Are Generative-based Graph Counterfactual Explainers Worth It?

Mario Alfonso Prado-Romero[1,[0000−0002−0491−3515]], Bardh Prenkaj[2,[0000−0002−2991−2279]], and Giovanni Stilo[3,[0000−0002−2092−0213]]

[1] Gran Sasso Science Institute
[2] Sapienza University of Rome
[3] University of L'Aquila

Abstract. Counterfactual Explanation (CE) methods have gained traction as a means to provide recourse for users of AI systems. While widely explored in domains like medical images and self-driving cars, Graph Counterfactual Explanation (GCE) methods have received less attention. GCE explainers generate a new graph similar to the original but with a different outcome according to the underlying prediction model. Notably, generative machine learning methods have achieved remarkable success in generating images with a particular art style and natural language processing. In this study, we thoroughly examine the capabilities of Generative GCE methods. Specifically, we analyse G-CounteRGAN, a graph-specific adaptation of the CounteRGAN method, and compare its performance against other generative explainers and a selection of search- and heuristic-based explainers in the literature. Contrarily to heuristic-based methods, we remark that generative approaches are extremely useful to generate multiple counterfactuals by sampling the learned latent space on the training data.

Keywords: Graph Counterfactual Explainability, · Generative Explanations · Graph-to-Image

1 Introduction

Explainability is crucial in sensitive domains to enable users and service providers to make informed and reliable decisions [10]. However, deep neural networks, commonly used for generating predictions, often suffer from a lack of interpretability, widely referred to as the "black-box" problem [22], hindering their wide adoption in domains such as healthcare and finance. On the other end of the spectrum of explainability, we find inherently interpretable "white-box" prediction models [16], which are preferred for decision-making purposes [34]. Alas,

marioalfonso.prado@gssi.it, prenkaj@di.uniroma1.it
corresponding: giovanni.stilo@univaq.it

black-box models demonstrate superior performance and generalisation capabilities when dealing with high-dimensional data [2,6,8,11,18,27,28,29,35,37].

Recently, deep learning (relying on GNNs [30]) has been beneficial in solving graph-based prediction tasks, such as community detection [41], link prediction [38], and session-based recommendations [40,42]. Despite their remarkable performance, GNNs are black boxes, making them unsuitable for high-impact and high-risk scenarios. The literature has proposed several post-hoc explainability methods to understand "what is happening under the hood" of the prediction models. Specifically, counterfactual explainability is useful to understand how modifications in the input lead to different outcomes. Similarly, a recent field in Graph Counterfactual Explainability (GCE) has emerged [25] that aims at producing counterfactuals for graphs.

We provide the reader with an example that helps clarify a counterfactual example in graphs. Suppose we have a social network where a specific user U posts an illicit advertisement, thus violating the Terms of Service (ToS). A counterfactual explanation of U's account suspension would be "*if the user had refrained from writing the post about selling illegal goods, her account would not have been banned*".

Generally, GCE methods can be search-, heuristic-, and learning-based approaches [25]. Search-based approaches find the counterfactual examples within the data distribution: i.e., for a graph, G is the dataset $\mathcal{G}$ and a given predictor Φ, they find a $G' \in \mathcal{G}$ s.t. $\Phi(G) \neq \Phi(G')$. In this way, the search for counterfactuals is bound by the data distribution, which is not necessarily the reality. In scenarios where access to the graph dataset $\mathcal{G}$ is unavailable, search-based methods fail to produce a valid counterfactual, undermining their usefulness as explainers. Heuristic-based approaches surpass this limitation because they perturb the original graph G into G' such that $\Phi(G) \neq \Phi(G')$ without accessing the dataset $\mathcal{G}$. In other words, G' can be outside the data distribution of $\mathcal{G}'$. A substantial drawback of heuristic-based approaches is defining the perturbation heuristic (e.g., rules), which comes after careful examination of the data. These heuristic approaches often require domain expertise to express how the input graph should be perturbed faithfully. For instance, producing valid counterfactuals for molecules requires knowledge about atom valences and chemical bonds. Contrarily, learning-based approaches learn the heuristic (e.g., reinforcement learning) based on the data. This kind of explainer is trained on some samples and can be used to produce a single counterfactual at inference time.

Within learning-based approaches, we identify generative strategies that allow us to produce multiple counterfactual examples from a learned latent space. After training, generative approaches do not need access to the oracle Φ since their latent space can be sampled to generate multiple valid counterfactuals for a particular input graph. Generative approaches are important since they learn the perturbation of the input autonomously (vs heuristics approaches), are not confined to the data distribution (vs search approaches), and do not rely on a learned mask to apply to the input to produce counterfactuals (vs learning approaches).

This work investigates the behaviour of generative GCE methods compared to search- and heuristic-based methods. We test SoA generative approaches against search and heuristic methods, including two baseline. We also adapt CounteRGAN [19] - namely G-CounteRGAN - on graph data, using the GRETEL framework [23,26] and extend the experimental work in [24]. In more detail, differently from [24], we discuss a critical yet important argument, i.e., *can generative approaches compete with search/heuristic baselines in producing valid counterfactuals in large synthetic datasets?* We argue that generative explainers are crucial to produce multiple counterfactuals by sampling their learned latent space. Additionally, differently from heuristic-based methods which have a hard-coded perturbation strategy, generative approaches can potentially learn this perturbation strategy and apply it to the sampled instances from the latent space. This leads us to believe that generative approaches have the potential to become akin to a Swiss-knife in GCE. Finally, we aim to provide readers with initial insights into applying generative approaches for GCE, which still need to be explored in the current literature.

The rest of the paper is organised as follows. Sec. 2 presents the related work in GCE, focusing on generative approaches. Sec. 3 describes the problem formulation. Sec. 4 illustrates how G-CounteRGAN works and its adaption to the graph domain. Sec. 5.1 describes the datasets and their characteristics. Sec. 5 depicts the experiments and extensive discussions. Finally, Sec. 6 concludes the paper.

2 Related Work

The eXplainable AI (XAI) literature distinguishes between inherently explainable and black-box methods [10]. Black-box methods can be further categorised into factual and counterfactual explanation methods. Here, we concentrate on counterfactual methods as categorised in [25] and exploit the same notation used in that survey.

