CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks

Ana Lucic[1] **Maartje ter Hoeve**[1] **Gabriele Tolomei**[2] **Maarten de Rijke**[1] **Fabrizio Silvestri**[2]
[1]University of Amsterdam [2]Sapienza University of Rome

Abstract

Existing methods for interpreting predictions from Graph Neural Networks (GNNs) have primarily focused on generating subgraphs that are especially relevant for a particular prediction. However, such methods do not provide a clear opportunity for recourse: given a prediction, we want to understand how the prediction can be changed in order to achieve a more desirable outcome. In this work, we propose a method for generating counterfactual (CF) explanations for GNNs: the minimal perturbation to the input (graph) data such that the prediction changes. Using only edge deletions, we find that our method, CF-GNNExplainer, can generate CF explanations for the majority of instances across three widely used datasets for GNN explanations, while removing less than 3 edges on average, with at least 94% accuracy. This indicates that CF-GNNExplainer primarily removes edges that are crucial for the original predictions, resulting in minimal CF explanations.

1 Introduction

Advances in machine learning (ML) have led to breakthroughs in several areas of science and engineering, ranging from computer vision, to natural language processing, to conversational assistants. Parallel to the increased performance of ML systems, there is an increasing call for the "understandability" of ML models [10]. Understanding *why* an ML model returns a certain output in response to a given input is important for a variety of reasons such as model debugging, aiding decison-making, or fulfilling legal requirements [9]. Having certified methods for interpreting ML predictions will help enable their use across a variety of applications [25].

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

Explainable AI (XAI) refers to the set of techniques "*focused on exposing complex AI models to humans in a systematic and interpretable manner*" [30]. A large body of work on XAI has emerged in recent years [3, 12]. Counterfactual (CF) explanations are used to explain predictions of individual instances in the form: "If X had been different, Y would not have occurred" [15, 32, 34]. CF explanations are based on CF examples: modified versions of the input sample that result in an alternative output (i.e., prediction). If the modifications recommended are also *actionable*, this is referred to as achieving recourse [16, 38].

To motivate our problem, consider an ML application for computational biology: drug discovery is a task that involves generating new molecules that can be used for medicinal purposes [35, 43]. Given a candidate molecule, a GNN can predict if this molecule has a certain property that would make it effective in treating a particular disease [13, 27, 42]. If the GNN predicts it does not have this desirable property, CF explanations can help identify the minimal change required in order for the molecule to have the desirable property. This could help us not only generate a new molecule that has this property, but also understand the molecular structures that contribute to this property.

Although GNNs have shown state-of-the-art results on tasks involving graph data [5, 47], existing methods for explaining the predictions of GNNs have primarily focused on generating subgraphs that are relevant for a particular prediction [1, 8, 19, 24, 28, 31, 40, 44–46]. However, *none of these methods are able to identify the minimal subgraph automatically* – they all require the user to specify the size of the subgraph, S, in advance. We show that even if we adapt existing methods to the CF explanation problem, and try varying values for S, such methods are not able to produce valid, accurate CF explanations, and are therefore not well-suited to solve the CF explanation problem. To address this gap, we propose CF-GNNExplainer, a method for generating CF explanations for GNNs.

Similar to other CF methods for tabular or image data proposed in the literature [16, 39], CF-GNNExplainer works by perturbing input data at the instance-level. Unlike previous methods, CF-GNNExplainer can generate CF explanations for graph data. In particular, our method iteratively removes edges from the original adjacency matrix based on matrix sparsification techniques, keeping track of the perturbation that leads to a change in prediction, and returning the perturbation with the smallest change w.r.t. the number of edges.

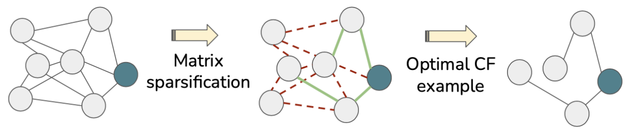

Figure 1: Intuition of counterfactual example generation by CF-GNNExplainer.

We evaluate CF-GNNExplainer on three public datasets for GNN explanations and measure its effectiveness using four metrics: fidelity, explanation size, sparsity, and accuracy. We find that CF-GNNExplainer is able to generate CF examples with at least 94% accuracy, while removing fewer than 3 edges on average. We make the following contributions:

- We formalize the problem of generating CF explanations for GNNs (Section 4).
- We propose CF-GNNExplainer, a novel method for explaining predictions from GNNs (Section 5).
- We propose an experimental setup for holistically evaluating CF explanations for GNNs (Section 6).

The implementation of CF-GNNExplainer is available at `https://anonymous.4open.science/r/cf-gnnexplainer-9EE3`.

2 Related Work

Several GNN XAI approaches have been proposed – a recent survey of the most relevant work is presented by Yuan et al. [45]. However, unlike our work, *none* of the methods in this survey generate CF explanations.

