are useful in graphs and nodes classification, link prediction, entity resolution, question answering, fraud detection, etc. Recently, interpretability methods for graph neural networks are becoming popular. For a taxonomic survey of these approaches, we refer to [26]. However, since the interpretability methods have not been thoroughly compared with each other, it is unclear whether the later methods outperform the earlier ones. The evaluation frameworks, datasets, metrics, and the GNNs employed were often not consistent across these works, neither they were widely compared beyond graphs and nodes classification tasks. Lack of ground truth, benchmarks, and errors introduced while evaluating the performance of GNN interpretability methods are other concerns [4].

We address the aforementioned challenges by re-implementing 15 recent GNN interpretability methods in a common environment and code base, using several real-world graph datasets from diverse domains, identical evaluation metrics, and novel ground truth, via employing a number of state-of-the-art GNNs and downstream tasks. With our system, gInterpreter, one can visualize important findings, such as which interpretability method is more suitable under what metrics and scenarios. A user may plug-and-play with different graph datasets, GNNs, downstream tasks, interpretability tools, and can also interactively update the test graph to realize the importance of its salient features (e.g., important nodes, edges, node- and edge-attributes) in the context of downstream applications. Thus, we believe that gInterpreter would benefit researchers, domain experts, and data scientists. A video demonstration of our system is available at YouTube - **https://youtu.be/z4R_v8Lbmsw**.

Differences with prior benchmarking efforts. Earlier [1, 4, 13, 18] and recently [9, 26] empirically studied GNN-based interpretability methods. Ours is a comprehensive evaluation that includes more recent interpretability methods, state-of-the-art GNNs, and downstream tasks beyond graphs and nodes classification (e.g., link prediction and entity resolution). To the best of our knowledge, the demonstration of our system, gInterpreter, is the first demonstration proposal about interactive performance profiling of 15 recent GNN interpretability methods. We select them as they are state-of-the-art instance-specific GNN interpretability methods from 4 different categories [26]: gradient-based, perturbation-based, surrogate, and decomposition methods. Additionally, we compare more recent GNN interpretability methods not considered in [26], e.g., counterfactuals [20]. We re-implement 15 interpretability methods from different

GRADES & NDA '23, June 18, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0201-3/23/06.
https://doi.org/10.1145/3594778.3594880

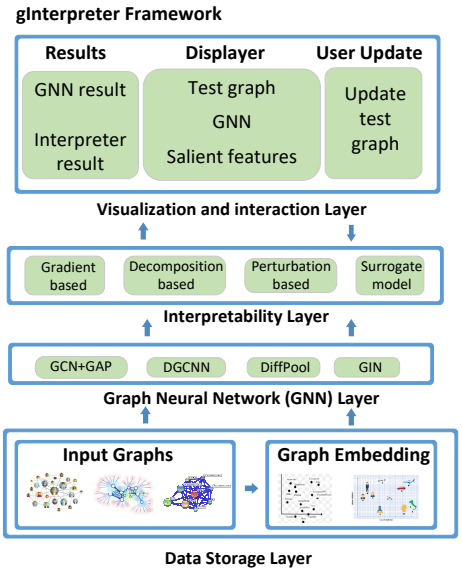

Figure 1: The architecture of the gInterpreter system

categories (e.g., model-specific vs. model-agnostic, forward vs. backward, perturbation-based vs. gradient-based vs. decomposition-based vs. surrogate models vs. counterfactuals), together with advanced GNNs employed for several downstream applications, and evaluate them via a diverse set of metrics, efficiency results, interactive approaches, and novel ground truth, compared to earlier works.

2 FRAMEWORK OVERVIEW

Figure 1 shows gInterpreter's architecture with four layers.

Data storage layer. The back-end stores training and test graphs and their embeddings. The graph structure is represented as an adjacency matrix. Node and edge features are stored as one-hot encodings. We store node embeddings for a graph in a dictionary: each node as a key and its embedding vector as the associated value.

