Robust Stochastic Graph Generator for Counterfactual Explanations

Mario Alfonso Prado-Romero[*1], **Bardh Prenkaj**[*2], **Giovanni Stilo**[3]

[1] Gran Sasso Science Institute
[2] Sapienza University of Rome
[3] University of L'Aquila
marioalfonso.prado@gssi.it, prenkaj@di.uniroma1.it, giovanni.stilo@univaq.it

Abstract

Counterfactual Explanation (CE) techniques have garnered attention as a means to provide insights to the users engaging with AI systems. While extensively researched in domains such as medical imaging and autonomous vehicles, Graph Counterfactual Explanation (GCE) methods have been comparatively under-explored. GCEs generate a new graph similar to the original one, with a different outcome grounded on the underlying predictive model. Among these GCE techniques, those rooted in generative mechanisms have received relatively limited investigation despite demonstrating impressive accomplishments in other domains, such as artistic styles and natural language modelling. The preference for generative explainers stems from their capacity to generate counterfactual instances during inference, leveraging autonomously acquired perturbations of the input graph. Motivated by the rationales above, our study introduces RSGG-CE, a novel **R**obust **S**tochastic **G**raph **G**enerator for **C**ounterfactual **E**xplanations able to produce counterfactual examples from the learned latent space considering a partially ordered generation sequence. Furthermore, we undertake quantitative and qualitative analyses to compare RSGG-CE's performance against SoA generative explainers, highlighting its increased ability to engendering plausible counterfactual candidates.

Introduction

Explainability is crucial in sensitive domains to enable users and service providers to make informed and reliable decisions (Guidotti et al. 2018). However, deep neural networks, commonly used for generating predictions, often suffer from a lack of interpretability, widely referred to as the *black-box* problem (Petch, Di, and Nelson 2021), 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 (Loyola-González 2019), which are preferred for decision-making purposes (Verenich et al. 2019). Alas, black-box models demonstrate superior performance and generalisation capabilities when dealing with high-dimensional data (Aragona et al. 2021; Ding et al. 2019; Feng, Tang, and Liu 2019; Huang et al. 2020; Madeddu, Stilo, and Velardi 2020; Prenkaj et al. 2021, 2020, 2023a; Verma, Mandal, and Gupta 2022; Wang, Yu, and Miao 2017).

Recently, deep learning (relying on GNNs (Scarselli et al. 2008)) has been beneficial in solving graph-based prediction tasks, such as community detection (Wu et al. 2022), link prediction (Wei et al. 2022), and session-based recommendations (Wu et al. 2019; Xu, Xi, and Wang 2021). 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 (Prado-Romero et al. 2023).

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 (Prado-Romero et al. 2023). Search-based approaches find counterfactual examples within the data distribution. Heuristic-based approaches perturb the original graph G into G' such that, for a certain prediction model Φ, namely oracle, $\Phi(G) \neq \Phi(G')$ without accessing the dataset $\mathcal{G}$. In other words, G' can be outside the data distribution of $\mathcal{G}'$. Heuristic-based approaches suffer the need to define the perturbation heuristic (e.g., rules), which might come after careful examination of the data and involve 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 generative "heuristic" based on the data. This kind of explainer is trained on samples and thus can be used to produce counterfactual instances at inference time.

In this work, we propose RSGG-CE, a **R**obust **S**tochastic **G**raph **G**enerator for **C**ounterfactual **E**xplanations, to produce counterfactual examples from the learned latent space considering a partially ordered generation sequence. RSGG-

[*] These authors contributed equally.

CE is not confined to the data distribution (vs. search approaches) and does not rely on a learned mask to apply to the input to produce counterfactuals (vs. other learning approaches). However, it learns the perturbation of the input autonomously (vs. heuristic approaches), relying on a partial order generation strategy. Moreover, it does not need access extensively to the oracle Φ since its latent space can be sampled to generate multiple candidate counterfactuals for a particular input graph. The following discusses the related works in Sec. . In Sec. , we present the method and the needed preliminary knowledge (see Sec.). Finally, in Sec. , we conduct the performance analysis, four ablation studies, and a qualitative anecdotal inspection. We add supplementary material (SM) to this paper available here[1].

Related Work

The literature distinguishes between inherently explainable and black-box methods (Guidotti et al. 2018). Black-box methods can be further categorised into factual and counterfactual explanation methods. Here, we concentrate on counterfactual methods as categorised in (Prado-Romero et al. 2023) and exploit the same notation used in that survey.

While many works provide counterfactual explanations for images/text (to point out some (Vermeire et al. 2022; Xu et al. 2023; Zemni et al. 2023)), only a few focus on graph classification problems (Abrate and Bonchi 2021; Liu et al. 2021; Ma et al. 2022; Nguyen et al. 2022; Numeroso and Bacciu 2021; Tan et al. 2022; Wellawatte, Seshadri, and White 2022). According to (Prado-Romero et al. 2023), 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 (Huang et al. 2023)). Here, we treat only instance-level and learning-based explainers.

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 $\Phi(G')$. Learning-based strategies can be divided into perturbation matrix (Tan et al. 2022), reinforcement learning (Nguyen et al. 2022; Numeroso and Bacciu 2021; Wellawatte, Seshadri, and White 2022), and generative approaches (Ma et al. 2022). After training, the learned latent space of generative approaches can be exploited as a sampling basis to engender plausible counterfactuals. (w.l.o.g.), generative methods learn a latent space that embeds original and non-existing estimated edges' edge probabilities (e.g., see (Ma et al. 2022)). This way, one can employ sampling techniques to produce counterfactual candidates w.r.t. the input instance. In the following, we report the most recent and effective SoA methods.

MEG (Numeroso and Bacciu 2021) and MACCS (Wellawatte, Seshadri, and White 2022) employ multi-objective reinforcement learning (RL) models (retrained for each input instance) to generate molecule counterfactuals. 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. MACDA (Nguyen et al. 2022) uses RL to produce counterfactuals for the drug-target affinity problem.

