Explainability in GNNs: A Step Towards Global Self-Explainable GNNs

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Wendy Zheng

Spring 2024

Technical Project Team Members

Yinhan He

Carter Bassler

Alexander Shen

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Jundong Li, Department of Computer Science

Explainability in GNNs: A Step Towards Global Self-Explainable GNNs

Wendy Zheng
Computer Science
University of Virginia
Charlottesville, Virginia, USA
ncd9cf@virginia.edu

ABSTRACT

Graph Neural Networks (GNNs) offer extensive capabilities regarding graph-based tasks. Despite existing efforts, the explainability of predictions made by GNNs is still underexplored. It is uncertain how GNNs obtain specific predictions for certain inputs, which can raise ethical concerns when using GNNs in critical real-world situations, such as healthcare and financial analysis. One approach to this problem is graph counterfactual explanations (GCE), i.e., finding the minimal modification to the input graph so that the GNN changes its prediction to an alternative class. Current GCE methods are mainly post hoc, meaning they rely on an external explainer model to interpret GNN output. However, since the explainer and the GNN are trained separately, this poses the question of whether the explainer is correctly learning because it lacks access to the learning trajectory of the GNN. Furthermore, global counterfactual explanations, which help explain many input graphs, give deeper insight into the model's average behavior, making them more useful for interoperability than local, i.e., individual, explanations. In this work, we propose **G**lobal **C**ounterfactual-based **S**elf-explainable **GNN** (GCSGNN). GCSGNN is self-explainable; the model is simultaneously learning to predict graph labels and explain the predictions. The framework learns the latent patterns of the data and the modifications to those patterns that cause the GNN to change its predictions. As GCSGNN works in the latent space, it includes a decoder to map the latent space back to the input space to make the explanations more human-interpretable. This approach allows the explanations to reveal the essence of the GNN prediction scheme more effectively. Moreover, using latent patterns allows for global explanations, which enhances the human interpretability of the acquired graph explanations.

1 INTRODUCTION

Graph Neural Networks (GNNs) have many real-world applications, such as in finance [10], medical diagnosis [1], and traffic forecasts [6], achieving substantial performance in handling graph-structured data. However, GNNs lack explainability, as they are often considered black-box models, meaning it is difficult to understand how the model obtained its output for a specific input graph. Thus, there is a significant concern when using GNNs in critical fields where a small error can result in dangerous consequences.

Current works provide various approaches to explain GNN predictions [5, 7–9, 11], with one of them being graph counterfactual explanations (GCE). GCE aims to find the minimal modification to an input graph predicted as undesired such that the model would change its prediction to the desired class given the input graph with modifications (i.e., the counterfactual graph). The modification here can be a combination of adding or removing edges and nodes or changing the attributes of nodes and edges. The undesired and desired classes are predefined depending on real-world rules. For example, if given the chemical composition of a harmful drug, the counterfactual would be the minimal changes to make the drug harmless, the desired class.

Although existing GCE methods provide sufficient explanations for model outputs, they have two main limitations: (1) **External Explainer Model**. Most GCE approaches use an external explainer model to generate counterfactuals and treat the GNN as an oracle [7, 8, 12]. The explanations and the predictions are learned separately, raising the question of whether the explainer model is correctly learning because it lacks access to the learning trajectory of the GNN. Therefore, the explainer model can misinterpret the true explanations and generate poor-quality counterfactuals. (2) **Lack of Global Explainability**. Global explanations are where each explanation can apply to many input graphs, providing insight into the patterns in the GNN predictions. Local explanations offer counterfactuals for each graph, but using a local explanation to explain a different graph might not make sense.

As a solution, we propose GCSGNN (**G**lobal **C**ounterfactual-based **S**elf-explainable **GNNs**). GCSGNN counteracts the first limitation as it is self-explainable, which means it is simultaneously learning to predict and explain. The built-in interpretability removes the need for an external explainer, and the explanations and predictions are tightly coupled, i.e., the learning trajectory of the predictor influences the explainer module and vice versa). Thus, the model can better learn the significant information in the graphs, providing better quality counterfactuals. The proposed framework also overcomes the second limitation by learning the significant sub-patterns in the latent embeddings; it aims to recognize the class-specific features that cause the model to classify it as a particular class and replace it with counterfactual sub-patterns to change the model prediction. Thus, GCSGNN gives a global approach to generating counterfactuals by learning the patterns in GNN predictions.