GNN layer. We employ 7 recent GNNs for graph and node classification, link prediction, and entity resolution tasks: GCN+GAP, DGCNN, DiffPool, GIN, SEAL, LGLP, and GraphER (§3). We use torch_geometric, dive into graphs (DIG), and deep graph libraries (DGL) for GNN implementation. The pretrained model configuration and parameters are stored as .pt files.

Interpretability layer. We use 15 instance-specific GNN interpretability methods from 5 categories: gradient-, perturbation-, and decomposition -based, surrogate models, and counterfactuals (§3). We also implement them using DIG, DGL, and torch_geometric libraries.

Visualization and interaction layer. The output layer of gInterpreter reports the performance and efficiency results of the selected GNN and the interpretability method in a tabular format. Additionally, it provides an interactive displayer (built using NetworkX, dash, and pyvis libraries) to visualize the test graph, GNN model, important neurons, and salient graph features according to a selected interpretability method. The user may modify the test graph (while ensuring that the new graph is valid for that domain) and realize the importance of salient features related to a downstream task.

3 GNN AND INTERPRETABILITY METHODS

We briefly introduce recent graph neural networks and their interpretability methods that have been employed in our system.

Graph neural networks (GNNs). Recent GNNs are convolution-based, known as graph convolutional neural networks (GCNs), conducting recursive neighborhood aggregation. We consider 4 state-of-the-art GNNs designed for node and graph classification, 2 GNNs for link prediction, and 1 GNN for entity resolution.

GCN+GAP consists of graph convolutional layers following Kipf and Welling [7], creating node embeddings useful for node classification. For graph classification, node features from the last GCN layer are aggregated (known as a READOUT function), e.g., with a global average pooling (GAP) layer [13] to compute the entire graph's representation, followed by dense layers and a softmax classifier.

DGCNN [29] uses graph convolutional layers, followed by a *SortPooling* layer, classic convolutional and dense layers, and then a softmax classifier. The SortPooling layer sorts nodes based on node features from the last GCN layer and selects the top-k nodes.

DiffPool [25] learns a soft clustering of nodes at each DiffPool layer, which forms the coarsened input for the next DiffPool layer. Every DiffPool layer contains a collection of GCN layers for node embeddings generation and their probabilistic assignments to different clusters. Embedding vectors from the last DiffPool layer are aggregated by taking maximum across each embedding dimension. This is followed by dense layers and a softmax classifier.

GIN [23] employs injective neighborhood aggregation and READOUT functions to generate a more powerful GNN.

SEAL [28] extracts local enclosing subgraphs around links as input, and applies GNN learning to predict how likely the links exist.

LGLP [2] converts the original graph into a line graph, and conducts GNN-based node classification in the line graph, which solves the link prediction problem in the original graph.

GraphER [8] trains an Entity Record Graph Convolutional Network (ER-GCN), which embeds semantic and structural information into token embeddings, thus conducting token-centric entity resolution.

GNN interpretability methods. We study 9 recent *forward* and another 6 *backward* interpretability methods.

Forward interpretability methods are GNN model-agnostic, relying on learning evidence about graphs or nodes passed through the GNN model. They could be *perturbation*-based (e.g., GNNExplainer [24], PGExplainer [10], GraphMask [14], and SubgraphX [27]), that is, masking some node features and/or edge features and analyzing the resulting changes when the modified graphs are passed through the GNN model. They might also employ a simple, interpretable *surrogate model* to approximate the predictions of a complex GNN model. Examples include PGM-Explainer [21], RelEx [30], GraphLime [5], and DnX [11]. Additionally, we consider one counterfactuals-based GNN interpretability method, CF^2 [20].

