Explainable GNNs in Biomedicine

Nicolas Arturo Perez Zambrano

Delft University of Technology

TU Delft

Explainable GNNs in Biomedicine

by

Nicolas Arturo Perez Zambrano

Nicolas Arturo	Student Number
Perez Zambrano	5081270

Thesis advisor:	Elvin Isufi
Daily supervisor:	Megha Khosla
Project Duration:	September, 2024 - May, 2025
Research group:	Multimedia Computing
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

Cover:	Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA under CC BY-NC 2.0 (Modified)
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Graph Neural Networks have become ubiquitous in machine learning research, and their use has also given rise to expectations of what a model can do and how we can understand it. Explainability has become one of the key tools for solving these problems, but explainability often needs to consider domain-specific requirements to be useful. To the best of our knowledge, no domain-specific requirements have been set for the biomedicine domain when working with biomedicine. In this thesis, we seek to understand what standards are needed by the domain, set automatic metrics to meet them, and evaluate these metrics in problems common to biomedicine. Thanks to working on the gene disease association and the proteins classification task, we are able to provide some insights into what constraints require what explainers. We find that no single explainer is able to outperform all the other explainers in all metrics. This work offers practical recommendations for selecting appropriate explainers for specific biomedical applications. It identifies key directions for developing domain-specific explainability approaches that address the unique needs of biomedical research.

Contents

1 Introduction

1.1. The Need for Explainable AI in Biomedicine

In recent years, Artificial Intelligence (AI) has made remarkable strides in various fields, including biomedicine. AI has already achieved human-level performance in multiple tasks such as skin cancer classification [19]. As AI models become increasingly complex and powerful in the analysis of biological data, the need for transparency and interpretability becomes paramount. The need for transparency has led to the emergence of Explainable AI (XAI), a field aimed at developing AI systems whose decisions can be understood and interpreted by humans [79]. XAI encompasses a range of techniques, all aimed at providing human-interpretable insights into the decision-making processes of complex AI models.

The importance of explainability in biomedicine cannot be overstated. A recent review found that approximately 30% of existing articles in the field of explainability focus on the medical domain [53]. This emphasis reflects the critical nature of transparency in the biomedical domain, where decisions can have life-changing consequences. More complicated models are needed in biomedicine, so XAI helps bridge the gap between more complex models and makes them understandable.

The frequent use of XAI methods in biomedicine is in contrast to the little effort put into adapting XAI approaches to the biomedicine field. It has been argued that most explainers should be tuned to each domain in which they are used [36], but most methods are rarely adapted to bioinformatics [81] [1]. This vacuum is the primary focus of this thesis.

For the remainder of this thesis, we will use the following definitions;

- Model: Machine learning system that outputs a value for an input.
- Explainer: System that gives information that explains why a model gives the output it gave for the given input.
- Explanation: The resulting output of the explainer for a given model.

1.2. Graph Neural Networks in Biomedicine

Graph Neural Networks (GNNs) have emerged as powerful tools in biomedical research, offering a natural way to model and analyze complex biological systems. GNNs are specialized artificial neural networks that are adapted to work with graphs as input, which is naturally extended to biomedicine due to structures such as molecules that can be represented as graphs [58]. The inherent graph structure of many biological phenomena makes GNNs particularly well-suited for a wide range of applications in this field.

Key areas where GNNs have made significant contributions include:

- Molecular Structure and Drug Discovery: GNNs are used to predict drug target affinities and molecular properties, accelerating the drug discovery process [73].

- Genomics: GNNs facilitate the prediction of gene-disease associations and gene expression analysis, improving our understanding of genetic factors in diseases [47].
- Cellular and Tissue-level Analysis: GNNs are used in the classification of cell types from single-cell RNA sequencing data and brain connectome analysis [70].

Despite their success, the application of GNNs in biomedicine faces several challenges, particularly in terms of interpretability and biological plausibility. The complexity of these models often makes their decision-making processes opaque, which is problematic in healthcare applications where transparency is crucial [36].

1.3. Explainability in Graph Neural Networks

While XAI is crucial across various AI domains, it faces particular challenges when applied to Graph Neural Networks (GNNs).

- Structural Complexity: The interplay between nodes and edges in graphs makes it difficult to isolate the impact of individual elements on the model's decision. [57]
- Non-linearity: The multiple transformations in GNNs make it difficult to find the relation between input features and output predictions.
- Consistency: Most explainers use randomness in explanations, meaning the highlighted features can change on the same input per run [53]

Benchmark methods have come up as an approach to mitigate these issues when applying XAI methods to GNNs. The techniques vary depending on the method, but the idea behind them is to provide a standard template for comparing XAI approaches, to understand how well they are honestly explaining. The need for benchmarking methods arises because XAI methods are not always human-understandable [44] [34], and theoretical results do not always match practical results in XAI [3].

Some of these methods include the BAGEL benchmark proposed by Rathee et al. [57], and GraphFramEx proposed by Amara et al [2].

1.4. Addressing the Needs of XAI in Biomedical GNNs

Through our work, we have found that authors would often highlight the following needs, and addressing these needs forms the core of this work. The needs are as follows:

- Domain Specificity: Many existing explainers are domain-agnostic, failing to account for the unique requirements of biomedical applications [36] [81].
- Interpretability for non-technical users: As highlighted by Lotsch et al. [45], explanations must cater to both statistical experts and domain practitioners.
- Biological Plausibility: Explanations must respect the constraints and realities of biomedical systems, avoiding biologically impossible interpretations [80] [78] [81].
- Fairness and Bias: There is a need to identify and address potential biases in AI healthcare applications [32].
- Consistency: Most explainers use some randomization, which means each output might be different [3], and having inconsistent results is not considered acceptable by humans [28].

1.5. Research questions

How can the previously explained needs of biomedicine be addressed when deciding what explainer to use?

As Karim et al. [36], each domain has different needs when adapting explainers towards it. We want to understand how different explainability metrics align with the needs of biomedicine that we outlined in the previous section. We establish various metrics that are relevant to the domain, and show how they perform when used in domain-specific datasets.

How do dataset properties affect the explanations in biomedicine?

Class imbalance is most common in biological problems [24]. Biological systems also have their structure that might not be understood by models, leading to biologically impossible outcomes [80]. We seek to understand the effect that the data itself has on the output of the explainers, and if we can control for it.

How differently do gradient-based, perturbation-based, and surrogate explainers perform between node and graph classification problems in biomedical applications?

Rathee et al. [57] found that there was a difference in explainer performance between graph and node classification tasks. We seek to understand what effect is kept in biomedical applications, using a consistent set of metrics in both problems. We additionally investigate the impact that different GNN models have on explainer performance.

What strategies should be used when applying general-use GNN explainers in the biomedicine domain?

Based on the previous research questions, we have found some steps for setting up explainers in biomedicine. This thesis should be the first step in adapting explainers to meet the various needs of biomedicine and setting up future steps.

1.6. Structure of thesis

The thesis is organized into the following chapters:

- Chapter 1 introduces the problem and explains the needs of biomedicine when using XAI, highlighting why we need to adapt general explainer use when using the tools in biomedicine.
- Chapter 2 introduces the explainers used, as well as giving information on why we picked the datasets that we chose.
- Chapter 3 introduces the evaluation metrics that will be used, as well as why we chose these metrics. This section addresses what is needed to use AI in biomedicine and outlines what is required to meet trustworthiness.
- Chapter 4 gives detailed information on the experiments taken and how they differ from the original papers used as a starting point.
- Chapter 5 gives results of the experiments, as well as providing an in-depth analysis of them. This chapter also answers the research questions about the performance of the different explainer types, as well as how the dataset influences the explanation.
- Chapter 6 provides concluding words and summarizes many of the findings. It includes guidelines on how to apply explainers to biomedicine based on current findings.

The code used for this thesis can be found here: `https://github.com/PriXAI/XAI-GNN-Biomed-Thesis`

2

Explainers and Datasets

We will now focus on explaining the explainers commonly used in the literature, as well as introducing the types of problems we will be working with. You can find more detailed information on the datasets used for each problem type in Chapter 4.

Kakkad et al. [35] divide GNN explainability into various categories. The two main branches are counterfactual methods, which aim to identify the slightest change in the input that alters the output, and factual methods, which focus on determining the input features that have the most significant influence on the prediction.

We can further divide Factual methods into self-interpretable and post-hoc methods. Self-interpretable methods have explainability built directly into the model.

We focus on post-hoc methods, which work by using the model's input and its output. We took this decision to make our analysis applicable in more cases and integrate it with existing solutions.

We can further divide post-hoc explainers into three groups, based on the categorization of Kakkad et al. [35] and Rathee et al. [57].

- Gradient-based (G): Gradients measure the rate of change, meaning we can understand the relative importance of each feature by understanding the gradient of the prediction compared to the input.
- Perturbation-based (P): Perturbation methods perturb the input, using the difference between the new prediction and the original to score the features that were modified.
- Surrogate-based (S): Surrogate methods fit a simple interpretable model around the prediction, then the output of the interpretable model is used as an explanation.

2.1. Explainers

Explainers focus on the three elements that make up a graph: nodes, features, and edges. Some explainers combine these elements, and when used to explain nodes and features, we will refer to them as graph explanations for brevity.

We present the list of explainers used during this thesis, as well as highlighting some of their use in bioinformatics. In contrast to the work done by GNNX-Bench [40], we give a bigger focus to gradient methods.

2.1.1. GNNExplainer (P)

GNNExplainer [74] is a method that works by learning a soft mask for the edges and node features of the input graph. It is a perturbation-based approach. It optimizes this mask to maximize the mutual information between the prediction of the complete graph and the prediction of the subgraph induced by the mask, while keeping the subgraph small. This approach effectively identifies the most critical

substructures and features for a given prediction. This method is one of the first explainer methods and is frequently used in the literature [46] [57] [20]. It was applied in drug repurposing to highlight key molecular structures [69].

The way GNNExplainer works is by optimizing the following:

$$\max_{G_s \subseteq G_c} MI(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S) \tag{2.1}$$

Where G_c represents the original graph, $G_S \subseteq G_c$, $v_j \in G_s$ are the nodes chosen to belong in G_s, $X_S = \{x_j | v_j \in G_S\}$ are the features of G_S for each corresponding node, Y are the original labels, v is the node to explain, and MI is mutual information between two random variables. This optimization aims to find the smallest subgraph that maximizes the mutual information with the model's prediction.

The original paper later simplified the equation to something more tractable. We can remove $H(Y)$ since it is fixed for a trained model. GNNExplainer also treats the subgraph as a random graph variable $G_S \sim G$, focusing on optimizing this. Lastly, GNNExplainer assumes that the equation is convex and applies Jensen's inequality, leading to the following equation:

$$\min_{G_s \subseteq G_c} H(Y|G = \mathbb{E}_G[G_S], X = X_S). \tag{2.2}$$

2.1.2. Integrated Gradients (G)

Integrated Gradients, commonly shortened to IG [66], calculate the importance of each input feature by integrating the gradients along a straight path from a baseline input (typically a zero input) to the actual input. It is a gradient-based method. It accumulates these gradients to attribute the prediction difference to each input feature. The authors demonstrated its application on a molecular graph convolution architecture [37], showing its ability to identify essential atoms and bonds in molecular property prediction tasks.

The IG attribution for an input x with baseline x' is defined as:

$$\text{IG}(x) = (x - x') \odot \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x} d\alpha \tag{2.3}$$

In practice, we approximate this integral through a Riemann sum [66]:

$$\text{IG}(x) \approx (x - x') \odot \frac{1}{N} \sum_{k=1}^{N} \frac{\partial f(x' + \frac{k}{N}(x - x'))}{\partial x} \tag{2.4}$$

Where N is the number of steps in the Riemann approximation, and $\odot$ represents element-wise multiplication,

2.1.3. Saliency (G)

Saliency introduced by Simonyan et al. [61] is a simple, gradient-based attribution method that computes the gradient of the output with respect to the input features. In the context of GNNs, we can apply Saliency to both node features and graph structure. Saliency maps were used as an auditing tool for a model to detect COVID-19 in chest radiographs [15].

The saliency attribution for an input x is defined as:

$$\text{Attribution}(x) = \left| \frac{\partial f(x)}{\partial x} \right| \odot x \tag{2.5}$$

Where x is the input, $f(x)$ is the model output, $\odot$ represents element-wise multiplication, $|\cdot|$ is the absolute value operation, and $\frac{\partial f(x)}{\partial x}$ is the gradient of the output with respect to the input.

In the context of GNNs, Saliency can be applied to both node features and graph structure. For node classification tasks, we compute:

$$S(v) = \left| \frac{\partial f(G, v)}{\partial h_v} \right| \tag{2.6}$$

where G is the graph, v is the node of interest, and h_v is the feature vector of node v. For graph-level tasks, we aggregate node-level saliencies:

$$S(G) = \sum_v \left| \frac{\partial f(G)}{\partial h_v} \right| \tag{2.7}$$

2.1.4. InputXGradient (G)

This explainer multiplies the input features with the output gradients for that input value. It is a gradient-based method. InputXGradient is a very simple explainer, since it relies only on the input and the default gradients, but it can tell the direction and magnitude of an explanation [59]. Li et al. used the method to better understand molecular properties [43].

InputXGradient is defined as follows:

$$\text{Attribution}(x) = x \odot \frac{\partial f(x)}{\partial x} \tag{2.8}$$

Where x is the input, $f(x)$ is the model output, $\odot$ represents element-wise multiplication, and $\frac{\partial f(x)}{\partial x}$ is the gradient of the output with respect to the input.

