CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks

Lucic, A.; ter Hoeve, M.; Tolomei, G.; de Rijke, M.; Silvestri, F.

DOI
10.1145/1122445.1122456
Publication date
2021
Document Version
Final published version
Published in
DLG-KDD'21

Link to publication

Citation for published version (APA):
Lucic, A., ter Hoeve, M., Tolomei, G., de Rijke, M., & Silvestri, F. (2021). CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. In *DLG-KDD'21: Deep Learning on Graphs, August 14–18, 2021, Online* Article 3 ACM. https://doi.org/10.1145/1122445.1122456

CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks

Ana Lucic
University of Amsterdam
Amsterdam, Netherlands
a.lucic@uva.nl

Maartje ter Hoeve
University of Amsterdam
Amsterdam, Netherlands
m.a.terhoeve@uva.nl

Gabriele Tolomei
Sapienza University of Rome
Rome, Italy
gabriele.tolomei@uniroma1.it

Maarten de Rijke
University of Amsterdam
Amsterdam, Netherlands
derijke@uva.nl

Fabrizio Silvestri
Sapienza University of Rome
Rome, Italy
fsilvestri@diag.uniroma1.it

ABSTRACT

Given the increasing promise of Graph Neural Networks (GNNs) in real-world applications, several methods have been developed for explaining their predictions. So far, these methods 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 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 our method primarily removes edges that are crucial for the original predictions, resulting in minimal CF explanations.

ACM Reference Format:
Ana Lucic, Maartje ter Hoeve, Gabriele Tolomei, Maarten de Rijke, and Fabrizio Silvestri. 2021. CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. In *DLG-KDD'21: Deep Learning on Graphs, August 14–18, 2021, Online.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

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 [7]. 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 [5]. Having certified methods for interpreting ML predictions will help enable their use across a variety of applications [18].

Explainable AI (XAI) refers to the set of techniques "*focused on exposing complex AI models to humans in a systematic and interpretable manner*" [21]. A large body of work on XAI has emerged in recent years [2, 8]. Counterfactual (CF) explanations are used to explain predictions of individual instances in the form: "If X had been different, Y would not have occurred" [25]. CF explanations are based on CF examples: modified versions of the input sample that result in an alternative output response (i.e., prediction). If the modifications recommended are also clearly *actionable*, this is referred to as achieving recourse [12, 28].

To motivate our problem, we consider an ML application for computational biology. Drug discovery is a task that involves generating new molecules that can be used for medicinal purposes [26, 33]. 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 [9, 19, 32]. If the GNN predicts it does not have this desirable property, CF explanations can help identify the minimal change one should make to this molecule, such that it has this desirable property. This could help us not only generate a new molecule with the desired property, but also understand what structures contribute to a molecule having this property.

Although GNNs have shown state-of-the-art results on tasks involving graph data [3, 38], existing methods for explaining the predictions of GNNs have primarily focused on generating subgraphs that are relevant for a particular prediction [1, 4, 14, 16, 20, 22, 30, 34, 36, 37]. However, none of these methods are able to *identify the minimal subgraph automatically* since they all require the user to specify in advance the size of the subgraph, S. 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 explanations, and are therefore not well-suited to solve the CF explanation problem. To address this gap, we propose CF-GNNExplainer, which generates CF explanations for GNNs.

Similar to other CF methods proposed in the literature [12, 29], CF-GNNExplainer works by perturbing data at the instance-level. In this work, the instances are nodes in the graph since we focus on node classification. In particular, our method iteratively removes edges from the original adjacency matrix based on matrix sparsification techniques, keeping track of the perturbations that lead to a change in prediction, and returning the perturbation with the smallest change w.r.t. the number of edges. We evaluate CF-GNNExplainer on three public datasets for GNN explanations and measure its effectiveness using four metrics: fidelity, explanation size, sparsity, and accuracy. CF-GNNExplainer is able to generate CF examples with at least 94% accuracy, while removing fewer than three edges on average. In this work, we (i) formalize the problem of generating CF explanations for GNNs, and (ii) propose a novel method for generating CF explanations for GNNs.

