

# 000 BEYOND EDGE DELETION: A COMPREHENSIVE AP- 001 PROACH TO COUNTERFACTUAL EXPLANATION IN 002 GRAPH NEURAL NETWORKS 003

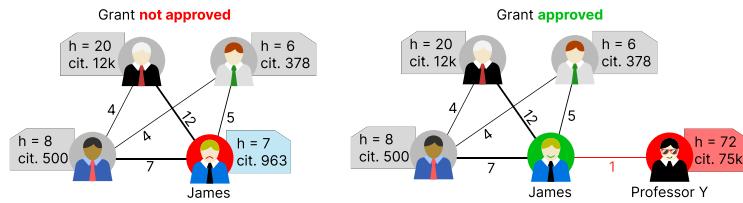
004  
005  
006 **Anonymous authors**  
007 Paper under double-blind review  
008  
009  
010  
011  
012

## ABSTRACT

013 Graph Neural Networks (GNNs) are increasingly adopted in domains like molec-  
014 ular biology and social network analysis, yet their black-box nature hinders inter-  
015 pretability and trust. This is especially problematic in high-stakes applications,  
016 such as predicting molecule toxicity, drug discovery, or guiding financial fraud  
017 detections, where transparent explanations are essential. Counterfactual expla-  
018 nations – minimal changes that flip a model’s prediction – offer a transparent lens  
019 into GNNs behavior. In this work, we introduce Xplore, a novel technique that  
020 significantly broadens the counterfactual search space. It consists of gradient-  
021 guided perturbations to adjacency and node feature matrices. Unlike most prior  
022 methods, which focus solely on edge deletions, our approach belongs to the grow-  
023 ing class of techniques that optimize edge insertions and node-feature perturba-  
024 tions, here jointly performed under a unified gradient-based framework, enabling  
025 a richer and more nuanced exploration of counterfactuals. To quantify both struc-  
026 tural and semantic fidelity, we introduce a cosine similarity metric on learned  
027 graph embeddings, addressing a key limitation of traditional distance-based met-  
028 rrics, demonstrating that Xplore produces more coherent and minimal counterfa-  
029 ctuals. Empirical results on nine real-world and five synthetic benchmarks show  
030 up to +56.3% improvement in validity and +52.8% in fidelity over state-of-the-art  
031 baselines, while retaining competitive runtime.

## 1 INTRODUCTION

032 Explainability is crucial in  
033 fields like healthcare and fi-  
034 nance, where transparent,  
035 accountable decisions are  
036 necessary Guidotti et al.  
037 (2018). While deep neu-  
038 ral networks are power-  
039 ful, their black-box na-  
040 ture limits interpretability  
041 Petch et al. (2021), hin-  
042 dering trust in sensitive  
043 applications. White-box  
044 models are more inter-  
045 pretable Loyola-González  
046 (2019) and better suited for  
047 regulated settings Verenich  
048 et al. (2019), but often underperform compared to black-box models on complex, high-dimensional  
049 tasks Aragona et al. (2021); Ding et al. (2019).



050 Figure 1: (left) James works with his research lab in publishing sci-  
051 entifically good papers with decent impact for the community; however,  
052 his grant application is not approved. (right) James writes a single  
053 not-so-interesting paper with Professor Y, who happens to be highly  
054 influential with a broader community, making James’s grant applica-  
055 tion approved. *James is happy, but at what cost?*

056 GNNs have gained significant attention Wei et al. (2022), but like other deep learning models, they  
057 lack interpretability. To mitigate this, post-hoc methods, particularly counterfactual explanations,  
058 aim to reveal how input changes affect predictions. Recently, Graph Counterfactual Explainability  
059 (GCE) has recently emerged as a key area Prado-Romero et al. (2023b). Consider a research collab-

oration network, where nodes represent scientists, and edges indicate co-authorships on published papers. Suppose a young researcher, James, is recommended for a prestigious grant based on their position in the collaboration graph. A counterfactual explanation might state:

*"If James had not co-authored a paper with Professor Y, he would not have been considered for the grant."*

This explanation highlights how specific connections within the graph structure influence high-stakes decisions. The counterfactual framework reveals how adding or removing certain collaborations alters the model's prediction, *offering insights into network-driven career advancements and systemic biases in academic recognition* – see Figure 1.

GCE techniques can be broadly categorized into search-based, heuristic-based, and learning-based approaches Prado-Romero et al. (2023b). Search-based methods identify counterfactuals by searching within the existing data distribution. Heuristic-based approaches modify the input graph  $G$  to create a perturbed version  $G'$ , ensuring that a prediction model  $\Phi$  produces different classifications ( $\Phi(G) \neq \Phi(G')$ ). These methods, however, require carefully designed heuristics, which often rely on domain expertise. For instance, generating chemically valid counterfactuals for molecular graphs necessitates an understanding of atomic valences and bonding rules. Learning-based strategies, in contrast, automate the discovery of meaningful perturbations by training on data, allowing for the generation of counterfactual examples at inference time.

In this work, we introduce XPlore, an extension of Lucic et al. (2022), designed to enhance counterfactual generation for GNNs. Unlike the original method, which only removes edges to modify graphs, XPlore introduces additional flexibility by incorporating edge additions and node feature perturbations. This expansion allows for a more comprehensive exploration of graph modifications, improving the quality of counterfactual examples. Our approach employs a specialized loss function that autonomously guides perturbations, minimizing unnecessary modifications, a huge improvement over heuristic-driven methods. Moreover, rather than incorporating generative modeling into the learning objective, XPlore directly leverages gradient-based optimization to identify the closest counterfactual instance efficiently.

Our contributions are as follows: (1) **Comprehensive Input Perturbation:** Unlike previous methods that only remove edges, our framework permits both edge additions and node feature modifications. With the sole exception of extra self-loop additions, this allows us to perturb the entirety of the original input data that the oracle processes. (2) **Gradient-Guided Optimization:** By exploiting the properties of the loss, we find the closest counterfactual through directed modifications. This guarantees that the counterfactual explanation is not only minimal but also consistent with the oracle's learned decision boundaries. (3) **Mitigation of Out-of-Distribution Issues:** The incorporation of edge additions along with node feature modifications helps alleviate the out-of-distribution problem Chen et al. (2023). This ensures that the generated counterfactuals retain key discriminative features relevant to the target class. (4) **Enhanced Search Space for Counterfactual Motifs:** We allow for both edge additions and node feature perturbations expands the search space, providing deeper insights into how structural and feature-level changes influence the oracle's prediction process and help uncover truly counterfactual motifs.

## 2 RELATED WORK

We distinguish between inherently explainable and black-box methods, focusing on counterfactual explanations for graph classification. While counterfactuals are well-established in images and text Vermeire et al. (2022); Zemni et al. (2023), their application to graphs is less common Liu et al. (2021); Ma et al. (2022); Nguyen et al. (2022); Numeroso & Bacciu (2021); Tan et al. (2022). As categorized in Prado-Romero et al. (2023b), graph counterfactual explanation methods fall into heuristic search and learning-based approaches. This work focuses on instance-level, learning-based explainers that identify minimal perturbations to flip a prediction, providing actionable, data-point-specific insights.

Learning-based strategies often use perturbation matrices Tan et al. (2022), reinforcement learning Numeroso & Bacciu (2021), or generative models Ma et al. (2022); Prado-Romero et al. (2024). Some notable methods include CF-GNNExp. Lucic et al. (2022), which learns a binary perturbation

matrix; CF<sup>2</sup> Tan et al. (2022), which balances factual and counterfactual reasoning through multi-objective optimization; and CLEAR Ma et al. (2022), which uses a Variational Autoencoder (VAE) to generate counterfactuals. Other approaches, such as RS GG-CE Prado-Romero et al. (2024) and D4Explainer Chen et al. (2023), rely on advanced techniques like Generative Adversarial Networks (GANs) and discrete denoising diffusion, respectively. We are also aware of emerging research on time-related graph counterfactuality Prenkaj et al. (2024); Qu et al. (2024), but this is outside the scope of our work.

Differently from SoTA, we perform graph- and node-level explanations. In the former, we identify changes needed to alter the prediction for an entire graph. In the latter, we leverage loss information from individual nodes, allowing us to perturb both the local graph structure and node features with the targeted objective of changing a single node’s label. Here, an oracle performs node-wise classification, and our method focuses on modifying the inputs that directly influence the prediction of the targeted node.

121

### 122 3 METHOD

#### 124 3.1 PROBLEM FORMULATION

126 **Graph Counterfactual Explanation.** Suppose we have a well-trained GNN serving as an oracle  
 127  $\Phi : \mathcal{G} \rightarrow \mathcal{Y}$  and an original graph instance  $G \in \mathcal{G}$ , with predicted label  $\Phi(G)$ . The objective  
 128 of counterfactual explanation is to find a perturbed graph  $G'$ , obtained by a counterfactual model  
 129  $\mathcal{E} : \mathcal{G} \rightarrow \mathcal{G}$ , that minimally deviates from  $G$  while ensuring that the oracle’s prediction changes, i.e.,  
 130  $\Phi(G') \neq \Phi(G)$  Prado-Romero et al. (2023b). Let  $\Delta(G, G')$  denote the distance function measuring  
 131 the difference between  $G$  and  $G'$ ; then the problem can be stated as:

$$132 \quad G^* = \arg \min_{G' \in \mathcal{G}'} \Delta(G, G') \text{ s.t. } \Phi(G) \neq \Phi(G'). \quad (1)$$

134 As done in Lucic et al. (2022), we generate counterfactuals by reformulating the  
 135 above hard-constrained formulation into a soft-  
 136 unconstrained optimization by minimizing the  
 137 loss function:

$$139 \quad L(G, G') = L_{\text{pred}}(G, G' | \Phi) + \beta L_{\text{dist}}(G, G'). \quad (2)$$

141 Here,  $L_{\text{pred}}$  is a prediction loss that encourages  
 142  $\Phi(G') \neq \Phi(G)$  and  $L_{\text{dist}}$  is a distance loss that  
 143 promotes similarity between  $G'$  and  $G$ , with  
 144 the trade-off controlled by  $\beta$ . Thus, the optimal  
 145 counterfactual example for  $G$  is obtained  
 146 by solving Equation (2).

147 **Node Counterfactual Explanation.** Our  
 148 method can be extended to node-level counterfactual  
 149 explanations. In this scenario, rather  
 150 than modifying only the target node, we perturb the entire graph, including all node features. More-  
 151 over, the loss formulation can be adapted to encompass a set of nodes by imposing a soft constraint  
 152 that preserves the class labels of nodes other than the target through the use of binary cross-entropy  
 153 with logits as the loss function. This framework naturally generalizes to multi-node classification  
 154 tasks, where the objective is to generate counterfactual explanations for a set of nodes rather than a  
 155 single node.

#### 156 3.2 PROPOSED METHOD

158 We propose XPlore to sieve the less-constrained counterfactual search space compared to the base  
 159 model Lucic et al. (2022), aiming at better explanations while reducing the out-of-distribution effect  
 160 shown in Chen et al. (2023). XPlore, not only drops instance edges but also adds them, enabling the  
 161 discovery of counterfactual explanations for which a class change is unattainable by edge removal  
 alone (e.g. TCR in Section B; see Figure 2). Moreover, by modifying node features, our approach

Table 1: Notation used in the Section 3.

Symbol	Description
$\mathcal{E}, \Phi$	Counterfactual and oracle model
$G, G'$	Original and counterfactual graphs
$A, \tilde{A}$	Adjacency matrices (original and counterfactual)
$P, \tilde{P}$	Original and new perturbation matrices for edges
$N, \tilde{N}$	Binary and continuous node feature perturb. matrices
$X, W$	Node feature and weight matrices
$D$	Degree matrix
$\Gamma$	Probability weight matrix with values $\gamma_{i,j} \in [0, 1]$
$\sigma(\cdot)$	Sigmoid activation function
$\text{softmax}(\cdot)$	Softmax function
$\mathcal{T}_\alpha(\cdot)$	Entry-wise threshold: $\mathcal{T}_\tau(X) = \mathbb{1}\{\sigma(X) \geq \alpha\}$
$\odot$	Element-wise (Hadamard) product
$\mathbb{1}\{\cdot\}$	Indicator function
$K$	Number of optimization iterations
$L_{\text{pred}}, L_{\text{dist}}$	Prediction and distance losses

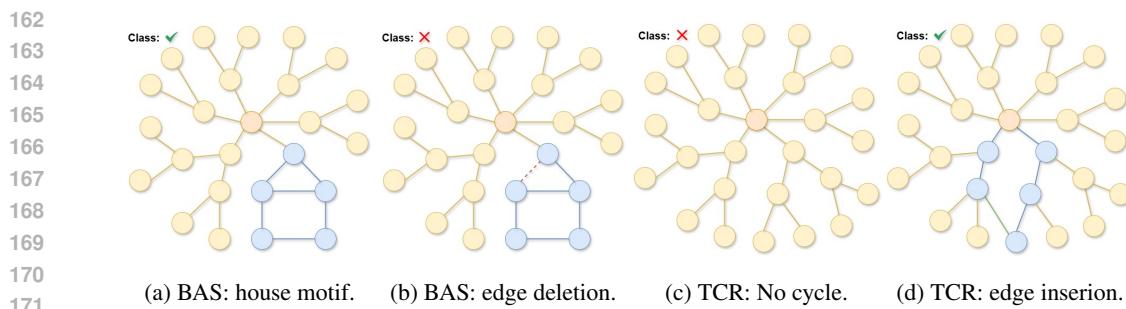


Figure 2: Illustration of counterfactual explanations in graph classification. *a* and *b*: The class changes only after an **edge deletion** (red, dashed). *c* and *d*: the predicted class changes only after an **edge addition** (green). XPlore enables both types of perturbations, providing a broader search space than edge deletion only methods.

can identify counterfactuals where a class change is unachievable solely through edge modifications (or any amount thereof), but rather via feature adjustments. We explored two modalities for handling features: i.e., (1) applying a gating mechanism to retain or discard node features, similarly to the edge-dropping process introduced in Lucic et al. (2022); (2) allowing for continuous adjustment in node features, enabling their values to increase/decrease smoothly. We provide formal guarantees on convergence, on  $\ell_1$ -minimality of perturbations and semantic fidelity in Section A.