While many works provide counterfactual explanations for images/text (to point out some [5,31,32,44,36,43,47]), only a few focus on graph classification problems [1,14,17,20,21,33,39]. According to [25], GCE works are categorised into search (heuristic) and learning-based approaches. We are aware that a new branch of global (model-level) counterfactual explanations is being developed (see [12]). Nevertheless, we concentrate only on instance-level explanations.

Search (heuristic) approaches. Search-based methods for generating counterfactual explanations rely on a specific criterion, such as the similarity between instances, to search for a suitable counterfactual within the dataset. On the other hand, heuristic-based methods adopt a systematic approach to modify the input graph until a valid counterfactual is obtained.

In this work, we adopted DDBS and OBS [1] because they are the cornerstone for graph counterfactuality in the context of brain networks. These methods represent the brain as a graph, where vertices correspond to well-established regions of interest (ROIs) and edges represent connections between co-activated

ROIs. DDBS and OBS employ a bidirectional search heuristic: first, they perturb the edges of the input graph G until a counterfactual graph G' is reached; then, they rollback certain perturbations made in the initial stage, aiming to reduce the distance between G and G' while maintaining the counterfactual condition. OBS selects edges for perturbation randomly, while DBS queries the dataset to identify the most common edges for each class and leverages this information for perturbation purposes. We point the reader to [3,14,39] for other search-based methods.

Learning-based approaches. The methods belonging to this category share a three-step pipeline: 1) generating masks that indicate the relevant features given a specific input graph G; 2) combining the mask with G to derive a new graph G'; 3) feeding G' to the prediction model (oracle) Φ and updating the mask based on the outcome $\Phi(G')$. Without loss of generality, learning-based strategies can be further divided into perturbation matrix [33], reinforcement learning [20,21,39], and generative approaches [17] In the following, we report the most recent and effective ones.

CF^2 [33] generates factual explanations by balancing factual and counterfactual reasoning. This method generates a factual graph, a subgraph of the original input. Then, it produces a counterfactual by removing the factual subgraph from the input, similarly to [3]. CF^2 also considers the simplicity of the counterfactual (i.e., a smaller explanation size is preferred). Despite being a factual method, CF^2 is included here because it inherently exhibits a counterfactual property through the elimination of the factual subgraph.

MEG [21] and MACCS [39] employ multi-objective reinforcement learning (RL) models to generate counterfactuals for molecules. Their domain-specificity limits their applicability and makes them difficult to port on other domains. The reward function incorporates a task-specific regularisation term that influences the choice of the next action to perturb the input. Similarly, MACDA [20] uses RL to produce counterfactuals for the drug-target affinity prediction problem.

CLEAR [17] is a generative method that relies on a variational autoencoder (VAE). The encoder maps each graph G to a latent representation Z, and the decoder generates the counterfactual based on Z. The generation of counterfactuals is conditioned on G and a desired class $c \neq \Phi(G)$. As per how a VAE works, the generated counterfactuals are complete graphs with stochastic weights on the edges. Therefore, the authors use a sampling procedure to produce valid counterfactuals. Notice, however, that during the decoding process, the order of the vertices in G differs from that in G'. Hence, a graph matching step between G and G' is necessary, which can be time-consuming due to approximating the NP-hard problem [15].

As mentioned in Sec. 1, the method proposed in this work is an adaptation of CounteRGAN [19]. Although originally not proposed for graphs, CounteRGAN is a generative approach for image-based counterfactual explanations for a specific explainee class $c \in C$. It employs a GAN with residual connections and an underlying classifier to generate valid and plausible counterfactuals for the input. This way, the generator network learns to produce counterfactuals for the

input with outcome c. Unlike CLEAR, it does not sample the latent space of the generator to produce multiple counterfactuals for the same input instance. Instead, CounteRGAN binarises the real values of the generator's latent space according to a user-defined threshold, thus giving a unique solution. Nevertheless, CounteRGAN requires multiple trained GAN architectures, each of whose generators are responsible for explaining one class $c \in C$, making it unfeasible for multi-class classification problems[4].

3 Problem Formulation

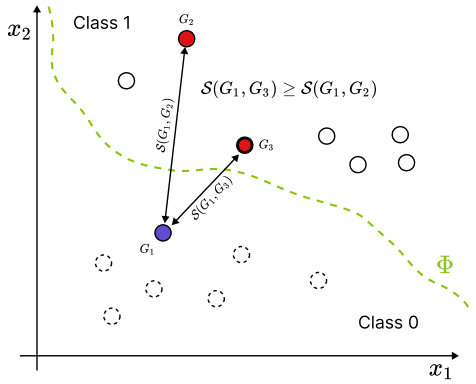

Fig. 1. Given the separation line learned from the prediction model Φ, we can classify instances into two classes. We show three instances G_1, G_2, and G_3 such that $\Phi(G_1) = 0$ and $\Phi(G_2) = \Phi(G_3) = 1$. Because G_2 and G_3 are classified differently from G_1, they are valid counterfactuals. However, our goal is to find counterfactuals whose similarity is the highest according to a particular function $\mathcal{S}$. Here, $\mathcal{S}(G_1, G_3) \geq \mathcal{S}(G_1, G_2)$ which makes G_3 the counterfactual with minimal changes w.r.t. G_1.

Here, we provide the ground formalisation on which our method is based. The literature has formulated a plethora of definitions of what is a counterfactual explanation. Generally, given a graph G, a counterfactual explanation G' satisfies the condition $\Phi(G) \neq \Phi(G')$, where Φ is a particular prediction model. Nevertheless, this counterfactual definition is under-specified since we are interested in having a counterfactual G' with minimal changes w.r.t. the original graph G and not just one that lies on the other side of the decision boundary (see Fig. 1). We rely on the notation introduced in [25] which is a general formulation for multi-class classification problems easily adaptable to a binary classification scenario.