2 PROBLEM FORMULATION

Let $f(A, X; W)$ be any GNN, where A is the adjacency matrix, X is the feature matrix, and W is the learned weight matrix of f (i.e., A and X are the inputs of f, and f is parameterized by W). We define the *subgraph neighbourhood* 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 neighbourhood 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 define a node v as a tuple of the form $v = (A_v, x)$, where x is the feature vector for v.

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 $f(x) \neq f(\bar{x})$ [31]. 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. The resulting optimal CF explanation is $\Delta_x^* = \bar{x}^* - x$ [15].

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. $\bar{A_v}$ is obtained by removing some edges from A_v, such that $f(v) \neq f(\bar{v})$.

Following Wachter et al. [31] and Lucic et al. [15], we generate CF examples by minimizing a loss function of the form:

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

where v is the original node, 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 $\mathcal{L}_{pred}$. The goal is to find $\bar{v}^*$ that minimizes Equation 1: this is the optimal CF example for v.

3 METHOD

Here we propose CF-GNNExplainer, a method for generating CF explanations of the form $\bar{v} = (\bar{A_v}, x)$, 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 [13] for node classification:

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

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 [13].

3.1 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. [24] for training sparse neural networks, where the objective is to zero out weights in W. In contrast, 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. [24], 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.

3.2 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 Equation 2 for our illustrative one-layer case to isolate A_v:

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

To generate CFs, we propose a new function g, which is based on f, but it is parameterized by P instead of by 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 Equation 2), and call this $\bar{D_v}$:

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

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).

3.3 Loss Function Optimization

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

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

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 $\bar{v}$. Since we do not perturb the feature values, this corresponds to the difference between A_v and $\bar{A_v}$, i.e., 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 Equation 1 with respect to the intermediate $\hat{P}$, *not* the binary P.

3.4 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,

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 $v = (x, A_v)$, trained GNN model $f$, CF model $g$, loss function $\mathcal{L}$, learning rate $\alpha$, trade-off parameter $\beta$, number of iterations $K$, distance function $d$.