2.1.5. Deconvolution (G)

Deconvolution [77] computes the gradient of the target output with respect to the input, but during backpropagation, it only propagates nonnegative gradients. It is a gradient-based method. This approach aims to highlight features that positively contribute to the target class. It was used in a comparative study of different explainers for a gene expression problem by Budhkar et al. [7].

The Deconvolution method works by backpropagating the activation of a target output neuron through the network, but with modified gradient computation rules. For a given layer l, the Deconvolution attribution is computed as:

$$\text{Attribution}_l = \text{ReLU}(\text{Deconv}_{l+1} \cdot W_l^T) \tag{2.9}$$

Where Deconv_{l+1} is the Deconvolution of the next layer, W_l is the weight matrix of the current layer, and ReLU is the rectified linear unit activation function.

The key difference between Deconvolution and standard backpropagation is in how it handles the ReLU activation function:

$$\text{Deconv}(\text{ReLU}(x)) = \text{ReLU}(\text{Deconv}(x)) \tag{2.10}$$

During the backward pass, Deconvolution keeps only the positive gradients, effectively ignoring the paths where ReLU would have set the activations to zero in the forward pass.

2.1.6. Guided Backpropagation (G)

Guided Backpropagation [62] is similar to Deconvolution; this method also focuses on backpropagation but combines it with the ReLU activation function's gradient. It is a gradient-based method. It only propagates positive gradients for positive activations, aiming to visualize what the network is looking for, rather than what it has found. This method was also used in the comparative study by Budhkar et al. [7].

2.1.7. Parameterized Explainer for Graph Neural Network (PGExplainer) (P)

Parameterized Explainer for Graph Neural Network (PGExplainer) [46] focuses on explaining the general model instead of being done per instance. It is a perturbation-based explainer. PGExplainer assumes that all graphs have some underlying structure, meaning each graph can be described as $G_o = G_s + \Delta G$, where G_o is the original graph, G_s is the "important" subgraph, and ΔG is information relevant to the task. That is why PGExplainer uses the following expression to find G_s:

$$\max_{G_s \subseteq G_c} \text{MI}(Y_o, G_s) = H(Y_o) - H(Y_o|G = G_s) = \min_{G_s \subseteq G_c} H(Y_o|G = G_s), \tag{2.11}$$

Here, Y_o is the prediction of the GNN model. $H(Y_o)$ remains fixed with a trained model. As such, PGExplainer only has to optimize $\min_{G_s \subseteq G_c} H(Y_o|G = G_s)$. PGExplainer also focuses on predicting the relevant edges that make up G_s.

Due to difficulties with the objective function, PGExplainer takes multiple steps to make it more tractable. It assumes that the edges in the explanatory graph are conditionally independent and also relaxes the optimization from binary to continuous variables on the edges.

Furthermore, PGExplainer takes several steps to differentiate itself from GNNExplainer, which shares a similar objective function. Most importantly, GNNExplainer needs to be trained per input graph. In contrast, PGExplainer is first trained and then used per instance, meaning a trained model is considerably more efficient for each explanation needed. Additionally, it includes multiple regularization terms. It has constraints for the explanation size and to guarantee connectedness.

The original paper already works on a biological dataset, using the MUTAG dataset for parts of their experiments [46].

2.1.8. GraphLIME (S)

GraphLIME (Graph Local Interpretable Model-agnostic Explanations) [31] is an adaptation of the LIME algorithm for graph-structured data. The objective of this method is to explain individual predictions of any graph neural network model by learning an interpretable model locally around the prediction. Costi et al. [13] used the method to help predict diabetes.

Given a node v in graph G and a GNN model f, GraphLIME generates an explanation by solving the following optimization problem:

$$\xi(v) = \arg\min_{o \in O} o(f, X_n) \tag{2.12}$$

Where:

- O is a class of interpretable models, specifically the Hilbert-Schmidt Independence Criterion (HSIC) Lasso
- f is the GNN model
- X_n is the sampling local information of the node v

GraphLIME works by perturbing the subgraph around the node of interest and observing how the model's predictions change. Then, a linear model is fit to this local behavior. The coefficients of this linear model serve as an explanation, indicating the importance of each feature.

2.1.9. Zorro (P)

Zorro [22] is a powerful explanation method specifically designed for Graph Neural Networks (GNNs). It aims to produce sparse, stable, and faithful explanations by optimizing a novel objective based on Rate-Distortion Theory. The Zorro method optimizes the following objective:

$$\arg\max_{x \in X_n} \mathcal{F}(V_n, \{f\}) \text{ or } \arg\max_{v \in V_n} \mathcal{F}(\{v\}, F_n) \tag{2.13}$$

Where:

- $\mathcal{F}$ is the RDT Fidelity
- V_n is the set of all nodes not included in the current subgraph. It starts with all of the features.
- F_n is the set of all the features not included in the current subgraph. Start with all of the features.

At each step, the algorithm selects the subset that yields the most significant improvement in RDT Fidelity, whether it is a node or a candidate feature. You can find the definition of RDT Fidelity in Chap. 3.

Zorro generates explanations by learning a mask over the input graph's edges and node features. This mask is optimized to preserve the model's prediction while being as sparse as possible.

2.1.10. DummyExplainer (D)

DummyExplainer is not an actual explanation method but serves as a benchmark for the quality of the explanation. Since it does not fit into the three explainer categories, we define it in its own category, DummyExplainer (D). It randomly assigns importance values to each potential mask value. By comparing other explainers with this random baseline, you can ensure that the explanations provided by different methods are helpful [20].

2.2. Explainer Intuition

To make it easier to understand each explainer, we will briefly go over the methods and provide concise explanations for each of them. The idea of this section is to get an intuition of why each of the methods works.

- **GNNExplainer (P)**: Find the minimal subgraph and feature set that has the same information as the original prediction.
- **Integrated Gradients (G)**: Given a baseline input (typically zero), what are the features that give the most significant difference in the prediction?
- **Saliency (G)**: Calculate the gradient of the output using the input features to find the elements that have the most significant influence.
- **InputXGradient (G)**: Same as Saliency, but now captures direction instead of only magnitude.
- **Deconvolution (G)**: Backwards propagation using only positive output gradients to find input feature importance.
- **Guided Backpropagation (G)**: Backwards propagation using only positive input gradients to find feature importance.
- **PGExplainer (P)**: Find minimal subgraph and feature set with same information as original prediction, but focused on edges and learning over multiple inputs.
- **GraphLIME (S)**: Creates a locally interpretable model that approximates the complex model's behavior around a specific prediction.
- **Zorro (P)**: Do greedy selection on feature that improves RDT fidelity.
- **DummyExplainer (D)**: Assigns random importance values to each feature.

2.3. Datasets

Graph neural networks work with problems of classification and regression, just like standard machine learning. Each of these problems can also be divided into node, edge, or graph prediction levels. We focus on node and graph classification, as we found these types to be more common in biomedicine.

2.3.1. Node Classification

For node classification, we focus on the work done by eXplainable Gene–Disease Associations via Graph Neural Networks (XGDAG) [47]. XGDAG is a paper that works in the task of gene-disease association (GDA) discovery. This task focuses on identifying genes likely associated with a given disease, using multiple types of network data [47].

XGDAG is relevant for explainability because the authors use explainable methods as a step for gene prediction. After using a GNN model, XGDAG uses various explanations in the given output. These explanations build a ranked order of the genes. Through this approach, they were able to beat other state-of-the-art papers in GDA. You can find more detailed information about XGDAG in Chapter 4.

2.3.2. Graph Classification

For the graph classification problem, we choose to focus on the Proteins dataset as defined by TU-Dataset [50]. This dataset is used to predict whether a protein is an enzyme or not.

This dataset was chosen in particular because, as of the time of writing, it is the most commonly used dataset in biology, according to Papers with Code [11].

We used the work done by GNNX-Bench [40] as a starting point. GnnX-Bench is a benchmarking study of GNN explainers, focusing on perturbation-based methods. Their work provides a starting point for model architectures and offers insights for comparison. A more detailed explanation of what is being used from the GNNX-bench can be found in Chap. 4.

3

Evaluation Metrics for XAI in Biomedicine

With the growing power of AI, requirements are also getting stricter for its use [12]. Despite this, all of the needs for AI use can be summarized in one simple word: Trustworthiness [36] [12] [17].

Sadly, trustworthiness is a word that has multiple definitions [36] [12] [17] [53] [35] [14]. This issue is expounded when you take into account that these definitions can be mutually exclusive to each other. For example, the definition of trustworthy AI given by [42] includes fairness, but as the authors [39] find, some of the definitions of fairness are mutually incompatible in most scenarios.

As such, we need to work on narrowing down the scope to something that we can work on.

Based on the research, we have grouped the needs of XAI in biomedicine into three distinct categories that we will focus on addressing:

- Faithful model explanations
- Human understandable
- Consistency
- Time

Additionally, we will explain the out-of-scope needs and provide a summary of the previously included points.

3.1. Faithful model explanations

If an explanation says that a model uses certain features, those features should be significant to the model. Guaranteeing that explanations are faithful to the given model can help us understand how the explainers change depending on the problem and the datatype [81]. Additionally, we can understand whether the explanations are consistent with the underlying data and are biologically plausible [80] [78] [81]. However, there are multiple ways of measuring faithfulness, which we will discuss in the following sections.

3.1.1. RDT-Fidelity

To measure the faithfulness of explainers that provide feature or node attributions, we will be using rate distortion-based fidelity (RDT-Fidelity) as proposed in Zorro [22]. RDT-Fidelity works by randomizing the features that were not considered relevant by the explainer and comparing the prediction with the randomized and the original features. So, if the explainer says something is important, changing the nonimportant features should not affect the prediction.

RDT-Fidelity is defined as follows:

$$\mathcal{F}(\mathcal{S}) = \mathbb{E}_{Y_s|Z\sim\mathcal{N}}\left[\mathbb{I}_{f(X)=f(Y_s)}\right] \tag{3.1}$$

$$Y_S = X \odot M(S) + Z \odot (\mathbb{1} - M(S)), Z \sim \mathcal{N} \tag{3.2}$$

Where S is the explanation, $M(S)$ is the explanation mask, f is the GNN, X is the input, $\mathcal{N}$ is the noise distribution, and Y_s is the perturbed input.

For our use case, we are slightly modifying the method. The first modification is to allow for soft masks; this allows the metric to be used with more explainers. The second is to allow as direct input a node and feature mask set, as many methods output both directly. Lastly, for the case of regression problems, we will measure the difference between the scores in order to make the value more understandable when analyzing.

3.1.2. Alternative Fidelity

An alternative way of measuring Fidelity has been proposed, one that works without random perturbation. This metric is actually two elements, Fidelity+ and Fidelity-, as described by Yuan et al. [75]. The idea of this approach is that, for Fidelity+, removing important features should change the prediction, and for Fidelity-, it is that keeping only the essential features should not change the prediction. These fidelities are measured as follows:

$$\text{Fidelity}_+ = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\hat{y}_i^{G_i\setminus S} = \hat{y}_i) \tag{3.3}$$

$$\text{Fidelity}_- = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\hat{y}_i^{G_i^S} = \hat{y}_i) \tag{3.4}$$

Where

- N is the total number of graphs in the dataset
- $\hat{y}_i$ is the original prediction of the model for graph i
- G_i represents the original complete graph
- S denotes the subgraph identified as important by the explainer
- $G_i \setminus S$ represents the graph with the important subgraph removed
- G_i^S represents only the important subgraph
- $\hat{y}_i^{G_i\setminus S}$ is the model's prediction when the important subgraph is removed
- $\hat{y}_i^{G_i^S}$ is the model's prediction when only the important subgraph is used

These also fit some of the requirements given by causality, where Fidelity- describes a sufficient explanation, and Fidelity+ describes a necessary explanation [2]. The values are bound within the range of 0 to 1; values with higher Fidelity+ are preferred, and lower Fidelity- values are preferred.

In the literature, we find an alternative name for this approach to Fidelity. Fidelity- is also known as being a Sufficient explanation. Fidelity+ is also known as being a Necessary explanation [2] [57].

Although Fidelity is the most intuitive explanation, it has a couple of problems related to it. Due to the removal and modification of information, the data the model is predicting on might come from a different distribution than the original data. Hooket et al. found that models are robust to information removal, in an image classification problem, removing 90% of the input still gave accuracy that was comparable to clean data, and above chance [30].

This robustness leads to an issue where the training data and the masked data might come from different distributions, and where the model remains accurate due to its innate properties. So, how

do we measure the distance between an explanation and the true dataset? Graphs expound these problems, where there are variations in input size and aggregation that make it more difficult to compare the distance between two graphs.

3.1.3. Kernel Distance

Graph kernels have emerged as a natural solution to the challenge of comparing graphs, offering a method to measure similarity between objects of varying structures. These techniques have found applications in diverse fields such as bio-informatics, chem-informatics, and neuroscience [41]. This allows us to understand how close the distributions of two graphs are, and can be used as a way of understanding how biologically plausible the explanations are.

Formally, a graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ is a function that maps a pair of graphs to a real number, satisfying the following properties for all graphs $G, G', G'' \in \mathcal{G}$:

1. Symmetry: $k(G, G') = k(G', G)$
2. Positive semi-definiteness: $\sum_{i,j} c_i c_j k(G_i, G_j) \geq 0$ for any $n \in \mathbb{N}$, graphs $G_1, \ldots, G_n \in \mathcal{G}$, and real numbers $c_1, \ldots, c_n \in \mathbb{R}$

After evaluating various graph kernel methods based on their expressivity, precision, and computational efficiency [41, 60], we selected the propagation kernel method [51] for our analysis. This choice was motivated by its favorable trade-off between accuracy [41] and speed [60].