Backward interpretability methods are model-specific and can be either *gradient-based* (i.e., backpropagating an importance signal from the output neuron of the model to the individual nodes of the input graph), or *decomposition-based* (i.e., distributing the prediction score in a backpropagation manner until the input layer). Thus, we observe which nodes, edges, and features contribute the most to the specific output label in the model. SA [1], GuidedBP [1], CAM [13], and Grad-CAM [17] are examples of gradient-based methods, whereas LRP [15] and ExcitationBP [13] are decomposition-based. Sensitivity Analysis (SA) uses the squared values of gradients as the importance of different input features.

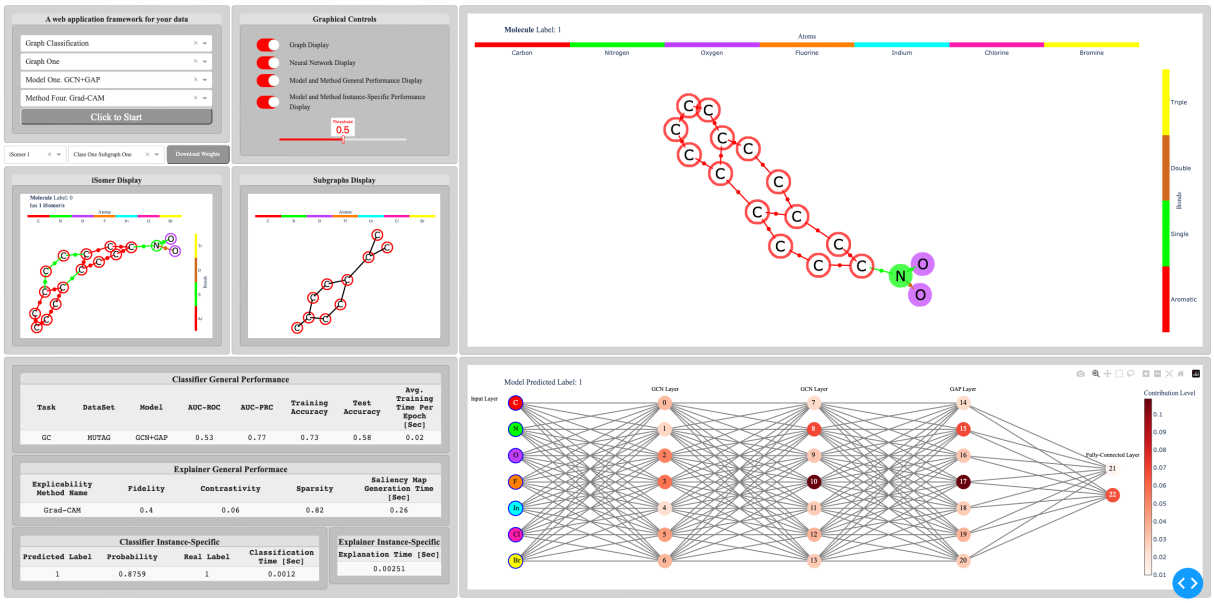

Figure 2: User interface of the glnterpreter **system: Graph classification with salient features**

GuidedBP follows a similar measure as SA, but it only backpropagates positive gradients, while zero-ing out negative gradients.

GRAD-CAM is class-discriminative, computing a coarse-grained feature importance map by associating the feature maps in the final convolutional layer with specific classes, based on the gradients of each class with respect to each feature map, and then using the weighted activations of the feature maps as an indication of which input features are important.

LRP, ExcitationBP decompose the output prediction score to different node importance scores following certain rules. Different from earlier gradient-based methods, they identify which input features contribute the most to the output, without focusing on its variation.

GNNExplainer learns soft masks for edges and node features to generate an evidence subgraph of the input graph and the masks are optimized to maximize the mutual information between the predictions of the input graph and that of the evidence subgraph.

PGExplainer follows the same paradigm as GNNExplainer which maximizes the mutual information between the predictions of the input graph and that of the evidence subgraph; however, it only generates edge masks by using a deep neural network to parameterize the generation process of the evidence subgraph. Due to its parameterized generation process, PGExplainer can explain multiple instances collectively and also works in an inductive setting.