**Original Adjacency matrix perturbations.** Let  $A \in \{0, 1\}^{n \times n}$  be the adjacency matrix – a square matrix whose elements indicate whether pairs of vertices are adjacent or not in the graph. We initially define the perturbation similarly to Lucic et al. (2022), having the CF generated matrix  $\hat{A} = P \odot A$ , with  $P = \mathbf{1}_{n \times n}$ , a binary perturbation matrix full of ones, with same dimensionality of  $A$ ; and  $\odot$  denoting element-wise (Hadamard) product (a real-valued  $\bar{P}$  is first generated, then a threshold with a entry-wise sigmoid transformation  $\sigma(\cdot)$  is performed to obtain a binary  $P$ , either discarding or retaining edges:  $\mathcal{T}_{0.5}(\bar{P}) = \mathbf{1}\{\sigma(\bar{P}) \geq 0.5\}$ ). To include self message passing, we add self-loops in the form of the identity matrix during CF search. We define  $X$  as the node feature matrix (so that  $G = (A, X)$ ),  $W$  as the weight matrix and  $\bar{D}$  as the degree matrix, a diagonal matrix containing the number of edges attached to each vertex, based on  $P \odot A + I$ . The original goal is to only remove edges zeroing out entries in the adjacency matrix, so find  $P$  (which acts as a gate) that minimally perturbs  $A$  and use it to compute  $\hat{A}$ . Hence, the original counterfactual generating model, parameterized by  $P$ , is:

$$\mathcal{E}(A, X, W; P) = \text{softmax}[\bar{D}^{-\frac{1}{2}}(P \odot A + I)\bar{D}^{-\frac{1}{2}}XW], \quad (3)$$

note that here, differently than Lucic et al. (2022), we perform the CF search not on a subgraph neighbourhood of a node, but rather on the whole graph. For undirected graphs, only the upper triangular part of  $\bar{P}$  is parametrized and it is symmetrized at each step to obtain  $\hat{A}$ . This ensures that adjacency perturbations remain symmetric.

**Edge-masking with Free Insertions** The original formulation of the framework has some downsides when it comes to updating  $P$ : when the oracle prediction is obtained with the associated loss, and new edges need to be added. Essentially, gradients for adjacency and perturbation matrix entries are computed and flow back, but edges not present in  $A$  cause respective gradients in  $P$  to be zeroed out – *edges not initially present in A cannot be inserted during CF search by design choice, and gradients of missing edges indicating the direction of optimal change, even if are back-propagated until P, will not lead to any update to it, thus introducing confusion for other updates that may be co-adapted/correlated to them*. To enable the inclusion of any edge, we swap the roles of  $A$  and  $P$ : initializing  $\bar{P} \leftarrow A$ , hence making it store the original adjacencies, but making it able to be updated by the iterative gradients; and making  $\bar{A} = \mathbf{1}_{n \times n}$  be a matrix full of ones, fully connecting the graph. We account for the original missing edges by removing them from  $\bar{P}$  (setting their values to zero), now  $\hat{A}$  is computed as  $\hat{A} = \bar{P} \odot \bar{A}$  plus self loops. By this swap, we give the explainer the possibility to freely add edges by updating the whole graph accordingly to the loss.  $\bar{A}$  now becomes a (redundant) matrix of ones whereas  $\bar{P}$  stores the graph connections, ready to be perturbed. We add to  $P$  a small Gaussian noise ( $std : \sigma = 0.1$ ) to break the symmetry in the gradient updates.

We also assign a probability weight to missing edges of  $\bar{P}$ : we define a matrix  $\Gamma \in [0, 1]^{n \times n}$ .  $\Gamma$  is added to  $\bar{P}$  at initialization:  $\bar{P} \leftarrow \bar{P} + \Gamma$ . The idea is to manually adjust each  $\gamma_{ij} \in [0, 1]$  so that by adding  $\Gamma$  it is possible to induce edge presence. For example we may be interested in a counterfactual explanation that has specific connections: we can directly encode them in  $\Gamma$ . Note that we swapped  $P$  and  $A$  to be coherent with the original formulation: as  $A$  now consists of a matrix of ones, it is redundant for updating  $\bar{P}$  and can be simply omitted. Now, the explainer is as in Equation (4):

$$\mathcal{E}(X, W; \bar{P}) = \mathcal{S}[\bar{D}^{-1/2} \hat{A} \bar{D}^{-1/2} X W], \quad (4)$$

where  $\mathcal{S}$  is either the element-wise sigmoid or the softmax function, depending on whether the task is binary or multi-class; and  $\bar{D}$  is based on  $\hat{A}$ , slightly different from Equation (3), ensuring that values in the diagonal are ones, specifically when self-loop are already present – due to  $\hat{A}$  having no more the diagonal entries fixed at zero as it depends on  $\bar{P}$  for storing perturbed adjacencies and not on  $A$  any more.

---

**Algorithm 1** XPlore.

---

```

Input graph  $G = (A, X)$ , trained oracle  $\Phi$ ,  

explainer  $\mathcal{E}$ , loss function  $L$ , learning rate  $\alpha$ ,  

number of iterations  $K$ , matrix  $\Gamma$ , node features  

gate flag, distance function  $d$ .  

 $y \leftarrow \Phi(G)$   

 $\bar{P} \leftarrow A + \mathcal{N}(0, 0.01) + \Gamma$   

 $\bar{N} \leftarrow \mathbf{1}_{n \times f}$   

 $G^* = []$   

for  $K$  iterations do  

     $G', G^* = \text{GET\_CF\_EXAMPLE}()$   

     $L \leftarrow L(G, G', \Phi)$  {► Eqns. 7-8}  

     $\bar{P} \leftarrow \bar{P} - \alpha \nabla_{\bar{P}} L$  {► Update  $\bar{P}$ }  

     $\bar{N} \leftarrow \bar{N} - \alpha \nabla_{\bar{N}} L$  {► Update  $\bar{N}$ }  

return  $G^*$   

func GET_CF_EXAMPLE ()  

     $\hat{A} \leftarrow \max(\mathcal{T}_{0.5}(\bar{P}), I)$  {►  $\hat{A}$  stores per-  

turbed adj.}  

    if gate then {► Gate node feat.}  

         $N \leftarrow \mathcal{T}_{0.5}(\sigma(\bar{N}))$   

    else {► Freely perturb node feat.}  

         $N \leftarrow \bar{N}$   

     $N \leftarrow N \odot X$   

     $G'_{cand} \leftarrow (\hat{A}, N)$   

    if  $\Phi(G) \neq \Phi(G'_{cand})$  then  

         $G' \leftarrow G'_{cand}$   

        if not  $G^*$  then  

             $G^* \leftarrow G'$  {► First CF}  

        else if  $d(G, G') \leq d(G, G^*)$  then  

             $G^* \leftarrow G'$  {► Closer CF}  

return  $G', G^*$ 

```

---

where  $\Phi_{\ell-1}(G)$  outputs the logits given the input  $G$ , and  $\ell$  is the number of layers.  $G'$  is  $\mathcal{E}(G)$ . The  $L_{dist}$  in Equation (2) is the element wise distance between  $G$  and  $G'$ , corresponding to the sums of the two  $L_p$ -norms of  $A$  and  $A'$ , and  $X$  and  $X'$ :

$$L_{\text{dist}}(G, G') = \|A - A'\| + \|X - X'\|. \quad (8)$$

**Node Features Perturbation.** Similarly to what has been done for the edge perturbation matrix  $\bar{P}$ , we can introduce a perturbation matrix  $\bar{N} = \mathbf{1}_{n \times f}$ ,  $f$  number of node features, to perform continuous perturbation on the node features. We can apply two different mechanisms (Algorithm 1): i.e., (1) gate the features values with a sigmoid, by applying the sigmoid and then the threshold to have a binary mask, and (2) let them freely change, updating them proportionally to the gradient information of the loss. In results, these mechanisms added on top of the edges addition, are defined as XPlore w/ gating and XPlore w/ freedom. Hence, the final formulation of our explainer considering all perturbations is:

$$N = \mathcal{T}_{0.5}(\bar{N}) \text{ if } \text{gate} \text{ else } \bar{N} \quad (5)$$

$$\mathcal{E}(X, W; \bar{P}, \bar{N}) = \mathcal{S}[\bar{D}^{-1/2} \hat{A} \bar{D}^{-1/2} (X \odot N) W]. \quad (6)$$

**Loss function optimization.** We generate  $\bar{P}$  by minimizing Equation (2), defining the prediction loss  $L_{\text{pred}}$  as in Equation (7): for  $L_{\text{logits}}$  we employ the cross-entropy (CE) loss for single-label classification tasks, encompassing binary and multi-class scenarios, and the binary cross-entropy with logits loss for multi-label classification tasks:

$$L_{\text{pred}}(G, G' | \Phi) = -\mathbf{1}[\Phi(G) = \Phi(G')] \cdot L_{\text{logits}}(\Phi_{\ell-1}(G), \Phi_{\ell-1}(G')), \quad (7)$$

270 3.3 ALGORITHMIC IMPLEMENTATION  
271

272 We summarize the details of our method XPlore<sup>1</sup> in Algorithm 1. Given a graph in the test set  
 273  $G$ , its prediction is obtained from the GNN oracle  $\Phi$ .  $\bar{P}$  is initialized as the adjacency matrix  $A$   
 274 and summed with  $\Gamma$ , giving missing edges a probability value of  $\gamma_{i,j} \in [0, 1]$ .  $N$  is assigned to a  
 275 matrix of ones. XPlore is run for  $K$  iterations, at each one, an optimization step is performed to  
 276 find a valid counterfactual and improve it, there is no stopping criterion as the same CF is iter-  
 277 atively modified and potentially improved across successive iterations, as conceived by Lucic et al.  
 278 (2022). Equation (6) is used to find a counterfactual example.  $P$  is computed by applying a sigmoid  
 279 transformation on  $\bar{P}$  and then a threshold to obtain a binary matrix. Although hard threshold  $\mathcal{T}$  is  
 280 applied in forward pass, gradients are back-propagated through this non-differentiable step via the  
 281 straight-through estimator (STE) Bengio et al. (2013), which treats thresholding as identity during  
 282 backpropagation (cf. Section A.2). We add self-loops by summing the identity matrix. According  
 283 to the node perturbation mechanism, node features are either gated or simply multiplied by the node  
 284 features perturbation matrix  $\bar{N}$ . The candidate graph produced  $G'_{\text{cand}}$  is fed to the GNN oracle  $\Phi$ .  
 285 If the output prediction is different from the initial node, a valid counterfactual is found. The closer  
 286 counterfactual is returned as the optimal CF example  $G^*$  after  $K$  iterations upon success.  $\bar{P}$  and  $\bar{N}$   
 287 are updated based on the loss, which is calculated according to Equations (2) and (6) to (8). This  
 288 setting allows to perform updates freely by perturbing edges and node features and adding missing  
 289 ones. The optimal explanation is retrieved as  $\Delta_G^* = G - G^*$ .

290 4 EXPERIMENTS  
291292 4.1 EXPERIMENTAL SETUP  
293

294 We train separate GCN oracles for each dataset using an 80-20 train-test split, with architectural  
 295 and hyperparameter details provided in Appendices C.1 and C.2, respectively. We compare XPlore  
 296 against CF<sup>2</sup> Tan et al. (2022), CLEAR Ma et al. (2022), RSGG-CE Prado-Romero et al. (2024),  
 297 D4Explainer Chen et al. (2023), and iRand Prado-Romero et al. (2023a). For iRand, we set the edge  
 298 perturbation probability ( $p$ ) to 0.01 and the number of iterations ( $t$ ) to 3.

299 Following the protocol from Prado-Romero et al. (2023b), we use a comprehensive evaluation suite,  
 300 including *Oracle Accuracy*, *Validity*, *Fidelity*, *Sparsity*, *Graph Edit Distance (GED)*, and *Oracle*  
 301 *Calls*. Additionally, we use the cosine similarity (CS), which measures the semantic similarity  
 302 between original and counterfactual graphs. Unlike structural metrics such as GED or sparsity, CS  
 303 captures alignment in graph meaning by comparing embeddings.

304 Given a set of graph embedders  $E$  ( $|E| = M$ ), we compute vector embeddings  $e_{ji} = E_j(G_i)$  and  
 305  $e'_{ji} = E_j(G'_i)$  for each graph  $G_i$  ( $|G| = N$ ) and its counterfactual  $G'_i$ , using  $E_j \in E$ . CS is defined  
 306 as:

$$307 \quad \text{CS}(e_{ji}, e'_{ji}) = \frac{1}{MN} \sum_{j=1}^M \sum_{i=1}^N \frac{e_{ji} \cdot e'_{ji}}{|e_{ji}| |e'_{ji}|} \in [-1, 1] \quad (9)$$

310 The set  $E$  includes following embedders: Feather-G Rozemberczki & Sarkar (2020),  
 311 Graph2Vec Narayanan et al. (2017), NetLSD Tsitsulin et al. (2018), WaveletCharacteristic Wang  
 312 et al. (2021), IGE Galland & Lelarge (2019), LDP Cai & Wang (2022), GeoScattering Gao et al.  
 313 (2019), GL2Vec Chen & Koga (2019), SF de Lara & Pineau (2018), and FGDS Verma & Zhang  
 314 (2017). Definitions for standard metrics are detailed in Section C.3.

315 4.2 EXPERIMENTAL RESULTS  
316

317 **XPlore on average achieves +17.3% percentage improvement in validity, and +15.0% on fi-**  
 318 **dельty on across the board vs. the second-best in graph classification.** Table 3 shows the  
 319 comparison between XPlore and other SoTA methods on all used datasets. We show the val-  
 320 idity, fidelity, sparsity, and oracle calls since they describe the goodness of the explainers in  
 321 finding valid counterfactuals that are also cheap in terms of querying the underlying predi-  
 322 tor. Note that XPlore is the best in 14/14 datasets in terms of validity and 14/14 regarding fi-

323 <sup>1</sup><https://anonymous.4open.science/r/XPlore>

Table 3: Comparison of Xplore with SoTA methods (validity  $\uparrow$  – up; fidelity  $\uparrow$  – down). **Bold** is best-performing; underline is second-best. **Xplore is best in 14/14 on validity, and 14/14 on fidelity (of which 1/14 on par with RSGG-CE).** Table 8 in the Appendix shows a detailed comparison on other metrics.  $\dagger$  indicates the second-best explainer.