The majority of existing GNN XAI methods provide an explanation in the form of a subgraph of the original graph that is deemed to be important for the prediction [1, 8, 19, 24, 28, 31, 40, 44–46]. We refer to these as *subgraph-generating methods*. Such methods are analogous to popular XAI methods such as LIME [29] or SHAP [22], which identify relevant features for a particular prediction for tabular, image, or text data. All of these methods require the user to specify the size of the explanation, S, in advance: the number features (or edges) to keep. In contrast, CF-GNNExplainer generates CF explanations, which can automatically find the size of the explanation. Although both types of techniques are meant for explaining GNN predictions, they are solving fundamentally different problems: CF explanations generate the minimal perturbation such that the prediction changes, which subgraph-retrieving methods identify a relevant (and not necessarily minimal) subgraph that matches the original prediction.

CF examples are also related to adversarial attacks [36]: they both represent instances obtained from minimal perturbations to the input, which induce changes in the prediction made by the learned model. One difference between the two is in the intent: adversarial examples are meant to fool the model, while CF examples are meant to explain the prediction [21]. In the context of graph data, adversarial attack methods make minimal perturbations to the *overall graph* with the intention of degrading overall model performance, as opposed to attacking individual nodes. In contrast, we are interested in generating CF examples for individual nodes, as opposed to identifying perturbations to the overall graph. We confirm that the CF examples produced by CF-GNNExplainer are informative and not adversarial by measuring the accuracy of our method (see Section 6.3).

There exists a substantial body of work on CF explanations for tabular, image, and text data [16, 34, 39]. Some methods treat the underlying classification model as a black-box [11, 18, 20], whereas others make use of the model's inner workings [14, 21, 37, 38, 41]. All of these methods are based on perturbing feature values to generate CF examples – they are not equipped to handle graph data with relationships (i.e., edges) between instances. In contrast, CF-GNNExplainer provides CF examples for graph data.

3 Background

3.1 Graph Neural Networks

Graphs are structures that represent a set of entities (nodes) and their relations (edges). GNNs operate on graphs to produce representations that can be used in downstream tasks such as graph or node classification. The latter is the focus of this work. We refer to Battaglia et al. [2] and Chami et al. [4] for an extensive overview of existing GNN methods.

Let $f(A, X; W)$ be any GNN, where A is an $n \times n$ adjacency matrix, X is an $n \times p$ feature matrix (with p features), and W are the learned weights of f. In other words, A and X are the inputs of f, and f is parameterized by W.

A node's representation is learned by iteratively updating the node's features based on its neighbors' features. The number of layers in f determines which neighbors are included: if there are ℓ layers, then the node's final representation only includes neighbors that are at most ℓ hops away from that node in the graph $\mathcal{G}$ – the rest of the nodes in $\mathcal{G}$ are not relevant for the computation of the node's final representation. We define the *subgraph neighborhood* of a node v as a tuple of the nodes and edges relevant for the computation of $f(v)$ (i.e., those in the ℓ-hop neighborhood of f): $\mathcal{G}_v = (A_v, X_v)$, where A_v is the subgraph adjacency matrix and X_v is the node feature matrix for nodes that are at most ℓ hops away from v. We then define a node v as a tuple of the form $v = (A_v, x)$, where x is the feature vector for v.

3.2 Matrix Sparsification

CF-GNNExplainer uses matrix sparsification to generate CF examples, inspired by Srinivas et al. [33]. They propose a method for training sparse neural networks: given a weight matrix W, a binary sparsification matrix is learned which is multiplied element-wise with W such that some of the entries in W are zeroed out. In [33], the objective is to remove entries in the weight matrix in order to reduce the number of parameters in the model. In our case, instead of learning a sparsification matrix to *zero out weights*, we want to *zero out entries in the adjacency matrix* (i.e., remove edges) in order to generate CF explanations for GNNs. That is, we want to remove the important edges – those that are crucial for the prediction.

4 Problem Formulation

In general, a CF example $\bar{x}$ for an instance x according to a trained classifier f is found by perturbing the features of x such that $f(x) \neq f(\bar{x})$ [41]. An optimal CF example $\bar{x}^*$ is one that minimizes the distance between the original instance and the CF example, according to some distance function d, and the resulting optimal CF explanation is $\Delta_x^* = \bar{x}^* - x$ [21].

For graph data, it may not be enough to simply perturb node features, especially since they are not always available. This is why we are interested in generating CF examples by perturbing the graph structure instead. In other words, we want to change the relationships between instances, rather than change the instances themselves. Therefore, a CF example for graph data has the form $\bar{v} = (\bar{A}_v, x)$, where x is the feature vector and $\bar{A}_v$ is a perturbed version of A_v, the adjacency matrix of the subgraph neighborhood of a node v. $\bar{A}_v$ is obtained by removing some edges from A_v, such that $f(v) \neq f(\bar{v})$. Following Wachter et al. [41] and Lucic et al. [21], we generate CF examples by minimizing a loss function of the form:

$$\mathcal{L} = \mathcal{L}_{pred}(v, \bar{v} \mid f, g) + \beta \mathcal{L}_{dist}(v, \bar{v} \mid d), \qquad (1)$$

where v is the original node and f is the original model; g is the CF model that generates $\bar{v}$; and $\mathcal{L}_{pred}$ is a prediction loss that encourages $f(v) \neq f(\bar{v})$. $\mathcal{L}_{dist}$ is a distance loss that encourages $\bar{v}$ to be close to v, and β controls how important $\mathcal{L}_{dist}$ is compared to $\mathcal{L}_{pred}$. We want to find $\bar{v}^*$ that minimizes Eq. 1: this is the optimal CF example for v.