Our main contribution is that we propose the novel model GCSGNN, which provides global self-explanable graph counterfactual explanations. It allows for built-in interpretability of the model, addressing the limitations of post-hoc GCE methods.

2 RELATED WORKS

2.1 Counterfactual-based Self-Explainable Networks for Tabular Data

Current works typically combine the neural network and the counterfactual explainer model into one pipeline and simultaneously train the two. For example, CounterNet [3] comprises three simple

multilayer perceptrons (MLPs): an encoder, a predictor, and a generator. The encoder maps the input data to the latent space to get the latent embeddings, which are then passed to the predictor to obtain the prediction. Lastly, the generator uses latent embeddings and the prediction to generate counterfactuals. VCNet [4] extends CounterNet by replacing the counterfactual generator MLP with a conditional variational autoencoder (cVAE). In these works, the explainer component is decoupled from the neural network in that its outputs do not influence the rest of the model. On the other hand, GCSGNN unifies the counterfactual generation process with the encoder and decoder, meaning that the explanations can affect the learning process of these components.

2.2 Global Prototype-based Self-Explainable GNNs

The field of global self-explainable GNNs is mainly understudied. Some recent works use prototypes as a global explanation method. Prototypes are exemplar graphs that are representative of all the data. ProtGNN [13] learns the latent representation of prototypes by minimizing the similarity between the embeddings of the input graphs and the prototype embeddings. The authors design the loss objective to ensure that the prototype embeddings are diverse and representative of the data. Lastly, the authors use the Monte Carlo tree search algorithm to project the prototype embeddings onto the input graphs to get the graph representations of the prototypes. ProtoVAE [2] builds on top of ProtGNN by incorporating an autoencoder. By doing so, the framework uses the decoder to obtain the graph representation of the prototype embeddings and eliminate the prototype projection step. Like these methods, GCSGNN uses learnable parameters to learn the significant inherent sub-patterns in the latent embeddings and counterfactual sub-patterns to generate counterfactual latent embeddings.

3 METHODOLOGY

3.1 Model Overview

Fig. 1 gives an overview of GCSGNN. The framework has four components: a graph encoder f_e, a graph decoder f_d, a graph predictor f_p, and the Global Counterfactual Generation module f_c. We define the ith input graph with n nodes as $G_i = (X_i, A_i, E_i)$ where $X_i \in \mathbb{R}^{n \times d_X}$, $A_i \in \mathbb{R}^{n \times n}$, and $E_i \in \mathbb{R}^{n \times n \times d_E}$. Here, d_X is the dimension of the node attributes, and d_E is the dimension of the edge attributes. G_i has the label $y_i \in \{0, 1\}$.

First, f_e maps the input graph G_i to a graph embedding h_i of length d_e. Then, f_c applies the learnable masks M and counterfactual sub-patterns S to generate the counterfactual latent embeddings $\hat{h}_i$. $M \in \mathbb{R}^{d_m \times d_e \times d_s}$ where d_m is a hyperparameter equal to the number of classes c times the number of masks / sub-patterns per class s and d_s is the sub-pattern dimension. It learns the significant inherent sub-patterns of each class in the graph embeddings. $S \in \mathbb{R}^{d_m \times d_s}$ learns the counterfactual sub-patterns to replace the inherent ones. Each sub-pattern in S has a one-to-one correspondence to a mask in M. Lastly, the graph decoder reconstructs the original input graphs G' and the counterfactual graphs $\hat{G}$, and the graph predictor classifies each embedding.