CF^2 (Tan et al. 2022) balances factual and counterfactual reasoning to generate explanations. Like other factual-based approaches, it identifies a subgraph in the input, then presents the remainder as a counterfactual candidate by removing this subgraph (Bajaj et al. 2021). Notably, CF^2 favours smaller explanations for simplicity.

CLEAR (Ma et al. 2022) uses a variational autoencoder (VAE) to encode the graphs into its latent representation Z. The decoder generates counterfactuals based on Z, conditioned on the explainee class $c \neq \Phi(G)$. Generated counterfactuals are complete graphs with stochastic edge weights. To ensure validity, the authors employ a sampling process. However, decoding introduces node order differences between G and G'. Thus, a graph matching procedure (NP-hard (Livi and Rizzi 2013)) between the two is necessary.

While unrelated to graphs, (Nemirovsky et al. 2022) rely on a GAN. They produce counterfactual candidates by training the generator to elucidate a user-defined class. (Prado-Romero, Prenkaj, and Stilo 2023) adapt this into G-CounteRGAN by treating the adjacency matrix as black-and-white images and employing 2D image convolutions.

What is our contribution to the literature? We design a novel approach to generate counterfactuals by leveraging the latent space of the generator network to reconstruct the input's topology. The discriminator guides this process, which compels the generator to learn the production of counterfactuals aligned with the opposite class.

Firstly, we tackle the limitations of factual-based methods that remove subgraph components to craft counterfactual candidates, a strategy found in (Bajaj et al. 2021; Tan et al. 2022). However, this falters when dual classes clash (e.g., acyclic vs cyclic graphs). In such cases, shifting from cyclic to acyclic mandates edge removal, while acyclic to cyclic requires edge addition, i.e., creating a cyclic graph from acyclic needs an added edge for a loop. Since our method combines the generated residual weighted edges with the original edges, we empower both edge addition and removal operations. Unlike images, graphs lack node ordering, rendering standard 2D convolutions inadequate due to the significance of node adjacency. We integrate Graph Convolution Networks (GCNs) to address this, naturally capturing node neighbourhoods via message-passing mechanisms (Feng et al. 2022). Our approach is zero-shot counterfactual generation. Prevalent techniques, especially those rooted in Reinforcement Learning (RL) (Nguyen et al. 2022; Numeroso and Bacciu 2021; Wellawatte, Seshadri, and White 2022), require recalibration at inference time to generate counterfactuals for previously unseen graphs. Differently, we learn a latent graph representation enabling stochastic estimations of the graph's topology, allowing us to reconstruct and generate counterfactuals without retraining.

[1]https://aiimlab.org/blog/2023/12/19/AAAI_24_Robust_Stochastic_Graph_Generator_for_Counterfactual_Explanations.html

Lastly, our model introduces an innovative strategy - i.e., partial-order sampling - using estimated edge probabilities acquired from the generator network. To the best of our knowledge, this is the first work that proposes a partial-order (Dan and Djeraba 2008) sampling approach on estimated edges. This aids in identifying sets of edges that should be sampled first for effective counterfactual generation (see Sec. and Algorithm 1).

Preliminaries

This section briefly overviews the fundamental concepts and techniques relevant to our study on robust counterfactual explanations. We introduce the concepts of graphs, adjacency matrices, and graph counterfactuals.

Graphs and Adjacency Matrix A graph, denoted as $G = (X, A)$, is a mathematical structure consisting of node features $X \in \mathbb{R}^{n\times d}$ and an adjacency matrix $A \in \mathbb{R}^{n\times n}$ which represents the connectivity between nodes. For an undirected weighted graph, the adjacency matrix A is symmetric, and its elements are defined as

$$A\left[v_i, v_j\right] = \begin{cases} w\left(v_i, v_j\right) & \text{if } \left(v_i, v_j\right) \text{ is an edge in } G \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $w\left(v_i, v_j\right) \in \mathbb{R}$ is the weight vector of the edge incident to the nodes v_i and v_j. For directed graphs, the adjacency matrix may exhibit asymmetry, thereby indicating the directionality of the edges. We focus on undirected graphs and denote the graph dataset with $\mathcal{G} = \{G_1, \ldots, G_N\}$.

Graph Counterfactuals Given a black-box (oracle) predictor $\Phi : G \to Y$, where, w.l.o.g., $Y = \{0, 1\}$, according to (Prado-Romero et al. 2023), a counterfactual for G is defined as

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

where $\mathcal{G}'$ is the set of all possible counterfactuals, and $\mathcal{S}(G, G')$ calculates the similarity between G and G'. (Prenkaj et al. 2023b) reformulate Eq. 2 and take a probabilistic perspective to produce a counterfactual that is quite likely within the distribution of valid counterfactuals by maximising

$$\mathcal{E}_\Phi(G) = \underset{G'\in\mathcal{G}'}{\arg\max}\, P_{cf}\left(G' \mid G, \Phi\left(G\right), \neg\Phi\left(G\right)\right) \tag{3}$$

where $\neg\Phi(G)$ indicates any other[2] class from the one predicted for G. In this work, we find valid counterfactuals by solving a specialisation of Eq. 3.

Method

Here we introduce RSGG-CE, namely **R**obust **S**tochastic **G**raph **G**enerator for **C**ounterfactual **E**xplanations, a supervised explanation method that exploits the generator's learned latent space to sample plausible counterfactuals for a particular explainee graph $G = (X, A)$. RSGG-CE distinguishes between the training and inference phases as depicted in Figure 1. For this reason, we first discuss the training phase and, later, the inference one.

[2]The provided formulation supports multi-class classification problems. For simplicity, we are in binary classification.