5 Method: CF-GNNExplainer

To solve the problem defined in Section 4, we propose CF-GNNExplainer, which generates $\bar{v} = (\bar{A}_v, x)$ given a node $v = (A_v, x)$. Our method can operate on any GNN model f. To illustrate our method and avoid cluttered notation, let f be a standard, one-layer Graph Convolutional Network [17] for node classification:

$$f(A, X; W) = \text{softmax}\left[\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW\right], \qquad (2)$$

where $\tilde{A} = A + I$, I is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ are entries in the degree matrix $\tilde{D}$, X is the node feature matrix, and W is the weight matrix [17].

5.1 Adjacency Matrix Perturbation

First, we define $\bar{A}_v = P \odot A_v$, where P is a binary perturbation matrix that sparsifies A_v. Our aim is to find P for a given node v such that $f(A_v, x) \neq f(P \odot A_v, x)$. To find P, we build upon the method by Srinivas et al. [33] for training sparse neural networks (see Section 3). Our objective is to zero out entries in the adjacency matrix (i.e., remove edges). That is, we want to find P that minimally perturbs A_v, and use it to compute $\bar{A}_v = P \odot A_v$. If an element $P_{i,j} = 0$, this results in the deletion of the edge between node i and node j. When P is a matrix of ones, this indicates that all edges in A_v are used in the forward pass.

Similar to Srinivas et al. [33], we first generate an intermediate, real-valued matrix $\hat{P}$ with entries in $[0, 1]$, apply a sigmoid transformation, then threshold the entries to arrive at a binary P: entries above 0.5 become 1, while those below 0.5 become 0. In the case of undirected graphs (i.e., those with symmetric adjacency matrices), instead of generating $\hat{P}$ directly, we first generate a perturbation vector which we then use to populate $\hat{P}$ in a symmetric manner.

5.2 Counterfactual Generating Model

We want our perturbation matrix P to only act on A_v, not $\tilde{A}_v$, in order to preserve self-loops in the message passing of f (i.e., we always want a node representation update to include the node's representation from the previous layer). Therefore we first rewrite Eq. 2 for our illustrative one-layer case to isolate A_v:

$$f(A_v, X_v; W) = \\ \text{softmax}\left[(D_v + I)^{-1/2}(A_v + I)(D_v + I)^{-1/2}X_vW\right] \tag{3}$$

To generate counterfactuals, we propose a new function g, which is based on f, but it is parameterized by P instead of W. We update the degree matrix D_v based on $P \odot A_v$, add the identity matrix to account for self-loops (as in $\tilde{D}_v$ in Eq. 2), and call this $\bar{D}_v$:

$$g(A_v, X_v, W; P) = \\ \text{softmax}\left[\bar{D_v}^{-1/2}(P \odot A_v + I)\bar{D_v}^{-1/2}X_vW\right] \tag{4}$$

In other words, f learns the weight matrix while holding the data constant, while g is optimized to find a perturbation matrix that is then used to generate new data points (i.e., CF examples) while holding the weight matrix (i.e., model) constant. Another distinction between f and g is that the aim of f is to find the optimal set of weights that generalizes well on an unseen test set, while the objective of g is to generate an optimal CF example, given a particular node (i.e., $\bar{v}$ is the output of g).

5.3 Loss Function Optimization

We generate P by minimizing Eq. 1, adopting the negative log-likelihood (NLL) loss for $\mathcal{L}_{pred}$:

$$\mathcal{L}_{pred}(v, \bar{v}|f, g) = \\ -\mathbb{1}\left[f(v) = f(\bar{v})\right] \cdot \mathcal{L}_{NLL}(f(v), g(\bar{v})). \tag{5}$$

Since we do not want $f(\bar{v})$ to match $f(v)$, we put a negative sign in front of $\mathcal{L}_{pred}$, and include an indicator function to ensure the loss is active as long as $f(\bar{v}) = f(v)$. Note that f and g have the same weight matrix W – the main difference is that g also includes the perturbation matrix P.

For $\mathcal{L}_{dist}$, we take d to be the element-wise difference between v and $\bar{v}$, corresponding to the difference between A_v and $\bar{A_v}$ – the number of edges removed. For undirected graphs, we divide this value by 2 to account for the symmetry in the adjacency matrices. When updating P, we take the gradient of Eq. 1 with respect to the intermediate $\hat{P}$, *not* the binary P.

5.4 CF-GNNExplainer

We call our method CF-GNNExplainer and summarize its details in Algorithm 1: given an instance in the test set v, we first obtain its original prediction from f and initialize $\hat{P}$ as a matrix of ones, J_n, to initially retain all edges. Next, we run CF-GNNExplainer for K iterations. To find a CF example, we use Eq. 4. First, we compute P by thresholding $\hat{P}$ (see Section 5.1). Then we use P to obtain the sparsified adjacency matrix that gives us a candidate CF example. This example is then fed to the original GNN, f, and if f predicts a different output than for the original node, we have found a valid CF example, $\bar{v}$. We keep track of the "best" CF example (i.e., the most minimal according to d), and return this as the optimal CF example $\bar{v}^*$ after K iterations. Between iterations, we compute the loss following Eqn 1 and 5, and update $\hat{P}$ based on the gradient of the loss. In the end, we retrieve the optimal CF explanation $\Delta_v^* = v - \bar{v}^*$.