GraphMask learns a parameterized classifier that, for every edge in every GNN layer, predicts if that edge can be dropped without sufficiently changing the prediction of the model.

SubgraphX uses Monte Carlo tree search to select the most important subgraph with Shapley value-based formulation.

PGM-Explainer uses an interpretable Bayesian network generated from node features perturbation.

RelEx explains in two steps – first by perturbation-based learning of a local differentiable approximation for the GNN model, and then learning an interpretable mask over the local approximation.

GraphLime fits a simpler, nonlinear surrogate model to the local dataset surrounding a node to explain node classification.

Distill n' Explain (DnX) learns a surrogate GNN via knowledge distillation, and then extracts explanations by solving a convex program.

CF^2 uses causal inference theory and generates both necessary and sufficient (counterfactual and factual) explanations.

Evaluation metrics. We consider several metrics including efficiency. *Fidelity* [26] computes the decrease in accuracy by masking important (or salient) input features having attribution values greater than a threshold. *Contrastivity* [13] computes the normalized difference of saliency maps across different classes, reporting how class-specific the explanations are. *Sparsity* [13] measures the size of the explanation set. Besides individual test instance-specific results, we demonstrate GNN model-specific aggregate measure by visualizing important frequent subgraphs induced by salient nodes from a set of test instances. When the ground-truth interpretability result is available (e.g., for synthetically generated graphs) [4], we report accuracy, precision, recall, and F1-measures of the employed interpretability methods. Finally, users can interactively update the input test graph to understand the importance of its salient features.

4 DEMONSTRATION

Demonstration and audience. We demonstrate with a web application. Our demonstration will fill the gap between black-box deep learning algorithms and explanation of their results, by interpreting technical complexities of ML pipelines. glnterpreter can be useful in selecting the best interpretability method for a certain task and dataset by comparing the results with ground truth (if available) or any domain-specific knowledge. All the interpretability methods covered by glnterpreter compute on-the-fly explanations, except SubgraphX [27] which is expensive.

User interface and interactiveness. Figure 2 presents the graphical user interface of our web application. When a user initiates this web application, it offers two input columns that have a set of interactive choices. In the left column, users can choose the task, the test graph/node(s) to select, the pre-trained GNN model to be applied for the task, and finally the interpretability method.

Table 1: Performance of GNN graph classification models on the *MUTAG* dataset

Model	AUC-ROC	AUC-PRC	Accuracy	Avg. Time/Epoch
GCN + GAP	0.80	0.81	0.71	0.05 sec
DGCNN	0.91	**0.82**	**0.85**	0.12 sec
DIFFPOOL	0.56	0.75	0.67	**0.04** sec
GIN	**0.92**	**0.82**	0.84	0.07 sec

In the second part of the input section, users can control the outputs to be returned, e.g., displaying the test object, the GNN model architecture, general as well as instance-specific statistics derived from the selected GNN and the interpretability method. For the input section, the application also offers a threshold slider to be set by the user on the intended value that determines the important components/attributes of the selected object accordingly.

gInterpreter permits users to consider isomers of the selected graph (which are structurally similar to the input graph with some differences in node and edge attributes, and have different class labels than the input graph), thus users can visualize the importance of salient features in the selected graph, compared to its isomers, in the context of a downstream task. Moreover, users can visualize active subgraphs induced by salient nodes from many test graphs w.r.t. the selected interpretability method.

Based on selections made by the user, the displayer provides three interactive 3D network visualization panels: (1) the selected graph, (2) its isomers (if available), and (3) active subgraphs that are frequently present based on the selected interpretability method. Over these interactive panels, users can hover the cursor and obtain annotations of each object, for instance, the number of connections for a node, the type of connections between two objects, and the intrinsic characteristics of objects.