	TCR	BAS	BZR	AIDS	ENYMES	Fingerprint	COLORS-3	TG	MUTAG	COX2	BBBP	PROTEINS	COLLAB	TRIANGLES
iRand	27.92	50.70	27.16	0.00	26.67	0.09	42.99	36.16	2.66	<u>69.81</u>	19.76	18.87	4.60	6.38
CF <sup>2</sup>	50.04	45.78	19.75	0.10	68.33	24.52	52.07	49.86	0.00	24.20	<u>25.26</u>	16.35	<u>52.66</u>	37.13
CLEAR	50.68	50.96	<u>60.49</u>	16.75	83.17	72.73	0.00	58.40	35.11	22.06	22.90	0.00	0.00	89.99
RSGG-CE $\dagger$	<u>67.90</u>	<u>91.04</u>	21.23	<u>19.80</u>	<u>98.33</u>	<u>90.46</u>	<u>94.57</u>	<u>89.28</u>	<u>56.91</u>	<b>99.36</b>	22.90	<u>58.67</u>	0.00	<u>99.84</u>
D4Explainer	44.82	35.16	20.00	0.10	68.00	24.52	0.00	49.86	9.57	22.06	21.24	0.00	0.00	31.11
CF-GNNExpl	32.40	44.18	19.75	0.10	68.33	24.52	52.07	49.86	10.11	22.06	22.31	16.35	<u>52.66</u>	37.13
<b>Xplore</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>32.30</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>67.55</b>	<b>99.36</b>	<b>81.51</b>	<b>65.41</b>	<b>100.0</b>	<b>100.0</b>	

	TCR	BAS	BZR	AIDS	ENYMES	Fingerprint	COLORS-3	TG	MUTAG	COX2	BBBP	PROTEINS	COLLAB	TRIANGLES
iRand	0.279	0.507	0.262	0.000	0.238	0.001	0.300	0.356	0.005	<u>0.677</u>	<u>0.275</u>	0.183	0.006	0.053
CF <sup>2</sup>	0.500	0.457	0.188	0.001	0.633	0.133	0.398	0.499	0.005	0.229	0.253	0.151	<u>0.262</u>	0.364
CLEAR	0.507	0.510	<u>0.595</u>	0.168	0.797	0.515	0.000	0.584	0.309	0.208	0.229	0.000	0.000	0.892
RSGG-CE $\dagger$	<u>0.679</u>	<u>0.910</u>	0.202	<u>0.198</u>	<u>0.930</u>	<u>0.653</u>	<u>0.824</u>	<u>0.888</u>	<u>0.516</u>	<b>0.968</b>	0.229	<u>0.556</u>	0.000	<u>0.987</u>
D4Explainer	0.448	0.446	0.190	0.001	0.632	0.133	0.000	0.499	0.021	0.208	0.212	0.000	0.000	0.302
CF-GNNExpl	0.324	0.442	0.188	0.001	0.633	0.133	0.398	0.499	0.005	0.208	0.223	0.151	<u>0.262</u>	0.364
<b>Xplore</b>	<b>1.000</b>	<b>1.000</b>	<b>0.980</b>	<b>0.323</b>	<b>0.942</b>	<b>0.730</b>	<b>0.896</b>	<b>0.994</b>	<b>0.548</b>	<b>0.968</b>	<b>0.803</b>	<b>0.627</b>	<b>0.570</b>	<b>0.988</b>

This shows that a simple modification on the learning objective and node feature manipulation enhances the faithfulness of the explainer. Xplore maintains a relatively low number of oracle class indicating that in at most 14.692 (see TG) iterations, the optimal counterfactual is found (see line 9 of Algorithm 1). Additionally, we show the relationship between GED and CS in Figure 3b. However, we are interested in those explainers that might induce more edit changes to the counterfactual yet maintaining high semantic similarity. Note that Xplore generally exhibits high GED values although it consistently achieves strong CS scores, underlining our intuition that semanticity plays an important role in counterfactuality. We reserve more thorough experimentation for future work. Lastly, we show the runtime (s) at inference time for all methods to show that Xplore is at least on par with SoTA explainers (see Figure 4).

**Xplore outperforms CF-GNNExpl on node classification, improving per-class explainability by +2.37 validity points.** We compare both methods on the BAS dataset, where nodes are labeled based on motif membership. Despite prior claims Lucic et al. (2022), generating effective node-level counterfactuals remains difficult, particularly in class-specific scenarios. As summarized in Table 4, Xplore shows consistent improvements, especially when explaining nodes associated with house motifs via edge removals. However, its performance declines when explanations require adding such motifs. Although Xplore occasionally misleads the oracle by exploiting out-of-distribution effects (see Section 4.3, CF0 in Table 4), handling edge additions remains a notable limitation, in contrast to the relatively reliable edge deletion strategy.

Table 4: Node-level performance in BAS. CF0 and CF1 are the nodes not belonging to house-shaped motifs, and those belonging to house-shaped motifs, respectively.

	Oracle Acc. $\uparrow$	Validity $\uparrow$	CF0 $\uparrow$	CF1 $\uparrow$
CF-GNNExpl	100%	2.42%	0.0%	40%
<b>Xplore</b>	100%	<b>4.74%</b>	<b>0.016%</b>	<b>100%</b>

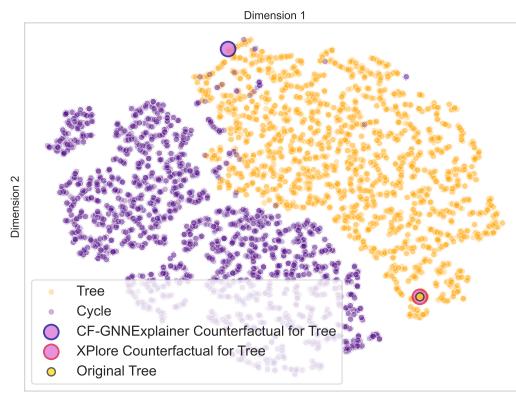
378    4.3 RESIDUAL OUT-OF-DISTRIBUTION INFLUENCE  
 379

380 Chen et al. (2023) observed that CF-GNNExpl leverages the out-of-distribution (OOD) effect that  
 381 influences the oracle’s prediction, by deriving explanatory subgraphs while omitting additional po-  
 382 tential edges.

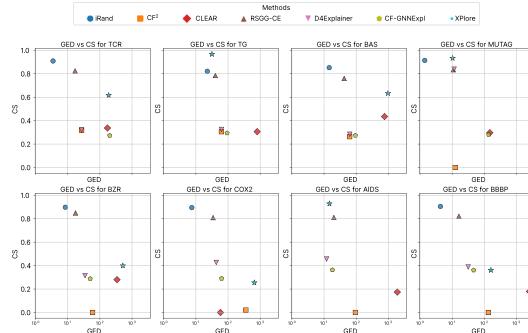
383 Consequently, the extracted explanation lacks  
 384 discriminative information for the counterfac-  
 385 tional class but is still classified as counterfac-  
 386 tional due to this OOD effect, ultimately mis-  
 387 leading the oracle and compromising reliabil-  
 388 ity. Although XPlore’s ability leads to gen-  
 389 erate more comprehensive and hence robust ex-  
 390 planations, the OOD effect remains present.  
 391 As shown in Figure 3b, we reduce OOD ef-  
 392 fect exploitation by increasing cosine simili-  
 393 arity, yet, this phenomenon persists and has not  
 394 been completely eliminated. t-SNE projection  
 395 of the Wavelet Characteristic embedding space  
 396 reveals that XPlore’s counterfactual exhibits  
 397 substantial semantic overlap with the original  
 398 instance, remaining in close proximity in an  
 399 embedding space that integrates both topolog-  
 400 ical and node attribute information (Figure 3a).  
 401 While preserving in-distribution behavior can  
 402 be advantageous in some contexts, here it in-  
 403 dicates that the explainer is capitalizing on the  
 404 OOD effect. In other words, the explainer  
 405 leverages the oracle’s decision boundary with-  
 406 out truly capturing the underlying counterfac-  
 407 tional distribution, given the oracle’s limited ex-  
 408 pressive power. Both instances lie outside the  
 409 Cycle distribution and thus benefit from the  
 410 OOD effect; however, XPlore’s counterfactual  
 411 is noticeably semantically identical to the origi-  
 412 nal, thanks to slight perturbations in the node  
 413 feature space. This reflects a previously ob-  
 414 served limitation: the inability to consistently  
 415 include edges additions in the generation of  
 416 counterfactual graphs, a constraint that may be  
 417 closely linked to the unstable nature of the or-  
 418 acle’s decision boundary and its capacity to ef-  
 419 fectively comprehend the underlying data dis-  
 420 tributions. We refer the reader to Leemann et al.  
 421 (2024) for non-adversarial counterfactual ex-  
 422 planations.

423    4.4 ABLATION STUDIES

424 **Free node perturbation leads to better counterfactual validity.** Since XPlore performs free node  
 425 perturbations, we study the effect of this freedom by constraining it and assessing whether simpler  
 426 mechanisms work in the same manner. Therefore, we test (1) XPlore w/ gating that performs edge  
 427 additions and node feature manipulation by gating them, similarly to the original edge drop mecha-  
 428 nism Lucic et al. (2022), either retaining or discarding them; and (2) a variant that does not have the  
 429 gating mechanism and cannot freely manipulate node features, called XPlore w/o freedom & gating.  
 430 To be consistent with the previous nomenclature, we rename XPlore in XPlore w/ freedom. We  
 431 show the validity of these variants in Table 5. Thanks to the better manipulation of graph elements,  
 432 all three variants have higher validity w.r.t. the original CF-GNNExplainer. The additional ability



(a) t-SNE projection of Wavelet Characteristic em-  
 beddings for TCR, comparing CFs generated by CF-  
 GNNExpl and XPlore for the Tree motif. While both  
 methods exploit the OOD effect, XPlore’s CF overlaps  
 with the original instance, preserving its semantic con-  
 tent, whereas CF-GNNExpl’s counterfactual diverges  
 more distinctly.



(b) Relationship between GED and CS. **XPlore consistently achieves strong CS scores w.r.t. the competitors along-side same GED values in log-scale**, indicating that counterfactuals retain meaningful semantic relations to original graphs, partially mitigating OOD.

Figure 3: OOD Effect and GED-CS Relationship.

to alter node features generally leads to even better results. The flexibility given to this modulation allows to tune XPlore to match the explainability criteria and find better counterfactuals.

In Table 6, we report metrics for  $\gamma = 0$  and  $\gamma = 0.01$  respectively, where  $\gamma$  is the value used to populate the  $\Gamma$  matrix entries that correspond to missing edges of  $A$ .  $\Gamma$  is added to  $P$  at initialization. Recall that an edge is present if and only if  $\sigma(v_i, v_j) > 0.5$  s.t.  $v_i, v_j$  are nodes. A positive  $\gamma$  increases the likelihood that edges are present at initialization, since  $\sigma(\gamma + \epsilon) > 0.5$ . However, because of the added Gaussian noise, presence is not guaranteed. This behavior possibly results in a higher number of counterfactuals found, but at the cost of higher GED, higher sparsity, and lower CS. This functionality may be needed if some edge is required to be present in the CF explanation; this prior knowledge is injected in the form of edge probabilities in matrix  $\Gamma$ .

Table 6: Metrics comparison over TCR dataset for different values of hyperparameter  $\gamma$ . We emphasize in bold the variants of XPlore per metric; underlined is the second-best.

Table 5: Validity (%) for different variants of XPlore ( $\gamma = 0.01$ ). Bold values are best.

	w/o freedom + gating	w/ gating	w/ freedom	CF-GNNExpl
TCR	99.680	65.580	<b>100.00</b>	50.040
TG	99.660	<b>100.00</b>	<b>100.00</b>	49.860
BAS	94.340	<b>100.00</b>	<b>100.00</b>	44.180
MUTAG	45.745	60.106	<b>67.553</b>	10.106
BZR	98.519	<b>100.00</b>	<b>100.00</b>	19.753
COX2	<b>99.358</b>	97.859	98.073	22.056
AIDS	0.300	19.850	<b>30.300</b>	0.100
BBBP	38.892	38.794	<b>79.794</b>	22.315
ENZYMES	76.500	<b>99.667</b>	98.833	68.333
PROTEINS	17.071	64.960	<b>65.409</b>	16.352
Fingerprint	26.105	<b>98.232</b>	94.742	24.523
COLLAB	55.640	<b>100.00</b>	93.380	52.660
COLORS-3	67.943	<b>100.00</b>	94.476	52.067
TRIANGLES	39.873	90.436	<b>100.00</b>	37.127

Table 6: Metrics comparison over TCR dataset for different values of hyperparameter  $\gamma$ . We emphasize in bold the variants of XPlore per metric; underlined is the second-best.