Training Let $\mathbb{D}(Y|X, A)$ denote a GCN discriminator that produces $Y = \{0, 1\}$ based on the plausibility assessment of the graph represented by the adjacency matrix A and node features X. Let $\mathbb{G}$ represent a Graph Autoencoder (GAE) (Kipf and Welling 2016) with $\text{ENC}(X, A)$ serving as the representation model for latent node-related interaction components (i.e., encoder), and $\text{DEC}(Z)$ acting as an inner-product decoder. Conversely to the original Residual GAN, (Zhang et al. 2020), where the generator $\mathbb{G}$ is trained on sampled Gaussian noise, we exploit the Residual GAN introduced in (Nemirovsky et al. 2022), where $\mathbb{G}$ is trained on instances sampled from the data distribution and optimised according to Eq. 4:

$$\begin{aligned}\mathcal{L}(\mathbb{D}, \mathbb{G}) = &\underset{(X_i,A_i)\in\mathcal{G}}{\mathbb{E}} \underbrace{\Big[\log \mathbb{D}(Y \mid X_i, A_i)\Big]}_{\text{discriminator optimisation}} \\ + &\underset{\substack{(X_j,A_j)\in\mathcal{G},\\ \hat{X}_j, A_j+\hat{A}_j=\mathbb{G}(X_j,A_j)}}{\mathbb{E}} \underbrace{\Big[\log(1-\mathbb{D}(Y \mid \hat{X}_j, A_j+\hat{A}_j))\Big]}_{\text{generator optimisation}}\end{aligned} \tag{4}$$

where $\mathbb{G}(X, A) = \text{DEC}(\text{ENC}(X, A))$. For completeness purposes, the generator $\mathbb{G}$ returns the reconstructed adjacency tensor $\hat{A}$ and the reconstructed node features $\hat{X}$. Notice that, unlike vanilla GANs, the input to the Residual GAN's discriminator is $A_j + \hat{A}_j$ for a graph $(X_j, A_j) \in \mathcal{G}$. To overcome the constraint of (Nemirovsky et al. 2022), which fixes the latent space of the generator to be of the same size as the input space, we exploit a Graph Autoencoder (GAE) such that it encodes graphs into a point in the latent space $\mathcal{Z}$, and the decoder maps it into a new reconstructed graph belonging to the same space as the input. This also allows the generator to learn relationships between its input and output, enabling fine-grained regularisation of residuals and alleviating mode collapse. Moreover, to support edge additions and removals, we set the activation function of the generator to the hyperbolic tangent and sum it to the input adjacency matrix.

Let $\mathbb{I}[\Phi(X, A) \neq c]$ be an indicator function that returns 1 if $\Phi(X, A) \neq c$ for a graph $G = (X, A)$. Let also be $\mathcal{G}_c = \{(X, A) \mid (X, A) \in \mathcal{G} \wedge \neg\mathbb{I}[\Phi(X, A) \neq c]\}$. Conversely, we indicate with $\mathcal{G}_{\neg c} = \{(X, A) \mid (X, A) \in \mathcal{G} \wedge \mathbb{I}[\Phi(X, A) \neq c]\}$. Because we need to generate counterfactuals for a pretrained black-box oracle Φ in a particular class c, we modify Eq. 4 as follows:

$$\begin{aligned}\mathcal{L}_{\Phi,c}(\mathbb{D}, \mathbb{G}) = &\sum_{(X_r,A_r)\in\mathcal{G}_{\neg c}} \underbrace{\Big(\log \mathbb{D}(Y \mid X_r, A_r)\Big)}_{\text{discriminator optimisation on real data}} \\ + &\sum_{(X_g,A_g)\in\mathbb{G}(\mathcal{G}_c)} \underbrace{\Big(\mathbb{I}[\Phi(X_g, A_g) \neq c] \log \mathbb{D}(Y \mid X_g, A_g)\Big)}_{\text{discriminator optimisation on generated data}} \\ + &\sum_{\substack{(X_j,A_j)\in\mathcal{G}_c,\\ \hat{X}_j, A_j+\hat{A}_j=\mathbb{G}(X_j,A_j)}} \underbrace{\log\Big(1-\mathbb{D}(Y \mid \hat{X}, A_j+\hat{A}_j)\Big)}_{\text{generator optimisation}}\end{aligned} \tag{5}$$