Definition 1 (Graph Counterfactual). *Given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$ is the set*

[4] Note that, the authors tested CounteRGAN only on a binary classification scenario.

of edges, a prediction model Φ that classifies G into a class $c \in C$. The set of counterfactuals of G can be defined as in Eq. 1.

$$\begin{aligned} s(c', G) &\coloneqq \max_{G' \in \mathcal{G}', G \neq G'} \{\mathcal{S}(G, G') \mid \Phi(G') = c'\} \\ \mathcal{E}_\Phi(G) &= \bigcup_{c' \in C - \{c\}} \{G' \in \mathcal{G}' \mid G \neq G', \mathcal{S}(G, G') = s(c', G)\} \end{aligned} \tag{1}$$

The previous definition supposes a function $\mathcal{S}(G, G')$, which measures the similarity between the graph G and its counterfactual G'. Eq. 1 returns a set of minimally changed counterfactuals w.r.t. the input graph G for each class $c' \in C - \{c\}$ where $c = \Phi(G)$. Notice that adapting this equation to the binary scenario is straightforward.:

$$\mathcal{E}_\Phi(G) = \underset{G' \in \mathcal{G}', G \neq G', \Phi(G) \neq \Phi(G')}{\arg\max} S(G, G') \tag{2}$$

Notice that Eq. 2 produces a single[5] counterfactual instance G' that is the most similar to G. The "search" for the counterfactuals is conditioned such that the returned instance G' is different[6] from G.

The similarity function between two graphs, G and G', can consider the vertex attributes, edge features, and graph structure. Without loss of generality, G can be embedded [4] into a latent space (e.g., via graph convolutional layers). Then, this embedding can be exploited to calculate the similarity (e.g., cosine similarity) with G' that has been embedded into the same latent space. Notice that the choice of the similarity function induces the production of the counterfactual explanations, leading us to believe that a more principled way of generating graph counterfactuals is needed for future research.

4 G-CounteRGAN

Here, we discuss the theoretical novelty we introduced to adapt the original method to propose G-CounteRGAN. CounteRGAN [19] uses a GAN with residual connections [48] and a prediction model - hereafter oracle - to produce meaningful counterfactuals. CounteRGAN was originally proposed for generating grey-scale image counterfactuals. However, we adapted it for graphs and named it G-CounteRGAN. Notice that a graph can be transformed into an adjacency tensor $A \in \mathbb{R}^{|V| \times |V| \times d}$. We assume each edge can have an associated d-dimensional vector of (normalised) weights. More formally, let $F(v_i, v_j) \in \mathbb{R}^d$ denote the weight vector of edge $e = (v_i, v_j)$. Then, the adjacency tensor corresponding to graph $G = (V, E)$ is built as follows.:

$$A[i, j] = \begin{cases} F(v_i, v_j) & \text{if } e = (v_i, v_j) \in E \\ \vec{0}^{\,d} & \text{otherwise} \end{cases}$$

[5] Notice that multiple counterfactuals can maximise $\mathcal{S}$ w.r.t. G. In these cases, ties are broken accordingly, and a single G' is returned.

[6] DDBS and OBS [1] return the original instance when their search heuristics fail to produce a counterfactual. This paper aims to avoid this behaviour.

The adjacency tensor A can be seen as an image with d channels and rely on simple convolutional operations with multiple kernels to extract underlying features. Throughout the remainder of the paper, we denote with $\mathcal{A} = \{A_1, \ldots, A_n\}$ the dataset containing adjacency tensors of the original graphs; G, now, represents a generator network, and D is a discriminator network.

A residual GAN generates residuals rather than complete synthetic data. Given a generator network G and a discriminator network D, G and D are trained in a min-max game where G seeks to maximise and D minimises the following objective function:

$$\mathcal{L}_{\text{RGAN}}(G, D) = \mathop{\mathbb{E}}_{A \sim p_{data}} \big[\log D(A)\big] + \mathop{\mathbb{E}}_{A_z \sim p_z} \big[\log(1 - D(A_z + G(A_z)))\big] \tag{3}$$

where G's input $A_z \in \mathbb{Z}$ is a latent variable sampled from a probability distribution p_z. Notice that, unlike vanilla GANs, the input to RGAN's discriminator is $A_z + G(A_z)$ instead of $G(A_z)$ only. Notice that the RGAN restricts the latent space of the generator to be the same as the input space, which alleviates the mode collapse phenomenon that vanilla GANs suffer from.

G-CounteRGAN exploits a pre-trained oracle Φ to produce meaningful counterfactuals for a specific class c. If we can access Φ's gradients, G-CounteRGAN optimises the following objective function:

$$\mathcal{L}_{\text{G-CounteRGAN}}(G, D) = \mathcal{L}_{\text{RGAN}}(G, D) + \mathcal{L}_{\Phi}(G, c) + Reg(G, A) \tag{4}$$

where c is the class to explain, and $Reg(G, A)$ is a regularisation term that controls the sparsity of the residuals (i.e., feature perturbations). Notice that, in this scenario, to make the generator produce counterfactuals, Eq. 3 is modified such that the generator takes in original input data and not noise:

$$\mathcal{L}_{\text{RGAN}}(G, A) = \mathop{\mathbb{E}}_{A \sim p_{data}} \log D(A) + \mathop{\mathbb{E}}_{A \sim p_{data}} \log \Big(1 - D(A + G(A))\Big) \tag{5}$$

Since sampling instances from the data distribution might induce G to generate null residuals, $\mathcal{L}_{\Phi}(G, c)$ steers the generator away from this behaviour, making it produce plausible counterfactuals. In other words, because the oracle Φ is capable of classifying a generated instance $A + G(A)$ into the explainee class c or something different, i.e., $\neg c$, the generator G is induced to learn to generate examples that are valid counterfactuals, hence belonging to $\neg c$. In this way, Φ plays a crucial role in guiding the learning of the generator according to the following objective:

$$\mathcal{L}_{\Phi}(G, c) = \mathop{\mathbb{E}}_{A \sim p_{data}} \log \Big(\mathbb{1}[\Phi(A + G(A)) \neq c]\Big) \tag{6}$$

where $\mathbb{1}[\Phi(A+G(A)) \neq c]$ is an indicator function that returns 1 if the generator has produced a valid counterfactual w.r.t. class c; 0 otherwise.

Notice that Φ is a black-box model we try to explain. Since we cannot access Φ's gradients, we cannot optimise $\mathcal{L}_{\Phi}(G, c)$. Instead, we weigh the first term of

$\mathcal{L}_{\text{RGAN}}$ by the prediction scores of Φ. Hence, Eq. 4 is modified as follows:

$$\mathcal{L}_{\text{G-CounteRGAN}}(G,D) = \frac{\sum_{A\in\mathcal{A}}\left(\mathbb{1}[\Phi(A)=c]\cdot\log D(A)\right)}{\sum_{A\in\mathcal{A}}\mathbb{1}[\Phi(A)=c]} + \frac{1}{|\mathcal{A}|}\sum_{A\in\mathcal{A}}\log(1-D(A+G(A))) + Reg(G,A) \tag{7}$$

where $\mathbb{1}[\Phi(A)=c]$ is an indicator function that returns 1 if Φ correctly classifies the input adjacency matrix A into the explainee class c.