$f(v) = y$   # Get GNN prediction
$\hat{P} \leftarrow J_n$   # Initialization

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

Function GET_CF_EXAMPLE()
    $P \leftarrow$ threshold($\sigma(\hat{P}^{(k)})$)
    $\bar{A_v} \leftarrow P \odot A_v$
    $\bar{v}^{(k)}_{cand} \leftarrow (\bar{A_v}, x)$
    if $f(v) \neq f(\bar{v}^{(k)}_{cand})$ then
        $\bar{v}^{(k)} \leftarrow \bar{v}^{(k)}_{cand}$
        if $\mathcal{L}_{dist}(v, \bar{v}) \leq \mathcal{L}_{dist}(v, \bar{v}^{(k)})$ then
            $\bar{v}^* \leftarrow \bar{v}^{(k)}$   # Keep track of best CF
        end if
    end if
    return $\bar{v}^*$
```

J_n, to initially retain all edges. Next, we run CF-GNNExplainer for K iterations. To find a CF example, we use Equation 4. First, we compute P by thresholding $\hat{P}$ (see Section 3.1). 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 Equations 1 and 5, and update $\hat{P}$ based on the gradient of the loss. In the end, we retrieve the optimal CF explanation $\Delta_v^* = v - \bar{v}^*$.

4 EXPERIMENTAL SETUP

We use the tree-cycles, tree-grids, ba-shapes node classification datasets from Ying et al. [34] to run our experiments for generating CF examples. These datasets were created specifically for the task of explaining 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 [17]. tree-cycles consists of a binary tree base graph with cycle-shaped motifs, tree-grids also has a binary tree as its base graph, 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. Here there are 4 possible classes: top of house, middle of house, bottom of house, or not part of the house. We use the same dataset splits (80% train, 10% validation, 10% test) and training setup as Ying et al. to train a 3-layer GCN (hidden size = 20) for each node classification task. Our GCNs have at least 87% accuracy on the test set.

To evaluate our method, we compare against 4 baselines: random, only-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 3.1. 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): only-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. [34], 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 such subgraph-retrieving methods 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*). We perform a hyperparameter search over S and choose the setting that produces the most CF examples. Data and code for all experiments is available at `https://github.com/a-lucic/cf-gnnexplainer`.

5 RESULTS

We evaluate these CF examples in terms of four metrics: (i) *Fidelity*, (ii) *Explanation Size*, (iii) *Sparsity*, and (iv) *Accuracy*. The results are shown in Table 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 accurately finding minimal CF examples. In cases where the baselines outperform our method on a particular metric, these baselines either perform poorly on the rest of the metrics, or on other datasets.

5.1 Fidelity

Fidelity is defined as the proportion of nodes where the original predictions match the prediction for the explanations [36]. 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*. CF-GNNExplainer outperforms only-1hop and GNNExplainer across all three datasets, and outperforms rm-1hop

Table 1: Results comparing our method to the baselines. Below each metric, ▼ indicates a low value is desirable, while ▲ indicates a high value is desirable.

	TREE-CYCLES				TREE-GRID				BA-SHAPES			
Metric	*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
ONLY-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**