The propagation kernel measures the similarity between graphs based on the diffusion of node labels. For a graph $G = (V, E)$ with nodes V and edges E, the propagation kernel $k(G, G')$ between two graphs G and G' is defined as:

$$K(G, G') = \sum_{u \in G} \sum_{v \in G'} k(u, v) \tag{3.5}$$

Here, k is a node kernel that is resolved by binning. For this particular implementation, locally sensitive hashing was used [60]. An LSH is a function h where given $d(x, x') < \theta$ then $\Pr(h(x) = h(x')) > p_1$. That is, similar elements are stored in the same bin. Neumann et al. provide a more detailed explanation [51].

The particular LSH used in this case is detailed in the following section. Here, the matrix X represents the nodes and their features, where N is the number of nodes, and D is the number of attributes.

Algorithm 1 CALCULATE-LSH

Require: matrix $X \in \mathbb{R}^{N \times D}$, bin width w, metric M
  if M = H **then**
    $X \leftarrow \sqrt{X}$ {square root transformation}
  end if
  if M = H or M = L2 **then**
    $v \leftarrow$ RAND-NORM(D) {generate random projection vector} {sample from $\mathcal{N}(0, 1)$}
  else if M = TV or M = L1 **then**
    $v \leftarrow$ RAND-NORM(D)/RAND-NORM(D) {sample from Cauchy$(0, 1)$}
  end if
  $b = w *$ RAND-UNIF$()$ {random offset $b \sim \mathcal{U}[0, w]$}
  $h(X) = \lfloor (X * v + b)/w \rfloor$ {compute hashes}

Here, the metrics are as follows:

- H: Hellinger distance
- L2: Euclidean (L2) distance
- L1: Manhattan (L1) distance
- TV: Total variation distance

For our experiments, we used the Manhattan distance.

To apply this idea with masks, we will work on sampling from the explanation, where we will only use the top K features of a given explanation when generating the explanation subgraph, where K is a percent cut-off based on sorting the values. For our use case, we will assume that the top K features are those with the highest value, not the highest absolute value. With this, an explanation subgraph is generated that can then be compared with the propagation kernel method [51]. For our tests, we fix $K = 10\%$, based on the findings of Zheng et al. [80]. We train the embeddings on the original dataset.

To apply this method to explanation masks, we sample from the explanation by selecting the top K features based on their importance values. Specifically, we generate an explanation subgraph using only the features with the highest positive values, where K is a percentage cut-off (set to 10% in our experiments, following Zheng et al. [80]). This approach allows us to compare the original graph with its masked version using the propagation kernel method.

The process can be summarized as follows.

1. Generate explanation masks for a given graph.
2. Create subgraphs by retaining only the top $K\%$ of nodes/edges based on the mask values.
3. Convert the original graph and the subgraphs to a format compatible with the graph kernel library (e.g., GraKeL [60]).
4. Compute the propagation kernel between the original graph and each subgraph.
5. Analyze the resulting kernel values to quantify the difference between the original and masked graphs.

We chose this approach because of its applicability to graph problems. Other approaches, such as the one used by Qiu et al. [55], were not deemed valid as they required the input shape to remain constant. Constant input shape is something that rarely happens with graphs. The approach used by Zheng et al. [80], where they trained an Autoencoder model and compared the resulting vector representation of graphs, was considered. However, we decided not to use it as it required a training step in the middle and could be affected by the randomness of training. Another approach that could be used is ROAR [30], where a fraction of the input is removed based on the importance of the feature and a model is retrained on this new set of features; however, this approach was deemed too time-consuming to be viable.

3.2. Human understandable

One of the main priorities for biomedical use is to ensure that the human who uses the tool actually understands the result received, something often ignored in explainability [44] [34]. We have already outlined that human tests are out of scope, but we can still find alternatives that help measure usefulness for humans.

3.2.1. Entropy

An explanation that says everything has the same importance would return a perfect fidelity score. This explanation is not good, as it means that the explanation is not really explaining. So we want a way of measuring whether our explanations are small and focused on the most important features. This idea also lines up with human intuition; if you have to remember hundreds of numbers, you will not memorize them, but if you only need to remember 10 numbers, it is much easier to memorize them.

We follow the Entropy definition described in Zorro[22], where we calculate the Shannon entropy over the normalized distribution. Entropy defines the average level of information of a variable, with a higher value meaning that there is more uncertainty about it. The formal mathematical definition is the following:

$$H(p) = -\sum_{\phi \in M} p(\phi) \log p(\phi). \tag{3.6}$$

In our experiments, we found that certain explainers give negative masks, which makes sense as this means that a particular attribute worked against the prediction. To deal with this, we first find the

absolute value of all the mask values, working under the assumption that the further a value is from 0, the more important that value is.

3.2.2. Counterfactual

Jeyakumar et al. [34] conduct an Amazon Mechanical Turk study comparing which explanation method humans prefer. They find that their model, ExMatchina, is consistently rated as the best model. This model works by providing an example of similar input and contrasting it, making it easier to understand why the system took the decision it did.

Other authors have found that humans prefer contrasting explanations, where humans prefer explanations of why the current explanation happened instead of something else [49]. For XAI, counterfactuals ask the perfect question: What needs to be changed to achieve a different outcome? [10]. As Chou et al.[10] outline, multiple points are needed for a good counterfactual: proximity, plausibility, sparsity, diversity, and feasibility. Other metrics in the thesis have taken all of these points into account, with the exception of proximity.

Based on this, we will take an approach based on the work done by Ramon et al.[56]. The authors used the outputs of LIME and SHAP as a ranked way of deciding which feature to remove or to set to 0, and then measured the effectiveness of the counterfactuals based on the number of features that were changed before there was a change in the prediction. As such, the fewer features modified, the closer the new point is to the original one.

$$Min_i \; C(x \setminus E_i) \neq C(x) \tag{3.7}$$

Where:

- x is the original feature set
- E_i is a ranked ordered list of the most important i features according to the explainer
- $x \setminus E_i$ is the feature set with the i most important features removed, in this case by setting them to 0
- $C(x)$ is the classifier's prediction for instance x

This method forces all metrics to become hard masks due to their ranking. An alternative approach could be used, which uses soft masks as a gradient for exploration, but this method does not work in the case of hard masks. We decided that a strategy that works for all types of masks is more desirable.

3.3. Consistency

Most explainers use some randomization when generating their explanation [3] [53]. This randomness can lead to an issue where the most important feature changes when the same model is explained again. To find out whether this is an issue, we came up with two metrics to measure consistency and understand whether repeated explanations differ too much.

3.3.1. Repeatability

For the cases where we receive hard masks, we use the approach defined in LEAF [3] to compare the Jaccard similarity between multiple different explanations. We chose this approach due to its flexibility for working with various types of masks and due to its existing application for explainability.

The Jaccard similarity is a method of measuring the similarity between different sets by seeing how many overlapping elements are contained versus the total number of elements in both sets.

We extend the metric defined in LEAF [3] to soft masks by taking the top K important features as a set and comparing the generated sets between different soft masks. We name this metric repeatability because we take it to be the number of times it repeats the same important values. We extend the definition given by LEAF [3] of Jaccard similarity for features to work with the use of feature rankings, where for each explanation, we get the most important K features.

$$\text{Repeatability} = \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} J(S_i, S_j) \tag{3.8}$$

Where:

$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{3.9}$$

And:

- n is the number of predictions being compared
- S_i and S_j are sets of important features from different predictions
- For hard masks, S_i and S_j are the sets of features with mask value 1
- For soft masks, S_i and S_j are the sets of top K important features
- $J(S_i, S_j)$ is the Jaccard similarity between sets S_i and S_j

In our experiments, we fix $K = 30$.

3.3.2. Variance over Explanations (VoE)

The most intuitive metric to measure consistency is measuring the variance of the explanations. The issue we find is that each resulting mask is one-dimensional, so finding the variance between different masks leads to a large covariance matrix. For all of the masks, the approach taken is to find the sum of the variances from the covariance matrix, which we call Variance over Explanations (VoE). The VoE can be represented as:

$$\text{VoE} = \sum_{i=1}^{n} \sigma_{ii} \tag{3.10}$$

where σ_{ii} represents the variance of the i-th element (the i-th diagonal element of the covariance matrix) and n is the number of elements in the mask.

Although an in-depth analysis of the covariance matrix would provide more detailed information, this approach is chosen for its simplicity and generalizability across different problem types. It also offers a more straightforward interpretation compared to analyzing the full covariance matrix.

Due to the large size of the datasets involved, we calculate the covariance matrix using an online method. This approach allows for the incremental computation of the covariance matrix, updating it as new data points are processed, rather than requiring all data to be present in memory simultaneously. This method is beneficial for handling large-scale datasets efficiently.

The approach used follows Welford's online algorithm [71]. The algorithm is described as follows:

Let $X = \{x_1, x_2, ..., x_m\}$ be the set of m samples, each sample representing an explanation, where each x_i is a vector of dimensions d.

1. For the mean calculation:
$$\bar{x}_k = \bar{x}_{k-1} + \frac{1}{k}(x_k - \bar{x}_{k-1}) \tag{3.11}$$
where $\bar{x}_k$ is the mean after processing k samples.
2. For the $M2$ statistic (sum of squared differences):
$$M2_k = M2_{k-1} + (x_k - \bar{x}_{k-1})(x_k - \bar{x}_k) \tag{3.12}$$

3. The variance for each feature after processing the m samples is:

$$\sigma_i^2 = \frac{M2_{n,i}}{m-1} \tag{3.13}$$

where $M2_{m,i}$ is the i-th element of the final $M2$ vector.

4. Finally, the total variance is the sum of the variances for all features:

$$\text{Total Variance} = \sum_{i=1}^{d} \sigma_i^2 = \sum_{i=1}^{d} \frac{M2_{n,i}}{n-1} \tag{3.14}$$

3.4. Time

Lastly, we also consider the time to run each model. As highlighted by GraphframEx [2], most papers ignore this when performing a benchmark of metrics. As such, we include this information in our analysis. For explainers that need to be trained, in particular PGExplainer [46], we include the training time in the benchmark. We will measure the time to explain an input.

3.5. Needs that are out of scope

Lastly, we would like to highlight metrics or ideas that were not implemented here. We found multiple metrics that were highlighted in biomedical papers, but were either vague in their description or had requirements that were outside of the scope of this thesis. This list is non-exhaustive, but it contains some of the most essential points that were considered for internal use.

3.5.1. Fairness

The main issue in measuring fairness is that the existing methods work directly on the model or the data [4], not on the explainers. Furthermore, the methods that we could use for fairness would highlight the need to understand why the model took its decisions [48] [4], which we believe is already taken into account by the different faithfulness metrics used.

3.5.2. Causality

Causality has been highlighted as one of the limiting factors in using AI in medicinal fields [6], as well as making the argument that causality is what is needed to meet European standards, not explanations [26].

Causality was not integrated into the system, as there is simply no existing system that provides model-agnostic causal explanations, to the best of our knowledge.

This issue can be partially related to the idea of trying to understand the model versus understanding the data [9]. Most causal systems, when applied to AI problems, focus on trying to understand the true relations in the data [8], while our focus here is on understanding why a model makes certain decisions. We also highlight that this difficulty in finding causality is inherent in the problem. AI systems tend to work with observed data, and finding causality from that is very difficult [10].

Additionally, we would like to highlight that we are not dealing directly with causality but have a method to describe counterfactuals. Counterfactuals have been linked to causality, which is ”the extent to which an explanation of a statement to a human expert achieves a specified level of causal understanding with effectiveness, efficiency, and satisfaction in a specified context of use.” [29]. As such, although we do not use causality fully, we believe that we have incorporated enough of its ideas to satisfy biomedical needs.

3.5.3. Human agreement

Human agreement as a metric would focus on making things understandable for physicians [53] [27] [45]. Doing this would require human testing, which is beyond the scope of this.

Additionally, Counterfactual as a metric has been highlighted for their use with humans [49] [34]. Given this, we believe that we are partially incorporating human agreement into one of the existing metrics.

3.5.4. Privacy related metrics

Privacy and data governance have all been highlighted as important metrics to improve the trust of AI [12]. In this thesis, we are ignoring them because we are working with data and models that are publicly available.

3.6. Summary

This section summarizes the metrics we will use in our studies and serves as a shorthand for expressing intuition about how each approach works.

- Fidelity: Are the explanations accurate to what the model is using?
  - RDT-Fidelity: If we randomize the nonimportant metrics, does the prediction change?
  - Fidelity+: Does removing important features change the prediction?
  - Fidelity-: If we keep only important features, does the prediction change?
- Kernel distance: How similar are the explanations to the original data?
- Entropy: Are the explanations concise?
- Counterfactual: How good is the explanation for changing the prediction?
- Repeatability: Are the same features always highlighted?
- Variance over explanations: What is the variance of the resulting masks?
- Time: How long does the explainer take to run?

In each specific section, you can find information on why these metrics were chosen and how they help address some of the needs of biomedicine.

4

Experiments

This section gives a detailed overview of the experiments carried out. We will provide more information on where we are applying the explainers and how the code has changed compared to the original papers. The original papers are still the focus of this structure.

In most cases, the general structure is: Take the original paper, whenever it uses an explainer, swap the explainer out, and perform tests at that moment. That is, keep the comparison as close as possible to what the original paper did.