6 Experimental Setup

6.1 Datasets and Models

Given the challenges associated with defining and evaluating the accuracy of XAI methods [7], we first focus on synthetic tasks where we know the ground-truth explanations. Although there exist real graph classification datasets with ground-truth explanations [6], there do not exist any real node classification datasets with ground-truth explanations, which is the task we focus on in this paper. Building such a dataset would be an excellent contribution, but is outside the scope of this paper.

In our experiments, we use the TREE-CYCLES, TREE-GRIDS, BA-SHAPES datasets from Ying et al. [44]. These are synthetic datasets that were created specifically for the task of explaining node classification predictions from GNNs. Each dataset consists of (i) a base graph, (ii) motifs that are attached to random nodes of the base graph, and (iii) additional edges that are randomly added to the overall graph. They are all undirected graphs. The classification task is to determine whether or not the nodes are part of the motif. The purpose of these datasets is to have a ground-truth for the "correctness" of an explanation: for nodes in the motifs, the explanation is the motif itself [23]. The dataset statistics are available in Table 1.

TREE-CYCLES consists of a binary tree base graph with 6-cycle motifs, TREE-GRIDS also has a binary tree as its base graph, but with 3×3 grids as the motifs. For BA-SHAPES, the base graph is a Barabasi-Albert (BA) with house-shaped motifs, where each motif consists of 5 nodes (one for the top of the house, two in the middle, and two on the bottom). Here, there are four possible classes (not in motif, in motif: top, middle, bottom). We note that compared to the other two datasets, the BA-SHAPES dataset is much more densely connected

Algorithm 1 CF-GNNExplainer: given a node $v = (A_v, x)$ where $f(v) = y$, generate the minimal perturbation, $\bar{v} = (\bar{A}_v, x)$, such that $f(\bar{v}) \neq y$.

```
Input: node $v = (A_v, x)$, trained GNN model $f$, CF model $g$, loss function $\mathcal{L}$, learning rate $\alpha$, number of iterations $K$, distance function $d$.

$f(v) = y$    # Get GNN prediction
$\hat{P} \leftarrow J_n$    # Initialization
$\bar{v}^* = [\,]$

for $K$ iterations do
  $\bar{v}$ = GET_CF_EXAMPLE()
  $\mathcal{L} \leftarrow \mathcal{L}(v, \bar{v}, f, g)$    # Eq 1 & Eq 5
  $\hat{P} \leftarrow \hat{P} + \alpha \nabla_{\hat{P}} \mathcal{L}$    # Update $\hat{P}$
end for

Function GET_CF_EXAMPLE()
  $P \leftarrow \text{threshold}(\sigma(\hat{P}))$
  $\bar{A}_v \leftarrow P \odot A_v$
  $\bar{v}_{cand} \leftarrow (\bar{A}_v, x)$
  if $f(v) \neq f(\bar{v}_{cand})$ then
    $\bar{v} \leftarrow \bar{v}_{cand}$
    if not $\bar{v}^*$ then
      $\bar{v}^* \leftarrow \bar{v}$    # First CF
    else if $d(v, \bar{v}) \leq d(v, \bar{v}^*)$ then
      $\bar{v}^* \leftarrow \bar{v}$    # Keep track of best CF
    end if
  end if
  return $\bar{v}^*$
```

Table 1: Dataset statistics. The # edges in the motif indicates the size of the ground truth (GT) explanation.

	Tree Cycles	Tree Grid	BA Shapes
# classes	2	2	4
# nodes in motif	6	9	5
# edges in motif (GT)	6	12	6
# nodes in total	871	1231	700
# edges in total	1950	3410	4100
Avg node degree	2.27	2.77	5.87
Avg # nodes in A_v	19.12	30.69	304.40
Avg # edges in A_v	18.99	33.94	1106.24

– the node degree is more than twice as high as that of the tree-cycles or tree-grid datasets, and the average number of nodes and edges in each node's computation graph is order(s) of magnitude larger. We use the same experimental setup (i.e., dataset splits, model architecture) as Ying et al. [44] to train a 3-layer GCN (hidden size = 20) for each task. Our GCNs have at least 87% accuracy on the test set.

6.2 Baselines

Since existing GNN XAI methods give explanations in the form of relevant subgraphs as opposed to CF examples, it is not straightforward to identify baselines for our experiments that ensure a fair comparison between methods. To evaluate CF-GNNExplainer, we compare against 4 baselines: random, 1hop, rm-1hop, and GNNExplainer. The random perturbation is meant as a sanity check. We randomly initialize the entries of $\hat{P} \in [-1, 1]$ and apply the same sigmoid transformation and thresholding as described in Section 5.1. We repeat this K times and keep track of the most minimal perturbation resulting in a CF example. Next, we compare against baselines that are based on the 1-hop neighbourhood of v (i.e., its ego graph): 1hop keeps all edges in the ego graph of v, while rm-1hop removes all edges in the ego graph of v.