3.2 Global Counterfactual Generation

The Global Counterfactual Generation module consists of two components: Significant Inherent Sub-pattern Filtering (SISF) and Counterfactual Sub-pattern Substitution (CSS). For a graph embedding h_i, SISF masks the inherent sub-patterns in h_i with M to get the filtered embeddings $h'_i \in \mathbb{R}^{d_m \times d_e}$:

$$h'_{i,j} = h_i \times (I_{d_e} - M_j \times M_j^T), \forall j \in \{1, ..., d_m\} \tag{1}$$

Then, CSS uses S to replace the masked parts with the counterfactual sub-patterns to generate the counterfactual embeddings $\hat{h}_i \in \mathbb{R}^{d_m \times d_e}$:

$$\hat{h}_{i,j} = he_{i,j} + (S_j \times M_j^T), \forall j \in \{1, ..., d_m\} \tag{2}$$

3.3 Graph Decoder

The graph decoder in GCSGNN takes either the graph embedding or counterfactual embedding h_i and obtains its graph representation $G'_i = (X'_i, A'_i, E'_i)$. The implementation uses three MLPs: a shared decoder, a node attribute decoder, and an edge attribute decoder. The input embedding h_i is first passed to the shared decoder and then to the node attributes decoder and edge attributes decoder, respectively, to get the corresponding attributes as logits, which are continuous. As G'_i should have discrete values, we create one-hot vectors for the node and edge attributes such that the largest entry is assigned 1 and the other entries are assigned 0 to obtain X'_i and E'_i. Furthermore, because the edge attributes matrix and the adjacency matrix should correlate with each other, the adjacency matrix is constructed with the following:

$$A'_{ijk} = 1 - \sigma(E'_{i,jk,l=0}), \forall i \in \{1, ..., B\}\ \forall j, k \in \{1, ..., n\} \tag{3}$$

where B is the batch size. We use sigmoid to obtain the probability of each edge attribute, and the adjacency matrix equals 1 minus the probability that the edge attribute is 0.

3.4 Model Objective Function

The loss objective of the model consists of 6 components: graph prediction loss (L_g), counterfactual prediction loss (L_c), mask loss (L_m), intra-pattern distance (L_i), reconstruction loss (L_r), and graph counterfactual distance (L_d).

$$L = L_g + L_c + L_m + L_i + L_r + L_d \tag{4}$$

L_g aims to optimize the probability that the prediction from the graph predictor f_p matches the input graph label y_i. Thus,

$$L_g = \frac{1}{B} \sum_{i=1}^{B} -\log P(f_p(h_i) = y_i) \tag{5}$$

On the other hand, L_c forces the counterfactual embeddings to learn the class label $\hat{Y}_j \in \{0, 1\}$ assigned to each mask / sub-pattern:

$$L_c = \frac{1}{sB} \sum_{i=1}^{B} \sum_{j=1}^{d_m} -\log P(f_p(h'_{i,j}) = \hat{Y}_j) \tag{6}$$

L_m and L_i are used to influence the learnable parameters M and S. Because M is a selection matrix that selects the values to mask

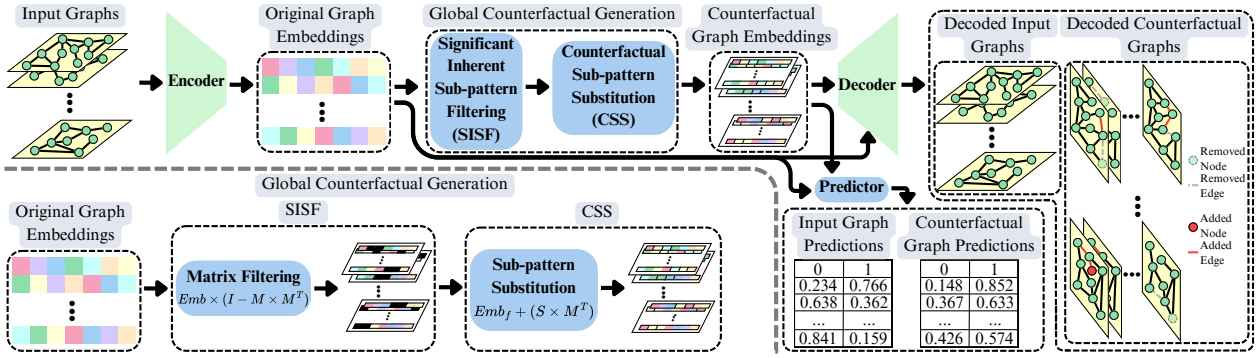

Figure 1: A overview of the proposed GCSGNN model.