GNNEXPLAINER	0.55	6.00	0.57	0.46	0.34	8.00	0.68	0.74	0.81	6.00	0.81	0.27
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

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. However, we will see that this corresponds to poor performance on the other metrics.

5.2 Explanation Size

Explanation Size is the number of removed edges. 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. Figures 1 to 5 show histograms of the *Explanation Size* for the five methods we test. 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 ONLY-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 neighbourhood is order(s) of magnitude larger. 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 neighbourhood, removing everything except the 1-hop neighbourhood, as is done in ONLY-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.

5.3 Sparsity

Sparsity measures the proportion of edges in A_v that are removed [36]. A value of 0 indicates all edges in A_v were removed. Since we want *minimal* explanations, we want a value close to 1. 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*.

5.4 Accuracy

Accuracy is the proportion of explanations that are "correct". Following Ying et al. [34] and Luo et al. [17], 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). We observe that our method 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 the motif nodes, contributing to the undesirable high *Fidelity* on those datasets. We observe *Accuracy* levels upwards of 94% for our method across *all* datasets, indicating that it is consistent in correctly removing edges that are crucial for the initial predictions.

5.5 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 ONLY-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.

6 RELATED WORK

Several GNN XAI approaches have been proposed – a recent survey of the most relevant work is presented by Yuan et al. [36]. However, unlike our work, *none* of the methods in this survey generate CF explanations. The vast majority of GNN explanation methods are based on retrieving a relevant subgraph of the original graph [1, 4, 14, 16, 20, 22, 30, 34, 37]. Other methods identify important node features [10] or similar examples [6]. Kang et al. [11] generate CF examples for GNNs, but their work focuses on a different task: link prediction. Other GNN XAI methods identify important node features [10] or similar examples [6]. Yuan et al. [35] and Schnake et al. [23] generate model-level explanations for GNNs, which differs from our work since we produce instance-level explanations. Adversarial attacks [27] are also related to CF examples, but there is a distinction in the intent: adversarial examples are meant to fool the model, while CF examples are meant to explain the prediction [15]. In the context of graph data, adversarial attack methods try to make minimal perturbations to the *overall graph* with the intention of degrading model performance. In contrast, we are interested in generating CF examples for individual nodes, as opposed to identifying perturbations to the overall graph.

Existing work on CF explanations for tabular, image, and text data is based on perturbing *feature values* to generate CF examples [12, 29]. However, they are not equipped to handle graph data with relationships (i.e., edges) between data points, unlike our method.

7 CONCLUSION