Our fourth baseline is based on GNNExplainer by Ying et al. [44], which identifies the S most relevant edges for the prediction (i.e., the most relevant subgraph of size S). To generate CF explanations, we remove the subgraph generated by GNNExplainer. We include this method in our experiments in order to have a baseline based on a prominent GNN XAI method, but we note that subgraph-retrieving methods like GNNExplainer are not meant for generating CF explanations. Unlike our method, GNNExplainer cannot automatically find a "minimal" subgraph and therefore requires the user to determine the number of edges to keep in advance (i.e., the value of S). As a result, we cannot evaluate how "minimal" its CF explanations are, but we can compare it against our method in terms of its ability to generate valid CF examples (*Fidelity*) and how accurate those CF examples are (*Accuracy*) (see Section 6.3). We report results on GNNExplainer for $S \in \{1, 2, 3, 4, 5, \text{GT}\}$, where GT is the size of the ground truth explanation (i.e., the number of edges in the motif, see Table 1).

6.3 Metrics

We generate separate CF examples for each node in the graph and evaluate in terms of four metrics.

Fidelity: is defined as the proportion of instances where the original predictions match the prediction for the explanations [26, 29]. Since we generate CF examples, we do not want the original prediction to match the prediction for the explanation, so we want a low value for *Fidelity*.

Explanation Size: is the mean number of removed edges for all instances. It corresponds to the $\mathcal{L}_{dist}$ term in Equation 1: the difference between the original A_v and the counterfactual $\bar{A}_v$. Since we want to have *minimal* explanations, we want a small value for this metric. Note that we cannot evaluate this metric for GNNExplainer.

Sparsity: measures the mean proportion of edges in A_v that are removed [45]. A value of 0 indicates all edges in A_v were removed. Since we want *minimal* explanations, we want a value close to 1. Note that we cannot evaluate this metric for GNNExplainer.

Accuracy: is the mean proportion of explanations that are "correct". Following Luo et al. [23], Ying et al. [44], we only compute accuracy for nodes that are originally predicted as being part of the motifs, since accuracy can only be computed on instances for which we know the ground truth explanations. Since we want *minimal* explanations, we consider an explanation to be correct if it *exclusively* involves edges that are inside the motifs (i.e., only removes edges that are within the motifs).

6.4 Hyperparameter Search

We experiment with different optimizers and hyperparameter values for the number of iterations K, the trade-off parameter β, the learning rate α, and the Nesterov momentum m (when applicable) and choose the setting that produces the most CF examples. We test the number of iterations $K \in \{100, 300, 500\}$, the trade-off parameter $\beta \in \{0.1, 0.5\}$, learning rate $\alpha \in \{0.005, 0.01, 0.1, 1\}$, and Nesterov momentum $m \in \{0, 0.5, 0.7, 0.9\}$. We test Adam, SGD and AdaDelta as optimizers. We find that for all three datasets, the SGD optimizer gives the best results, with $k = 500$, $\beta = 0.5$, and $\alpha = 0.1$. For the tree-cycles and tree-grid datasets, we set $m = 0$, while for the ba-shapes dataset, we use $m = 0.9$.

6.5 Complexity

CF-GNNExplainer has time complexity $O(KN^2)$, where N is the number of nodes in the subgraph neighbourhood and K is the number of iterations. We note that high complexity is common for local XAI methods (i.e., SHAP, GNNExplainer, etc), but in practice, one typically only generates explanations for a subset of the dataset. In total, we run approximately 375 hours of experiments on one Nvidia TitanX Pascal GPU with access to 12GB RAM.

7 Results

We evaluate CF-GNNExplainer in terms of the metrics outlined in Section 6.3. The results are shown in Table 2 and Table 3.[1] In almost all settings, we find that CF-GNNExplainer outperforms the baselines in terms of *Explanation Size*, *Subgraph Impact*, and *Accuracy*, which shows that CF-GNNExplainer satisfies our objective of finding accurate, minimal CF examples. In cases where other methods outperform CF-GNNExplainer on a particular metric, these methods perform poorly on the rest of the metrics, or on other datasets.

7.1 Main Findings

Fidelity: CF-GNNExplainer outperforms 1hop across all three datasets, and outperforms rm-1hop for tree-cycles and tree-grid in terms of *Fidelity*. We find that random has the lowest *Fidelity* in all cases – it is able to find CF examples for every single node. In the following subsections, we will see that this corresponds to poor performance on the other metrics.

Explanation Size: Figures 2 to 5 show histograms of the *Explanation Size* for the five methods we compare.

We see that across all three datasets, CF-GNNExplainer has the smallest (i.e., most minimal) *Explanation Sizes*. This is especially true when comparing to random and 1hop for the ba-shapes dataset, where we had to use a different scale for the x-axis due to how different the *Explanation Sizes* were. We postulate that this difference could be because ba-shapes is a much more densely connected graph; it has fewer nodes but more edges compared to the other two datasets, and the average number of nodes and edges in the subgraph neighborhood is order(s) of magnitude larger (see Table 1). Therefore, when performing random perturbations, there is substantial opportunity to remove edges that do not necessarily need to be removed, leading to much larger *Explanation Sizes*. When there are many edges in the subgraph neighborhood, removing everything except the 1-hop neighbourhood, as is done in 1hop, also results in large *Explanation Sizes*. In contrast, the loss function used by CF-GNNExplainer ensures that only a few edges are removed, which is the desirable behavior since we want minimal explanations.