	Validity $\uparrow$			Fidelity $\uparrow$			Sparsity $\downarrow$			GED $\downarrow$			Oracle Calls $\downarrow$			CS $\uparrow$		
	$\gamma = 0$	$\gamma = 0.01$	$\gamma = 0.1$	$\gamma = 0$	$\gamma = 0.01$	$\gamma = 0.1$	$\gamma = 0$	$\gamma = 0.01$	$\gamma = 0.1$	$\gamma = 0$	$\gamma = 0.01$	$\gamma = 0.1$	$\gamma = 0$	$\gamma = 0.01$	$\gamma = 0.1$	$\gamma = 0$	$\gamma = 0.01$	
w/o freedom + gating	50.04	<b>99.68</b>	70.86	0.500	<b>0.997</b>	0.709	0.745	3.578	3.996	<b>41.00</b>	201.6	224.3	46.00	24.09	16.16	0.309	0.273	0.406
w/ gating	65.58	65.58	49.18	0.656	0.656	0.492	<b>0.249</b>	4.931	5.491	<b>14.00</b>	278.5	308.5	12.36	3.706	<b>2.000</b>	<b>0.955</b>	0.426	0.293
w/ freedom	96.00	<b>100.0</b>	<b>100.0</b>	0.960	<b>1.000</b>	<b>0.251</b>	2.708	4.743	<b>14.00</b>	151.02	264.1	5.881	3.593	<b>3.570</b>	<b>0.956</b>	0.562	0.337	

## 5 CONCLUSION

We introduced XPlore, a novel counterfactual explainer for Graph Neural Networks (GNNs) that explores a more complete search space, allowing for both edge deletions/insertions and node feature perturbations. Our method is not a black box itself; it relies on a basic gradient-based optimization building block to perform an intuitive counterfactual search. Once an oracle is trained, our explainer is fast and lightweight, requiring no further training or intense computation. XPlore identifies minimal yet impactful modifications, ensuring high-quality counterfactual explanations while avoiding heuristic-driven biases. Through extensive benchmarking across various datasets, XPlore consistently outperformed existing state-of-the-art explainers in validity and fidelity. Our empirical evaluation highlights that XPlore achieves an average percentage gain of +17.3% in validity and +15.0% in fidelity over the second-best method across multiple datasets. We also rely on the cosine similarity of graph embeddings as a complementary metric, which demonstrates that XPlore captures structural modifications while maintaining semantic fidelity. This confirms the effectiveness of incorporating a more flexible perturbation space and a comprehensive loss function.

**Future work.** We acknowledge that mitigating out-of-distribution (OOD) effects and optimizing node feature perturbations remain key challenges. While our method has shown promise in this area, the capacity to reduce these effects is contingent on how effectively the oracle captures the data distribution. We posit that less robust oracles are more prone to OOD explanations, which gives us hope that a more robust oracle, such as one based on a diffusion model, could significantly reduce XPlore’s OOD effects. We will also investigate the link between counterfactual explanations and model robustness, aiming to create explainability methods that are both interpretable and resistant to adversarial manipulation. Another avenue would be that of refining the loss function to better balance minimal perturbations with explanation quality and developing more precise control over feature modifications.

## REFERENCES

- Dario Aragona, Luca Podo, Bardh Prenkaj, and Paola Velardi. Corona: a deep sequential framework to predict epidemic spread. In *Proc. of the 36th Annual ACM Symposium on Applied Computing*, pp. 10–17, 2021.

- 486    Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients  
 487    through stochastic neurons for conditional computation, 2013. URL <https://arxiv.org/abs/1308.3432>.  
 488
- 489    Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig,  
 490    Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):  
 491    235–242, 2000.  
 492
- 493    Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):  
 494    334–334, 1997.  
 495
- 496    Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and  
 497    Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl\_1):  
 498    i47–i56, 2005.
- 499    Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification,  
 500    2022. URL <https://arxiv.org/abs/1811.03508>.  
 501
- 502    Hong Chen and Hisashi Koga. Gl2vec: Graph embedding enriched by line graphs with edge features.  
 503    In *Neural Information Processing: 26th Int. Conf. ICONIP 2019*, pp. 3–14. Springer, 2019.  
 504
- 505    Jialin Chen, Shirley Wu, Abhijit Gupta, and Rex Ying. D4explainer: In-distribution explanations  
 506    of graph neural network via discrete denoising diffusion. *Adv. in Neural Inf. Processing Systems*,  
 507    36:78964–78986, 2023.  
 508
- 509    Nathan de Lara and Edouard Pineau. A simple baseline algorithm for graph classification, 2018.  
 510    URL <https://arxiv.org/abs/1810.09155>.  
 511
- 512    Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Cor-  
 513    win Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro com-  
 514    pounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal  
 515    chemistry*, 34(2):786–797, 1991.  
 516
- 517    Mucong Ding, Kai Yang, Dit-Yan Yeung, and Ting-Chuen Pong. Effective feature learning with  
 518    unsupervised learning for improving the predictive models in massive open online courses. In  
 519    *Proc. of the 9th Int. Conf. on Learning Analytics & Knowl.*, pp. 135–144, 2019.  
 520
- 521    Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without  
 522    alignments. *Journal of molecular biology*, 330(4):771–783, 2003.  
 523
- 524    Alexis Galland and Marc Lelarge. Invariant embedding for graph classification. In *ICML 2019  
 525    Workshop on Learning and Reasoning with Graph-Structured Data*, Long Beach, United States,  
 526    June 2019. URL <https://hal.science/hal-02947290>.  
 527
- 528    Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *Proc. of  
 529    the 36th Int. Conf. on Machine Learning*, volume 97, pp. 2122–2131. PMLR, 2019.  
 530
- 531    Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino  
 532    Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys  
 533    (CSUR)*, 51(5):1–42, 2018.  
 534
- 535    Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and general-  
 536    ization in graph neural networks, 2019. URL <https://arxiv.org/abs/1905.02850>.  
 537
- 538    Tobias Leemann, Martin Pawelczyk, Bardh Prenkaj, and Gjergji Kasneci. Towards non-adversarial  
 539    algorithmic recourse. In *World Conference on Explainable Artificial Intelligence*, pp. 395–419.  
 540    Springer, 2024.  
 541
- 542    Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrink-  
 543    ing diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD interna-  
 544    tional conference on Knowledge discovery in data mining*, pp. 177–187, 2005.  
 545
- 546    Yifei Liu, Chao Chen, Yazheng Liu, Xi Zhang, and Sihong Xie. Multi-objective explanations of gnn  
 547    predictions. In *2021 IEEE Int. Conf. on Data Mining (ICDM)*, pp. 409–418. IEEE, 2021.  
 548

- 540 O. Loyola-González. Black-box vs. white-box: Understanding their advantages and weaknesses  
 541 from a practical point of view. *IEEE Access*, 7:154096–154113, 2019.
- 542
- 543 A. Lucic, M.A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri. Cf-gnnexplainer: Counter-  
 544 factual explanations for graph neural networks. In *Int. Conf. on AI and Statistics*, pp. 4499–4511.  
 545 PMLR, 2022.
- 546 Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. CLEAR: Generative  
 547 counterfactual explanations on graphs. In *Adv. in Neural Inf. Processing Systems*, 2022.
- 548
- 549 Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in  
 550 silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*,  
 551 52(6):1686–1697, 2012.
- 552 Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu,  
 553 and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs, 2017. URL  
 554 <https://arxiv.org/abs/1707.05005>.
- 555 Michel Neuhaus and Horst Bunke. A graph matching based approach to fingerprint classification  
 556 using directional variance. In *International Conference on Audio-and Video-Based Biometric  
 557 Person Authentication*, pp. 191–200. Springer, 2005.
- 558
- 559 Tri Minh Nguyen, Thomas P Quinn, Thin Nguyen, and Truyen Tran. Explaining black box drug  
 560 target prediction through model agnostic counterfactual samples. *IEEE/ACM Trans. on Compt.  
 561 Biology and Bioinformatics*, 2022.
- 562
- 563 Danilo Numeroso and Davide Bacciu. Meg: Generating molecular counterfactual explanations for  
 564 deep graph networks. In *2021 Int. Joint Conf. on Neural Networks*, pp. 1–8. IEEE, 2021.
- 565
- 566 J. Petch, S. Di, and W. Nelson. Opening the black box: the promise and limitations of explainable  
 567 machine learning in cardiology. *Canadian Journal of Cardiology*, 2021.
- 568
- 569 Mario Alfonso Prado-Romero, Bardh Prenkaj, and Giovanni Stilo. Are generative-based graph  
 570 counterfactual explainers worth it? In *Joint European Conf. on Machine Learning and Knowl.  
 Disc. in Databases*, pp. 152–170. Springer, 2023a.
- 571
- 572 Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. A survey on  
 573 graph counterfactual explanations: Definitions, methods, evaluation, and research challenges.  
*ACM Comput. Surv.*, sep 2023b. ISSN 0360-0300.
- 574
- 575 Mario Alfonso Prado-Romero, Bardh Prenkaj, and Giovanni Stilo. Robust stochastic graph gener-  
 576 ator for counterfactual explanations. In *Proc. of the AAAI Conf. on Artificial Intelligence*, vol-  
 577 ume 38, pp. 21518–21526, 2024.
- 578
- 579 Bardh Prenkaj, Mario Villaizán-Vallelado, Tobias Leemann, and Gjergji Kasneci. Unifying evolu-  
 580 tion, explanation, and discernment: A generative approach for dynamic graph counterfactuals. In  
*Proc. of the 30th ACM SIGKDD Conf. on Knowl. Disc. and Data Mining*, pp. 2420–2431, 2024.
- 581
- 582 Zhan Qu, Daniel Gomm, and Michael Färber. Greedy and cody: Counterfactual explainers for  
 583 dynamic graphs. *arXiv preprint arXiv:2403.16846*, 2024.
- 584
- 585 Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recogni-  
 586 tion and machine learning. In *Joint IAPR international workshops on statistical techniques in  
 587 pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pp. 287–297.  
 Springer, 2008.
- 588
- 589 Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from  
 590 statistical descriptors to parametric models, 2020. URL <https://arxiv.org/abs/2005.07959>.
- 591
- 592 Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn,  
 593 and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments.  
*Nucleic acids research*, 32(suppl\_1):D431–D433, 2004.

- 594 Jeffrey J. Sutherland, Lee A. O’Brien, and Donald F. Weaver. Spline-fitting with a genetic algorithm:  
 595 A method for developing classification structure-activity relationships. *Journal of Chemical Inf.*  
 596 and Computer Sciences, 43(6):1906–1915, 2003.  
 597
- 598 Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang.  
 599 Learning and evaluating graph neural network explanations based on counterfactual and factual  
 600 reasoning. In *Proc. of the ACM Web Conf. 2022, WWW ’22*, pp. 1018–1027, 2022.  
 601
- 602 Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller.  
 603 Netlsd: Hearing the shape of a graph. In *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowl.*  
 604 *Discovery & Data Mining*. ACM, 2018.  
 605
- 606 I. Verenich, M. Dumas, M. La Rosa, and H. Nguyen. Predicting process performance: A white-box  
 607 approach based on process models. *Journal of Software: Evolution and Process*, 31(6):e2170,  
 608 2019.  
 609
- 610 Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on  
 611 graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.  
 612
- 613 Tom Vermeire, Dieter Brughmans, Sofie Goethals, Raphael Mazzine Barbossa de Oliveira, and  
 614 David Martens. Explainable image classification with evidence counterfactual. *Pattern Analysis*  
 615 and Applications, 25(2):315–335, 2022.  
 616
- 617 Lili Wang, Chenghan Huang, Weicheng Ma, Xinyuan Cao, and Soroush Vosoughi. Graph embed-  
 618 ding via diffusion-wavelets-based node feature distribution characterization. In *Proc. of the 30th*  
*619 ACM Int. Conf. on Information & Knowl. Management*, pp. 3478–3482, 2021.  
 620
- 621 Xuemei Wei, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, Qifeng Tang, and Kun Yuan. Dual  
 622 subgraph-based graph neural network for friendship prediction in location-based social networks.  
*ACM Trans. on Knowl. Disc. from Data*, 2022.  
 623
- 624 Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations  
 625 for graph neural networks. *Adv. in Neural Inf. Processing Systems*, 32, 2019.  
 626
- 627 Mehdi Zemni, Mickaël Chen, Éloi Zablocki, Hédi Ben-Younes, Patrick Pérez, and Matthieu Cord.  
 628 Octet: Object-aware counterfactual explanations. In *Proc. of the IEEE/CVF Conf. on Computer*  
 629 *Vision and Pattern Recognition (CVPR)*, pp. 15062–15071, June 2023.  
 630

## 631 A THEORETICAL FOUNDATIONS OF GRADIENT-GUIDED COUNTERFACTUAL 632 PERTURBATIONS

633 This section is meant to give a mathematical justification for why our gradient-based procedure  
 634 converges and finds small ( $\ell_1$ -minimal) local perturbations to the graph and features.  
 635

636 **Appendix Roadmap.** We first establish convergence of projected gradient descent (A.1),  
 637 then handle non-smooth thresholding (A.2), prove the  $\ell_1$ -minimality bound (A.3), derive the  
 638 edge-insertion/deletion condition (A.4), show our prediction loss is  $L$ -smooth (A.5), choose  
 639 sparsity-accuracy trade-offs (A.6), analyze computational complexity (A.7), extend to weighted  
 640 and directed graphs (A.8), and finally discuss search-space expressivity (A.9).  
 641

642 In Section 3, we described a soft, differentiable objective  $L(\cdot)$  (Equation (2)) that trades off  
 643 between (a) changing the model’s prediction and (b) paying an  $\ell_p$  “cost” for every edit. What  
 644 remains is to show that if we optimize this objective with a natural algorithm, we end up at a valid  
 645 counterfactual that also doesn’t make too-large edits.

646 Hence, we here justify why the joint gradient-based optimization over edge masks  $P$  and feature  
 647 perturbations  $N$  (cf. Algorithm 1 and eqs. (2) and (6)) converges to a meaningful counterfactual  
 648 and yields nearly  $\ell_1$ -minimal changes.