We also provide detailed information on the datasets used and information on how they were obtained. We seek to explain the dataset properties to understand what might have an effect on the explainers, as other researchers have found that explainer performance changes per dataset [57] [2] [40].

4.1. XGDAG Experiments

XGDAG [47] is a novel methodology for disease gene discovery that leverages graph-structured data using Graph Neural Networks (GNNs) along with an explainability phase to determine the ranking of candidate genes. The method frames gene discovery as a positive-unlabeled (PU) learning problem.

4.1.1. Dataset

The XGDAG paper uses multiple datasets, each constructed with different restrictions depending on the closeness of the association between genes desired or the original source for the GDA.

The dataset used by the authors is a protein-protein interaction (PPI) network, where the nodes are proteins with an edge between them if there is an interaction. The authors closely followed the steps taken by NIAPU [64], where they took the human PPI network contained in BioGRID [63], and kept only the genes that were present in both BioGRID and DisGeNET [52]. They did two rounds of this curated data, for training, they kept a curated set of associations, which means GDAs from reliable sources, and for validation, they used all associations.

XGDAG does not treat GDA as a traditional classification problem, as it is not a binary classification problem. There are cases where there is a known association between the gene and the disease, but in other cases, the gene can have an association with the disease, but the association is unknown. To deal with this, XGDAG takes an approach known as a positive-unlabeled (PU) learning problem.

In this type of problem, there are positive and unlabeled samples, some of which are negative and some of which are unknown positive. In the use case, a positive use case means that the gene is associated with the disease. The resulting labels are positive (P), likely positive (LP), weakly negative (WN), likely negative (LN), and reliably negative (RN).

To label these unlabeled genes, XGDAG used NIAPU [64] to assign pseudo-labels to unknown genes through a Markov diffusion process. This approach works through a gene similarity matrix based on features that are specific to known disease genes. This set of features is a mixture of network-based

features and biology-informed features, giving more specific information to each disease. After this, a Markovian diffusion process is used to distribute label probabilities across the graph until a stationary distribution is obtained to assign the pseudo-labels.

In our work, we use the version of XGDAG with an underlying binary classification system. The binary system was chosen to make comparisons easier with other explanatory methods, although the authors of XGDAG highlighted that it has an adverse effect on the results [47]. For comparison, we focus on three diseases: malignant neoplasm of breast (C0006142), colorectal carcinoma (C0009402), and liver cirrhosis (C0023893). Each one is a graph, and its properties can be found in table 4.1.

In the dataset, each node represents a gene, which is the instructions for generating a protein. These diseases were picked because of the high number of seed genes. Seed genes are those labeled 0, meaning they are the genes that are known to be associated with the given disease. What counts as a seed gene varies depending on the type of association, as a more curated set has more strict requirements and a smaller number of seed genes. For testing, the XGDAG authors used a curated seed gene set during feature creation and training. They saw if other genes known to be associated were recovered while evaluating the system. The edges represent protein interactions between genes [65].

Disease	Nodes/Genes	Edges	Node features	Pos/Neg Labels
C0009402	13 328	138 334	6	13,286 / 42
C0023893	13 328	138 334	6	13,304 / 24
C0006142	13 328	138 334	6	13,288 / 40

Table 4.1: Disease Dataset Characteristics

The majority class is the Pos labels, which are 1 in the dataset. The minority class is the 0 labels, which are the genes related to the relevant disease. The high imbalance of the data that is seen in Table 4.1 was noted by the authors, and data imbalance is known to have a significant effect on model performance [47]. Given that class imbalance is a common issue in biological works [24], we believe XGDAG provides a great venue to explore this dataset property.

These three diseases have the same number of nodes, edges, and features because they are all built with the same dataset. The difference seems to be in the actual features, as these are built depending on the seed genes associated with the given disease. We also note the large size of the graphs, there is only one dataset common in the literature that has a larger node and a larger edge count [2] [57]. This allows the XGDAG experiments to be a good test of the effect that a large dataset can have on using explainers, another factor that is common in biology [27].

These dataset properties defined in Table 4.1 are different from the properties described in the supplementary materials of XGDAG [47], as the number of nodes and edges differs for all three diseases.

We assume that this difference is because we are working with the version of the data that was further modified to be a binary classification problem, as already highlighted. This version of the dataset exists to compare with the DIAMOnD [23] method; at the time of XGDAG's writing, it was one of the state-of-the-art methods for gene prioritization.

We do have the issue that there are six node attributes, as the original paper lists 4. The four features listed are heat diffusion, balanced diffusion, NetShort, and NetRing. Heat diffusion and balanced diffusion are network diffusion-based features, while NetShort and NetRing are topological features informed by the seed genes in the graph. You can find detailed explanations about them in the original paper [47]. Information about what the other two features represent is unknown.

4.1.2. Model architecture

In the original XGDAG paper, the authors use explainers as a middle step to rank priority nodes in various types of disease. This use of explainers meant that there were predictions, an explainer explaining the predictions, and then an aggregation of the results of the explanations [47].

After pseudo-labeling, the XGDAG approach uses a GraphSAGE model [25] on the generated dataset.

GraphSAGE learns the embedding of a node based on the node's features and neighbors. The equation for a GraphSAGE layer is:

$$x_i' = \sigma\left(W_1 x_i + W_2 \cdot \text{AGGREGATE}(x_j, \forall j \in \mathcal{N}(i))\right) \tag{4.1}$$

Where:

- W_1 and W_2 are learnable weight matrices
- x_i is the feature vector for node i
- $\mathcal{N}(i)$ is the neighborhood of node i
- AGGREGATE is a function that combines neighbor information (e.g., mean, max, or LSTM)
- σ is a non-linear activation function

In the paper, σ is a ReLU function [21]. The authors trained and found that a seven-layer GraphSAGE GNN is the optimal size to avoid oversmoothing.

4.1.3. Explainer implementation

With the trained GraphSAGE model, XGDAG runs the model on the graph and gets a prediction for the individual nodes. These predictions are restricted to only the genes related to the disease and to using only the immediate neighbors of the gene being predicted.

For each node, XGDAG then obtains the number of times it appeared in an explanation, as well as the sum of the node importance values that were given per explanation. The nodes are only stored if they were marked as LP. Storing the nodes in this way creates a ranked list of them, where a node is ranked higher the more times it appears in an explanation. In case two nodes appear the same number of times, we give priority to the node with the higher node importance score. A visual example of this process can be found in Figure 4.1, obtained directly from the paper [47].

For explanations, the authors of XGDAG used three explainability methods: GNNExplainer [74], GraphSVX [18], and SubgraphX [76]. For our experiments, we focus on GNNExplainer as it is one of the most commonly used explainers in the field [57] [2], as well as the explainers previously listed in chapter 2.

For the training of the PGExplainer model, we split the evaluation dataset and use 80% for training. We picked this train split given the standard use of it in machine learning. The original paper contains no information or restrictions on how PGExplainer should be trained [46].

For all the other explainers, a similar approach was taken for their hyperparameters, focusing on the values given by the respective original authors, following any indications given.

4.1.4. Replication study

We then use the ranking of candidate genes to decide which nodes to classify as related to the disease first. The original authors used different K values to see what effect this had on the final accuracy. The ranking of the nodes given their explanations is essential for our use case, as it also tells us whether the explainers follow real biological insight or if the explanation generated has no backing in reality.

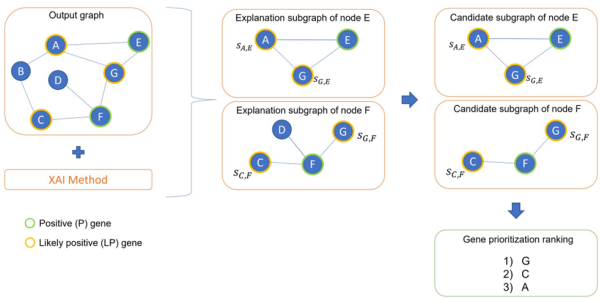

Figure 4.1: XGDAG explainability phase

To handle this replication, we had to make some changes related to the ranking. The original paper did two rankings: the first is the number of appearances, and the second is the value given per explanation. In our use case, we are not restricting the explanations to use only the immediate neighbors of the target node to be explained. Because the initial mask is often larger, we focus on the total score obtained, not on the number of appearances. Given this, we get a much larger number of our candidate genes and receive them in a different manner.

Given this difference, we will focus on the percentage of genes used. The original paper uses a simple top K genes, which is not representative of the total number of genes predicted as important. We believe that using it as a percentage is more informative. The change in the presentation of genes does mean a direct comparison with the author is not as simple as we use all of the genes in the graph, while the XGDAG authors use only a fraction of the genes. We will take into account the effect of counting genes differently during the analysis.

XGDAG measured the performance of this method by focusing on the resulting classification's F1 score. The F1 score summarizes and displays a trade-off between precision and recall.

With these changes in mind, we focus on repeating the other parts of the experiment. We use the ranked order of the genes as a list of positively labeled nodes with different thresholds, using percents instead of the top K genes in our case. From this "classification," we can obtain the precision, recall, F1 score, and AUC of the precision-recall curve.

4.2. Proteins Dataset Experiments

The Proteins dataset is a well-established benchmark in graph machine learning, specifically designed for the classification of proteins as enzymes or non-enzymes. It is part of the TU Dortmund University collection of graph benchmark datasets [50], which are widely used for evaluating graph kernel methods and graph neural networks.

4.2.1. Dataset

The dataset comprises 1,113 graphs, where each graph represents one protein. Each node in a graph corresponds to a secondary structure element, i.e., helices, sheets, and turns. The nodes are also enriched with physical and chemical information. In particular, they contain information on the hydrophobicity, the van der Waals volume, the polarity, and polarizability of the SSE represented by this node. Detailed explanation on this can be found in the work by Borgwardt et al. [5].

For the edges, there is an edge between nodes if they are neighbors in space within the protein structure, or if they are neighbors in the amino acid (AA) sequence. Each edge is labeled with its type [5].

Dobson and Doig originally compiled the Proteins dataset [16] to distinguish between enzymes and non-enzymes using structural properties. Enzymes are proteins that act as biological catalysts, accelerating chemical reactions without affecting their equilibrium [54]. The dataset is made up of 59% enzymes and 41% non-enzymes, making it slightly imbalanced towards the enzyme class.

The classification task for this dataset is to predict whether a protein functions as an enzyme (negative class) or not (positive class). This classification problem is particularly relevant in bioinformatics and drug discovery, as enzymes play crucial roles in metabolic pathways and are frequent targets for pharmaceutical interventions. Accurately classifying proteins can help identify potential drug targets and understand protein function in biological systems.

4.2.2. Model architecture

Following the benchmarking approach of GNN-X-Bench [40], we implemented four different Graph Neural Network (GNN) architectures for the protein classification task. Each model follows a general structure consisting of multiple GNN layers for feature extraction, followed by a max pooling operation to obtain a graph-level representation, and finally a multi-layer perceptron (MLP) for classification.

For all models, we used a consistent architecture with three GNN layers (each with a hidden dimension of 20), each being followed by batch normalization [33], and a ReLU operation. After the three layers pass through the dataset, the model performs a max pooling operation and a one-layer MLP for the

final prediction. We implemented four different types of GNN layers to compare their performance and explanatory capabilities.

Graph Convolutional Network (GCN)

The GCN [38] is a fundamental GNN architecture that performs message passing based on the normalized adjacency matrix. The propagation rule for each GCN layer is defined as:

$$\mathbf{H}^{(l+1)} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)} \tag{4.2}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$, and $\mathbf{W}^{(l)}$ is the learnable weight matrix for layer l.

GCN performs a localized first-order approximation of spectral graph convolutions, effectively aggregating features from a node's immediate neighborhood.

Graph Attention Network (GAT)

The GAT [67] introduces attention mechanisms that allow nodes to attend differently to their neighbors. The update rule for a GAT layer is:

$$\mathbf{x}_i' = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j \tag{4.3}$$

Where:

- $\mathbf{x}_i'$: The updated feature vector of node i after applying the GAT convolution operation. This value represents the node's features in the next layer.
- $\mathbf{x}_i$: The current feature vector of node i in the input layer.
- $\mathbf{x}_j$: The feature vector of node j, which is a neighbor of node i.
- $\mathcal{N}(i)$: The set of neighboring nodes of node i in the graph.
- $\mathbf{\Theta}$: The learnable weight matrix that linearly transforms the node features. This matrix is a shared transformation applied to all nodes.
- $\alpha_{i,j}$: The attention coefficient between node i and node j.

The attention coefficient $\alpha_{i,j}$ is as follows:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_1^T\mathbf{\Theta}\mathbf{x}_i + \mathbf{a}_2^T\mathbf{\Theta}\mathbf{x}_j\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_1^T\mathbf{\Theta}\mathbf{x}_i + \mathbf{a}_2^T\mathbf{\Theta}\mathbf{x}_k\right)\right)} \tag{4.4}$$

Where:

- $\mathbf{\Theta}$: The learnable weight matrix that transforms node features linearly.
- $\mathbf{x}_i$ and $\mathbf{x}_j$: Feature vectors of nodes i and j, respectively.
- $\mathbf{a}_1$ and $\mathbf{a}_2$: Learnable parameter vectors of the attention mechanism. These vectors are part of the attention mechanism that computes the importance of node relationships.
- LeakyReLU: The Leaky Rectified Linear Unit activation function, typically with a negative slope of 0.2. It adds non-linearity to the attention mechanism.
- $\mathcal{N}(i) \cup i$: The neighborhood of node i, including node i itself (accounting for self-loops).
- $\exp(\cdot)$: The exponential function, used to ensure the attention coefficients are positive.