Sparsity: CF-GNNExplainer outperforms all four baselines for all three datasets in terms of *Sparsity*. We note CF-GNNExplainer and rm-1hop perform much better on this metric in comparison to the other methods, which aligns with the results from *Explanation Size*.

[1]See the Appendix for tables with standard deviations.

Table 2: Results comparing our method (denoted CF-GNN) to RANDOM, 1HOP, and RM-1HOP. Below each metric, ▼ indicates a low value is desirable, while ▲ indicates a high value is desirable.

	TREE-CYCLES				TREE-GRID				BA-SHAPES			
Method	*Fid.* ▼	*Size* ▼	*Spars.* ▲	*Acc.* ▲	*Fid.* ▼	*Size* ▼	*Spars.* ▲	*Acc.* ▲	*Fid.* ▼	*Size* ▼	*Spars.* ▲	*Acc.* ▲
RANDOM	**0.00**	4.70	0.79	0.63	**0.00**	9.06	0.75	0.77	**0.00**	503.31	0.58	0.17
1HOP	0.32	15.64	0.13	0.45	0.32	29.30	0.09	0.72	0.60	504.18	0.05	0.18
RM-1HOP	0.46	2.11	0.89	—	0.61	2.27	0.92	—	0.21	10.56	0.97	**0.99**
CF-GNNEXPLAINER	0.21	**2.09**	**0.90**	**0.94**	0.07	**1.47**	**0.94**	**0.96**	0.39	**2.39**	**0.99**	0.96

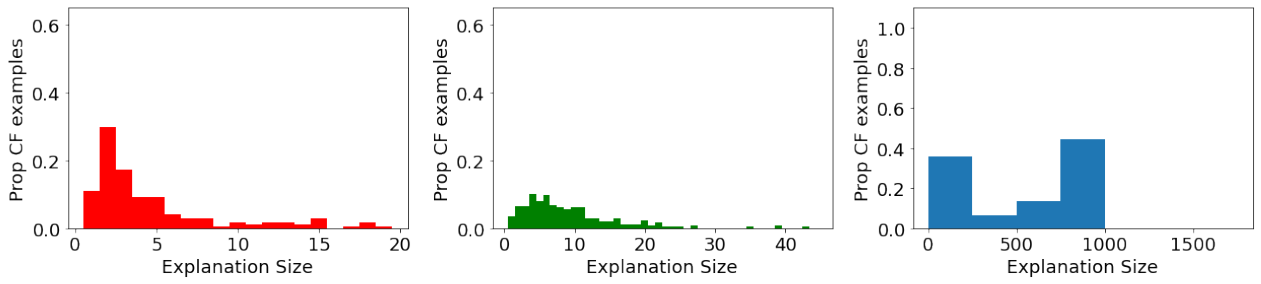

Figure 2: Histograms showing *Explanation Size* from RANDOM. Note the x-axis for BA-SHAPES goes up to 1500. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

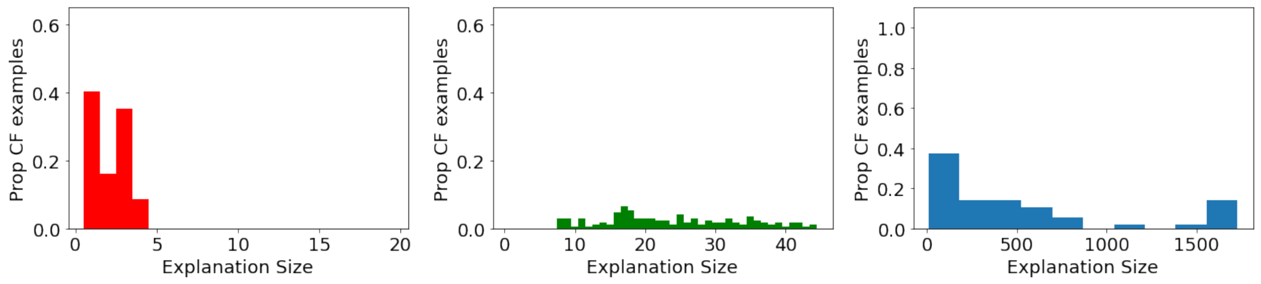

Figure 3: Histograms showing *Explanation Size* from 1HOP. Note the x-axis for BA-SHAPES goes up to 1500. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

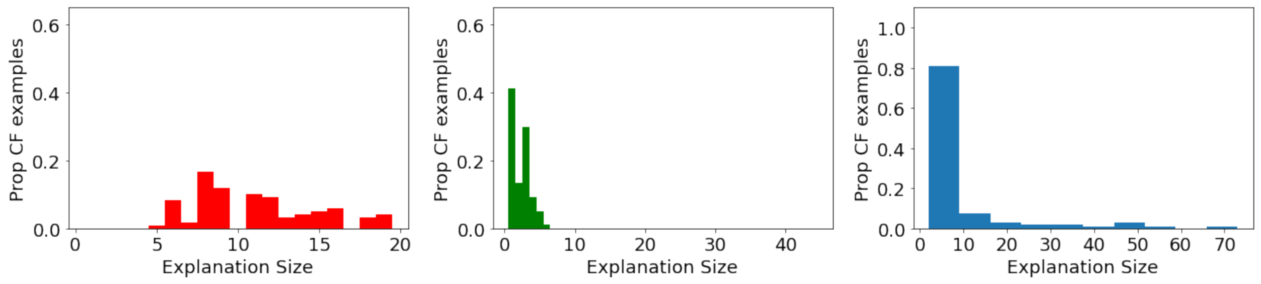

Figure 4: Histograms showing *Explanation Size* from RM-1HOP. Note the x-axis for BA-SHAPES goes up to 70. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