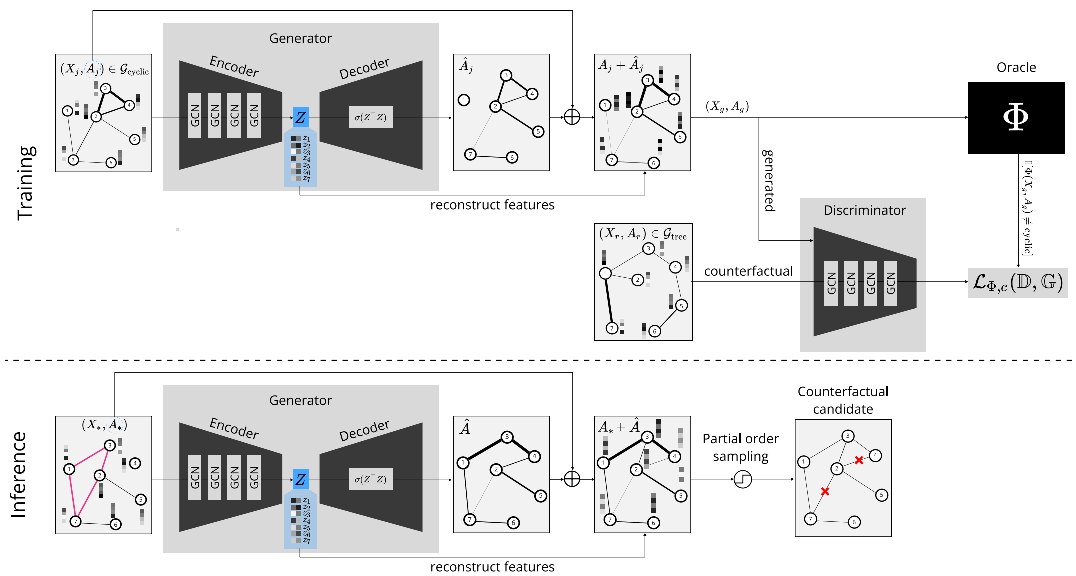

Figure 1: RSGG-CE's workflow during training (up) and inference (down) in a cyclic graph vs tree scenario. $(X_j, A_j) \in \mathcal{G}_{\text{cyclic}}$ is fed to the generator, which produces counterfactual residuals. The discriminator is trained on both real $(X_r, A_r) \in \mathcal{G}_{\text{tree}}$ and generated graphs (X_g, A_g). The generator optimisation is guided via Φ's classification. The generator explains input instances (X_*, A_*) at inference time by sampling edges according to a learned stochastic distribution of its latent space. In this scenario, we highlight the critical edges forming a cycle the generator needs to break so that the returned counterfactual candidate does not contain cycles. The sampled edges entail a counterfactual candidate that Φ needs to validate.

where $\mathbb{G}(\mathcal{G}_c) = \{\mathbb{G}(X, A) \mid (X, A) \in \mathcal{G} \ \land \neg\mathbb{I}[\Phi(X, A) \neq c]\}$ is the set of all the generated graphs. Since sampling instances from the data distribution might induce $\mathbb{G}$ to generate null residuals, integrating the accuracy of correct predictions from Φ - as shown in the second summation of Eq. 5 - steers the generator away from this behaviour, making it produce realistic counterfactuals (Guyomard et al. 2022) w.r.t. to an input graph $G_* = (X_*, A_*)$, i.e.,

$$\underset{\substack{(X,A)\in\mathcal{G},\\ \Phi(X,A)=\Phi(X_*,A_*)}}{\mathbb{E}} \left[\left|\left| \mathbb{G}(X_*, A_*) - (X, A) \right|\right|_2^2 \right] \tag{6}$$

where $G = (X, A)$ is a graph belonging to the same class as G_*. In other words, our approach produces counterfactuals close to the examples the generator has been trained, thus ensuring as little as possible perturbations w.r.t. G_* (see Sec.).

We train the generator exclusively on the graphs from the class we aim to explain. In this way, the graphs from the other classes are assigned as real instances for the discriminator training. Hence, by training the discriminator to differentiate between fake data (generated counterfactuals) and real data (corresponding to true counterfactual classes), the generator learns to produce counterfactuals conditioned on the graphs from the explainee class. For example, suppose we have a dataset containing acyclic (0) and cyclic (1) graphs. If we want to generate counterfactuals for the cyclic class, we feed the generator with cyclic instances. Meanwhile, we feed the discriminator with generated instances (labelled as fake data) and real trees (labelled as real data). Considering that the generator needs to fool the discriminator to maximise its objective function, it will learn how to mutate a cyclic into a plausible acyclic graph (e.g., by removing edges that form the cycles). This way, the generator's latent space can be exploited to generate counterfactual candidates for the input instance.

RSGG-CE can also be adapted for node classification where the dataset, the oracle, and the optimisation function (Eq. 5) must be modified accordingly. We invite the reader to check Sec. G of the SM for further details.

Inference and partial order sampling Differently from (Ma et al. 2022), we stochastically generate counterfactual candidates by sampling edges with partial order guided by the learned probabilities from the generator's latent space. The general sampling procedure is illustrated in Algorithm 1, where we sample the edges according to the order defined by the `partial_order` (line 3) function illustrated in Algorithm 2. We invite the reader to note that this approach is modular and that the function `partial_order(·,·)` can be specialised according to the application domain and prediction scenario. In Algorithm 1 for each edge set $\mathcal{E}$ of each partition group $\mathbb{O}$, we sample each edge according to their estimated probabilities and engender A' (line 7) and assess if it is a valid counterfactual (line 8) based on the verification guard o. As in (Abrate and Bonchi 2021), we default to the original input if we fail to produce a valid counterfactual (line 13). In Algorithm 2, we induce a partial order on the estimated edges, including non-existing ones in the original input. In line 1, we get the edges of the input instance, while in line 2, we get the set of non-existing ones. Afterward, in line 3, we build the corresponding partition groups by setting

Algorithm 1: Partial order sampling to produce a counterfactual.

Require: $G_* = (X_*, A_*), \mathbb{G} : \mathcal{G} \rightarrow \mathcal{G}, \Phi,$

$\hat{X}_*, A_* + \hat{A}_* = \mathbb{G}(X_*, A_*)$
$X_g, A_g \leftarrow \hat{X}_*, A_* + \hat{A}_*$
$\mathcal{P} \leftarrow$ `partial_order`(A_*)
$A' \leftarrow 0^{n \times n}$
for $\mathbb{O} \in \mathcal{P}$ **do**
  for $e = (u, v) \in \mathbb{O}.\mathcal{E}$ **do**
    $A'[u, v] \leftarrow$ `sample`$(e, A_g[u, v])$
    if $\mathbb{O}.o \wedge \Phi(X_g, A') \neq \Phi(X_*, A_*)$ **then**
      return (X_g, A')
    end if
  end for
end for
return (X_*, A_*)

Algorithm 2: Example of `partial_order`

Require: $A \in \mathbb{R}^{n \times n}$

$E \leftarrow$ `positive_edges`(A) ▷ *Get the set of edges from the adjacency matrix A*
$\neg E \leftarrow$ `negative_edges`(A) ▷ *Get the set of non-existing edges from the adjacency matrix A*
$\mathcal{P} \leftarrow \{(\mathcal{E}{=}E, o{=}0), (\mathcal{E}{=}\neg E, o{=}1)\}$ ▷ *Build the partial order of the existing and non-existing edges with group tuples consisting of edge set $\mathcal{E}$, and oracle verification guard o.*
return $\mathcal{P}$

the oracle verification guard $o = 1$ only for the non-existing group. Thus, the oracle will be called only once sampling finishes on the existing edge set.