648  
649

## A.1 CONVERGENCE TO LOCAL MINIMIZERS

650  
651  
652  
653

Let us stack all continuous perturbation variables into a single vector  $\theta = (\text{vec}(P), \text{vec}(N))$ , and write the soft objective Equation (2) (later for brevity we use  $L(\theta) = L(G; \theta)$ )

654  
655  
656

$$L(G; \theta) = L_{\text{pred}}(\Phi(G), E(G; \theta)) + \beta L_{\text{dist}}(\theta), \quad (10)$$

where

657  
658  
659

$$L_{\text{pred}}(\Phi, E(G; \theta)) = \text{CE}(\Phi(G), \Phi(E(G; \theta))) \quad (11)$$

for multi-class classification or

660  
661  
662

$$L_{\text{pred}}(\Phi, E(G; \theta)) = \text{BCE-Logits}(\Phi(G), \Phi(E(G; \theta))) \quad (12)$$

for multi-label classification, and

663  
664  
665

$$L_{\text{dist}}(\theta) = \|A - \hat{A}(\theta)\|_1 + \|X - X'(\theta)\|_1, \quad (13)$$

as in Equations (2), (7) and (8). To reflect our methodology, we optimize by projected gradient descent, as  $P$  is constrained to  $[0, 1]$ , while leaving  $N$  unconstrained:

666  
667  
668

$$\theta^{t+1} \leftarrow \begin{cases} P^{t+1} = \Pi_{[0,1]^{n^2}}(P^t - \eta \nabla_P L(P^t, N^t)), \\ N^{t+1} = N^t - \eta \nabla_N L(P^t, N^t) \quad (\text{no projection}). \end{cases} \quad (14)$$

669  
670  
671

where  $\Theta = [0, 1]^{n \times n} \times \mathbb{R}^{n \times f}$ , closed and convex parameter domain, enforces  $P \in [0, 1]^{n \times n}$  and  $N \in \mathbb{R}^{n \times f}$ .

672  
673  
674

**Assumption 1** Assume the loss  $L : \Theta \rightarrow \mathbb{R}$  satisfies:

675  
676  
677

1. *Differentiability:*  $L$  is smooth plus convex-nonsmooth (i.e.  $L_{\text{pred}}$  is  $C_1$ , and  $L_{\text{dist}}$  is proper, closed, convex, and can be smoothed by a fixed smooth approximation  $L_{\text{dist}, \epsilon}(\theta) = \lambda \sum_i \sqrt{\theta_i^2 + \epsilon^2}$  for a fixed  $\epsilon$ ).
2. *L-smoothness:*  $\|\nabla L(x) - \nabla L(y)\| \leq L\|x - y\|$  for all  $x, y \in \Theta$ . (This is the smoothness constant  $L$ ).
3. *Coercivity:*  $\{x \in \Theta : L(x) \leq c\}$  is bounded for any  $c \in \mathbb{R}$ .

678  
679  
680

**Theorem 1 (Convergence of PGD)** Under Assumption 1, if the step-size satisfies  $\eta \leq 1/L$ , then the iterates  $\theta_{t+1} = \Pi_\Theta(\theta_t - \eta \nabla L(\theta_t))$  obey

685  
686  
687

$$\sum_{t=0}^{\infty} \|\theta_{t+1} - \theta_t\|^2 < \infty \iff \lim_{t \rightarrow \infty} \|\theta^{t+1} - \theta^t\| = 0, \quad (15)$$

688  
689

and every limit point is Clarke-stationary.

690  
691  
692  
693  
694

and any limit point  $\theta^* = (P^*, N^*)$  fulfils the first-order optimality condition

$$0 \in \begin{pmatrix} \nabla_P L(P^*, N^*) \\ \nabla_N L(P^*, N^*) \end{pmatrix} + \begin{pmatrix} NC_{[0,1]^{n^2}}(P^*) \\ \{0\} \end{pmatrix}, \quad (16)$$

695  
696  
697

where  $NC$  is the normal-cone block encoding our constraint on  $P$ :  $v \in NC_{[0,1]^{n^2}}(P^*) \iff \langle v, P - P^* \rangle \leq 0 \quad \forall P \in [0, 1]^{n^2}$ .

698  
699  
700  
701

This means  $\theta^*$  is a stationary point of the soft objective (stationarity in  $N$  (zero gradient) and projected-stationarity in  $P$ ). Because  $L_{\text{dist}}$  has kinks at integer  $p$  entries ( $\ell_1$ ), we actually converge to a Clarke-stationary point. Clarke-stationary point means PGD still converges to a point where no small perturbation – even through the hard threshold – can locally decrease the loss. All standard PGD convergence guarantees for convex–nonsmooth+smooth composite objectives (e.g. see Bertsekas (1997)) carry over.

**Proof (derived from Bertsekas (1997) (ch. 2.3))**  $\Theta$  is closed and convex. Moreover, under our loss design the level sets  $\{\theta | L(\theta) \leq L(\theta_0)\}$  are bounded (or equivalently  $L$  is coercive), ensuring the standard projected-gradient-descent convergence arguments carry through.

A function  $L : \mathbb{R}^d \rightarrow \mathbb{R}$  is said to be  $L$ -smooth if its gradient is Lipschitz continuous with constant  $L$ . Concretely, for all  $\theta, \theta' \in \mathbb{R}^d$ ,

$$\|\nabla L(\theta') - \nabla L(\theta)\| \leq L\|\theta' - \theta\|.$$

Equivalently, this implies the Taylor-remainder bound

$$L(\theta') \leq L(\theta) + \nabla L(\theta)^\top (\theta' - \theta) + \frac{L}{2}\|\theta' - \theta\|^2,$$

for our purposes,  $L$  is  $L$ -smooth over the convex set  $\Theta$  ( $\theta, \theta' \in \Theta$ ). This says we can upper-bound the change in  $L$  by its first-order (linear) term plus a quadratic "error" that scale with  $\|\theta' - \theta\|^2$ .

Now set  $\theta' = \Pi_\Theta(\theta - \eta \nabla L(\theta))$  as the PGD update, that is: start at  $\theta$ , step "downhill" by  $\eta \nabla L(\theta)$ , then project back into the box  $\Theta$ .

We call

$$d(\theta) = \theta - \Pi_\Theta(\theta - \eta \nabla L(\theta))$$

the projected gradient (or "proximal residual"). It is the step we actually take, equivalently one can view  $\theta' - \theta = -d(\theta)$  as the update direction. When  $\|d(\theta)\|$  goes to zero, then  $\theta$  is a constrained stationary point, where "gradient+projection" induce no change.

Now, plugging  $\theta' = \Pi_\Theta(\theta - \eta \nabla L(\theta))$  into the smoothness bound gives two pieces:

**1. Linear piece:**  $\nabla L(\theta)^\top (\theta' - \theta) = -\nabla L(\theta)^\top d(\theta) \leq -\frac{1}{\eta}\|d(\theta)\|^2$ :

For any closed convex set  $\Theta$  and any point  $u$ , the projection  $\theta' = \Pi_\Theta(u)$  satisfies

$$(u - \theta')^\top (y - \theta') \leq 0 \quad \forall y \in \Theta,$$

as  $\theta'$  is defines as the unique minimizer of the convex problem  $\min_{y \in \Theta} \frac{1}{2}\|y - u\|^2$ , and the first-order (KKT) optimality condition for this strongly convex problem is exactly the equation above. Since  $\theta \in \Theta$ , we may plug  $y = \theta$  into this inequality:

$$(u - \theta')^\top (\theta - \theta') \leq 0,$$

and substitute  $u = \theta - \eta \nabla L(\theta)$  and  $\theta - \theta' = d(\theta)$ :

$$\begin{aligned} (\theta - \eta \nabla L(\theta) - \theta')^\top d(\theta) &\leq 0 \\ \implies (d(\theta) - \eta \nabla L(\theta))^\top &\leq 0. \end{aligned}$$

$$\begin{aligned} (d(\theta) - \eta \nabla L(\theta))^\top &= d(\theta)^\top d(\theta) - \eta \nabla L(\theta)^\top d(\theta) \\ &= \|d(\theta)\|^2 - \eta \nabla L(\theta)^\top d(\theta) \leq 0. \end{aligned}$$

$$\eta \nabla L(\theta)^\top d(\theta) \geq \|d(\theta)\|^2 \implies \nabla L(\theta)^\top d(\theta) \geq \frac{1}{\eta}\|d(\theta)\|^2.$$

Finally

$$\begin{aligned} \nabla L(\theta)^\top (\theta' - \theta) &= \nabla L(\theta)^\top (-d(\theta)) \\ &= -\nabla L(\theta)^\top d(\theta) \leq -\frac{1}{\eta}\|d(\theta)\|^2. \end{aligned}$$

This is the "linear piece" of the smoothness inequality giving us a guaranteed decrease proportional to  $\|d(\theta)\|^2$ .

**2. Quadratic piece:**  $\frac{L}{2}\|\theta' - \theta\|^2 = \frac{L}{2}\|d(\theta)\|^2$

Putting them together in the smoothness inequality:

$$L(\theta') \leq L(\theta) - \frac{1}{\eta}\|d(\theta)\|^2 + \frac{L}{2}\|d(\theta)\|^2.$$

756 Now if we choose the step size  $\eta \leq 1/L$ , then  
 757

$$758 -\frac{1}{\eta} + \frac{L}{2} \leq -\frac{1}{2\eta},$$

760 so overall

$$761 L(\theta') \leq L(\theta) - \frac{1}{2\eta} \|d(\theta)\|^2,$$

763 and rearranging and renaming  $\theta' = \theta^{t+1}$ ,  $\theta = \theta^t$ :

$$764 L(\theta^t) - L(\theta^{t+1}) \geq \frac{1}{2\eta} \|d(\theta^t)\|^2.$$

767 Telescoping (summing the inequalities) this descent bound over  $t = 0, \dots, T$  yields:  
 768

$$769 \sum_{t=0}^T [L(\theta^t) - L(\theta^{t+1})] \geq \frac{1}{2\eta} \sum_{t=0}^T [\|d(\theta^t)\|^2].$$

772 On the left the telescoping sum

$$773 L(\theta^0) - L(\theta^1) \geq \frac{1}{2\eta} \|d(\theta^0)\|^2,$$

$$774 L(\theta^1) - L(\theta^2) \geq \frac{1}{2\eta} \|d(\theta^1)\|^2,$$

775  $\dots$

776 collapses to  $L(\theta^0) - L(\theta^{T+1})$ , which is bounded above (since  $L$  is lower bounded). Hence the  
 777 right-hand sum of  $\|d(\theta^t)\|^2$  is finite, forcing  $\|d(\theta^t)\| \rightarrow 0$  as  $t \rightarrow \infty$ : so the projected gradient  
 778  $d(\theta^t) = \theta^t - \Pi(\theta^t - \eta \nabla L(\theta^t))$  vanishes as  $t \rightarrow \infty$ .

779 Any limit point  $\theta^*$  thus satisfies

$$780 \lim_{t \rightarrow \infty} \frac{1}{\eta} (\theta^t - \Pi_\Theta(\theta^t - \eta \nabla L(\theta^t))) = 0 \in \nabla L(\theta^*) + NC_\Theta(\theta^*),$$

781 i.e. the first-order (Karush-Juhn-Tucker) condition for stationarity on  $\Theta$ :  $NC_\Theta(\theta^*)$  is the normal  
 782 cone of the convex set  $\Theta$  at  $\theta^*$  that collects all vectors:

$$783 \mathbf{v} \in NC_\Theta(\theta^*) \iff \mathbf{v}^\top (\mathbf{y} - \theta^*) \leq 0 \quad \forall \mathbf{y} \in \Theta,$$

784 there exists some normal vector  $\mathbf{v} \in NC_\Theta(\theta^*)$  such that  $\nabla L(\theta^*) + \mathbf{v} = 0$ , or equivalently, it  
 785 is not possible to find any feasible direction  $\mathbf{y} - \theta^*$  along which the directional derivative of  $L$ ,  
 786  $\nabla L(\theta^*)^\top d$ , satisfies  $\nabla L(\theta^*)^\top d < 0$ , hence no feasible direction would decrease the loss further. In  
 787 other words, this is a constrained stationary point under the box constraint  $\Theta$  – no small perturbation  
 788 inside  $\Theta$  can locally lower the objective.

789 Here we assumed  $\eta \leq 1/L$ . In practice, however, computing  $L$  exactly for our GNN’s  
 790 prediction-loss is infeasible. Instead, we select the step size ( $\alpha$  is  $\eta$  in Algorithm 1) via a grid  
 791 search in  $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ . All choices empirically respected the surrogate smoothness  
 792 bound.

## 793 A.2 HANDLING NON-SMOOTH THRESHOLDING VIA SURROGATE GRADIENTS

794 Let  $P \in [0, 1]^{n \times n}$  be our continuous edge-importance matrix. We obtain a hard adjacency  
 795

$$796 \hat{A} = 1[P > 0.5] \in \{0, 1\}^{n \times n}$$

797 for the forward GNN pass. To propagate gradients back to  $P$ , we use the common PyTorch “detach”  
 798 trick (an instance of the straight-through estimator Bengio et al. (2013))

```
800 mask = (P >= 0.5).float() # no gradient
801 P_hard = P + (mask - P).detach() # forward uses mask; backward flows into P
802 outputs = GNN(P_hard, features)
```

803 Listing 1: STE-based hard-thresholding in PyTorch

In effect, the backward pass treats the binarization as if it were the identity function. This can be seen as a limiting case of using a steep sigmoid surrogate  $\sigma_\alpha(x) = 1/(1+e^{-\alpha(x-0.5)})$  with  $\alpha \rightarrow \infty$ . Under this STE, the composite loss

$$L(P) = L_{\text{pred}}(\hat{A}, X) + \beta \|P - P_0\|_1$$

is differentiable almost everywhere in  $P$ , and convergence to a Clarke-stationary point follows from standard projected-gradient arguments.  $P_0$  is the original adjacency mask.

Note that STE is a heuristic used in practice, and the previous convergence proofs refer to the continuous/relaxed problem, not the exact gradients through hard thresholding.

Note that we replace the non-differentiable hard threshold by the identity in the backward pass (i.e. STE). Using the identity in the backward pass induces a bias (dependent on the size of the downstream gradient) by propagating nonzero gradients where the true hard-threshold has zero derivative, whereas zeroing those gradients would irreversibly freeze masked entries – hence, to mitigate this, one may use a temperature-controlled sigmoid  $\sigma_\alpha$  (or even an annealed  $\alpha$ ) that smoothly trades off bias and trainability.