GraphSAGE

GraphSAGE [25] (SAmple and aggreGatE) is designed for inductive learning on large graphs by sampling and aggregating features from a node's local neighborhood. The update rule for a GraphSAGE layer is:

$$\mathbf{x}'_i = \mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \cdot \text{mean}_{j \in \mathcal{N}(i)} \mathbf{x}_j \tag{4.5}$$

where:

- $\mathbf{x}'_i$ is the updated feature vector of node i after the GraphSAGE operation
- $\mathbf{x}_i$ is the current feature vector of node i
- $\mathbf{x}_j$ is the feature vector of node j, which is a neighbor of node i
- $\mathcal{N}(i)$ is the set of neighboring nodes of node i in the graph
- $\mathbf{W}_1, \mathbf{W}_2$ are learnable weight matrices for transforming the node features
- $\text{mean}_{j \in \mathcal{N}(i)}$ is the mean aggregation function that averages the features across all neighbors

The GraphSAGE operator first aggregates information from a node's neighborhood using a mean pooling operation, and then combines this neighborhood representation with the node's features through learnable weight matrices. This formulation allows the model to learn both node-specific and neighborhood-based patterns in the data.

Graph Isomorphism Network (GIN)

GIN [72] is designed to maximize the representational power of GNNs, being as powerful as the Weisfeiler-Lehman graph isomorphism test. The update rule for a GIN layer is:

$$\mathbf{h}_i^{(l+1)} = \text{MLP}^{(l)} \left((1 + \epsilon^{(l)}) \cdot \mathbf{h} i^{(l)} + \sum j \in \mathcal{N}(i) \mathbf{h}_j^{(l)} \right) \tag{4.6}$$

Where $\epsilon^{(l)}$ is a learnable parameter or a fixed scalar (we used a learnable parameter), and MLP is a multi-layer perceptron. In our implementation, the MLP consists of two fully connected layers with a normalization layer between each.

GIN achieves injective aggregation by using sum pooling over the neighborhood and leveraging the universal approximation capabilities of MLPs.

Training and Hyperparameters

Following GNN-X-Bench [40], we trained all models using the Adam optimizer with a learning rate of 0.001. The models were trained for 100 epochs with early stopping (patience of 200 epochs) based on validation performance. We used a batch size of 128 and performed 10 different runs per model.

5

Results

In this section, we focus on answering how the different explainers perform in different types of graph problems, as well as understanding the effect that the dataset has on the explanations. We compare two biomedicine datasets, GDA and Proteins. We are also able to compare the three different explainer types (Perturbation, Gradient, and Surrogate) in the two different problem types, to see if there is some common trend between explainer types.

We hope that our findings help highlight essential caveats in the biomedical domain and provide insights into what actions to take when using explainers in biomedicine.

5.1. XGDAG results

We perform tests with XGDAG [47]. The paper used the NIAPU approach [64], applying explanations to the resulting prediction of Markovian diffusion to rank the importance of the nodes for further ranking. For our use case, we used the Markovian diffusion prediction and then analyzed the result, analyzing the usage of explainers by the authors of XGDAG [47].

The results are in tables 5.2 and 5.1. For each metric, we highlight the best-performing explainer.

We calculate the maximum Entropy by assuming that the entire mask is made of ones, which gives the ”noisiest” possible prediction. For most models, this value is 11.29, rounded to two decimal places. In the case of GraphLIME, it is 1.79 (rounded), because GraphLIME only works on features. For PGExplainer, it is higher given the size of the edge count, being 13.22. For our results, we normalize each value by dividing it by the corresponding maximum Entropy, so the result displayed in the table is the percent. We did this normalization to make the analysis easier. We made a similar change for the Counterfactual. The original values can be found in Table A.1.

For the Counterfactual value, there were multiple cases where the explainer found no way of changing the prediction. In those cases, we set the individual counterfactual value to be the number of elements that can be changed for the explanation given + 1.

We tested the repeatability 30 times for each node, and we performed the kernel distance experiment 20 times on each graph set. This last one is due to the randomness of the chosen kernel, as we want to know how reliable the method is in obtaining the Kernel Distance.

The last important detail to note is that we did not calculate the Kernel Distance in GraphLIME or PGExplainer. We were not able to calculate it due to restrictions on the given graph, which made a fix for these methods not available in time for the report. For similar reasons, we exclude Zorro from this result analysis.

Additionally, we add information on the tables on whether it is a gradient-based approach (G), a perturbation-based approach (P), or a surrogate method (S). We use the categorization of Kakkad et al. [35] and of Rathee et al. [57] for this. In case an approach is not categorized in the two previous papers, we have

Explainer	Time seconds ↓	RDT-Fidelity ↑	Fidelity- ↓	Fidelity+ ↑	Kernel Distance ↑
Deconvolution (G)	0.02 ± 0.01	0.78 ± 0.37	0.45 ± 0.5	0.01 ± 0.1	0.53 ± 0.18
DummyExplainer (D)	**0.01 ± 0.0**	0.62 ± 0.45	0.4 ± 0.49	0.42 ± 0.49	0.24 ± 0.1
GNNExplainer (P)	2.21 ± 0.12	0.65 ± 0.4	**0.19 ± 0.39**	0.4 ± 0.49	**0.54 ± 0.22**
GraphLIME (S)	1.9 ± 3.11	0.56 ± 0.46	0.45 ± 0.5	0.0 ± 0.0	N/A
GuidedBackprop (G)	0.02 ± 0.01	0.78 ± 0.37	0.45 ± 0.5	0.01 ± 0.1	0.51 ± 0.16
InputXGradient (G)	0.02 ± 0.01	**0.79 ± 0.35**	0.31 ± 0.46	0.21 ± 0.41	0.5 ± 0.15
IntegratedGradients (G)	1.02 ± 0.09	0.78 ± 0.35	0.26 ± 0.44	**0.5 ± 0.5**	0.48 ± 0.2
PGExplainer (P)	64.01 ± 12.83	0.7 ± 0.46	0.44 ± 0.5	0.0 ± 0.0	N/A
Saliency (P)	0.02 ± 0.01	0.56 ± 0.46	0.45 ± 0.5	0.0 ± 0.0	0.51 ± 0.2

Table 5.1: XGDAG Fidelity Metrics

Explainer	Repeatability ↑	VoE ↓	Entropy % ↓	Counterfactual % ↓
Deconvolution (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.49 ± 0.11	0.51 ± 0.50
DummyExplainer (D)	0.0 ± 0.0	6663.8 ± 4.11	0.98 ± 0.0	0.61 ± 0.44
GNNExplainer (P)	0.08 ± 0.08	388.77 ± 133.14	0.82 ± 0.06	0.51 ± 0.50
GraphLIME (S)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.25 ± 0.14	0.84 ± 0.38
GuidedBackprop (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.49 ± 0.11	0.51 ± 0.50
InputXGradient (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.37 ± 0.12	0.51 ± 0.50
IntegratedGradients (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.37 ± 0.01	**0.50 ± 0.50**
PGExplainer (P)	**1.0 ± 0.0**	**0.0 ± 0.0**	**0.14 ± 0.07**	0.80 ± 0.63
Saliency (P)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.49 ± 0.11	0.53 ± 0.50

Table 5.2: XGDAG Other Metrics

done the categorization based on the descriptions given in chapter 2. For the DummyExplainer, which is random, we will use (D) to distinguish it from the rest.

5.1.1. Metric Results

Time

For the time in seconds, DummyExplainer performed best, so it was the fastest in getting explanations. DummyExplainer's performance is expected; it just generates random values. The gradient methods tend to outperform all the other methods, which also makes sense, as they are a linear operation, instead of the search done by the perturbation and surrogate methods.

It is relevant to note that PGExplainer had the longest run time by far. We assume that this is mainly the fault of the training time, although it was done once per model to be explained, it is still a rather lengthy operation. The long time to use is contrary to what the authors of PGExplainer describe [46], but we believe that the more explanations that are needed, the smaller this difference in time.

RDT-Fidelity

For RDT-Fidelity, we have InputXGradient as the best-performing metric overall. In general, you can see that most gradient methods have similar RDT-fidelity mean and variance. A higher score for RDT-Fidelity implies that the explanation is more stable and more accurate to the original model.

The most relevant fact is that the worst-performing explainer in this case is not the DummyExplainer, but the Saliency maps. In fact, Saliency maps and GraphLIME had similar performance in both mean and variance. The fact that GraphLIME had the worst performance is interesting, as the original implementation of RDT-Fidelity is focused on feature explanations [57], which are what GraphLIME targets.

Fidelity+ and Fidelity-

We find that GNNExplainer is the best-performing model for Fidelity-, while IntegratedGradients is the best-performing model for Fidelity+. A low Fidelity- implies that the explanation is sufficient, so using

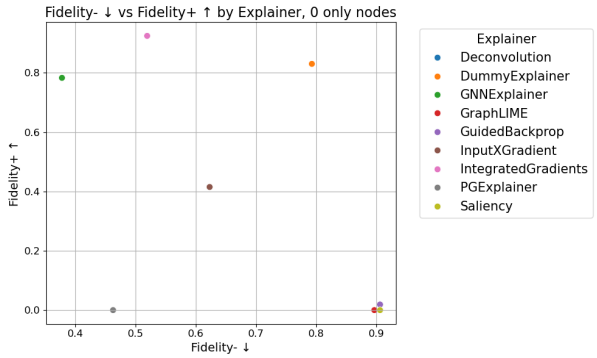

Figure 5.2: Fidelity comparison per explainer 0 nodes

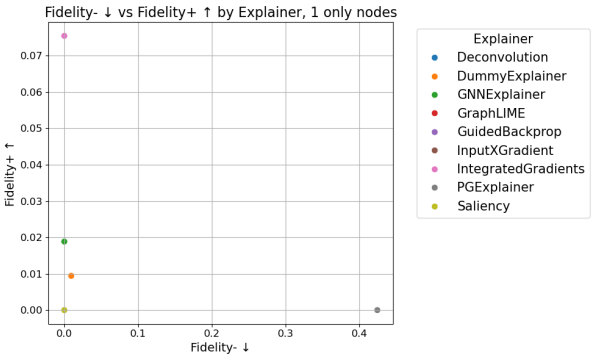

Figure 5.3: Fidelity comparison per explainer 1 nodes

only the elements highlighted by the explanation does not change the prediction of the model. A high Fidelity+ means the explanation is necessary, so removing the explanation changes the prediction of the model.

In figure 5.1 we can see how the two metrics trade off. In this plot, an explainer that goes to the upper left has the best performance, and the bottom right corner has the worst. We can notice that while some models have a trade-off, it tends to be the case that a model with a good Fidelity+ will have a better Fidelity-, and vice versa. In particular, we notice that IntegratedGradients and GNNExplainer are the best in these two metrics, with GNNExplainer having better Fidelity- (sufficient explanations) and IntegratedGradients having better Fidelity+ (necessary explanations).

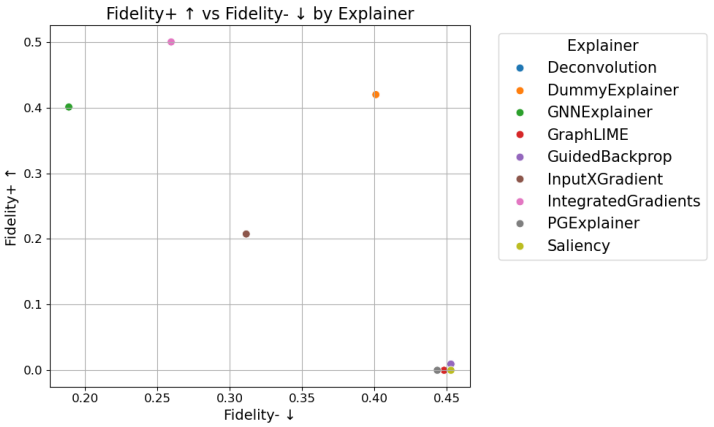

Figure 5.1: Fidelity comparison per explainer

We also notice that in the bottom right of figure 5.1, many of the explainers cluster together. This implies that they provided neither Sufficient nor Necessary explanations. There is at least one explainer for each class of explainers, so there is no simple explanation for why these four explainers perform so poorly. DummyExplainer is not among these four explainers. So, somehow, these four explainers were actively worse than a random explanation.

To find more information about why this happened, we filtered the responses on the positive and negative labels. XGDAG focused on predicting negative labels, so the elements the model predicted as 0. You can find those plots in figures 5.2 and 5.3. In XGDAG [47], the authors focused on the 0-labeled nodes, which are the ones that are related to the disease described in the specific graph.

It is very striking how significant the change in performance is for all models in both cases. You can see this difference in performance with the values used in the axes, with the plot focusing on 0 nodes having values that get closer to 1. The plot for the 0 nodes feels very similar to the plot with all the data, but with more extreme values. In the 0-node plot, we still notice that GNNExplainer and IntegratedGradients outperform most other models.

By comparison, in the 1-node plot, we see that all of the explainers perform worse. We believe that this is due to the model being trained to focus on the minority class, but we have difficulties understanding the exact effect. If the original model was overtrained to predict 0 labels, then the Fidelity+ metric would not be as high as it was, as the model would not change predictions when removing the original explanations.