Experimental Analysis

This section discusses three kinds of analyses[3]. First, we discuss the performances of RSGG-CE w.r.t. other SoA explainers (see Sec. F of the SM for the used evaluation metrics). In the second, we conduct four ablation studies to understand the robustness of RSGG-CE. In the third one, we conduct a qualitative anecdotal inspection. Experimental analysis is done by applying 10-fold cross-validations on real and synthetic datasets. We also analysed RSGG-CE's efficiency and convergence in Sec. E of the SM.

The *Tree-Cycles (TC)* (Ying et al. 2019) is an emblematic synthetic dataset. Each instance constitutes a graph comprising a central tree motif and multiple cycle motifs connected through singular edges. The dataset encompasses two distinct classes: i.e., one for graphs without cycles (0) and another for graphs containing cycles (1). The TC also allows control of the number of nodes, the number of cycles, and the number of nodes in them. For the performance comparison to the other SoA explainers, due to the computational complexity of some of them, we use 500 graphs with 28 nodes and randomly generate up to 3 cycles with varying sizes that go up to 7 nodes. We vary all the parameters for the ablation study as reported in Sec. . The hardness of this dataset depends on changing the oracle prediction; the explainer needs to learn to apply two opposite actions (i.e., remove or add edges to the input to generate acyclic or cyclic counterfactuals, respectively).

The *Autism Spectrum Disorder* (*ASD*) (Abrate and Bonchi 2021) is a real graph classification dataset obtained using functional magnetic resonance imaging (fMRI), where nodes represent brain Regions of Interest (ROI), and edges are co-activation between two ROIs. The two classes belong to individuals with *Autism Spectrum Disorder* (ASD) and *Typically Developed* (TD) individuals as the control group. Here, the graph instances can be disconnected.

In Table 1, we report the performance of RSGG-CE compared to other SoA learning-based explanation methods (i.e., MACCS (Wellawatte, Seshadri, and White 2022), CF^2 (Tan et al. 2022), CLEAR (Ma et al. 2022), and G-CounteRGAN (Prado-Romero, Prenkaj, and Stilo 2023)) for the TC and ASD datasets. Notably, a significant challenge arises when it comes to counterfactual explainability through learning-based explainers due to the inherent reliance on the overarching structure of the graph rather than specific nodes or edges. In this context, RSGG-CE takes the spotlight as an exceptional performer with a gain of 66.98% and 19.65% in Correctness over the second-performing method in TC and ASD, respectively. It outperforms all alternative methods, standing as the sole technique that improves, by a large margin, the Correctness of all the datasets without sacrificing the running time. Moreover, RSGG-CE also showcases superiority in terms of Graph Edit distance (GED) w.r.t. the other explainers. This further underscores RSGG-CE's capabilities in capturing the intricate structures in all the datasets and its wanted ability (see Sec. and Eq. 5) to generate counterfactual instances closer to the input one w.r.t. those generated by other explainers (see also Sec.).

Ablation Experiments

To understand the robustness of RSGG-CE, we conduct four ablation studies using the TC dataset by varying the number of nodes in the cycles, the number of cycles, the number of nodes in the graphs, and the number of instances in the dataset. For all the ablations, we use a dataset with 500 instances. We also include CF^2 since it is the best-performing learning-based explainer after RSGG-CE.

Robustness to the increasing number of nodes per cycle In this study, we fix the number of nodes to 128 and the number of cycles to 4 to assess how the number of nodes (from 3 to 28) in each cycle affects RSGG-CE's GED and Correctness (see Fig. 2). It is interesting to notice that the bigger the cycles become, the better RSGG-CE learns the edge probabilities since now the motifs are more evident. This means that, at inference time, the partial order sampling has higher chances of breaking cycles containing more nodes than those with a small number of them. For instance, if the number of nodes in a cycle is 3, the probability of cutting the

[3]The original code is on https://github.com/MarioTheOne/GRETEL. Newer versions of the project will be released on https://github.com/aiim-research/GRETEL.

		Methods				
		MEG †	CF^2 †	CLEAR ‡	G-CounteRGAN ‡	**RSGG-CE ‡**
TC	Runtime (s) ↓	272.110	<u>4.811</u>	25.151	632.542	**0.083**
	GED ↓	159.700	<u>27.564</u>	61.686	182.414	**11.000**
	Oracle Calls ↓	**0.000**	**0.000**	4341.600	1321.000	<u>121.660</u>
	Correctness ↑	<u>0.530</u>	0.496	0.504	0.504	**0.885**
	Sparsity ↓	2.510	0.496	1.110	3.283	**0.199**
	Fidelity ↑	<u>0.530</u>	0.496	0.504	0.504	**0.885**
	Oracle Acc. ↑	1.000	1.000	1.000	1.000	1.000
ASD	Runtime (s) ↓	×	**15.313**	275.884	969.255	<u>80.000</u>
	GED ↓	×	<u>655.661</u>	1479.114	3183.729	**234.853**
	Oracle Calls ↓	×	**0.000**	5339.455	1182.818	<u>794.805</u>
	Correctness ↑	×	0.463	<u>0.554</u>	0.529	**0.603**
	Sparsity ↓	×	<u>0.850</u>	1.917	4.125	**0.304**
	Fidelity ↑	×	**0.287**	<u>0.319</u>	0.265	**0.287**
	Oracle Acc. ↑	×	0.773	0.773	0.773	0.773

Table 1: Comparison of RSGG-CE with SoA methods. Metrics are reported on 10-fold cross-validations. Bold values are the best overall; underlined are second-best; × represents no convergence; † depicts a learning-based explainer, and ‡ a generative approach. *Correctness* and *GED* are most important metrics.