We propose CF-GNNExplainer, which generates CF explanations for any GNN. Our simple and effective method is able to generate CF explanations that are (i) minimal, 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. For future work, we plan to conduct a user study to determine if humans find CF-GNNExplainer useful in practice.

REFERENCES

[1] Federico Baldassarre and Hossein Azizpour. 2019. Explainability Techniques for Graph Convolutional Networks. *arXiv preprint arXiv:1905.13686* (May 2019).

[2] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. 2021. Benchmarking and Survey of Explanation Methods for Black Box Models. arXiv:2102.13076 [cs.AI]

[3] Andreea Deac, Yu-Hsiang Huang, Petar Veličković, Pietro Liò, and Jian Tang. 2019. Drug-Drug Adverse Effect Prediction with Graph Co-Attention. *arXiv:1905.00534 [cs, q-bio, stat]* (May 2019). http://arxiv.org/abs/1905.00534 arXiv: 1905.00534.

[4] Alexandre Duval and Fragkiskos D. Malliaros. 2021. GraphSVX: Shapley Value Explanations for Graph Neural Networks. (2021). arXiv:2104.10482 [cs.LG]

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

[6] Lukas Faber, Amin K Moghaddam, and Roger Wattenhofer. 2020. Contrastive Graph Neural Network Explanation. *ICML 2020 Workshop on Graph Representation Learning and Beyond* (2020), 6.

[7] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger. 2018. Explainable AI: The New 42?. In *CD-Make 2018*. 295–303.

[8] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. 2018. A Survey of Methods for Explaining Black Box Models. *arXiv preprint arXiv:1802.01933* (2018).

[9] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V. Chawla. 2021. Few-Shot Graph Learning for Molecular Property Prediction. In *Proceedings of The Web Conference*. arXiv:2103.10432 [q-bio.BM]

[10] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. *arXiv preprint arXiv:2001.06216* (Jan. 2020).

[11] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. 2019. ExplaiNE: An Approach for Explaining Network Embedding-based Link Predictions. *arXiv preprint arXiv:1904.12694* (2019).

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

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (Feb. 2017).

[14] Wanyu Lin, Hao Lan, and Baochun Li. 2021. Generative Causal Explanations for Graph Neural Networks. *arXiv preprint arXiv:2104.06643* (April 2021).

[15] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. 2020. FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles. arXiv:1911.12199 [cs.LG]

[16] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized Explainer for Graph Neural Network. (Nov. 2020). http://arxiv.org/abs/2011.04573

[17] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized Explainer for Graph Neural Network. *NeurIPS* (2020).

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

[19] Cuong Q. Nguyen, Constantine Kreatsoulas, and Kim M. Branson. 2020. Meta-Learning GNN Initializations for Low-Resource Molecular Property Prediction. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+)*. arXiv:2003.05996 [cs.LG]

[20] Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. 2019. Explainability Methods for Graph Convolutional Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 10764–10773.

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

[22] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2020. Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking. *arXiv preprint arXiv:2010.00577* (Oct. 2020).

[23] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T. Schütt, Klaus-Robert Müller, and Grégoire Montavon. 2020. XAI for Graphs: Explaining Graph Neural Network Predictions by Identifying Relevant Walks. *arXiv preprint arXiv:2006.03589* (June 2020).