At inference, drawing inspiration from the sampling technique proposed in CLEAR [17], counterfactuals are generated by selecting edges based on the edge probabilities learned by the generator in the latent space. It is worth noting that the sampling procedure maintains the node order due to our consideration of adjacency tensors instead of graphs. This deliberate choice aids in circumventing the graph-matching procedure, which significantly impacts the execution time of CLEAR (ref. Sec. 5). Notice that one can repeat the sampling procedure of the generator's latent space multiple times to engender several plausible counterfactuals. Among them, one could choose the one with the lowest Graph Edit Distance (GED) to indicate the smallest number of perturbation steps from the original graph to arrive to this particular counterfactual instance (see Sec. 5.2 and 5.3). However, sorting and selection criteria of the counterfactuals can be user-specified according to the application scenario.

We train the generator to exclusively accept instances belonging to the class we intend to explain[7]. Conversely, instances of the other classes are assigned as real instances for the discriminator. Consequently, by training the discriminator to distinguish between fake data (the generated counterfactuals) and real data (those corresponding to the true counterfactual classes), the generator learns to produce counterfactual instances conditioned on the instances from the explainee class.

Fig. 2 provides a visual representation of Eq. 7, which illustrates the training process of G-CounteRGAN[8]. For comprehensiveness, we also demonstrate how the counterfactuals are generated during the inference phase through a single forward step. It is important to note that the sampling procedure for the residuals enables us to generate multiple counterfactuals for a given input adjacency tensor.

5 G-CounteRGAN's performance analysis

Here, we assess the performances of generative approaches against the other SoA methods. First, we describe the adopted benchmarking datasets providing the

[7] This is an architectural drawback since one needs to train $|C|$ models (generators) to explain all classes. At inference, we need to switch between the pre-trained generators and access the one that explains a graph G of class $\Phi(G)$ to generate counterfactuals.

[8] The implementation can be found at `https://github.com/MarioTheOne/GRETEL`.

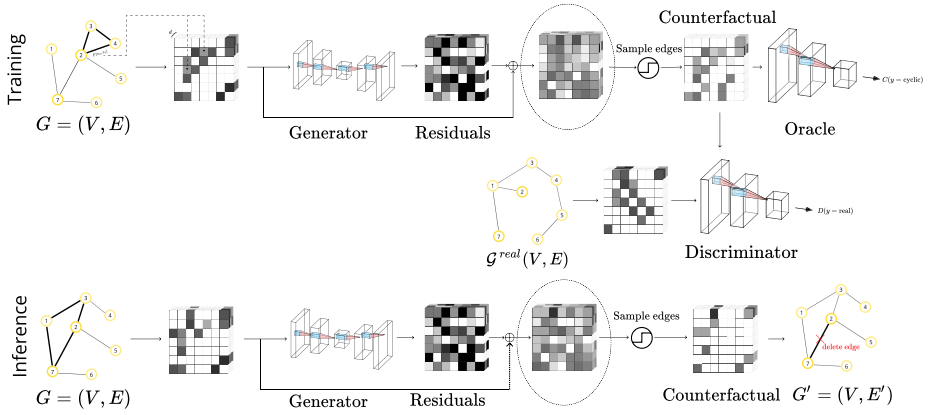

Fig. 2. G-CounteRGAN's workflow during training (up) and inference (down). The weighted adjacency tensor $A \in \mathbb{R}^{|V|\times|V|\times d}$ is transformed and fed to the generator, which produces counterfactual residuals. The discriminator is trained on both real and generated counterfactuals. The generator explains input instances at inference time by sampling edges according to a learned probabilistic distribution of its latent space.

details on their generation process. Then, we describe the evaluation metrics and the hyperparameters used to run each method (see Sec. 5.2). Finally, in Sec. 5.3, we provide a detailed discussion on the performances of the compared methods.

5.1 Benchmarking Datasets

As noticed in [25], comparing SoA methods is a cumbersome task due to the heterogeneity of datasets in the literature. Hence, we perform tests on three synthetic datasets generated according to [26,45]. We use synthetic datasets since they are structured and generated through a specific procedure, making them easier to control and modify. This paper uses datasets that distinguish between cyclic and acyclic (trees) graphs. Ying et al. [45] propose three simple steps to generate this type of dataset:

1. Generate a base graph with specific characteristics, including the vertex and edge number.
2. Generate well-known motifs or use handcrafted ones, ensuring the base graph still needs to contain the motif to be added.
3. Connect the chosen motifs to the base graph while controlling for additional similar motifs.

The resulting dataset can have two classes, graphs generated solely by the first step labelled 0 and those following all three steps labelled 1. Fig. 3 shows two instances from the two classes and highlights the motif added to the instance on the left to produce that on the right. We report the characteristics of the three datasets in Table 1. Notice that, due to the original (de)convolution operations of

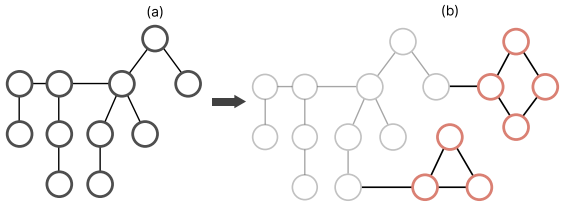

Fig. 3. Example of Tree-Cycles dataset instances. Instance (a) belongs to class 0; meanwhile, instance (b) belongs to class 1. The highlighted motif is a cyclic graph with a user-defined maximum number of vertices (here 4).

G-CounteRGAN's generator, the input adjacency tensor $A \in \mathbb{R}^{|V|\times|V|\times d}$ must satisfy $|V| \equiv 0 \pmod 4$ which is a substantial short-coming considering the unconstrained aspects of real-world graphs.