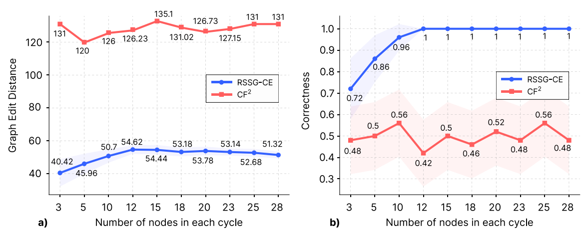

Figure 2: GED ↓ and Correctness ↑ trends when varying the number of nodes in each cycle on TreeCycles with 128 nodes and 4 cycles per instance.

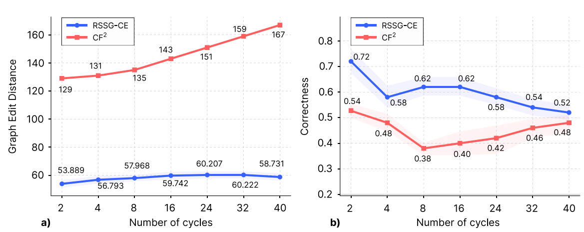

Figure 3: GED ↓ and Correctness ↑ trends when varying the number of cycles per instance on TreeCycles with 128 nodes and 3 nodes per cycle.

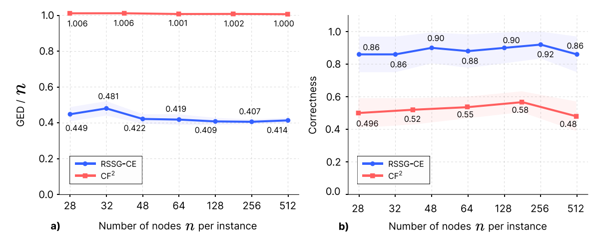

Figure 4: The trend of GED ↓ normalised by the number n of nodes instance, and the trend of Correctness ↑ when n increases.

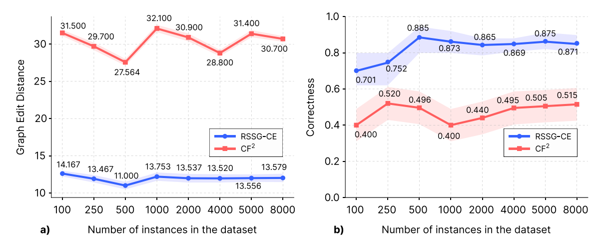

Figure 5: The trend of GED ↓ and Correctness ↑ when varying the number of instances in the dataset.

cycle is $2/3$. Specularly, if the number of nodes in a cycle tends to the number of nodes in the instance, the probability of cutting the cycle is nearer to 1. Therefore, Correctness has an increasing trend. Additionally, while RSGG-CE reaches a correctness of 1 when the number of nodes in cycles increases, it does not sacrifice the recourse cost (GED) needs to engender valid counterfactuals (see Fig. 2.a). Contrarily, since RSGG-CE's partial order sampling favours the existing edges in the original instance, with the increase in cycle sizes, the GED has a non-increasing trend. This is straightforward since finding a valid counterfactual can be done with fewer sampling iterations. For example, if we have a single ring (cyclic graph) per instance, the number of operations to engender a counterfactual is 1. Notice that CF^2 entails a zigzag Correctness trend, leading us to believe it does not scale with the increasing number of nodes in the cycle. This phenomenon is also supported regarding GED since CF^2 converges to a local minimum and tries to engender the same counterfactual regardless of the input instance (notice the standard error equal to zero).

Robustness to the increasing number of cycles In this investigation, we keep the cycle size fixed at 3 nodes and assess the impact of cycle quantity (ranging from 2 to 32) on RSGG-CE's GED and Correctness metrics (Fig. 3). It is interesting to notice that the GED (Fig. 3.a) is linearly

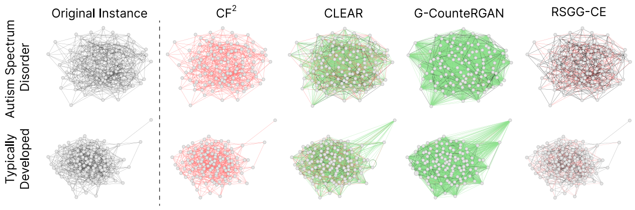

Figure 6: Qualitative comparison (best viewed in colour) of the counterfactuals candidate produced by CF^2, CLEAR, G-CounteRGAN, and RSGG-CE on the ASD real dataset. Concerning the original graph, green edges are additions, red are removals, and grey are maintained.

dependent on the number of cycles, thus reflecting the increased hardness of the problem (i.e., the need to break a higher number of cycles) but without slipping into an exponential trend. Similarly, the Correctness is slightly affected by the number of cycles in the graphs but maintains its linear trend. This plot indicates that RSGG-CE can cope with complex topologies without sacrificing too much in terms of performance. Additionally, CF^2 has an increasing trend in GED w.r.t. the change in the number of cycles also influences the Correctness (e.g., from 2 to 18).

Robustness to the number of nodes In this ablation, for each graph, we randomly generated up to 3 cycles with a varying size that goes up to 7 nodes, and we delve into the impact of graph number of nodes (ranging from 28 to 512) on RSGG-CE's GED and Correctness metrics (refer to Fig. 3). To better catch the general trend, in this case, we reported the recourse cost (GED) normalised by the number of nodes of the graph. Fig. 4.a depicts an outstanding linear trend w.r.t. the number of nodes of the graph. It must be noticed that the Correctness - reported in Figure 4.b is not affected by the increased dimensionality. Those results, in conjunction with the results obtained in the previous ablation study (see Fig. 3.b), let us state that the performances are solely affected by the complexity of the datasets. Similarly, CF^2 has a linear trend in normalised GED by n as RSGG-CE. Recall that CF^2 is a factual-based explainer that supports only edge removal operations. Now, because the GED / n ratio tends to 1, we argue that CF^2 perturbs the entire adjacency matrix of the original instance.