[24] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. 2016. Training Sparse Neural Networks. *arXiv preprint arXiv:1611.06694* (Nov. 2016).

[25] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. 2021. A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence. *IEEE Access* 9 (2021), 11974–12001.

[26] Jonathan M Stokes. 2020. A Deep Learning Approach to Antibiotic Discovery. *Cell* (2020), 29.

[27] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S. Yu, Lifang He, and Bo Li. 2018. Adversarial Attack and Defense on Graph Data: A Survey. *arXiv preprint arXiv:1812.10528* (2018).

[28] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable Recourse in Linear Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 10–19.

[29] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual Explanations for Machine Learning: A Review. *arXiv preprint arXiv:2010.10596* (2020).

[30] Minh N. Vu and My T. Thai. 2020. PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks. (2020).

[31] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2018. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 2 (2018), 841–888.

[32] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. 2020. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies* (Dec. 2020), S1740674920300305. https://doi.org/10.1016/j.ddtec.2020.11.009

[33] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. 2021. MARS: Markov Molecular Sampling for Multi-objective Drug Discovery. In *Proceedings of the International Conference on Learning Representations*. arXiv:2103.10432 [q-bio.BM]

[34] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. *arXiv preprint arXiv:1903.03894* (Nov. 2019).

[35] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. XGNN: Towards Model-Level Explanations of Graph Neural Networks. *arXiv preprint arXiv:2006.02587* (June 2020).

[36] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in Graph Neural Networks: A Taxonomic Survey. *arXiv preprint arXiv:2012.15445* (2020).

[37] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On Explainability of Graph Neural Networks via Subgraph Explorations. *arXiv preprint arXiv:2102.05152* (Feb. 2021).

[38] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 34, 13 (July 2018), i457–i466. https://doi.org/10.1093/bioinformatics/bty294 arXiv: 1802.00543.

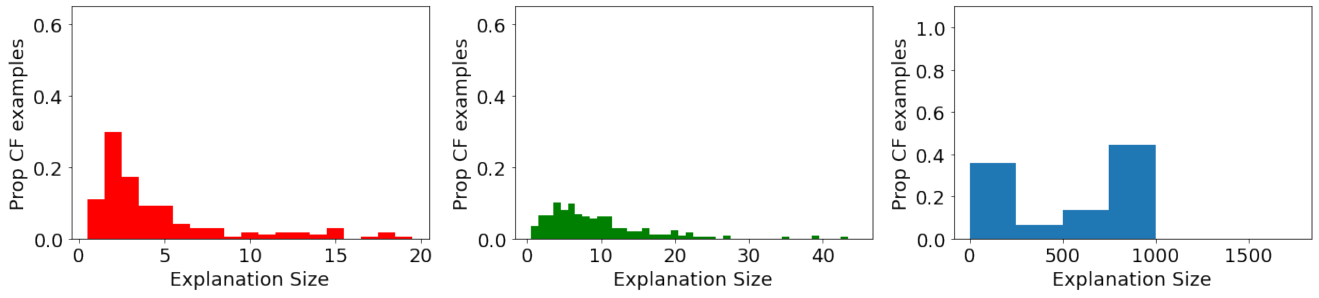

Figure 1: Histograms showing *Explanation Size* from RANDOM. Note the x-axis for BA-SHAPES goes up to 1500. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

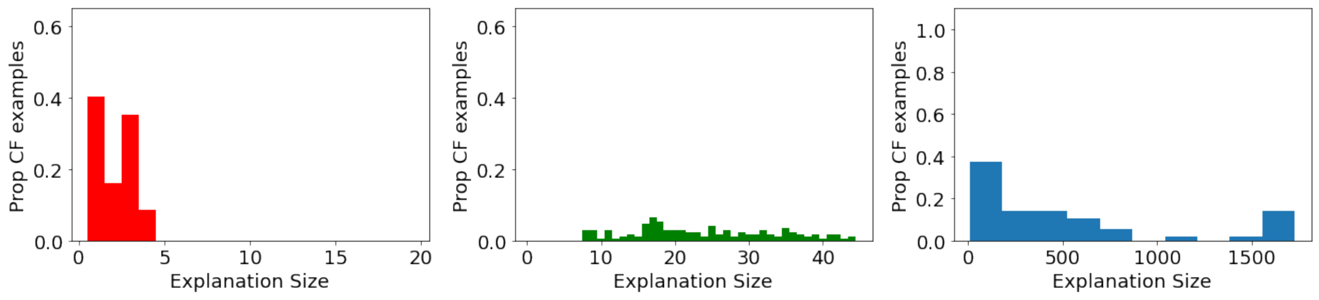

Figure 2: Histograms showing *Explanation Size* from ONLY-1HOP. Note the x-axis for BA-SHAPES goes up to 1500. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

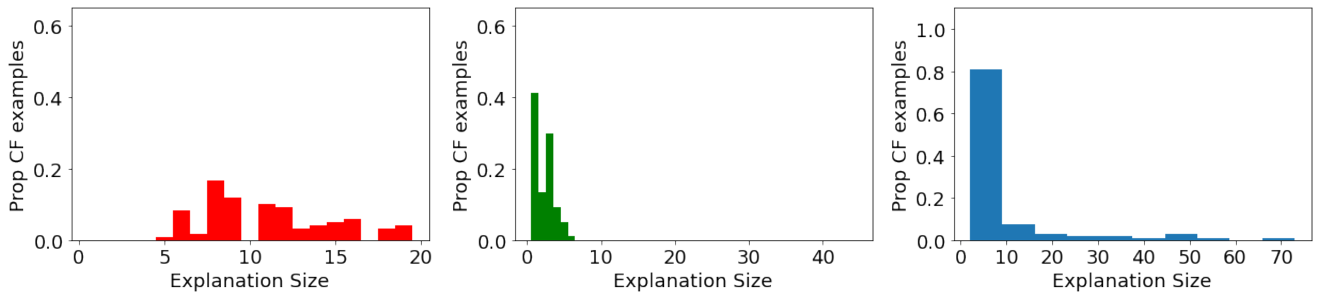

Figure 3: Histograms showing *Explanation Size* from RM-1HOP. Note the x-axis for BA-SHAPES goes up to 70. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

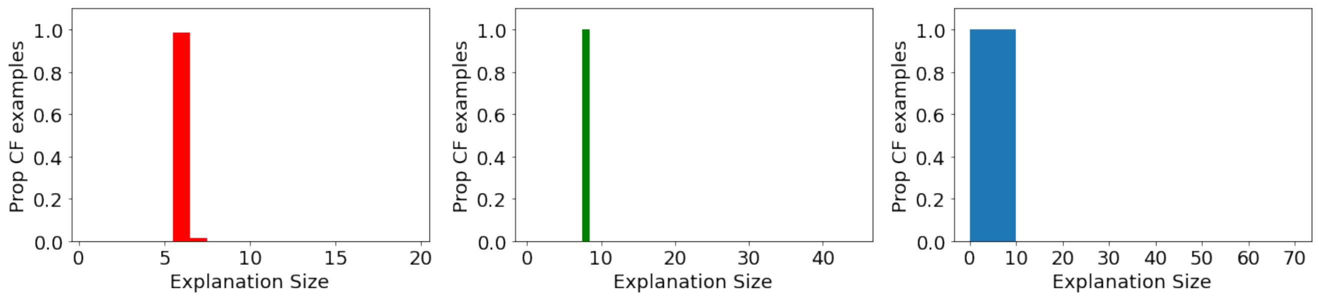

Figure 4: Histograms showing *Explanation Size* from GNNEXPLAINER. Note that the y-axis goes up to 1. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.

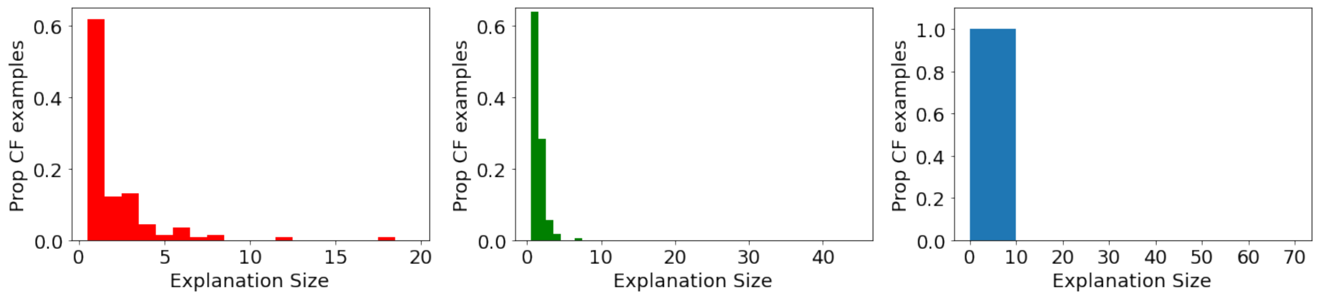

Figure 5: Histograms showing *Explanation Size* from our method, CF-GNNEXPLAINER. Note the x-axis for BA-SHAPES goes up to 70. Left: TREE-CYCLES, Middle: TREE-GRID, Right: BA-SHAPES.