Table 1. The dataset characteristics. $|\mathcal{G}|$ is the number of instances; $\mu(|V|)$ and $\sigma(|V|)$ represent the mean and std of the number of vertices per instance; $\mu(|E|)$ and $\sigma(|E|)$ represent the mean and std of the number of edges per instance; $|C_i|$ is the number of instances in class $i \in \{0, 1\}$. |Test set| represents the instances evaluated in each fold.

	$\lvert\mathcal{G}\rvert$	$\mu(\lvert V\rvert)$	$\sigma(\lvert V\rvert)$	$\mu(\lvert E\rvert)$	$\sigma(\lvert E\rvert)$	$\lvert C_0\rvert$	$\lvert C_1\rvert$	Class distr.	|Test set|
Tree-Cycles@28	500	28	0	27.566	0.621	252	248	0.504 : 0.496	50
Tree-Cycles@32	500	32	0	31.542	0.620	263	237	0.526 : 0.474	50
Tree-Cycles@48	500	48	0	47.568	0.627	253	247	0.506 : 0.494	50

5.2 Evaluation metrics and hyperparameter choice

We follow the suggestion in [25] to evaluate each method and use multiple metrics to show a complete and fair assessment. To this end, we exploit the following evaluation metrics:

- *Runtime* measures the time the explainer takes to produce a counterfactual. We perform this measure on the same hardware and software platform for an impartial comparison among the methods.
- *Graph Edit Distance (GED)* quantifies the structural distance between the graph G and its counterfactual G'. More formally, given a set of actions (vertex or edge addition/removals) $\{p_1, \ldots, p_n\} \in \mathcal{P}(G, G')$, that depict the path to transform G into G', with a corresponding cost $w(p_i)$, then $GED = \min_{\{p_1,\ldots,p_n\}\in\mathcal{P}(G,G')} \sum_{i=1}^{n} w(p_i)$.
- *Oracle calls* [1] quantifies the times the explainer asks the oracle to produce a counterfactual.
- *Oracle accuracy* evaluates the oracle's performances in predicting outcomes. For a given graph G and its true label y_G, it is equal to $\chi(G) = \mathbb{1}[\Phi(G) = y_G]$.

Algorithm 1 `iRand`: Iteratively permute edges to produce a counterfactual.

Require: $G = (V, E)$, $\alpha \in (0, 1)$, Φ, T $\in \mathbb{N}_+$
$K \leftarrow \lfloor \alpha \cdot |E| \rfloor$
for $i = 1$ **to** K **do**
  for $t = 1$ **to** T **do** ▷ *Try generating the counterfactual at most T times*
    $E' \leftarrow E$
    $E_i \leftarrow \text{sample}(V \times V, i)$ ▷ *Uniformly sample i pairs of vertices*
    for $(v_i, v_j) \in E_i$ **do** ▷ *Perform edge addition/removal operations*
      if $(v_i, v_j) \in E$ **then**
        $E' \leftarrow E' - \{(v_i, v_j)\}$ ▷ *Remove the edge if it exists*
      else
        $E' \leftarrow E' \cup \{(v_i, v_j)\}$ ▷ *Add it if it does not exist*
      end if
    end for
    $G' = (V, E')$
    if $\Phi(G') \neq \Phi(G)$ **then** ▷ *Valid counterfactual found*
      return G'
    end if
  end for
end for
return G ▷ *If everything fails, return the original G*

Table 2. Hyperparameters of the methods for each dataset.

	Methods			
	iRand	CF^2	CLEAR	CounteRGAN
TreeCycles	$\alpha = 0.15$	$\alpha = 0.6$	num_labels=2	batch_size = 25
	$T = 10$	$\lambda = 500$	epochs = 500	iterations = 250
		epochs = 100		discriminator_steps=3
		lr = 10^4		generator_steps=2
		batch_size = 25		num_labels=2
		features = 8		threshold = 0.5

- *Correctness* [9,26] verifies whether the explainer can produce a valid counterfactual. It is defined as $\mathbb{1}[\Phi(G) \neq \Phi(G')]$.
- *Sparsity* [46] measures the similarity between the input graph and its counterfactual according to the input attributes. We rely on sparsity's definition of [25] as $\frac{\mathcal{D}(G,G')}{|G|}$ where $\mathcal{D}(\cdot,\cdot)$ measures the distance between its input parameters. Notice that G and G' have vertex, edge, and graph attributes, which are taken into account by the distance function.
- *Fidelity* [46] measures how faithful the counterfactuals are to the oracle considering their correctness. It is defined as $\chi(G) - \mathbb{1}[\Phi(G') = y_G]$.

We compare two generative approaches (i.e., G-CounteRGAN and CLEAR [17]) with two SoA methods (i.e., OBS [1] and CF^2 [33]) and two baselines methods (i.e., DCE and iRand). DCE [7], first used as a baseline in [1], searches for a counterfactual G^* such that $G^* = \arg\max_{G' \in \mathcal{G}', \Phi(G) \neq \Phi(G')} \mathcal{S}(G, G')$. Moreover, we propose an additional baseline, iRand, to verify whether the learning-based

Table 3. Performances of G-CounteRGAN and SoTA methods on all datasets. † symbolises learning-based approaches; ‡ indicates generative approaches; ∗ depicts search (heuristic) methods. Bold values are the best overall; underlined are second-best on average per dataset.

Dataset	Method	Runtime ↓	GED ↓	Oracle calls ↓	Correctness ↑	Sparsity ↓	Fidelity ↑	Oracle accuracy ↑
Tree-Cycles@28	DCE ∗	0.125	42.570	501.000	**1.000**	0.766	**1.000**	1.000
	OBS ∗	**0.067**	49.444	139.545	0.965	0.889	0.965	1.000
	iRand ∗	0.218	**0.588**	484.599	0.569	**0.011**	0.569	1.000
	CF^2 †	0.581	27.566	**0.000**	0.496	0.496	0.496	1.000
	CLEAR ‡	22.121	64.006	**0.000**	0.504	1.152	0.504	1.000
	G-CounteRGAN ‡	4.120	271.822	**0.000**	0.524	4.893	0.524	1.000
Tree-Cycles@32	DCE ∗	**0.143**	50.112	501.000	**1.000**	0.788	**1.000**	1.000
	OBS ∗	**0.143**	57.542	159.260	0.964	0.905	0.964	1.000
	iRand ∗	0.339	**0.590**	627.342	0.575	**0.009**	0.575	1.000
	CF^2 †	0.412	31.542	**0.000**	0.474	0.496	0.474	1.000
	CLEAR ‡	30.227	80.351	**0.000**	0.526	1.265	0.526	1.000
	G-CounteRGAN ‡	0.298	359.698	**0.000**	0.504	5.659	0.504	1.000
Tree-Cycles@48	DCE ∗	0.214	82.000	501.000	**1.000**	0.858	**1.000**	1.000
	OBS ∗	**0.147**	89.268	237.678	0.935	0.934	0.935	1.000
	iRand ∗	1.627	**0.533**	1594.374	0.527	**0.006**	0.527	1.000
	CF^2 †	3.771	47.568	**0.000**	0.494	0.498	0.494	1.000
	CLEAR ‡	31.081	171.983	**0.000**	0.506	1.800	0.506	1.000
	G-CounteRGAN ‡	6.612	1121.550	**0.000**	0.506	117.737	0.506	1.000