Robustness by the number of instances In this last ablation, we randomly generated up to 3 cycles with varying sizes that go up to 7 nodes for each graph. We delve into the impact of varying the number of instances in the dataset (ranging from 100 to 8000) on RSGG-CE's GED and Correctness metrics (refer to Fig. 5). Notice that in Fig. 5.a, the GED is unaffected by the number of instances by exposing a linear constant trend for both RSGG-CE and CF2. As expected, the Correctness of RSGG-CE increases at the beginning and stabilises when the number of instances exceeds 250 (see Fig. 5.b). In general, the same Correctness trend is also confirmed for CF^2 with a lower value.

Qualitative Anecdotal Inspection

Here, we discuss anecdotally the quality of the counterfactual graphs generated by RSGG-CE and the SoA methods on the ASD dataset. Fig. 6 shows the counterfactual generation for CF^2, CLEAR, G-CounteRGAN, and RSGG-CE on two graphs belonging to the *Autistic Spectrum Disorder* and *Typically Developed* classes. For both instances, we show the original edges and illustrate how they get modified by each method. For visualisation purposes, we colour red the edge deletion operations, green the edge addition operations, and grey the original edge maintenance operations. CF^2 clearly shows its behaviour of removing the factual subgraph in both instances regardless of the class on ASD. However, it is peculiar that the subgraph $\tilde{G}$ corresponds to the original graph G, justifying the high GED reported in Table 1. CLEAR is the only SoA method that, in this example, evidently performs all three types of operations on the original edges. Although not as naive as CF^2's edge perturbation policy, CLEAR exposes a higher GED than CF^2 and RSGG-CE due to its tendency to over-generate non-existing edges. Interestingly, CLEAR is the only method that estimates self-loops (see instance of class *Typically Developed*) among the rest. G-CounteRGAN, as shown in Table 1, is the worst-performing strategy, exposing a tendency to produce a densely connected graph by adding non-existing edges. Lastly, RSGG-CE, although capable of edge addition/removal operations, here exhibits only removal ones. We argue that because RSGG-CE's produced valid explanation has a low GED, this dataset's partial order sampling strategy engenders a valid counterfactual only by examining the original edges. To this end, we believe that edge removals are primarily needed to produce valid counterfactuals in ASD.

We extend this qualitative analysis via a comprehensive visualisation technique that analyses the operations performed on each edge (see Sec. D of the SM).

Conclusion

In this study, we introduced RSGG-CE, a novel **R**obust **S**tochastic **G**raph **G**enerator for **C**ounterfactual **E**xplanations able to produce counterfactual examples from the learned latent space considering a partially ordered generation sequence. We showed, quantitatively and qualitatively, that RSGG-CE outperforms all SoA methods. RSGG-CE produces counterfactuals – *conditioned on the input* – from the learned latent space incorporating a partially ordered generation sequence. Additionally, the proposed partial order sampling offers an effective means for discerning existing and non-existing edges, contributing to the overall robustness of our model. One of the key findings of our study is the resilience of RSGG-CE in handling increasingly complex motifs within the graph instances. In the future, we want to leverage the ability of RSGG-CE to generate multiple counterfactual candidates to produce cohesive candidate explanations. Lastly, incorporating counterfactual minimality into the loss function of generative models might offer potential improvements in interpreting the generated explanations.

Acknowledgments

This work is partially supported by the 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, by the "ICSC – Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU, by European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP E13C22001060006, and by Territori Aperti (a project funded by Fondo Territori, Lavoro e Conoscenza CGIL CISL UIL).
All the numerical simulations have been realized on the Linux HPC cluster Caliban of the High-Performance Computing Laboratory of the Department of Information Engineering, Computer Science and Mathematics (DISIM) at the University of L'Aquila.

References

Abrate, C.; and Bonchi, F. 2021. Counterfactual graphs for explainable classification of brain networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2495–2504.

Aragona, D.; Podo, L.; Prenkaj, B.; and Velardi, P. 2021. CoRoNNa: a deep sequential framework to predict epidemic spread. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 10–17.

Bajaj, M.; Chu, L.; Xue, Z. Y.; Pei, J.; Wang, L.; Lam, P. C.-H.; and Zhang, Y. 2021. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems*, 34: 5644–5655.

Dan, A. S.; and Djeraba, C. 2008. Mathematical Tools for Data Mining-Set Theory, Partial Orders, Combinatorics.

Ding, M.; Yang, K.; Yeung, D.-Y.; and Pong, T.-C. 2019. 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*, 135–144.

Feng, J.; Chen, Y.; Li, F.; Sarkar, A.; and Zhang, M. 2022. How Powerful are K-hop Message Passing Graph Neural Networks. In *NeurIPS*.

Feng, W.; Tang, J.; and Liu, T. X. 2019. Understanding dropouts in MOOCs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 517–524.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5): 1–42.

Guyomard, V.; Fessant, F.; Guyet, T.; Bouadi, T.; and Termier, A. 2022. VCNet: A self-explaining model for realistic counterfactual generation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 437–453. Springer.

Huang, K.; Xiao, C.; Glass, L. M.; Zitnik, M.; and Sun, J. 2020. SkipGNN: predicting molecular interactions with skip-graph networks. *Scientific reports*, 10(1): 1–16.

Huang, Z.; Kosan, M.; Medya, S.; Ranu, S.; and Singh, A. 2023. Global counterfactual explainer for graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 141–149.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*.

Liu, Y.; Chen, C.; Liu, Y.; Zhang, X.; and Xie, S. 2021. Multi-objective Explanations of GNN Predictions. In *2021 IEEE International Conference on Data Mining (ICDM)*, 409–418. IEEE.