in the embeddings, each mask can only have a d_s number of ones; otherwise, the masks would modify unrelated parts. Therefore,

$$L_m = \frac{1}{sd_s}\left(\sum_{j=1}^{d_m}\sum_{l=1}^{d_s}\left(\left|\max_k M_{j,kl} - \mathbf{1}\right| + \left|\left(\sum_{k=1}^{d_e}|M_{j,kl}|\right) - \mathbf{1}\right|\right)\right) \tag{7}$$

L_i is to encourage diversity between the masks / sub-patterns of the same class by minimizing the dot product between different mask and sub-pattern pair. To do so, we first expand the sub-patterns by multiplying it with the masks: $sm = S \times M^T$. This ensures that the positioning of the sub-patterns are incorporated. Then, L_i is calculated with the following:

$$L_i = \frac{1}{s^2}\sum_{a=0}^{1}\left|sm_{\hat{Y}=a} \times sm^T_{\hat{Y}=a} - I_s(d_{\hat{Y}=a})\right| \tag{8}$$

where $d = \sum_{k=1}^{d_e} S^2_{j,kl}, \forall j \in \{1, ..., d_m\}\ \forall l \in \{1, ..., d_s\}$. Forcing the diagonal to equal the squared sum of each pair ensures that values in M are binary.

The last two components influence the decoder. L_r is the entropy loss between X_i, A_i, E_i and X'_i, A'_i, E'_i. It uses weights (i.e., w_A, w_X, w_E) as the graph data are sparse, meaning it consists mostly of zeros.

$$R_{A_{i,jk}} = A_{i,jk}\log(A'_{i,jk}) + (1 - A_{i,jk})\log(1 - A'_{i,jk})$$

$$L_{r,A} = \frac{1}{Bn^2}\sum_{i=1}^{B}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(-w_{A,jk} \cdot R_{A_{i,jk}}\right)$$

$$R_{X_{i,j}} = \log\left(\frac{\exp X'_{i,j,k=X_{i,j}}}{\sum_{k=0}^{d_X-1}\exp X'_{i,j,k}}\right)$$

$$L_{r,X} = \frac{1}{B \cdot \sum_{k=0}^{d_X-1} w_{X,k}}\sum_{i=1}^{B}\sum_{j=1}^{n}\left(-w_{X,X_{i,j}} \cdot R_{X_{i,j}}\right) \tag{9}$$

$$R_{E_{i,jk}} = \log\left(\frac{\exp E'_{i,jk,l=E_{i,jk}}}{\sum_{l=0}^{d_E-1}\exp E'_{i,jk,l}}\right)$$

$$L_{r,E} = \frac{1}{B \cdot \sum_{l=0}^{d_E-1} w_{E,l}}\sum_{i=1}^{B}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(-w_{E,E_{i,jk}} \cdot R_{E_{i,jk}}\right)$$

$$L_r = L_{r,A} + L_{r,X} + L_{r,E}$$

L_d aims to force the decoded counterfactual graphs $\hat{G}_i = (\hat{X}_i, \hat{A}_i, \hat{E}_i)$ to be more similar to $G' = (X', A', E')$.