The displayer also depicts the architecture of the selected GNN model. The network is represented in a 2D interactive panel. It allows users to hover on neurons, layers, and other components of a GNN. By hovering the cursor, users can observe the contribution of all neurons to any prediction, scaled and non-scaled contributions of the neuron, and the type of layer that the neuron belongs to.

The other output is to provide statistical information about the selections made by a user. The results can be divided into two categories: general and instance-specific. The general statistics demonstrate the mean performance of the selected model and the interpretability method over many test instances. The instance-specific part corresponds to displaying performance analysis of the selected model and the interpretability method on a selected object.

The application offers downloading of associated weight sets and configurational details of the pretrained GNN models.

Demonstration with the *MUTAG* dataset. The *MUTAG* (https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets) is an open-source dataset of mutagenic aromatic and heteroaromatic nitro compounds classified according to their mutagenic effects on a bacterium . For the graph classification task, we shall demonstrate gInterpreter with this dataset, highlighting which graph components have potential impacts on classification results. While in the following we discuss our demonstration scenario and result snippets for this dataset and graph classification task, during demonstration we shall also showcase other downstream tasks (e.g., link prediction, entity resolution).

Table 2: Performance of selected interpretability methods on the *MUTAG* dataset and graph classification task

GCN+GAP Classifier

Methods	Fidelity	Contrastivity	Sparsity
GNNExplainer	**0.481**	**0.789**	0.847
ExcitationBP	0.029	0.526	0.703
LRP	0.004	0.473	0.192
SA	0.148	0.052	0.687
GuidedBP	0.139	0.210	**0.879**

DGCNN Classifier

Methods	Fidelity	Contrastivity	Sparsity
GNNExplainer	**0.361**	**0.842**	**0.831**
ExcitationBP	0.283	0.170	0.801
LRP	0.253	0.478	0.639
SA	0.164	0.095	0.472
GuidedBP	0.151	0.193	0.779

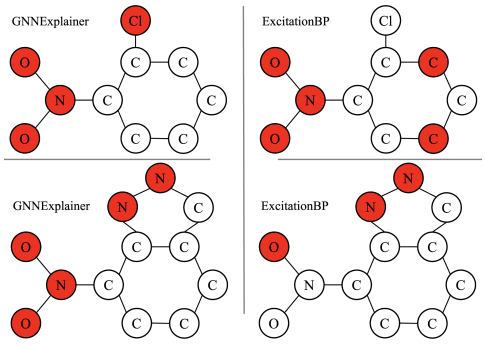

Figure 3: GNNExplainer and ExcitationBP interpretability results on two compounds from two different classes in the *MUTAG* dataset

Table 1 shows the performance of 4 GNN classification models. We notice that DGCNN and GIN outperform others in terms of AUC-ROC, AUC-PRC, and accuracy metrics, which is due to the benefits of SortPooling layer in DGCNN and injective nature of GIN. The average training time required per epoch is also modest for GIN.

Table 2 indicates that the performance of the same interpretability method is enhanced when combined with a better GNN classification model (e.g., DGCNN over GCN+GAP). Among the selected methods in Table 2, GNNExplainer achieves better interpretability results (i.e., higher fidelity, contrastivity, and sparsity) compared to other backward interpretability methods.

Figure 3 presents interpretability results of GNNExplainer and ExcitationBP on two compounds from two different classes in the *MUTAG* dataset. The dark (red color-filled) nodes are important nodes as identified by an interpretability method, where we set the importance threshold 0.5 on a scale of [0, 1]. We find that GNNExplainer is more discriminative compared to ExcitationBP, since the former can well-identify the differences between two compounds selected from two different classes.

We believe that our benchmarking and visualization results obtained from gInterpreter will be useful to both expert and non-expert users in the domain of interpretability for GNNs.

ACKNOWLEDGMENTS

The work presented in this article is supported by the Novo Nordisk Foundation grant NNF22OC0072415.