### A.3 $\ell_1$ -MINIMALITY BOUND OF PERTURBATIONS

Let again,

$$\theta = (\text{vec}(P), \text{vec}(N)), \quad \theta_0 = (\text{vec}(A), \text{vec}(1_{n \times f})),$$

and the well-known soft objective

$$L(G; \theta) = L_{\text{pred}}(\Phi(G), E(G; \theta)) + \beta \|\theta - \theta_0\|_1,$$

**Lemma 1** Assume  $L_{\text{pred}}$  is bounded below and set  $\Delta = L_{\text{pred}} - \inf_{\Theta} L_{\text{pred}}$ , the difference of  $L_{\text{pred}}$  with the infimum (greatest lower bound) of the prediction loss  $L_{\text{pred}}$  over all feasible  $\theta \in \Theta = [0, 1]^{n \times n} \times \mathbb{R}^{n \times f}$ , the best possible point in  $\Theta$  achievable by  $\theta^*$ . Then any stationary point  $\theta^*$  of  $L$  satisfies

$$\|\theta^* - \theta_0\|_1 \leq \frac{\Delta}{\beta},$$

where  $\theta_0$  is the original unperturbed  $(P, N)$ -vector.

**Proof 1** Since  $\theta_0 \in \Theta$ , stationarity (or just minimality) of  $\theta^*$  gives

$$L(\theta^*) \leq L(\theta_0) \implies L_{\text{pred}}(\theta^*) + \beta \|\theta^* - \theta_0\|_1 \leq L_{\text{pred}}(\theta_0).$$

Rearranging,

$$\begin{aligned} \beta \|\theta^* - \theta_0\|_1 &\leq L_{\text{pred}}(\theta_0) - L_{\text{pred}}(\theta^*) \\ &\leq L_{\text{pred}}(\theta_0) - \inf_{\Theta} L_{\text{pred}} = \Delta \end{aligned}$$

Dividing by  $\beta$ , we get the clean bound

$$\|\theta^* - \theta_0\|_1 \leq \frac{\Delta}{\beta}.$$

Thus, choosing  $\beta \geq \Delta/\epsilon$  forces  $\|\theta^* - \theta_0\|_1 \leq \epsilon$ , (i.e. the size of our edits (in  $\ell_1$ ) is at most the ratio of “how much we can lower the prediction loss” over “how costly each unit of edit is.”

**Practical choice of  $\beta$ .** In theory, the bound

$$\|\theta^* - \theta_0\|_1 \leq \frac{\Delta}{\beta}, \quad \Delta = L(\theta_0) - L(\theta^*)$$

guides  $\beta$  to achieve a target  $\ell_1$ -norm (sparsity). In practice  $\Delta$  is unknown, so we select  $\beta$  empirically – e.g. via grid search or cross-validation – by monitoring the resulting  $\|\theta^*\|_1$  (or edge count) and choosing the smallest  $\beta$  that attains the desired sparsity level  $\epsilon$ .

864 A.4 EDGE-INSERTION AND EDGE-DELETION CONDITION  
865

866 We now derive a simple, quantitative criterion under which a zero-entry  $p_{ij} = 0$  will be driven  
867 strictly positive and a one-entry  $p_{ij} = 1$  will be driven to zero, by projected gradient descent on the  
868 soft objective Equation (10).

869 Recall that  $L_{\text{dist}}$  penalizes deviations of both edges and node features, but here we focus on its  
870 effect on the edge variable  $p_{ij}$ .

871 We consider the soft objective  
872

$$873 L(P, N) = L_{\text{pred}}(P, N) + \beta L_{\text{dist}}(P, N),$$

874 where  
875

- 876 •  $P = [p_{ij}] \in \{0, 1\}^{n \times n}$  is the *final* binary adjacency mask,
- 877 •  $N$  is the node-feature matrix,
- 878 •  $\bar{P} = [\bar{p}_{ij}] \in [0, 1]^{n \times n}$  is the *original* continuous adjacency (here we assume no noise, in  
879 practice we add a small Gaussian noise to  $\bar{P}$ , which does not materially change the  $\pm 1$   
880 subgradients at the boundaries),
- 881 •  $L_{\text{dist}}(P, N)$  includes the edge penalty  $\sum_{i,j} |p_{ij} - \bar{p}_{ij}|$ , Equation (8) (plus any feature-  
882 penalty terms).

883 To decide whether an edge coordinate  $p_{ij}$  will switch its value under projected gradient descent, we  
884 inspect the one-dimensional update

$$885 p_{ij}^{(t+1)} = \Pi_{\{0,1\}} \left( p_{ij}^{(t)} - \eta(g_{ij} + \beta d_{ij}) \right),$$

886 where  
887

$$888 g_{ij} = \frac{\partial L_{\text{pred}}}{\partial p_{ij}}, \quad d_{ij} = \frac{\partial}{\partial p_{ij}} |p_{ij} - \bar{p}_{ij}| = \text{sign}(p_{ij} - \bar{p}_{ij}).$$

889 By the sign-definition,  
890

$$891 d_{ij} = \begin{cases} +1, & p_{ij} > \bar{p}_{ij}, \\ -1, & p_{ij} < \bar{p}_{ij}, \end{cases} \quad d_{ij} \in [-1, 1] \text{ if } p_{ij} = \bar{p}_{ij}.$$

892 **Insertion.** At the insertion boundary  $p_{ij} = 0$  with continuous  $\bar{p}_{ij} > 0$ , we have  $d_{ij} = -1$  if  
893  $\bar{p}_{ij} > 0$ , but to capture the switch from 0 to 1 we consider the subgradient at the boundary:  
894

$$895 p_{ij}^{(t+1)} = 1 \iff 0 - \eta(g_{ij} + \beta d_{ij}) \leq -1 \iff -g_{ij} > \beta d_{ij}.$$

896 Since at the 0-boundary we take  $d_{ij} = +1$  for the “hardest” subgradient,  
897

$$898 -g_{ij} > \beta$$

900 is required for insertion.  
901

902 **Deletion.** At the deletion boundary  $p_{ij} = 1$  with  $\bar{p}_{ij} < 1$ , we similarly take  $d_{ij} = -1$  and find  
903

$$904 p_{ij}^{(t+1)} = 0 \iff 1 - \eta(g_{ij} + \beta d_{ij}) \geq 1 \iff g_{ij} > \beta.$$

905 In both cases, an edge flips exactly when its *marginal benefit* or *cost* in the prediction loss exceeds  
906 the fixed penalty  $\beta$ , yielding a transparent sparsity threshold.  
907

908 ROLE OF THE  $\Gamma$  MATRIX  
909

910 Rather than using a uniform  $\ell_1$  penalty  $\beta \sum_{i,j} |\bar{P}_{ij} - A_{ij}|$ , we shift the target by  $\Gamma$  and write  
911

$$912 \min_{\bar{P}} L_{\text{pred}}(\bar{P}) + \beta \sum_{i,j} |\bar{P}_{ij} - (A_{ij} + \gamma_{ij})|.$$

918 By simple algebraic manipulation this is equivalent (up to an additive constant) to  
 919

$$920 \min_{\bar{P}} L_{\text{pred}}(\bar{P}) + \beta \sum_{i,j} w_{ij} |\bar{P}_{ij} - A_{ij}|, \quad w_{ij} \equiv \frac{1}{1 + \gamma_{ij}}.$$

923 In the latter form, the KKT subgradient for each  $(i, j)$  is  
 924

$$925 \underbrace{\frac{\partial L_{\text{pred}}}{\partial \bar{P}_{ij}}}_{\text{model term}} + \beta w_{ij} \text{sign}(\bar{P}_{ij} - A_{ij}),$$

928 so a larger  $\gamma_{ij} \Rightarrow$  smaller  $w_{ij} \Rightarrow$  smaller magnitude of the regularizer  $\Rightarrow$  cheaper to flip edge  $(i, j)$ ,  
 929 all **without** changing the global  $\beta$ .  
 930

### 931 A.5 SMOOTHNESS OF THE PREDICTION LOSS

933 We next show, at a high level, that our prediction loss  $L_{\text{pred}}(P, N)$  admits an  $L$ -Lipschitz continuous  
 934 gradient (i.e. is  $L$ -smooth) in the perturbation variables  $(P, N)$ .  
 935

936 **Lemma 2** *Let  $f_\theta$  be a GNN with fixed weights  $\theta$ , built from layers that are each Lipschitz continuous  
 937 (e.g. linear transforms, neighbor-aggregation, and elementwise activations such as ReLU). Define*

$$939 L_{\text{pred}}(P, N) = L(f_\theta(A \odot P, X + N), y),$$

940 where  $L$  is a twice-differentiable loss (e.g. cross-entropy). Then there exists a constant  $L > 0$   
 941 depending only on  $\theta$ , the GNN architecture, and  $L$ , such that for all  $(P, N)$  and  $(P', N')$ ,

$$943 \|\nabla L_{\text{pred}}(P, N) - \nabla L_{\text{pred}}(P', N')\| \\ 944 \leq L \|(P, N) - (P', N')\|.$$

946 Hence  $L_{\text{pred}}$  is  $L$ -smooth.  
 947

948 **Proof 2 (High-Level)** Note that each GNN layer can be written as the composition of:

- 950 • A linear map in  $(P, N)$  (adjacency mask enters via  $A \odot P$ , features via  $X + N$ ), which is  
 951 Lipschitz.
- 952 • An elementwise activation (e.g. ReLU or smooth ReLU), which is 1-Lipschitz.
- 953 • A final differentiable loss  $L$ , whose gradient is Lipschitz in the model's output.

956 By the chain rule, the gradient of the overall map  $(P, N) \mapsto L(f_\theta(\cdot), y)$  is Lipschitz, with constant  
 957

$$958 L \leq L_L \prod_{\ell=1}^L L_{\text{layer}, \ell},$$

961 where  $L_L$  is the loss's Lipschitz constant and each  $L_{\text{layer}, \ell}$  upper-bounds the Lipschitz constant of  
 962 layer  $\ell$ .  
 963

964 **Remark.** In practice we do not require the exact value of  $L$ , only the existence of such a bound to  
 965 invoke standard convergence results for projected gradient methods on smooth + nonsmooth objectives.  
 966

### 968 A.6 CHOOSING THE TRADE-OFF

970 In practice,  $\beta$  controls sparsity versus fidelity. A grid search (cf §5.3) typically selects  $\beta \in [0.1, 1]$   
 971 to yield few edits while guaranteeing  $\Phi(G') \neq \Phi(G)$ . The step size ( $\alpha$  is  $\eta$  in Algorithm 1) is as  
 well selected via a grid search in  $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ .

972 A.7 COMPUTATIONAL COMPLEXITY.  
973974 Each iteration of Algorithm 1 performs:  
975

- 976 • one forward+backward pass through the GNN, which on a graph with
- $n$
- nodes,
- $|E|$
- edges,
- 
- 977 hidden-dimensionality
- $d$
- and feature-dimensionality
- $f$
- takes

978 
$$O(|E| d + n d f),$$
  
979

980 since message-passing scales linearly in edges and feature multiplications scale in  $n f \times d$ ;  
981

- 982 • elementwise thresholding of the perturbation masks
- $P$
- and
- $N$
- , costing

983 
$$O(|E| + n f),$$
  
984

985 as we only inspect each edge entry and each feature entry once. And the STE back-pass  
986 adds no asymptotic overhead.  
987

Thus the dominant per-iteration cost is

989 
$$O(|E| d + n d f),$$
  
990

i.e. linear in the graph size under standard GNN architectures, ensuring the method scales to large  
991 graphs. Recall that  $K$  total iterations are performed in Algorithm 1.  
992993 A.8 EXTENSION TO WEIGHTED AND DIRECTED GRAPHS  
994995 Our theoretical developments extend straightforwardly to weighted or directed graphs. For weighted  
996 graphs, replace the binary mask  $P \in \{0, 1\}^{n \times n}$  by a continuous mask  $P \in [a, b]^{n \times n}$  (e.g.  $[0, 1]$ ),  
997 and substitute the unweighted  $\ell_1$  distance  
998

999 
$$\sum_{i,j} |p_{ij} - \bar{p}_{ij}|$$
  
1000

1001 with a weighted version  
1002

1003 
$$\sum_{i,j} w_{ij} |p_{ij} - \bar{p}_{ij}|,$$
  
1004

1005 where  $w_{ij} > 0$  can reflect edge-specific costs. All projected gradient and subgradient arguments  
1006 carry over by projecting onto the box  $[a, b]$  and using the weighted sign subgradient for the  $\ell_1$   
1007 term. For directed graphs, simply treat  $(i, j)$  and  $(j, i)$  as distinct entries in  $P$ , and the same “flip  
1008 when  $|\partial L_{\text{pred}} / \partial p_{ij}| > \beta w_{ij}$ ” threshold holds. Convergence to a KKT-stationary point and the  
1009  $\ell_1$ -minimality bound remain valid with these modifications, since they rely only on box-constraints  
1010 and separable  $\ell_1$  penalties.  
10111012 A.9 SEARCH-SPACE EXPRESSIVITY  
10131014 By jointly parameterizing the adjacency perturbation  $\bar{P} \in [0, 1]^{n \times n}$  and feature perturbation  
1015  $N \in \mathbb{R}^{n \times f}$  (see Section 3 and algorithm 1), our gradient-based explainer can, in principle, reach  
1016 any discrete graph–feature configuration (once binarized via sigmoid and thresholding). In practice,  
1017 however, the non-convex loss landscape can trap plain gradient descent in a local minimum in the  
1018 basin around the original graph – favoring small, local edits. To counteract this, we add a small  
1019 amount of Gaussian noise to  $\bar{P}$  at initialization. These random perturbations diversify the initial  
1020 gradient directions and help the optimizer “hop” across low-gradient regions, leading to a richer  
1021 exploration of structurally distinct counterfactuals without sacrificing convergence.  
10221023 Hence, even when the optimization converges, the explainer may fail to reach a counterfactual due  
1024 to either local minima or an ill-conditioned decision boundary in the oracle – e.g., the true label  
1025 region may be disjoint, vanishingly thin, or entirely absent near the input. This is reflected in our ex-  
periments: some generated counterfactuals either fail to flip the label or result in out-of-distribution  
inputs. These issues could be mitigated by improving the oracle’s inductive bias and generalization  
capacity.  
1026

## 1026 B DATASETS 1027

1028 To evaluate our approach against state-of-the-art GCE methods, we conducted experiments on 14  
1029 datasets (9 real and 5 synthetic), covering diverse domains and involving multi-class graph classifi-  
1030 cation tasks. Table 7 illustrates the characteristics of the datasets used in this paper.