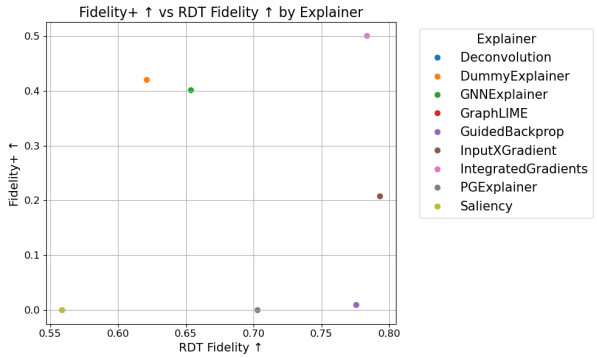

Figure 5.4: Fidelity+ vs RDT-Fidelity

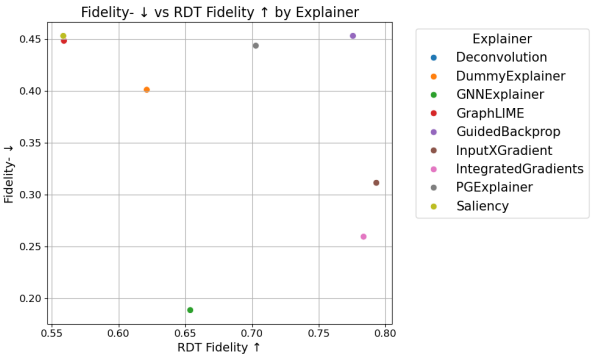

Figure 5.5: Fidelity- vs RDT-Fidelity

The last thing to note is that PGExplainer was the only explainer not to change its results when comparing 0 vs. 1 predictions, at least not substantially. The explainer was never able to find the necessary explanations (Fidelity +), which is also strange when you consider that the explainer had a good performance for its RDT-Fidelity score. So, it seems that the explanation was able to follow model behavior (RDT-Fidelity), but was unable to provide the necessary information to keep the same prediction (Fidelity+). We believe that this difference is due to how the two different methods use noise, with RDT-Fidelity using random noise for the non-target nodes [22], and Fidelity+ setting them to 0 [2].

We also believe that part of this issue comes from the fact that PGExplainer is the only method that focuses on the edges. All other explainers work with nodes, features, or both. The original authors of XGDAG highlighted the importance of the underlying structure [47], and the features are built partly on top of the connections between the genes (nodes) themselves. As such, we believe that changes in the edges are another one of the main reasons why PGExplainer has such consistent performance, compared to all the other explainers.

We find that DummyExplainer has a surprisingly high Fidelity+, which is equivalent to a Necessary explanation. That means that randomly picking elements tends to give you the genes necessary to predict a disease. This performance does not really make sense if seen in a vacuum, but we believe that this is due to the class imbalance in the dataset and how the model was trained to adapt for this.

You can see a graph comparing RDT-Fidelity with Fidelity+ in figure 5.4, and a plot comparing RDT-Fidelity with Fidelity- in figure 5.5.

For the comparison between RDT-Fidelity and Fidelity+, there seems to be a curve similar to exponential growth. It appears that higher RDT-Fidelity values only affect Fidelity+ when reaching more extreme values, at which point the differences are more pronounced. This effect might imply that RDT-Fidelity is more sensitive and is better at expressing minor differences in accuracy. You can also notice a similar curve downwards when comparing Fidelity- with RDT-Fidelity. We believe this helps reinforce that RDT-Fidelity is more sensitive compared to the other Fidelity measurements.

We also want to highlight that in both cases, GNNExplainer and DummyExplainer were the only explainers that were outside of the curve. Given this result, the property of the curve might only apply to gradient-based methods. PGExplainer is the other perturbation method that we could compare with, which might confirm or deny this idea. However, we believe that PGExplainer overfitted on the training set; as such, we cannot use it properly for comparison.

Entropy

For the entropy values, we notice that PGExplainer and GraphLIME have the explanations with the lowest Entropy. Lower Entropy means the explanations had fewer elements highlighted. Explanations with fewer elements means that the explanations are smaller and easier to understand [57].

There are only 6 features in the dataset. The current Entropy that GraphLIME obtained means it consistently picked 2 features as relevant. The authors built 2 of the features using the biological information of what genes were already known to be related to the disease. So, in a way, GraphLIME might be

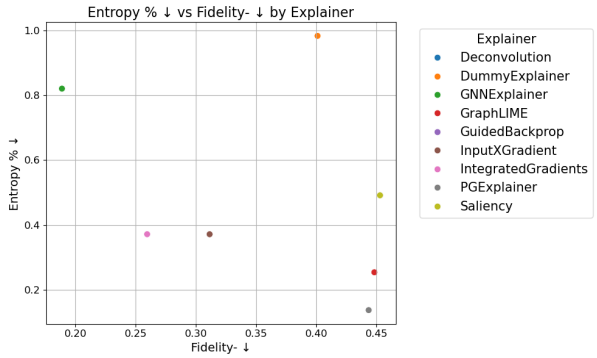

Figure 5.6: Entropy vs Fidelity-

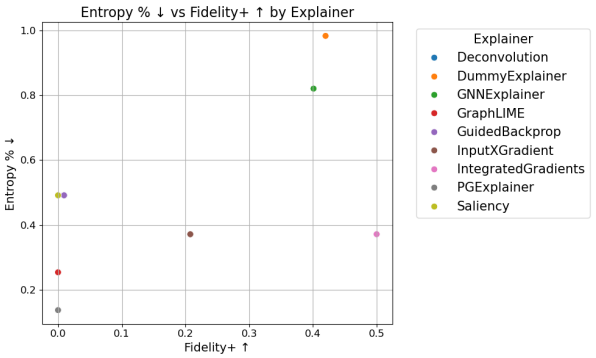

Figure 5.7: Entropy vs Fidelity+

”cheating” because it uses features that already see the seed genes. However, given the issue with the features described in chapter 4, we cannot confirm if this is the case.

The Entropy itself is not a sufficient comparison, as an explainer can highlight none of the features and get a perfect entropy. The more interesting comparisons are when you compare Entropy with the Fidelity values. The comparison with Fidelity- can be found in figure 5.6, and the comparison with Fidelity+ can be found in figure 5.7.

In figure 5.7, we notice a slight trend where a higher Entropy leads to a higher Fidelity+. This finding is in line with previous findings [57], and makes intuitive sense. The less information a system has, the less likely it is to be able to replicate the original approach closely. The only explainer that somewhat breaks the trend is IntegratedGradients.

The results of PGExplainer are now more interesting. The explainer would consistently predict a few edges, but those edges would often be wrong. We believe that this is because we overfit the PGExplainer model on the training data. Our results on repeatability and variance later on help reinforce this idea.

We quickly want to mention that the high Entropy for GNNExplainer is partly due to the lack of fine-tuning on our end. GNNExplainer allows for control of the size of the subgraph and feature explanations, using these as a hyperparameter [74]. The original paper describes that these features should be informed based on prior knowledge of the dataset, but in our use case, we decided not to optimize it. Proper guidelines on how to optimize these hyperparameters might be needed.

Kernel Distance

A higher Kernel Distance (closer to 1) means that the subgraph generated from the explanations is a graph that is close to the domain of the original graphs. Having a higher Kernel Distance is important because it means that the explanations are more likely to be possible in reality. For the sake of comparison, we also calculated the Kernel Distance between the three graphs we are using for the comparison. This value was $0.3230.314$.

For Kernel Distance, GNNExplainer has masks that are closer to the original plot. This might come from the fact that it’s perturbation-based. All of the explainers obtained a lower Kernel Distance value compared to the distance between the graphs of the original dataset. The only method that performed worse was DummyExplainer, which is good as it means that the methods are able to give realistic explanations.

We find that the Kernel distance that uses only the original graphs is relatively low, considering most of the graphs are the same. This might indicate that the Kernel that we used is too sensitive to differences in graphs and that we need to find alternatives.

Repeatability and Variance over Explanations (VoE)

For repeatability and VoE, we have similar results. High repeatability means that the top-K features were the same across multiple runs, and low VoE implies that the explanation had a low overall variance. As such, both metrics measure how consistent the explanations are.

Only DummyExplainer and GNNExplainer give non-perfect results; all of the other explainers get the perfect score for the corresponding metric. We expected to have non-perfect results with DummyExplainer. The VoE being different for GNNExplainer is consistent with other studies [57], but the fact that PGExplainer has a variance over the explanations of 0 is concerning. Given the lack of information on how to train the explainer in the original PGExplainer paper [46], we believe that we overfit the training data during training. We followed the parameters set up by the author, but in this case, each explainer was trained on only one graph, training on multiple nodes. We assume that the original parameters were meant for multiple graphs.

Counterfactual

A low Counterfactual means that the given explanation is able to rank the features in a way that makes it easier to change the prediction. A low Counterfactual implies that we can use the explainers to find what is necessary to change the outcome.

For the Counterfactual, IntegratedGradients performed the best. By comparison, PGExplainer and GraphLIME were the worst-performing models, doing worse than even DummyExplainer. However, this result does not give all the information.

The reason this is happening is because the explainers were either able to find a Counterfactual immediately or were not able to find it at all. You can see this result in figure 5.8. In the violin plot, it is easy to notice that for almost all explainers, the results cluster either at 0 or at 100 percent. The only exceptions are DummyExplainer, PGExplainer, and GraphLIME.

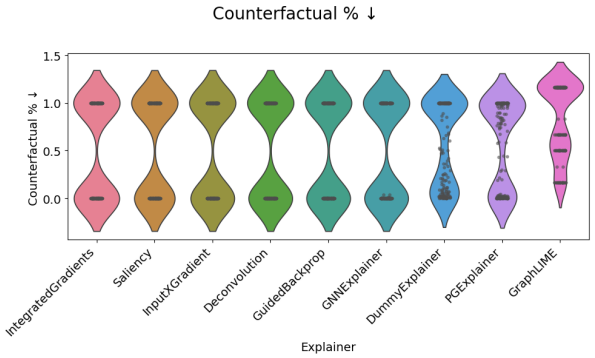

Figure 5.8: Counterfactual % XGDAG

The reason the values are so extreme is because of the class imbalance the model had to learn. This imbalance can be seen for the counterfactual values for 0 labeled nodes and for 1 labeled nodes, as can be seen in Figures 5.9 and 5.10, respectively. If the original label was 0, finding a Counterfactual tended to be trivial. If the original label was 1, finding a Counterfactual was impossible for most models.

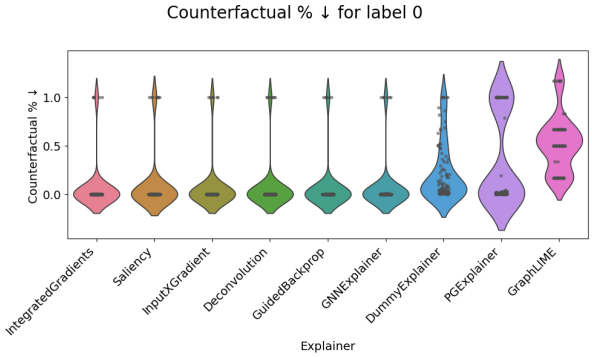

Figure 5.9: Counterfactual 0 labeled nodes

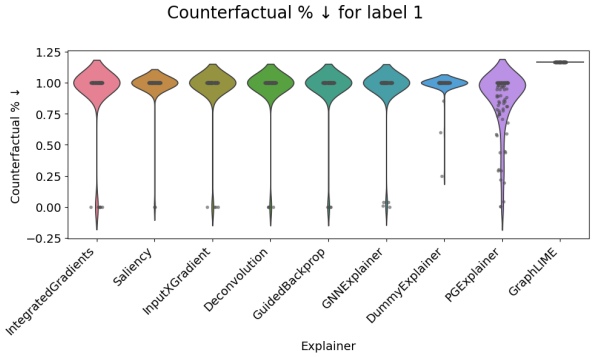

Figure 5.10: Counterfactual 1 labeled nodes

The reason behind these results makes some sense. If a gene has no relation to the target disease (1 labeled), having it be responsible for the disease is not an easy change. However, genes related to a disease are sensitive, so slight changes might mean that they stop being related to the disease (0 labeled).