(generative) methods generalise better than a random explainer. In detail, iRand is an iterative approach that works as depicted by the Algorithm 1. At each iteration, it adds/removes i edges from the input graph sampled from the Cartesian product of its vertex set V. If the current iteration produces a valid counterfactual, the procedure returns it. If the "search" for the counterfactual does satisfy the condition in line 14 for T attempts, then iRand defaults to the input graph G similarly to [1]. We expect this baseline performs a smaller number of changes (perturbations) on the synthetic dataset since, intuitively, transforming a tree into a graph requires the addition of a single edge (see Sec. 5.3). Lastly, Table 2 shows the hyperparameters used for those methods that incorporate them in their explainability strategy.

5.3 Discussion and performance analysis

Table 3 depicts the performance of SoA methods on the test set (i.e., 10% of $|\mathcal{G}|$). For each method, we report averages on 10-fold cross-validation. Notice that all methods share the same folds[9] and the same portion of the train-test sets on each fold. Oracles have been pre-trained on the entire dataset to mimic the black-box behaviour where the explainer has no clue how the oracle was trained. Hence, we report the same oracle accuracy for all explainers in each fold. For the Tree-Cycles, we also can rely on an omniscient oracle that performs a graph visit and verifies whether a vertex is visited twice, indicating the presence of a cycle. This oracle never fails to identify trees and graphs, thus reaching perfect accuracy. Excluding the oracles' performances allows the reader to understand

[9] The folds contain the same instances.

each explainer's limitations and benefits better. Having defined the same view of data and underlying oracle, we can reasonably compare all explainers. Additionally, trained methods (i.e., CLEAR, CF^2, G-CounteRGAN) do not need to access the oracle at inference (test) time. However, to be consistent with other methods, we report the number of oracles calls at training time for them.

In Tree-Cycles, all methods exhibit the same correctness and fidelity. Recall that fidelity measures the faithfulness of counterfactual explanations to the oracle's predictions. Since the oracle always predicts the correct class, fidelity consistently aligns with the oracle's prediction (yielding a value of 1 for $\chi(G)$). The $\mathbb{1}[\Phi(G') = y_G]$ component depends on the explainer's ability to generate valid counterfactuals. In this scenario, misclassifications can be attributed to the explainer's limitations in producing valid counterfactuals due to the oracle's infallibility.

DCE has the highest correctness across the board for all Tree-Cycles dataset variants since it guarantees finding a valid counterfactual. Due to its inherent searching mechanism, DCE might suffer from higher GED scores. As expected, iRand has correctness slightly above the chance level, constituting the baseline for other learning methods that systematically fail to surpass. Additionally, it has the best GED. We argue that one needs to add a single edge to produce a counterfactual for a tree (i.e., a graph). Similarly, to produce a counterfactual for a graph (i.e., a tree), one must remove at most $|E| - |V| + 1$ edges. Nevertheless, we noticed that iRand is bad at breaking cycles, thus defaulting to return the original instance (line 20 of Algorithm 1). As per how GED is defined, if the "counterfactual" returned is the original instance, then GED is equal to 0, obscuring the real performances. CF^2 is the second best in terms of GED, but it is the worse w.r.t. the other learning methods in terms of correctness (i.e., ~ 0.488 on average). The low GED of CF^2 results from its factual-based reasoning to generate counterfactuals. In other words, CF^2 first finds the factual graph and then produces the counterfactual as the remainder (i.e., without the factual) on the original input. The two generative methods, CLEAR and G-CounteRGAN, have the worst GED across the board, respectively, at the penultimate and last place. For G-CounteRGAN, because it does not support vertex additions/removals, we found that the model can produce valid counterfactuals for a given tree in input by simply connecting each vertex with another one, thus producing a complete graph. Contrarily, G-CounteRGAN cannot cut down edges when the input is a cyclic graph. Due to this shortcoming, the correctness of the chance level is justified (i.e., half of the time, the explainer is right at generating a valid counterfactual). CLEAR, on the other hand, has better GED, correctness, and sparsity than G-CounteRGAN. We believe a GAN approach requires far more epochs to perform equally well as a VAE (see Table 2). However, CLEAR has the highest runtime across the board due to the graph-matching phenomenon after sampling from the latent space. CF^2 performs 4.5k oracle calls, CLEAR 225k, and G-CounteRGAN 337.5k.

Table 3 delineates that learning-based explainers cannot compete with a simple search (heuristic) approaches to produce valid counterfactuals. This lack of

performance might undermine the usefulness of generative approaches in explainability, especially in critical domains. However, we argue that optimising these models and training them for longer times would result in improved performances [13]. Generative counterfactual models possess enormous potential for future research since one can generate multiple counterfactual examples by sampling the generator's latent space [17,19]. We believe that more graph-suitable convolution operations on G-CounteRGAN (e.g., GCNs) would improve the performances of the overall architecture since they integrate message-passing mechanisms within the neighbourhood of a particular vertex and do not view the graph as a flattened structure as is the case with mere 2D convolutional operations.