$$D_{\hat{X}_{ij},a} = \frac{1}{d_X}\sum_{k=1}^{n}\left(\sum_{l=0}^{d_X-1}(\hat{X}_{\hat{Y}=a,\hat{y}=a,ijkl} - X'_{\hat{y}=a,ikl})^2\right)^{1/2}$$

$$D_{\hat{A}_{ij},a} = \frac{1}{n}\left(\sum_{k=1}^{n}\sum_{l=1}^{n}\left(\hat{A}_{\hat{Y}=a,\hat{y}=a,ijkl} - A'_{\hat{y}=a,ikl}\right)^2\right)^{1/2}$$

$$D_{\hat{E}_{ij},a} = \frac{1}{ndE}\sum_{k=1}^{n}\sum_{l=1}^{n}\left(\sum_{m=0}^{d_E-1}\left(\hat{E}_{\hat{Y}=a,\hat{y}=a,ijklm} - E'_{\hat{y}=a,iklm}\right)^2\right)^{1/2}$$

$$L_d = \frac{1}{Bns}\sum_{a=0}^{1}\sum_{i=1}^{B}\sum_{j=1}^{s} D_{\hat{X}_{ij},a} + D_{\hat{A}_{ij},a} + D_{\hat{E}_{ij},a} \tag{10}$$

where $\hat{y}$ is the list of the counterfactual graph labels, i.e., the opposite of the original labels y.

4 CONCLUSION

This paper explores a new approach to graph counterfactual explanations: GCSGNN. It uses learnable parameters and the loss objective to learn to simultaneously generate counterfactuals and make predictions to couple the two processes tightly. However, some areas require future work. One issue is that the current framework may be unstable as it is likely dependent on the initial random state of the parameters. A potential solution is to investigate the theoretical foundation behind generating counterfactuals to obtain the best values. Another possible direction is to unify further counterfactual generation and graph prediction by using the explanations to determine the graph predictions. We hope our contributions will pioneer a new avenue for global self-explainable counterfactual-based models.

5 ACKNOWLEDGEMENTS

This endeavor would not have been possible without Yinhan's contributions to developing the framework and his guidance. In addition, I am grateful to Professor Jundong Li for his mentorship. Lastly, I would like to thank Carter Bassler and Alexander Shen for all their efforts throughout the process.

REFERENCES

[1] David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. 2021. Graph-Based Deep Learning for Medical Diagnosis and Analysis: Past, Present and Future. *Sensors* 21, 14 (July 2021), 4758. https://doi.org/10.3390/s21144758

[2] Srishti Gautam, Ahcene Boubekki, Stine Hansen, Suaiba Amina Salahuddin, Robert Jenssen, Marina MC Höhne, and Michael Kampffmeyer. 2022. ProtoVAE: A Trustworthy Self-Explainable Prototypical Variational Model. arXiv:2210.08151 [cs.LG]

[3] Hangzhi Guo, Thanh Hong Nguyen, and Amulya Yadav. 2023. CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations. arXiv:2109.07557 [cs.LG]

[4] Victor Guyomard, Françoise Fessant, Thomas Guyet, Tassadit Bouadi, and Alexandre Termier. 2022. VCNet: A self-explaining model for realistic counterfactual generation. arXiv:2212.10847 [cs.AI]

[5] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. arXiv:2001.06216 [cs.LG]

[6] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* 207 (Nov. 2022), 117921. https://doi.org/10.1016/j.eswa.2022.117921

[7] Ana Lucic, Maartje ter Hoeve, Gabriele Tolomei, Maarten de Rijke, and Fabrizio Silvestri. 2022. CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. arXiv:2102.03322 [cs.LG]

[8] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2022. Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking. arXiv:2010.00577 [cs.CL]

[9] Yong-Min Shin, Sun-Woo Kim, and Won-Yong Shin. 2024. PAGE: Prototype-Based Model-Level Explanations for Graph Neural Networks. arXiv:2210.17159 [cs.LG]

[10] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. 2022. A Review on Graph Neural Network Methods in Financial Applications. arXiv:2111.15367 [q-fin.ST]

[11] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. arXiv:1903.03894 [cs.LG]

[12] Jiaxing Zhang, Zhuomin Chen, Hao Mei, Dongsheng Luo, and Hua Wei. 2023. RegExplainer: Generating Explanations for Graph Neural Networks in Regression Task. arXiv:2307.07840 [cs.LG]

[13] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. 2021. ProtGNN: Towards Self-Explaining Graph Neural Networks. arXiv:2112.00911 [cs.LG]