Given our results for Counterfactual, we believe that a good venue for research would be to test genes for which a counterfactual value was found, for those genes that are not related to the disease (so 1 labeled). Given that the focus of this problem is to find new links between known genes and diseases

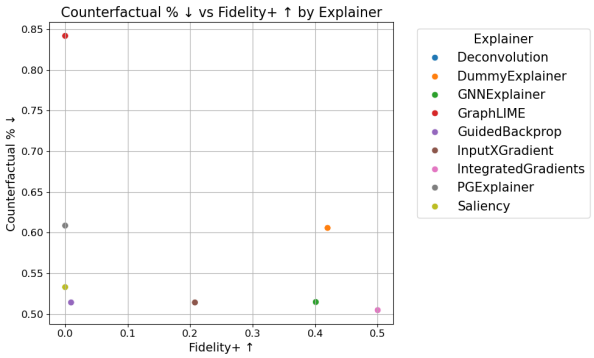

Figure 5.11: Fidelity+ vs Counterfactual all

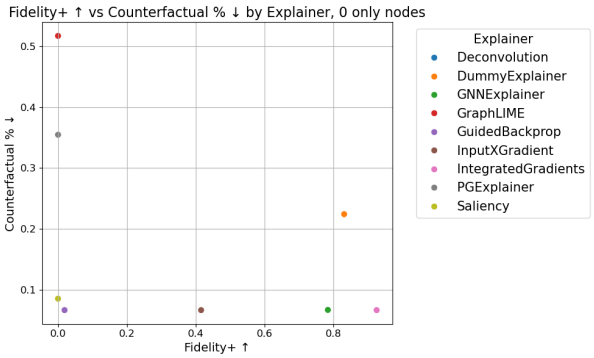

Figure 5.12: Fidelity+ vs Counterfactual 0 labeled nodes

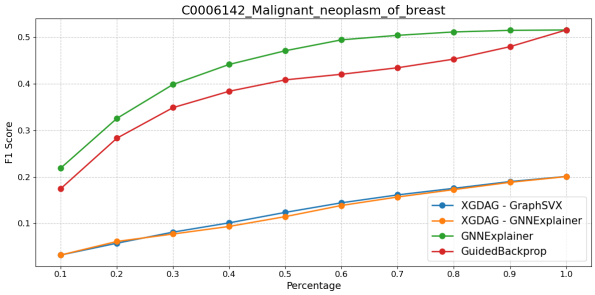

Figure 5.13: Percent Malignant Neoplasm of Breast F1 score XGDAG vs New results

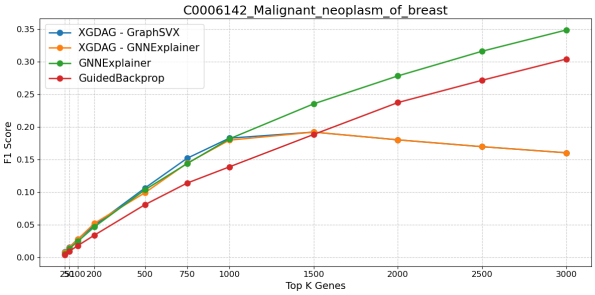

Figure 5.14: Top K Malignant Neoplasm of Breast F1 score XGDAG vs New results

[47], these might be a relevant venue for future exploration. In particular, the results of PGExplainer might be most interesting, as that was the best-performing explainer when filtered for 1 labeled nodes.

Lastly, we want to compare these results with Fidelity+, as that metric also sees how efficient the mask is at changing the prediction. We have the plot comparing the two results in figure 5.11, and we also do the plot filtering for only 0 labeled nodes in figure 5.12.

There is a significant decay in quality in the scores of Fidelity+ and Counterfactual when you filter to only the 0-labeled nodes. We note that a high Fidelity+ tends to correlate with a lower Counterfactual %. Based on these plots, we would define that only the explainers on the bottom horizontal line (GuidedBackprop, InputXGradient, GNNExplainer, and IntegratedGradients) should be used for finding Counterfactual values.

5.1.2. Replication results

First, we compare the performance scores obtained using our methods to the performance scores obtained in the original paper. We focus on the F1 score as the metric to compare, as that was the value on which the original authors focused the most. Given that it provides a trade-off between precision and recall, it also provides more information than using either value separately. The exact scores for the four metrics, recall, precision, F1, and AUC, can be found in the Appendix A.

We have a comparison that uses the top K genes, using the K values presented by XGDAG. Additionally, we transform the values to percents of nodes predicted, given that we did not restrict our methods to the immediate neighborhood.

The F1 score values for the C0006142 malignant neoplasm of the breast can be found as a percent in figure 5.13, and with the top K values in figure 5.14. This plot compares some of our best-performing explainers versus the results obtained in XGDAG. The Figures for the other two diseases can be found in the Appendix A.

The F1 score for the different explainers presented in Figure 5.14 shows that our approaches are competitive with XGDAG. Our results imply that you do not need to filter on immediate neighboring

nodes to find relevant genes, as the explainers seem to highlight the relevant genes no matter what.

We see that by using more nodes, we are able to achieve great performance in Figure 5.13. While the number of genes is much higher when using the overall graph, it seems that using all of the genes in the graph is still a valid strategy for finding relevant genes.

However, we believe that these results using the entire graph highlight that most of the nodes in the graph are somewhat related to the disease, not that our approach is a good alternative to filtering on neighbors. This result can imply that either the entire graph is relevant, or that the list of seed genes is too extensive and needs stricter requirements when finding a link to the disease. We believe that this issue is also in the Figure that compares all of the different explainers we tested with, which can be found in Figure 5.15.

The only part that matters in Figure 5.15 is that DummyExplainer is a competitive explanation model to retrieve new genes. We believe that this happens because the list of test genes overlaps very closely with the graph. As such, predicting all of the genes as being related to the disease is a valid solution, which does not make sense. This reinforces our belief that the underlying graph is the most crucial factor in the prediction of new candidate genes, not the explainers used.

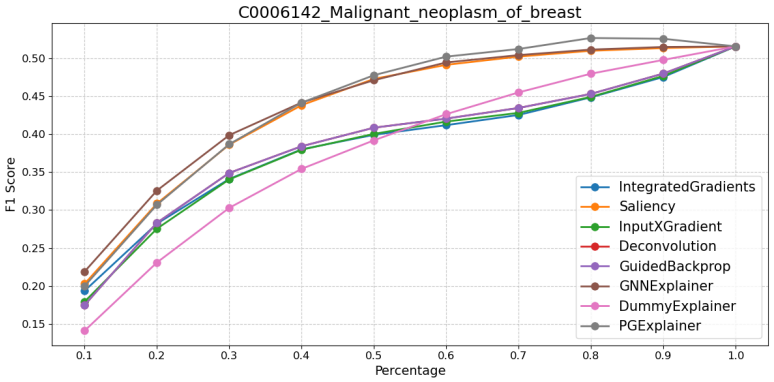

Figure 5.15: Malignant Neoplasm of Breast All Explainers

5.1.3. Conclusion of XGDAG results

Based on the initial results, we can observe the complexity of the biological field in explanations. On the metric side of things, we can see many trade-offs between Fidelity and Entropy that have often been outlined in the literature [57] [2], which helps to reinforce the importance of these particular metrics.

There did not seem to be a common grouping between the different strategies. The two perturbation methods (GNNExplainer and PGExplainer) would often get contrary results. If PGExplainer had low entropy results with low Fidelity, GNNExplainer would get high entropy results with high Fidelity. There was no trend like this with the gradient methods either, so it does seem that each method is really fine-tuned towards specific needs.

IntegratedGradients seemed to be the method that found the best trade-off between a high fidelity score and a lower Entropy score, so this approach is likely best when you need to find a human explanation. We postulate that the sensitivity axiom defined in the original paper of IntegratedGradients [66] is responsible for this, as it is the most significant difference from other guided backpropagation methods. The sensitivity axiom says that if two inputs differ on a feature, and that causes a different prediction, that feature must be used.

Counterfactual explanations seem to correspond to being necessary explanations (Fidelity+ in our case). This finding is in line with the findings of Amara et al. [2] and Wachter et al. [68], which makes some sense. Removing the features that are highlighted changes the prediction, whether you do it one by one or in a batch. The advantage that Counterfactual seems to provide is that it is more sensitive to changes in the graph, compared to the results obtained from Fidelity+.

We believe the most important finding was not the results of the explainers themselves, but the difference in performance between the majority and the minority class. We found this result in multiple metrics. This result highlights the idea that the use of explainers needs to take into account the domain where they are being used, as the dataset and the model trained on the dataset can have a significant influence on the explainers [81] [1] [36]. Users should consider what target class they are trying to

Explainer	Time seconds ↓	RDT-Fidelity ↑	Fidelity- ↓	Fidelity+ ↑	Mask Kernel Distance ↑
Deconvolution (G)	0.01 ± 0.0	0.87 ± 0.26	0.08 ± 0.27	0.69 ± 0.46	**0.0007 ± 0.0001**
DummyExplainer (D)	**0.0 ± 0.0**	0.86 ± 0.27	0.21 ± 0.41	0.19 ± 0.39	0.0005 ± 0.0
GNNExplainer (P)	0.85 ± 0.15	0.87 ± 0.27	0.35 ± 0.48	0.0 ± 0.0	0.0006 ± 0.0001
GuidedBackprop (G)	0.01 ± 0.0	0.87 ± 0.26	0.08 ± 0.27	0.69 ± 0.46	**0.0007 ± 0.0001**
InputXGradient (G)	0.01 ± 0.0	0.86 ± 0.28	0.41 ± 0.49	0.38 ± 0.48	**0.0007 ± 0.0001**
IntegratedGradients (G)	0.31 ± 0.08	**0.87 ± 0.28**	0.4 ± 0.49	0.33 ± 0.47	0.0006 ± 0.0001
Saliency (G)	0.01 ± 0.0	0.77 ± 0.34	**0.02 ± 0.12**	**0.7 ± 0.46**	**0.0007 ± 0.0001**

Table 5.3: Proteins Fidelity Metrics

Explainer	Repeatability ↑	VoE ↓	Entropy % ↓	Counterfactual % ↓
Deconvolution (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.86 ± 0.06	0.75 ± 0.33
DummyExplainer (D)	0.02 ± 0.03	120.44 ± 119.39	0.97 ± 0.0	0.8 ± 0.29
GNNExplainer (P)	0.35 ± 0.21	10.84 ± 20.48	0.95 ± 0.03	0.7 ± 0.34
GuidedBackprop (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.86 ± 0.06	0.75 ± 0.33
InputXGradient (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	**0.69 ± 0.08**	0.74 ± 0.33
IntegratedGradients (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.71 ± 0.07	**0.73 ± 0.34**
Saliency (G)	**1.0 ± 0.0**	**0.0 ± 0.0**	0.86 ± 0.06	0.75 ± 0.33

Table 5.4: Proteins Other Metrics

predict when using any explainer.

5.2. Protein Results

Results for the Proteins dataset [50] can be found in tables 5.3 and 5.4. The values were rounded to 2 decimal places for the sake of fitting them in the table. In case one of multiple rounded values has a better unrounded performance, we will highlight the best version. The tables presented contain the information for all the GNN models; the results per model can be found in the appendix A.

Zorro and GraphLIME were not used in these comparisons, as those methods focus on node-level explanations. We were not able to use PGExplainer due to time constraints.

Maximum Entropy per graph was calculated assuming the entire mask is made of ones, giving the ”noisiest” possible prediction. The Entropy obtained was divided by this maximum Entropy, giving the Entropy % value that is shown in table 5.4.

We take a similar approach for Counterfactual %, where we take the number of nodes in the respective graph as the maximum counterfactual value. In case the explanation was not able to find a Counterfactual value, we set the Counterfactual value to be number of nodes $+ 1$.

We ran each explainer 30 times for each input to calculate the Repeatability and VoE values. For the Kernel Distance experiment, we initialized a new Kernel 20 times for the masks each explainer gave for each model type. This last one is to account for the randomness of the chosen kernel.

We also calculate the Kernel Distance within graphs. We define this as the distance between the graphs of the original dataset, without any modification. For the Proteins dataset, the resulting Kernel Distance within graphs was 0.0.

5.2.1. Metric Results

Time

The overall trend of the results is similar to what we found when comparing with the XGDAG experiments. Gradient methods are fast; perturbation methods take longer.

RDT-Fidelity GNN Model	All	Saliency	Deconvolution
GAT	0.8032	0.739018	0.830982
GCN	0.8359	0.814911	0.853661
GIN	0.8389	0.652232	0.849464
SAGE	0.9264	0.893036	0.928839

Table 5.5: Proteins RDT-Fidelity

RDT-Fidelity

In this case, RDT-Fidelity gives quite different results. The most striking fact is that most of the explainers have a similar score. The fact that most of the explainers are so similar can imply that the underlying model is much more consistent, compared to the variance obtained in the XGDAG results. This is reinforced when you consider that DummyExplainer had these consistent results as well, giving random values to the mask had no visible effect on the RDT-Fidelity score.

We want to note that in this case, GNNExplainer had an RDT-Fidelity score that was similar to the Perturbation methods, which differs from the results obtained from the XGDAG experiments. Given how RDT-Fidelity is calculated, this might imply that the underlying model remains much more resistant to noise compared to the model used for the XGDAG method. Given that DummyExplainer also had such a high RDT-Fidelity score, there is some concern about how well the RDT-Fidelity was able to capture the nuance of the explanations.

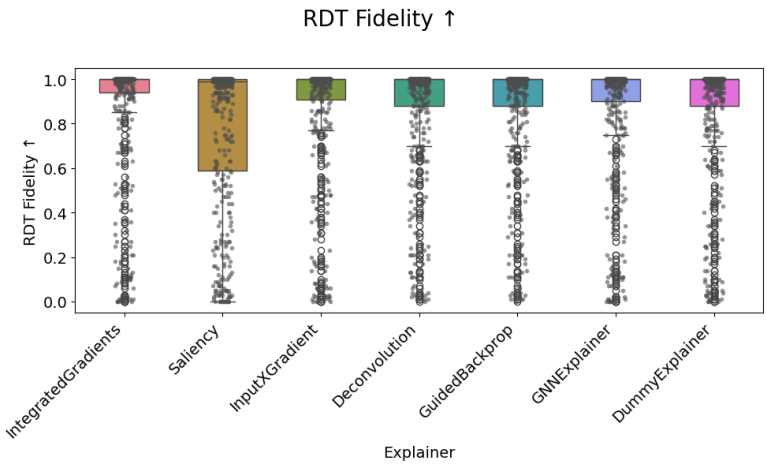

Figure 5.16: RDT-Fidelity per explainer proteins

The only outlier in these results is the Saliency maps, which had the worst performance. As can be seen in Figure 5.16, the difference comes in large part due to the higher variance. This result is also in line with our previous results, where the Saliency explainer was the worst performer while we were doing experiments with the XGDAG method.