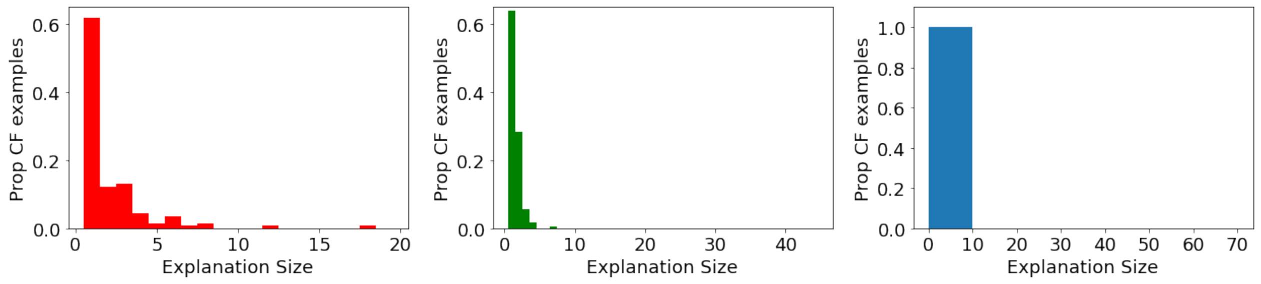

Figure 5: Histograms showing *Explanation Size* from CF-GNNEXPLAINER. Note the x-axis for BA-SHAPES goes up to 70. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

Table 3: Results comparing our method (denoted CF-GNN) to GNNExplainer. GNNExplainer cannot find S automatically, so we try varying values of S. GT indicates the size of the ground truth explanation for each dataset. CF-GNN finds S automatically. Below each metric, ▼ indicates a low value is desirable, while ▲ indicates a high value is desirable.

	tree-cycles				tree-grid				ba-shapes			
	Fid.	*Size*	*Spars.*	*Acc.*	*Fid.*	*Size*	*Spars.*	*Acc.*	*Fid.*	*Size*	*Spars.*	*Acc.*
Method	▼	▼	▲	▲	▼	▼	▲	▲	▼	▼	▲	▲
GNNExp ($S = 1$)	0.65	1.00	0.92	0.61	0.69	1.00	0.96	0.79	0.90	1.00	0.94	0.52
GNNExp ($S = 2$)	0.59	2.00	0.85	0.54	0.51	2.00	0.92	0.78	0.85	2.00	0.91	0.40
GNNExp ($S = 3$)	0.56	3.00	0.79	0.51	0.46	3.00	0.88	0.79	0.83	3.00	0.87	0.34
GNNExp ($S = 4$)	0.58	4.00	0.72	0.48	0.42	4.00	0.84	0.79	0.83	4.00	0.83	0.31
GNNExp ($S = 5$)	0.57	5.00	0.66	0.46	0.40	5.00	0.80	0.79	0.81	5.00	0.81	0.27
GNNExp ($S = \text{GT}$)	0.55	6.00	0.57	0.46	0.35	11.83	0.53	0.74	0.82	6.00	0.79	0.24
CF-GNNExplainer	**0.21**	2.09	0.90	**0.94**	**0.07**	1.47	0.94	**0.96**	**0.39**	2.39	0.99	**0.96**

Accuracy: We observe that CF-GNNExplainer has the highest *Accuracy* for the tree-cycles and tree-grid datasets, whereas rm-1hop has the highest *Accuracy* for ba-shapes. However, we are unable to calculate the accuracy of rm-1hop for the other two datasets since it is unable to generate *any* CF examples for motif nodes, contributing to the low *Coverage* on those datasets. We observe *Accuracy* levels upwards of 94% for CF-GNNExplainer across *all* datasets, indicating that it is consistent in correctly removes edges that are crucial for the initial predictions in the vast majority of cases (see Table 2).

7.2 Comparison to GNNExplainer

Table 3 shows the results comparing our method to GNNExplainer. We find that our method outperforms GNNExplainer across all three datasets in terms of both *Fidelity* and *Accuracy*, for all tested values of S. However, this is not surprising since GNNExplainer is not meant for generating CF explanations, so we cannot expect it to perform well on a task it was not designed for. We cannot compare *Explanation Size* or *Sparsity* fairly since GNNExplainer requires the user to input the value of S.