1031 **Table 7:** Dataset characteristics.  
1032

	TCR	TG	BAS	MUTAG	BZR	COX2	AIDS	BBBP	ENZYMES	PROTEINS	Fingerprint	COLLAB	COLOR-3	TRIANGLES
Avg # of Nodes	28	64	64	17.93	35.75	41.22	15.69	25.95	32.63	39.06	74.49	61.31	20.85	
Node Attr.	—	1	4	—	3	3	4	18	29	2	2.06	4	—	
Avg # of Edges	27.75	65.01	64.01	19.79	38.36	43.45	16.20	24.06	62.14	72.82	5.76	2457.78	91.03	32.74
# of Graphs	5000	5000	5000	188	405	467	2000	2039	600	1113	2149	5000	105000	45000
# of Classes	2	2	2	2	2	2	2	2	6	2	15	3	11	10
Motif Category	Cycle	Grid	House	—	—	—	—	—	—	—	—	Color	Triangle	Synthetic
Oracle Test. Acc.	100.00%	99.72%	100.00%	88.30%	99.01%	98.72%	99.80%	99.41%	95.67%	98.02%	74.97%	69.60%	91.68%	99.18%

1033 AIDS Riesen & Bunke (2008) consists of graphs representing molecular compounds. These graphs  
1034 are derived from the AIDS Antiviral Screen Database of Active Compounds. This data set consists  
1035 of two classes (active and inactive), which represent molecules with or without activity against HIV.  
1036 The molecules are converted into graphs in a straightforward manner by representing atoms as nodes  
1037 and the covalent bonds as edges. Nodes are labeled with the number of the corresponding chemical  
1038 symbol and edges by the valence of the linkage. There are 2,000 elements in total (1,600 inactive  
1039 elements and 400 active elements).

1039 **BAShapes** Ying et al. (2019) is a synthetic dataset consisting of a base graph and motifs connected  
1040 on the base. The base graph is a Barabasi-Albert (BA) graph with house-shaped motif attached to  
1041 it. The resulting graph is further perturbed by adding  $0.1N$  random edges. Following the generation  
1042 done in Lucic et al. (2022), there are 8 nodes on the base graph with 5 edges connecting them. Each  
1043 base graph has 7 motives connected to it.

1044 **BBBP** (Blood-Brain Barrier Penetration) Martins et al. (2012) is a dataset widely used in drug  
1045 discovery and neurological research to develop machine learning models that predict blood-brain  
1046 barrier permeability. The blood-brain barrier is a protective membrane that shields the central nervous  
1047 system by regulating the passage of solutes. Its presence is a critical consideration in drug  
1048 development, whether for designing molecules that target the central nervous system or for identifying  
1049 compounds that should be restricted from crossing the barrier. BBBP contains binary labels  
1050 for 2,053 curated molecules, indicating whether a compound can penetrate the blood-brain barrier.  
1051 Specifically, 1,570 molecules can penetrate the barrier, while 483 cannot.

1052 **BZR and COX2** Sutherland et al. (2003) have different molecular compounds, 467 cyclooxygenase-  
1053 2 (COX-2) inhibitors and 405 benzodiazepine receptor (BZR) ligands, respectively. These datasets  
1054 are widely used in quantitative structure-activity relationships (QSAR) studies, which attempt to  
1055 correlate the biological activities of compounds with their structural attributes, to help elucidate the  
1056 mechanism by which they act and to predict the activities of novel derivatives.

1057 **COLLAB** is a social network dataset comprising 5000 scientific collaborations derived from 3 public  
1058 collaboration datasets Leskovec et al. (2005), namely, High Energy Physics, Condensed Matter  
1059 Physics and Astro Physics.

1060 **COLORS-3** Knyazev et al. (2019) is a synthetic dataset consisting of 10500 random graphs with  
1061 11 classes where features of each node are assigned to one of the three colors (red, green or blue),  
1062  $\mathbf{p} \in \mathbb{R}^3$ . The dataset is extended to higher  $n$ -dimensional cases  $\mathbf{p} \in \mathbb{R}^n$ .

1063 **ENZYMES** Borgwardt et al. (2005) is a bio-informatics dataset consisting of a dataset of 600 enzymes,  
1064 constructed from the Protein Data Bank Berman et al. (2000) and labeled with their corresponding  
1065 enzyme class labels from the BRENDA enzyme database Schomburg et al. (2004). It includes 100 proteins  
1066 from each of six classes (EC 1, EC 2, EC 3, EC 4, EC 5, EC 6), which represent proteins out of the six enzyme  
1067 commission top level hierarchy (EC classes). The proteins are converted into graphs by representing the secondary  
1068 structure elements of a protein with nodes and edges of an attributed graph. Nodes are labeled with their type (helix, sheet, or loop)  
1069 and their amino acid sequence. Every node is connected with an edge to its three nearest neighbors in space. Edges  
1070 are labeled with their type and the distance they represent in angstroms.

1071 **Fingerprint** Riesen & Bunke (2008) is a computer vision dataset consisting of 2149 fingerprints that  
1072 are converted into graphs by filtering the images and extracting regions that are relevant Neuhaus &

1080 Bunke (2005). In order to obtain graphs from fingerprint images, the relevant regions are binarized  
 1081 and a noise removal and thinning procedure is applied. This results in a skeletonized representation  
 1082 of the extracted regions. Ending points and bifurcation points of the skeletonized regions are  
 1083 represented by nodes. Additional nodes are inserted in regular intervals between ending points and  
 1084 bifurcation points. Finally, undirected edges are inserted to link nodes that are directly connected  
 1085 through a ridge in the skeleton. Each node is labeled with a two-dimensional attribute giving its  
 1086 position. The edges are attributed with an angle denoting the orientation of the edge with respect to  
 1087 the horizontal direction.

1088 **MUTAG** Debnath et al. (1991) is a widely used dataset consisting of 188 nitroaromatic chemi-  
 1089 cal compounds divided into two classes according to their mutagenic effect on a *Salmonella ty-*  
 1090 *phimurium* bacterium. Input graphs represent chemical compounds, vertices stand for atoms and  
 1091 are labeled by the atom type (represented by one-hot encoding), while edges between vertices rep-  
 1092 resent bonds between the corresponding atoms.

1093 **PROTEINS** Borgwardt et al. (2005) is a bio-informatics dataset comprising 1113 proteins from the  
 1094 dataset of enzymes (59%) and non-enzymes (41%) Dobson & Doig (2003). Proteins are modeled  
 1095 as feature vectors which indicate for each amino acid its fraction among all residues, its fraction of  
 1096 the surface area, the existence of ligands, the size of the largest surface pocket and the number of  
 1097 disulphide bonds.

1098 **Tree-Cycles (TCR)** Ying et al. (2019) is an emblematic synthetic dataset. Each instance constitutes  
 1099 a graph comprising a central tree motif and multiple cycle motifs connected through singular edges.  
 1100 The dataset encompasses two distinct classes: i.e., one for graphs without cycles (0) and another for  
 1101 graphs containing cycles (1). The TC also allows control of the number of nodes, the number of  
 1102 cycles, and the number of nodes in them.

1103 **Tree-Grid (TG)** is a synthetic dataset similar to *Tree-Cyles*, in which n-by-n grid motifs are attached  
 1104 to the main tree motif in place of cycle motifs. We used 5000 graphs with 64 nodes and randomly  
 1105 attached a 3-by-3 grid.

1106 **TRIANGLES** Knyazev et al. (2019) is a synthetic dataset comprising 45000 graphs. The task is to  
 1107 count the number of triangles in the graph, node degree features as one-hot vectors are added to all  
 1108 graphs, so that the oracle model can exploit both graph structure and features.

## 1112 C DETAILED EXPERIMENTAL SETUP

### 1114 C.1 ORACLES AND TRAINING

1116 The oracle Graph Convolutional Network (GCN) models for different datasets were trained with  
 1117 varying architectures and hyperparameters, all using RMSprop optimization and CrossEntropyLoss,  
 1118 with an 80%-20% train-test split. Most datasets (BAS, AIDS, BZR, COX2, TCR, TG, and EN-  
 1119 ZYMES) used 3 convolutional layers and 1 dense layer, BBBP (5 conv, 3 dense), COX2, MUTAG  
 1120 (2 conv, 2 dense), TRIANGLES (5 conv, 2 dense), COLORS-3 (2 conv, 1 dense), COLLAB, PRO-  
 1121 TEINS (3 conv, 2 dense), Fingerprint (5 conv, 5 dense) . Learning rates varied between 0.001 and  
 1122 0.01, with training epochs ranging from 20 to 1000, and batch sizes typically 32, except BBBP,  
 1123 ENZYMES, PROTEINS (64), Fingerprint (128), COLLAB (256), TRIANGLES (1024). The ex-  
 1124 periments were conducted on an 8GB NVIDIA RTX 4060 GPU with an Intel Core i9-13900HX and  
 1125 32GB RAM, while node explanation experiments used a 4GB NVIDIA RTX 3050 GPU with an  
 1126 Intel Core i7-11800H and 16GB RAM.

### 1129 C.2 HYPERPARAMETERS SEARCH

1131 We perform a hyperparameter search according to these combination of parameters:  $\alpha \in$   
 1132  $\{0.001, 0.01, 0.1, 1\}$ ,  $\beta \in \{0, 0.5, 1\}$ ,  $K \in \{50, 100, 500, 5000\}$ ,  $\gamma \in \{0, 0.1, 0.01, 0.001\}$ . The  
 1133 optimal configuration found was  $\alpha = 0.1$ ,  $\beta = 0.5$ ,  $K = 50$  and  $\gamma = 0$ ; if accuracy is preferred over  
 other metrics (see Section 4.4), then  $\gamma = 0.01$  performed best over the other choices.

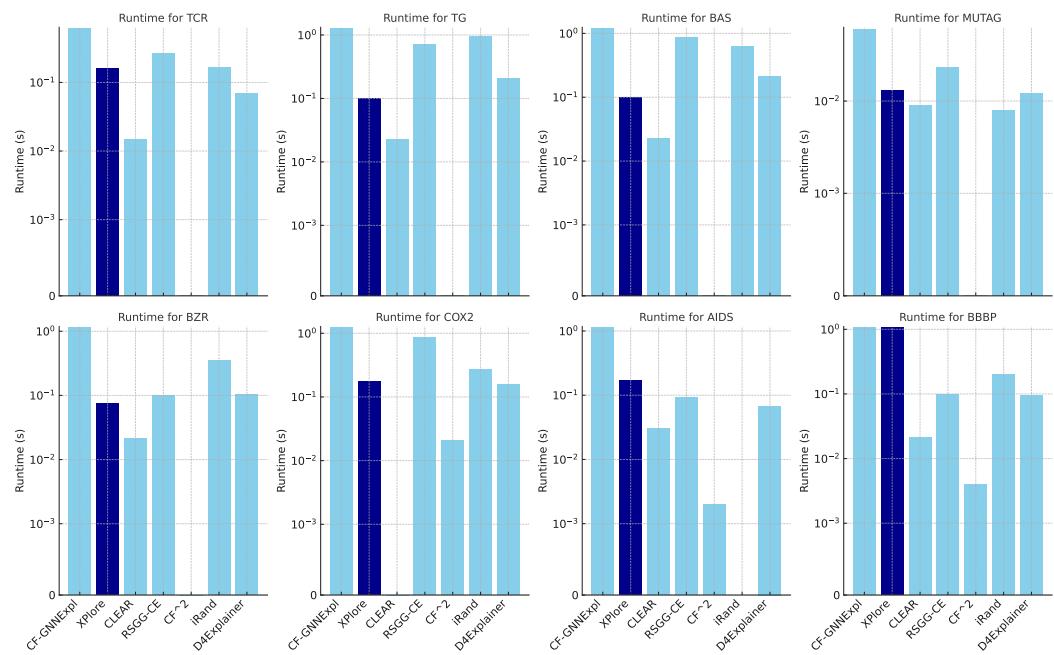


Figure 4: Runtime (s) at inference across multiple datasets. **Xplore (dark blue)** maintains a competitive runtime, making it the most reliable explainer in practice.

### C.3 METRICS

**Oracle Accuracy** tells us how accurate the model  $\Phi$  is at predicting the ground truth labels,  $\frac{1}{N} \sum_{i=1}^N \mathbf{1}[\Phi(G_i) = y_i]$ .

**Validity** illustrates the capability of the explainer to cross the decision boundary of  $\Phi$  given an input  $G_i$ ,  $\frac{1}{N} \sum_{i=1}^N \mathbf{1}[\Phi(G'_i) \neq \Phi(G_i)]$ .

**Fidelity** measures how different the generated counterfactual explanations are to the original instance,  $\frac{1}{N} \sum_{i=1}^N \mathbf{1}(\Phi(G_i) = y_i) - \mathbf{1}(\Phi(G'_i) = y_i)$ .

**Sparsity** measures the ratio between the number of structural features modified to obtain a valid counterfactual explanation and the number of structural features in the original instance.

**Graph Edit Distance (GED)** tracks the modification applied to the graph of the valid counterfactual explanation, i.e., number of added/removed nodes/edges.

**Oracle Calls** is the number of times the explainer calls  $\Phi$  to generate a valid counterfactual explanation.

•**Runtime to counterfactual:** the time in seconds required by the explainer to generate a valid counterfactual explanations.

## D ADDITIONAL EXPERIMENTS

Table 8 shows all the performances of Xplore against the SoTA methods. Figure 4 shows the runtime of all explainers, showcasing that Xplore is lightweight and only second to CLEAR. However, it shows that Xplore is more efficient than its baseline CF-GNNExpl. which is, notice, more restrictive in perturbing the input graph (i.e., only allowing edge deletions). Contrarily, Xplore allows for both edge additions and removals as well as node perturbations.

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

Table 8: Extended comparison of XPlore with SoTA methods on the used datasets. Standard deviation is reported. Bold is best-performing; underline is second-best. Highlighted columns depict the most important metrics as they evaluate the faithfulness of the explainer to produce counterfactuals. When an explainer cannot produce valid counterfactuals in a dataset, it does not make sense to evaluate the other metrics (see –).

Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
TCR	iRand	27.92% ± 44.86%	0.279 ± 0.449	<b>0.062</b> ± 0.023	10.666 ± 0.084
	CF <sup>2</sup>	50.04% ± 50.00%	0.500 ± 0.500	0.491 ± 0.000	<b>0.000</b> ± 0.000
	CLEAR	50.68% ± 50.00%	0.507 ± 0.500	3.037 ± 0.193	<b>0.000</b> ± 0.004
	RSGG-CE	67.90% ± 46.69%	0.679 ± 0.467	0.305 ± 0.057	8.890 ± 0.169
	D4Explainer	44.82% ± 49.73%	0.448 ± 0.497	0.491 ± 0.000	<b>0.000</b> ± 0.012
	CF-GNNExpl	32.40% ± 46.80%	0.324 ± 0.460	0.745 ± 0.000	46.000 ± 0.014
XPlore		<b>100.00%</b> ± 0.00%	<b>1.000</b> ± 0.000	2.708 ± 0.90	<b>3.593</b> ± 0.009
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
BAS	iRand	50.70% ± 50.00%	0.507 ± 0.500	<b>0.112</b> ± 0.044	43.600 ± 0.293
	CF <sup>2</sup>	45.78% ± 49.82%	0.457 ± 0.499	0.496 ± 0.000	<b>0.000</b> ± 0.001
	CLEAR	50.96% ± 49.99%	0.510 ± 0.500	6.007 ± 0.325	<b>0.000</b> ± 0.035
	RSGG-CE	91.04% ± 28.56%	0.910 ± 0.280	0.331 ± 0.108	23.585 ± 1.069
	D4Explainer	44.56% ± 49.70%	0.446 ± 0.497	0.499 ± 0.000	<b>0.000</b> ± 0.014
	CF-GNNExpl	44.18% ± 49.66%	0.442 ± 0.497	0.748 ± 0.000	46.000 ± 0.283
XPlore		<b>100.00%</b> ± 0.00%	<b>1.000</b> ± 0.000	7.600 ± 7.557	<b>2.514</b> ± 0.047
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
BZR	iRand	27.16% ± 44.48%	0.262 ± 0.451	<b>0.104</b> ± 0.045	25.373 ± 0.213
	CF <sup>2</sup>	19.75% ± 39.81%	0.188 ± 0.403	0.516 ± 0.007	<b>0.000</b> ± 0.002
	CLEAR	60.49% ± 48.89%	0.595 ± 0.503	4.937 ± 0.843	<b>0.000</b> ± 0.012
	RSGG-CE	21.23% ± 40.90%	0.207 ± 0.414	<b>0.258</b> ± 0.003	2.000 ± 0.015
	D4Explainer	20.00% ± 40.00%	0.190 ± 0.403	0.523 ± 0.000	<b>0.000</b> ± 0.015
	CF-GNNExpl	19.75% ± 39.81%	0.188 ± 0.403	0.758 ± 0.003	46.000 ± 0.224
XPlore		<b>100.00%</b> ± 0.00%	<b>0.980</b> ± 0.198	6.595 ± 3.616	2.244 ± 0.029
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
AIDS	iRand	0.00% ± 0.00%	0.000 ± 0.000	—	—
	CF <sup>2</sup>	0.10% ± 3.16%	0.001 ± 0.032	3.870 ± 0.011	<b>0.000</b> ± 0.004
	CLEAR	16.75% ± 37.34%	0.168 ± 0.373	13.442 ± 1.834	<b>0.000</b> ± 0.008
	RSGG-CE	19.80% ± 39.85%	0.198 ± 0.398	<b>0.258</b> ± 0.004	2.000 ± 0.027
	D4Explainer	0.10% ± 0.032%	0.001 ± 0.032	0.488 ± 0.031	<b>0.000</b> ± 0.001
	CF-GNNExpl	0.10% ± 3.16%	0.001 ± 0.032	0.745 ± 0.005	46.000 ± 0.059
XPlore		<b>32.30%</b> ± 45.96%	<b>0.323</b> ± 0.460	2.445 ± 2.002	5.851 ± 0.058
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
ENZYMEs	iRand	26.67% ± 44.22%	0.238 ± 0.445	<b>0.074</b> ± 0.046	29.069 ± 0.046
	CF <sup>2</sup>	68.33% ± 46.52%	0.633 ± 0.502	0.657 ± 0.034	<b>0.000</b> ± 0.004
	CLEAR	83.17% ± 43.42%	0.797 ± 0.402	27.942 ± 24.021	<b>0.000</b> ± 0.017
	RSGG-CE	98.33% ± 12.80%	<b>0.930</b> ± 0.292%	0.396 ± 0.152	19.400 ± 2.801
	D4Explainer	68.00% ± 46.65%	0.632 ± 0.499	0.663 ± 0.049	<b>0.000</b> ± 0.066
	CF-GNNExpl	68.33% ± 46.52%	0.633 ± 0.502	0.657 ± 0.034	2.000 ± 0.002
XPlore		<b>100.00%</b> ± 0.00%	<b>0.942</b> ± 0.291	2.394 ± 1.538	2.615 ±
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
Fingerprint	iRand	0.09% ± 3.05%	0.001 ± 0.030%	<b>0.031</b> ± 0.000	3.500 ± 0.007
	CF <sup>2</sup>	24.52% ± 43.02%	0.133 ± 0.400	0.412 ± 0.052	<b>0.000</b> ± 0.000
	CLEAR	72.73% ± 44.53%	0.515 ± 0.537	15.517 ± 14.400	<b>0.000</b> ± 0.001
	RSGG-CE	90.46% ± 29.38%	0.653 ± 0.566	0.329 ± 0.142	6.479 ± 0.048
	D4Explainer	24.52% ± 43.02%	0.133 ± 0.400	0.412 ± 0.052	<b>0.000</b> ± 0.014
	CF-GNNExpl	24.52% ± 43.02%	0.133 ± 0.400	0.412 ± 0.052	2.000 ± 0.002
XPlore		<b>100.00%</b> ± 0.00%	<b>0.730</b> ± 0.487	0.872 ± 0.591	4.168 ± 0.052
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
COLORS-3	iRand	42.99% ± 49.51%	0.300 ± 0.571	<b>0.076</b> ± 0.064	66.175 ± 62.151
	CF <sup>2</sup>	52.07% ± 49.96%	0.398 ± 0.568	0.598 ± 0.046	<b>0.000</b> ± 0.001
	CLEAR	0.00% ± 0.00%	0.000 ± 0.000	—	—
	RSGG-CE	94.57% ± 22.66%	0.824 ± 0.471	<b>0.312</b> ± 0.079	4.040 ± 0.848
	D4Explainer	0.00% ± 0.00%	0.000 ± 0.000	—	—
	CF-GNNExpl	52.07% ± 49.96%	0.398 ± 0.568	0.598 ± 0.046	2.000 ± 0.007
XPlore		<b>100.00%</b> ± 0.00%	<b>0.871</b> ± 0.453	0.536 ± 0.057	12.204 ± 0.155
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
TG	iRand	36.16% ± 48.05%	0.356 ± 0.485	<b>0.175</b> ± 0.086	69.177 ± 0.0542
	CF <sup>2</sup>	49.86% ± 50.00%	0.499 ± 0.500	0.496 ± 0.000	<b>0.000</b> ± 0.001
	CLEAR	58.40% ± 49.29%	0.584 ± 0.493	6.353 ± 0.400	<b>0.000</b> ± 0.003
	RSGG-CE	89.28% ± 30.94%	0.888 ± 0.324	0.316 ± 0.091	19.152 ± 0.911
	D4Explainer	49.86% ± 50.00%	0.499 ± 0.500	0.500 ± 0.001	<b>0.000</b> ± 0.013
	CF-GNNExpl	49.86% ± 50.00%	0.499 ± 0.500	0.784 ± 0.000	46.000 ± 0.414
XPlore		<b>100.00%</b> ± 0.00%	<b>0.994</b> ± 0.106	<b>0.248</b> ± 0.004	14.692 ± 0.236
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
MUTAG	iRand	2.66% ± 16.09	0.005 ± 0.163	<b>0.035</b> ± 0.011	4.200 ± 0.004
	CF <sup>2</sup>	0.00% ± 0.00%	0.005 ± 0.318	—	—
	CLEAR	35.11% ± 47.73%	0.309 ± 0.506	2.438 ± 0.375	<b>0.000</b> ± 0.001
	RSGG-CE	56.91% ± 49.52%	<b>0.516</b> ± 0.550	0.264 ± 0.004	2.000 ± 0.002
	D4Explainer	9.57% ± 29.42%	0.021 ± 0.309	0.526 ± 0.016	<b>0.000</b> ± 0.007
	CF-GNNExpl	10.11% ± 30.14%	0.005 ± 0.318	0.758 ± 0.005	46.000 ± 0.147
XPlore		<b>67.55%</b> ± 46.82%	<b>0.548</b> ± 0.613	0.459 ± 0.036	2.866 ± 0.089
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
COX2	iRand	69.81% ± 45.91%	<b>0.677</b> ± 0.499	<b>0.087</b> ± 0.049	22.482 ± 0.192
	CF <sup>2</sup>	24.20% ± 42.83%	0.229 ± 0.435	4.354 ± 0.235	<b>0.000</b> ± 0.005
	CLEAR	22.06% ± 41.46%	0.208 ± 0.421	<b>0.512</b> ± 0.001	<b>0.000</b> ± 0.002
	RSGG-CE	<b>99.36%</b> ± 7.99%	<b>0.968</b> ± 0.231	0.761 ± 0.366	92.914 ± 1.102
	D4Explainer	22.06% ± 41.46%	0.208 ± 0.421	0.518 ± 0.003	<b>0.000</b> ± 0.025
	CF-GNNExpl	22.06% ± 41.46%	0.208 ± 0.421	0.756 ± 0.001	46.000 ± 0.198
XPlore		<b>99.36%</b> ± 7.99%	<b>0.968</b> ± 0.238	7.682 ± 3.704	<b>11.388</b> ± 0.568
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
BBBP	iRand	19.76% ± 39.82%	0.275 ± 0.460	<b>0.059</b> ± 0.037	12.928 ± 0.190
	CF <sup>2</sup>	25.26% ± 43.45%	0.253 ± 0.434	0.510 ± 0.024	<b>0.000</b> ± 0.002
	CLEAR	22.90% ± 42.02%	0.229 ± 0.420	51.889 ± 32.182	<b>0.000</b> ± 0.022
	RSGG-CE	22.90% ± 42.02%	0.229 ± 0.420	0.258 ± 0.006	2.000 ± 0.131
	D4Explainer	21.24% ± 40.90%	0.212 ± 0.409	0.524 ± 0.010	<b>0.000</b> ± 0.038
	CF-GNNExpl	22.31% ± 41.64%	0.223 ± 0.416	0.758 ± 0.005	46.000 ± 0.185
XPlore		<b>81.51%</b> ± 38.82%	<b>0.803</b> ± 0.412	2.217 ± 1.206	4.055 ± 0.030
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
PROTEINS	iRand	18.87% ± 39.13%	0.183 ± 0.394	<b>0.107</b> ± 0.078	75.524 ± 4.036
	CF <sup>2</sup>	16.35% ± 36.98%	0.151 ± 0.375	0.651 ± 0.025	<b>0.000</b> ± 0.001
	CLEAR	22.90% ± 42.02%	0.229 ± 0.420	51.889 ± 32.182	<b>0.000</b> ± 0.022
	RSGG-CE	<b>58.67%</b> ± 49.24%	<b>0.556</b> ± 0.527	0.664 ± 0.477	78.757 ± 3.667
	D4Explainer	0.00% ± 0.00%	0.000 ± 0.000	—	—
	CF-GNNExpl	16.35% ± 36.98%	0.151 ± 0.375	0.651 ± 0.025	2.000 ± 0.002
XPlore		<b>65.41%</b> ± 47.57%	<b>0.627</b> ± 0.511	<b>0.569</b> ± 0.039	2.798 ± 0.006
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
COLLAB	iRand	4.60% ± 20.95%	0.006 ± 0.213	<b>0.018</b> ± 0.021	58.696 ± 2.505
	CF <sup>2</sup>	52.66% ± 49.93%	0.262 ± 0.634	0.886 ± 0.061	<b>0.000</b> ± 0.003
	CLEAR	0.00% ± 0.00%	0.000 ± 0.000	—	—
	RSGG-CE	0.00% ± 0.00%	0.000 ± 0.000	—	—
	D4Explainer	0.00% ± 0.00%	0.000 ± 0.000	—	—
	CF-GNNExpl	<b>52.66%</b> ± 49.93%	<b>0.262</b> ± 0.624	0.886 ± 0.061	<b>2.000</b> ± 0.004
XPlore		<b>100.00%</b> ± 0.00%	<b>0.570</b> ± 0.709	0.810 ± 0.068	3.604 ± 0.021
Dataset	Method	Validity↑	Fidelity↑	Sparsity↓	Oracle Calls↓
TRIANGLES	iRand	6.38% ± 24.44%	0.053 ± 0.247	<b>0.123</b> ± 0.080	71.421 ± 3.346
	CF <sup>2</sup>	37.13% ± 48.31%	0.364 ± 0.488	0.621 ± 0.032	<b>0.000</b> ± 0.001
	CLEAR	89.99% ± 30.01%	0.892 ± 0.311	53.517 ± 36.786	<b>0.000</b> ± 0.019
	RSGG-CE	<b>99.84%</b> ± 4.00%	0.987 ± 0.141	0.325 ± 0.080	5.161 ± 0.378
	D4Explainer	31.11% ± 46.29%	0.302 ± 0.468	0.619 ± 0.038	<b>0.000</b> ± 76.890
	CF-GNNExpl	37.13% ± 48.31%	0.364 ± 0.488	0.621 ± 0.032	46.000 ± 0.056
XPlore		<b>100.00%</b> ± 0.00%	<b>0.988</b> ± 0.138	0.532 ± 0.053	3.366 ± 0.026