Livi, L.; and Rizzi, A. 2013. The graph matching problem. *Pattern Analysis and Applications*, 16: 253–283.

Loyola-González, O. 2019. Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View. *IEEE Access*, 7: 154096–154113.

Ma, J.; Guo, R.; Mishra, S.; Zhang, A.; and Li, J. 2022. CLEAR: Generative Counterfactual Explanations on Graphs. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Madeddu, L.; Stilo, G.; and Velardi, P. 2020. A feature-learning-based method for the disease-gene prediction problem. *International Journal of Data Mining and Bioinformatics*, 24(1): 16–37.

Nemirovsky, D.; Thiebaut, N.; Xu, Y.; and Gupta, A. 2022. CounteRGAN: Generating counterfactuals for real-time recourse and interpretability using residual GANs. In Cussens, J.; and 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*, volume 180 of *Proceedings of Machine Learning Research*, 1488–1497. PMLR.

Nguyen, T. M.; Quinn, T. P.; Nguyen, T.; and Tran, T. 2022. Explaining Black Box Drug Target Prediction through Model Agnostic Counterfactual Samples. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Numeroso, D.; and Bacciu, D. 2021. Meg: Generating molecular counterfactual explanations for deep graph networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Petch, J.; Di, S.; and Nelson, W. 2021. Opening the black box: the promise and limitations of explainable machine learning in cardiology. *Canadian Journal of Cardiology*.

Prado-Romero, M. A.; Prenkaj, B.; and Stilo, G. 2023. Revisiting CounteRGAN for Counterfactual Explainability of Graphs. In Maughan, K.; Liu, R.; and Burns, T. F., eds., *The First Tiny Papers Track at ICLR 2023, Tiny Papers @ ICLR 2023, Kigali, Rwanda, May 5, 2023*. OpenReview.net.

Prado-Romero, M. A.; Prenkaj, B.; Stilo, G.; and Giannotti, F. 2023. A Survey on Graph Counterfactual Explanations: Definitions, Methods, Evaluation, and Research Challenges. *ACM Comput. Surv.*

Prenkaj, B.; Aragona, D.; Flaborea, A.; Galasso, F.; Gravina, S.; Podo, L.; Reda, E.; and Velardi, P. 2023a. A self-supervised algorithm to detect signs of social isolation in the elderly from daily activity sequences. *Artificial Intelligence in Medicine*, 135: 102454.

Prenkaj, B.; Distante, D.; Faralli, S.; and Velardi, P. 2021. Hidden space deep sequential risk prediction on student trajectories. *Future Generation Computer Systems*, 125: 532–543.

Prenkaj, B.; Velardi, P.; Distante, D.; and Faralli, S. 2020. A reproducibility study of deep and surface machine learning methods for human-related trajectory prediction. In *Proc. of the 29th ACM Int. Conf. on Inf. & Knowl. Management*, 2169–2172.

Prenkaj, B.; Villaizan-Vallelado, M.; Leemann, T.; and Kasneci, G. 2023b. Adapting to Change: Robust Counterfactual Explanations in Dynamic Data Landscapes. *arXiv preprint arXiv:2308.02353*.

Scarselli, F.; Gori, M.; Tsoi, A.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Trans. on Neural Networks*, 20(1): 61–80.

Tan, J.; Geng, S.; Fu, Z.; Ge, Y.; Xu, S.; Li, Y.; and Zhang, Y. 2022. Learning and Evaluating Graph Neural Network Explanations Based on Counterfactual and Factual Reasoning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, 1018–1027. New York, NY, USA: Association for Computing Machinery. ISBN 9781450390965.

Verenich, I.; Dumas, M.; La Rosa, M.; and Nguyen, H. 2019. Predicting process performance: A white-box approach based on process models. *Journal of Software: Evolution and Process*, 31(6): e2170.

Verma, H.; Mandal, S.; and Gupta, A. 2022. Temporal deep learning architecture for prediction of COVID-19 cases in India. *Expert Systems with Applications*, 195: 116611.

Vermeire, T.; Brughmans, D.; Goethals, S.; de Oliveira, R. M. B.; and Martens, D. 2022. Explainable image classification with evidence counterfactual. *Pattern Analysis and Applications*, 25(2): 315–335.

Wang, W.; Yu, H.; and Miao, C. 2017. Deep model for dropout prediction in MOOCs. In *Proceedings of the 2nd international conference on crowd science and engineering*, 26–32.

Wei, X.; Liu, Y.; Sun, J.; Jiang, Y.; Tang, Q.; and Yuan, K. 2022. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*.

Wellawatte, G. P.; Seshadri, A.; and White, A. D. 2022. Model agnostic generation of counterfactual explanations for molecules. *Chemical science*, 13(13): 3697–3705.

Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; and Tan, T. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 346–353.

Wu, X.; Xiong, Y.; Zhang, Y.; Jiao, Y.; Shan, C.; Sun, Y.; Zhu, Y.; and Yu, P. S. 2022. Clare: A semi-supervised community detection algorithm. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2059–2069.

Xu, L.; Xi, W.; and Wang, C. 2021. Session-based Recommendation with Heterogeneous Graph Neural Networks. In *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*, 1–8. IEEE.

Xu, Z.; Lamba, H.; Ai, Q.; Tetreault, J.; and Jaimes, A. 2023. Counterfactual Editing for Search Result Explanation. *arXiv preprint arXiv:2301.10389*.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Zemni, M.; Chen, M.; Zablocki, E.; Ben-Younes, H.; Pérez, P.; and Cord, M. 2023. OCTET: Object-Aware Counterfactual Explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15062–15071.

Zhang, L.; Long, C.; Zhang, X.; and Xiao, C. 2020. Risgan: Explore residual and illumination with generative adversarial networks for shadow removal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 12829–12836.