7.3 Summary of results

Evaluating on four distinct metrics for each dataset gives us a more holistic view of the results. We find that for all three datasets, CF-GNNExplainer can generate CF examples for the majority of nodes in the test set, while only removing a small number of edges. For nodes where we know the ground truth (i.e., those in the motifs) we achieve at least 94% *Accuracy.*

Although random can generate CF examples for every node, they are not very minimal or accurate. The latter is also true for 1hop – in general, it has the worst scores for *Explanation Size*, *Sparsity* and *Accuracy.* GNNExplainer performs at a similar level, indicating that although it is a prominent GNN XAI method, it is not well-suited for solving the CF explanation problem.

rm-1hop is competitive in terms of *Explanation Size*, but it performs poorly in terms of *Fidelity* for the tree-cycles and tree-grid datasets, and its *Accuracy* on these datasets is unknown since it is unable to generate *any* CF examples for nodes in the motifs. These results show that our method is simple and extremely effective in solving the CF explanation task, unlike existing GNN XAI methods.

8 Conclusion

In this work, we propose CF-GNNExplainer, a method for generating CF explanations for any GNN. Our simple and effective method is able to generate CF explanations that are (i) minimal, both in terms of the absolute number of edges removed (*Explanation Size*), as well as the proportion of the subgraph neighborhood that is perturbed (*Sparsity*), and (ii) accurate, in terms of removing edges that we know to be crucial for the initial predictions. We evaluate our method on three commonly used datasets for GNN explanation tasks and find that these results hold across all three datasets. We find that existing GNN XAI methods are not well-suited to solving the CF explanation task, while CF-GNNExplainer is able to reliably produce minimal, accurate CF explanations.

For future work, we plan to incorporate node feature perturbations in our framework and extend CF-GNNExplainer to accommodate graph classification tasks. We also plan to investigate the potential of adapting graph attack methods for generating CF explanations, as well as conduct a user study to determine if humans find CF-GNNExplainer useful in practice.

References

[1] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, May 2019.

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, October 2018.

[3] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models, 2021.

[4] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2021.

[5] Andreea Deac, Yu-Hsiang Huang, Petar Veličković, Pietro Liò, and Jian Tang. Drug-drug adverse effect prediction with graph co-attention. *arXiv preprint arXiv:1905.00534*, May 2019.

[6] 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*, 34:786–797, 1991.

[7] Finale Doshi-Velez and Been Kim. Towards a Rigorous Science of Interpretable Machine Learning. *arXiv preprint arXiv:1702.08608v2*, 2017.

[8] Alexandre Duval and Fragkiskos D. Malliaros. Graphsvx: Shapley value explanations for graph neural networks. 2021.

[9] EU. Regulation (EU) 2016/679 of the European Parliament (GDPR). *Official Journal of the European Union*, L119:1–88, 2016.

[10] Randy Goebel, Ajay Chander, Katharina Holzinger, Freddy Lecue, Zeynep Akata, Simone Stumpf, Peter Kieseberg, and Andreas Holzinger. Explainable AI: The new 42? In *CD-Make 2018*, pages 295–303, 2018.

[11] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, May 2018.

[12] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933*, 2018.

[13] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V. Chawla. Few-shot graph learning for molecular property prediction. In *Proceedings of The Web Conference*, 2021.

[14] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. DACE: Distribution-aware counterfactual explanation by mixed-integer linear optimization. *IJCAI*, pages 2855–2862.

[15] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-Agnostic Counterfactual Explanations for Consequential Decisions. *AISTATS*, May 2019.

[16] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: Definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2020.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, February 2017.

[18] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, December 2017.

[19] Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. *arXiv preprint arXiv:2104.06643*, April 2021.

[20] Ana Lucic, Hinda Haned, and Maarten de Rijke. Why does my model fail? Contrastive local explanations for retail forecasting. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 90–98, 2020.

[21] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. Focus: Flexible optimizable counterfactual explanations for tree ensembles. *arXiv preprint arXiv:1911.12199*, 2020.

[22] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774. Curran Associates, Inc., 2017.

[23] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *NeurIPS*, 2020.

[24] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *arXiv preprint arXiv:2011.04573*, November 2020.

[25] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint arXiv:1706.07269*, 2017.

[26] Christoph Molnar. *Interpretable Machine Learning*. 2019.

[27] Cuong Q. Nguyen, Constantine Kreatsoulas, and Kim M. Branson. Meta-learning gnn initializations for low-resource molecular property prediction. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

[28] Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10764–10773, Long Beach, CA, USA, June 2019. IEEE.

[29] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[30] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[31] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. *arXiv preprint arXiv:2010.00577*, October 2020.

[32] Lisa Schut, Oscar Key, and Rory McGrath. Generating Interpretable Counterfactual Explanations By Implicit Minimisation of Epistemic and Aleatoric Uncertainties. *AISTATS*, 2021.

[33] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. Training sparse neural networks. *arXiv preprint arXiv:1611.06694*, November 2016.

[34] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9: 11974–12001, 2021.

[35] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180:688–702, 2020.

[36] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S. Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[37] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 465–474, 2017.

[38] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 10–19, 2019.

[39] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[40] Minh N. Vu and My T. Thai. PGM-Explainer: Probabilistic graphical model explanations for graph neural networks. 2020.

[41] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31 (2):841–888, 2018.

[42] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, December 2020.

[43] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. In *Proceedings of the International Conference on Learning Representations*, 2021.

[44] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. *arXiv preprint arXiv:1903.03894*, November 2019.

[45] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*, 2020.

[46] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. *arXiv preprint arXiv:2102.05152*, February 2021.

[47] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, July 2018.