Finally, hereafter, we discuss anecdotally the counterfactual graphs gener-

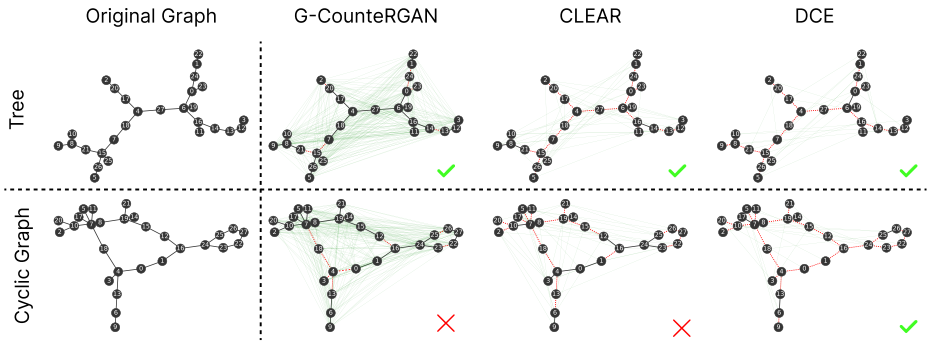

Fig. 4. Counterfactual produced by G-CounteRGAN, CLEAR, and the optimistic baseline DCE on Tree-Cycles@28. As in the original graph, green edges are additions, red ones are removals, and black ones are maintained. An × denotes an invalid counterfactual, and a ✓is valid.

ated by the compared methods. Fig. 4 illustrates three methods that generate counterfactuals for both classes (i.e., tree and cyclic graph). We select the counterfactual with the lowest GED from the original graph for G-CounteRGAN and qualitatively assess how the other two methods cope with the same instance. Notice that G-CounteRGAN (in the tree-to-graph scenario) learns to produce an almost complete graph which is correct but very far away[10] from the original graph, confirming our intuition about its high GED scores. In the other scenario - i.e., graph-to-tree - both generative methods fail (see green edges) to produce a valid counterfactual; hence, their chance-level correctness. DCE never fails to produce a valid counterfactual (see Table 3), although the counterfactuals may seem cluttered visually. We notice that generating a counterfactual for a cyclic graph is a difficult task since the explainer needs to understand what are the edges that need to be removed such that the cycles are broken and what edges

[10] One needs a single edge between any pair of vertices (v_i, v_j) in a tree $G = (V, E)$, s.t. $e = (v_i, v_j) \notin E$, to form a cycle, hence producing a valid counterfactual.

can be safely added without introducing new cycles, provided that the method does not support vertex additions/removals. Despite its apparent simplicity, this synthetic case represents a hard and prototypical problem that must be included in all future investigations of the GCE research field.

6 Conclusion

We defined the GCE problem and discussed the latest developments in the field. Specifically, we thoroughly analysed the effectiveness of current generative models applied to GCE generation. We detailed the implementation of G-CounteRGAN, an adaptation of the CounteRGAN explainer to the graph domain, by considering the adjacency tensor as a special grey-scale image with d channels. By exploiting the learned latent space of the generator, we can produce multiple counterfactual explanations by sampling the edge probability encoded therein. To assess the effectiveness of this approach, we compared it against another generative explainer and other heuristic and search-based GCE methods. Our quantitative and qualitative analysis revealed that most learning-based explainers need help understanding the nature of the problem in our test dataset. Lastly, the integration of graph-based convolution operations could improve the performances of G-CounteRGAN, leading to the rise of a potential new field in GCE, that of Generative GCE (i.e., GenGCE).

Acknowledgement

This work is partially supported by European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics - Prot. IR0000013 - Avviso n. 3264 del 28/12/2021, XAI: Science and technology for the eXplanation of AI decision - ERC Advanced Grant 2018 G.A. 834756 and by the HPC & Big Data Laboratory of DISIM, University of L'Aquila (`https://www.disim.univaq.it/`).

References

1. Abrate, C., Bonchi, F.: Counterfactual graphs for explainable classification of brain networks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 2495–2504 (2021)
2. Aragona, D., Podo, L., Prenkaj, B., Velardi, P.: Coronna: a deep sequential framework to predict epidemic spread. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. pp. 10–17 (2021)
3. Bajaj, M., Chu, L., Xue, Z.Y., Pei, J., Wang, L., Lam, P.C.H., Zhang, Y.: Robust counterfactual explanations on graph neural networks. Advances in Neural Information Processing Systems **34**, 5644–5655 (2021)
4. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE transactions on knowledge and data engineering **30**(9), 1616–1637 (2018)

5. Dabkowski, P., Gal, Y.: Real time image saliency for black box classifiers. Advances in neural information processing systems **30** (2017)
6. Ding, M., Yang, K., Yeung, D.Y., Pong, T.C.: Effective feature learning with unsupervised learning for improving the predictive models in massive open online courses. In: Proceedings of the 9th international conference on learning analytics & knowledge. pp. 135–144 (2019)
7. Faber, L., Moghaddam, A.K., Wattenhofer, R.: Contrastive graph neural network explanation. In: Proc. of the 37th Graph Repr. Learning and Beyond Workshop at ICML 2020. p. 28. Int. Conf. on Machine Learning (2020)
8. Feng, W., Tang, J., Liu, T.X.: Understanding dropouts in moocs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 517–524 (2019)
9. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery pp. 1–55 (2022)
10. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 1–42 (2018)
11. Huang, K., Xiao, C., Glass, L.M., Zitnik, M., Sun, J.: Skipgnn: predicting molecular interactions with skip-graph networks. Scientific reports **10**(1), 1–16 (2020)
12. Huang, Z., Kosan, M., Medya, S., Ranu, S., Singh, A.: Global counterfactual explainer for graph neural networks. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. pp. 141–149 (2023)
13. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16. pp. 491–507. Springer (2020)
14. Liu, Y., Chen, C., Liu, Y., Zhang, X., Xie, S.: Multi-objective explanations of gnn predictions. In: 2021 IEEE International Conference on Data Mining (ICDM). pp. 409–418. IEEE (2021)
15. Livi, L., Rizzi, A.: The graph matching problem. Pattern Analysis and Applications **16**, 253–283 (2013)
16. Loyola-González, O.: Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. IEEE Access **7**, 154096–154113 (2019)
17. Ma, J., Guo, R., Mishra, S., Zhang, A., Li, J.: CLEAR: generative counterfactual explanations on graphs. In: NeurIPS (2022), http://papers.nips.cc/paper_files/paper/2022/hash/a69d7f3a1340d55c720e572742439eaf-Abstract-Conference.html
18. Madeddu, L., Stilo, G., Velardi, P.: A feature-learning-based method for the disease-gene prediction problem. International Journal of Data Mining and Bioinformatics **24**(1), 16–37 (2020)
19. Nemirovsky, D., Thiebaut, N., Xu, Y., Gupta, A.: Countergan: Generating counterfactuals for real-time recourse and interpretability using residual gans. In: Cussens, J., Zhang, K. (eds.) Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands. Proceedings of Machine Learning Research, vol. 180, pp. 1488–1497. PMLR (2022), https://proceedings.mlr.press/v180/nemirovsky22a.html
20. Nguyen, T.M., Quinn, T.P., Nguyen, T., Tran, T.: Explaining black box drug target prediction through model agnostic counterfactual samples. IEEE/ACM Transactions on Computational Biology and Bioinformatics (2022)

21. Numeroso, D., Bacciu, D.: Meg: Generating molecular counterfactual explanations for deep graph networks. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2021)
22. Petch, J., Di, S., Nelson, W.: Opening the black box: the promise and limitations of explainable machine learning in cardiology. Canadian Journal of Cardiology (2021)
23. Prado-Romero, M.A., Prenkaj, B., Stilo, G.: Developing and evaluating graph counterfactual explanation with gretel. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. pp. 1180–1183 (2023)
24. Prado-Romero, M.A., Prenkaj, B., Stilo, G.: Revisiting countergan for counterfactual explainability of graphs. In: Maughan, K., Liu, R., Burns, T.F. (eds.) The First Tiny Papers Track at ICLR 2023, Tiny Papers @ ICLR 2023, Kigali, Rwanda, May 5, 2023. OpenReview.net (2023), https://openreview.net/pdf?id=d0m0Rl15q3g
25. Prado-Romero, M.A., Prenkaj, B., Stilo, G., Giannotti, F.: A survey on graph counterfactual explanations: Definitions, methods, evaluation, and research challenges. ACM Comput. Surv. (2023). https://doi.org/10.1145/3618105, https://doi.org/10.1145/3618105
26. Prado-Romero, M.A., Stilo, G.: Gretel: Graph counterfactual explanation evaluation framework. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 4389–4393 (2022)
27. Prenkaj, B., Distante, D., Faralli, S., Velardi, P.: Hidden space deep sequential risk prediction on student trajectories. Future Generation Computer Systems **125**, 532–543 (2021)
28. Prenkaj, B., Aragona, D., Flaborea, A., Galasso, F., Gravina, S., Podo, L., Reda, E., Velardi, P.: A self-supervised algorithm to detect signs of social isolation in the elderly from daily activity sequences. Artificial Intelligence in Medicine **135**, 102454 (2023)
29. Prenkaj, B., Velardi, P., Distante, D., Faralli, S.: A reproducibility study of deep and surface machine learning methods for human-related trajectory prediction. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 2169–2172 (2020)
30. Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. on Neural Networks **20**(1), 61–80 (2008)
31. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
32. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings (2014), http://arxiv.org/abs/1312.6034
33. Tan, J., Geng, S., Fu, Z., Ge, Y., Xu, S., Li, Y., Zhang, Y.: Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In: Proceedings of the ACM Web Conference 2022. p. 1018–1027. WWW '22, Association for Computing Machinery, New York, NY, USA (2022), https://doi.org/10.1145/3485447.3511948
34. Verenich, I., Dumas, M., La Rosa, M., Nguyen, H.: Predicting process performance: A white-box approach based on process models. Journal of Software: Evolution and Process **31**(6), e2170 (2019)

35. Verma, H., Mandal, S., Gupta, A.: Temporal deep learning architecture for prediction of covid-19 cases in india. Expert Systems with Applications **195**, 116611 (2022)
36. Vermeire, T., Brughmans, D., Goethals, S., de Oliveira, R.M.B., Martens, D.: Explainable image classification with evidence counterfactual. Pattern Analysis and Applications **25**(2), 315–335 (2022)
37. Wang, W., Yu, H., Miao, C.: Deep model for dropout prediction in moocs. In: Proceedings of the 2nd international conference on crowd science and engineering. pp. 26–32 (2017)
38. Wei, X., Liu, Y., Sun, J., Jiang, Y., Tang, Q., Yuan, K.: Dual subgraph-based graph neural network for friendship prediction in location-based social networks. ACM Transactions on Knowledge Discovery from Data (TKDD) (2022)
39. Wellawatte, G.P., Seshadri, A., White, A.D.: Model agnostic generation of counterfactual explanations for molecules. Chemical science **13**(13), 3697–3705 (2022)
40. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 346–353 (2019)
41. Wu, X., Xiong, Y., Zhang, Y., Jiao, Y., Shan, C., Sun, Y., Zhu, Y., Yu, P.S.: Clare: A semi-supervised community detection algorithm. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 2059–2069 (2022)
42. Xu, L., Xi, W., Wang, C.: Session-based recommendation with heterogeneous graph neural networks. In: International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021. pp. 1–8. IEEE (2021). https://doi.org/10.1109/IJCNN52387.2021.9533519, https://doi.org/10.1109/IJCNN52387.2021.9533519
43. Xu, Z., Lamba, H., Ai, Q., Tetreault, J., Jaimes, A.: Counterfactual editing for search result explanation. arXiv preprint arXiv:2301.10389 (2023)
44. Yang, L., Kenny, E., Ng, T.L.J., Yang, Y., Smyth, B., Dong, R.: Generating plausible counterfactual explanations for deep transformers in financial text classification. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 6150–6160. International Committee on Computational Linguistics, Barcelona, Spain (Online) (Dec 2020). https://doi.org/10.18653/v1/2020.coling-main.541, https://aclanthology.org/2020.coling-main.541
45. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. pp. 9240–9251 (2019), https://proceedings.neurips.cc/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html
46. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: A taxonomic survey. IEEE Transactions Pattern Analaysis and Machine Intelligence **45**(5), 5782–5799 (2023). https://doi.org/10.1109/TPAMI.2022.3204236, https://doi.org/10.1109/TPAMI.2022.3204236
47. Zemni, M., Chen, M., Zablocki, E., Ben-Younes, H., Pérez, P., Cord, M.: Octet: Object-aware counterfactual explanations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15062–15071 (June 2023)

48. Zhang, L., Long, C., Zhang, X., Xiao, C.: Ris-gan: Explore residual and illumination with generative adversarial networks for shadow removal. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 12829–12836 (2020)