The reasoning behind this can be hinted at in the summarized table 5.5. Here, we show the RDT-Fidelity values of all the explainers averaged, Saliency only, and Deconvolution only. It is striking that Saliency performs worst no matter the GNN, but it is also considerably worse when a GIN architecture is involved. As of writing, we cannot find an explanation for this difference in behavior. We believe that the fact that an absolute value is used for Saliency maps is the reason for this, as it is the most significant difference in procedure compared to other gradient methods. Still, we have not found an explanation for why this function ends up having such influence on RDT-Fidelity.

Fidelity+ and Fidelity-

For these metrics, we find more variance in the values. The most notable fact is that Saliency maps outperform all the other methods in both Fidelity+ and Fidelity-.

The Figure in 5.17 leads to one other question: why do Saliency maps perform so well with Fidelity+ and Fidelity- but struggle with RDT-Fidelity? We do not have a good answer to this question at the moment.

Based on the figure 5.17, we see three clusters forming: Saliency, Deconvolution, and GuidedBackpropagation; InputXGradient and IntegratedGradients; DummyExplainer and GNNExplainer. These

clusters were recurring in our tests with the protein dataset. Deconvolution is not as visible in Figure 5.17, but we can see that it is part of this cluster when looking at Figures 5.18 and 5.19.

Another interesting point is the results of the DummyExplainer. This explainer had a good RDT-Fidelity and Fidelity- score, while having a poor Fidelity+ score. Why was it so difficult to have the model change predictions? Keeping random elements still somehow keeps the same label. The fact that the model was able to keep accurate predictions with random changes means that either the edges or the number of nodes are the most essential features for the models, and none of the explainers managed to capture that information appropriately. This result also tells us that our approach is lacking in topology-based method analysis, which could be limited due to our greater focus on identifying nodes and node features. Additional tests with explainers that give edge explanations might be needed, or some way of finding out if this is the case. This result also clashes with some of the findings of BAGEL [57], where they state that high RDT-Fidelity correlates with high stability, given that the random explanation had high RDT-Fidelity.

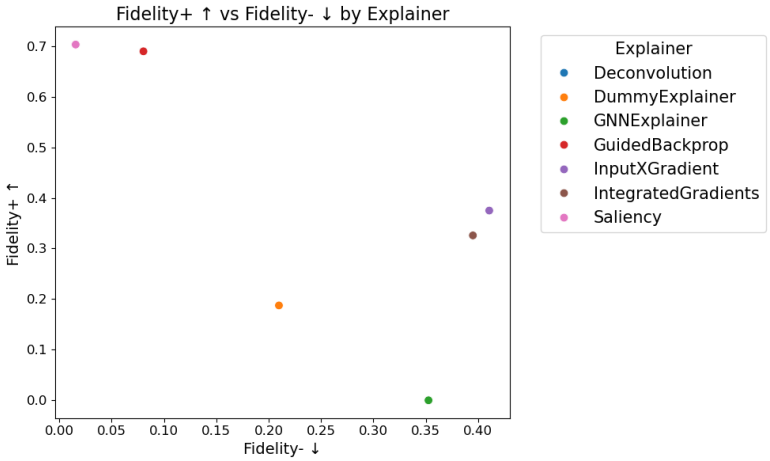

Figure 5.17: Fidelity + vs - per explainer

In figure 5.17 we can see a comparison between Fidelity+ and Fidelity-. We find similar results to those in GraphFramEx [2], where Saliency maps were able to outperform the other explainers in the two metrics. A significant difference compared to GraphFramEx is that most of the explainers are not able to find a necessary explanation, as the average Fidelity+ is not close to 1. The reason for this might be related to the underlying model, as we will highlight that the model has a large influence in the explanation.

We also work on finding the results per GNN model, to see if there is any appreciable difference. The results can be found in figures 5.18 and 5.19. We can see that Saliency maps perform best no matter the underlying model, and that gradient methods tend to outperform the perturbation method, which lines up with the results found by GraphFramEx [2]. One thing to note is that GIN and SAGE models tended to have more faithful explanations, no matter the explainer.

Lastly, we analyze the results based on whether the prediction matched the true label (correct) or whether it did not (incorrect). Figures 5.22 and 5.23 contain the corresponding plots.

While the general clustering of the solutions seems similar, there appears to be a big difference in the explanation quality. It seems that in cases where the model's prediction is incorrect, the explanation suffers. A wrongly predicted label will receive a worse prediction. This is confusing, as the explainers are not using the target data in any way, so why would this have an effect? We cannot find an explanation for this, but we believe that future use of explainers must be cautious of model accuracy.

The issue of explanation fidelity suffering seems to mainly affect the best-performing cluster of Saliency, Deconvolution, and GuidedBackpropagation. In Figures 5.21 and 5.20, you can see the influence there, mainly by the difference in the axis values. We also find that the explainers have worse performance for the minority class, true labels constitute 32.14% of the dataset.

This difference in performance depending on the underlying label does not seem to be the cause of the difference in performance depending on whether the model's prediction was correct. When filtering the dataset for correct predictions, true labels constituted 26.05% of the dataset. A difference, but nothing close to what is seen in the plots. As such, we have no explanation for the difference in faithfulness given a different underlying label or given that the model wrongly predicted the label.

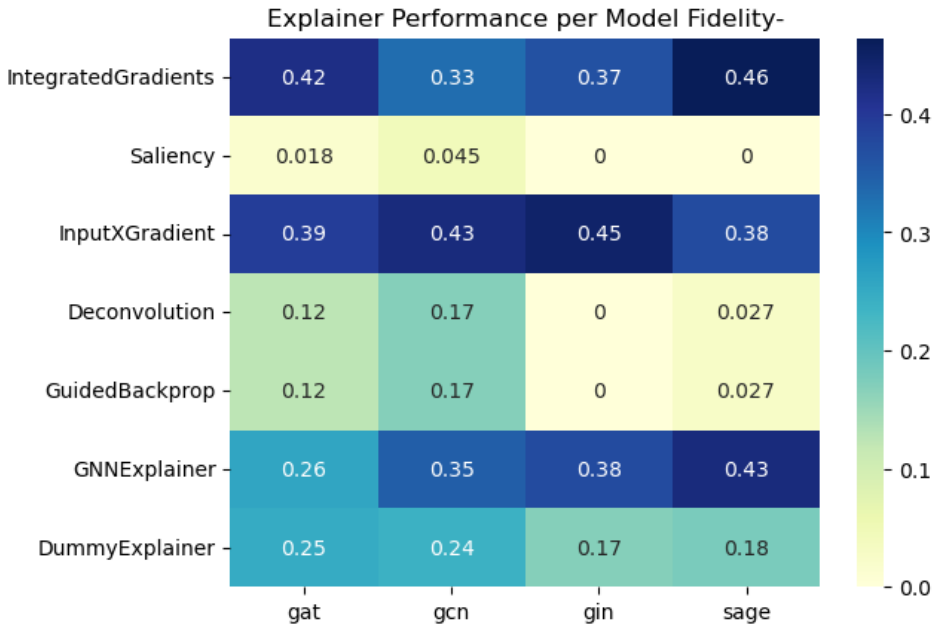

Figure 5.18: Proteins Performance per model Fidelity-

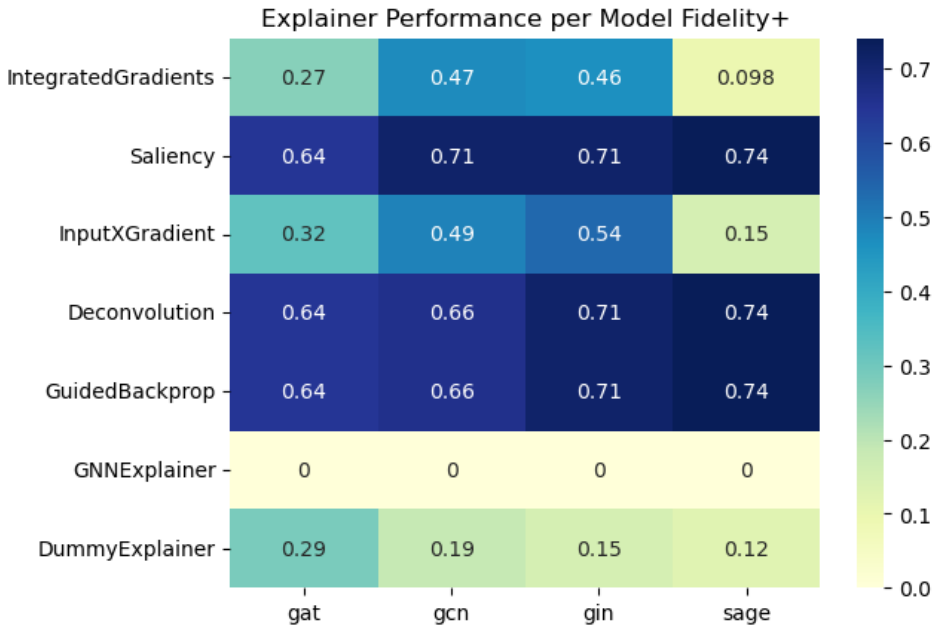

Figure 5.19: Proteins Performance per model Fidelity+

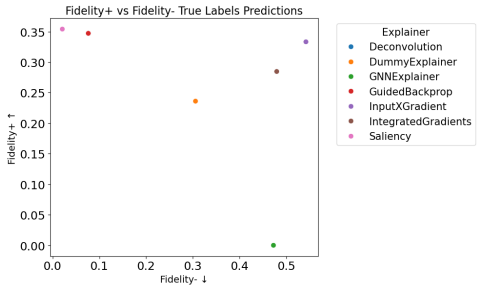

Figure 5.20: Fidelity+ vs Fidelity- for True Original Label

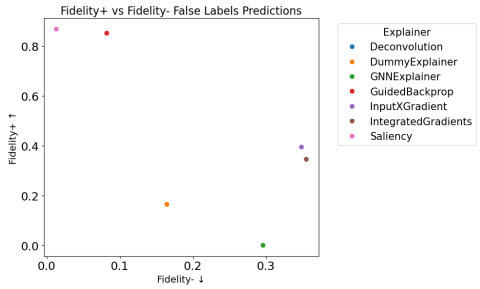

Figure 5.21: Fidelity+ vs Fidelity- for False Original Label

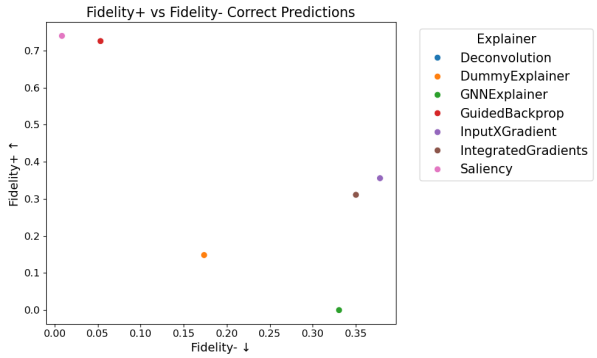

Figure 5.22: Fidelity+ vs Fidelity- for Correct Predictions

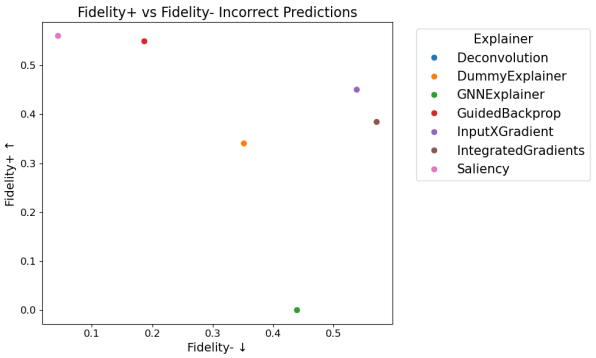

Figure 5.23: Fidelity+ vs Fidelity- for Incorrect Predictions

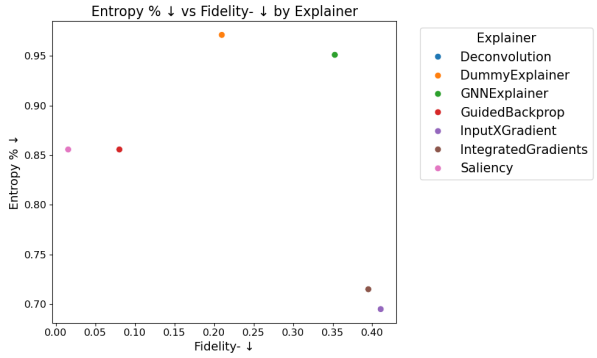

Figure 5.24: Proteins Entropy vs Fidelity-

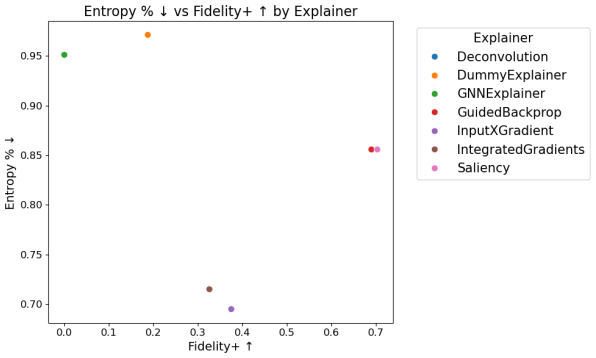

Figure 5.25: Proteins Entropy vs Fidelity+

We do notice that Saliency maps remain the best explainer regardless, which aligns with the results obtained by GraphFrameEx [2].

Entropy

For the Entropy scores, we can see that InputXGradient was the best-performing method. We observe that the value is, on average, higher than in the XGDAG experiments. The higher Entropy score is probably because this is a graph classification task, so all of the information is relevant to the final prediction.

To see more detailed information on how the Entropy affects the fidelity scores, we can find the figures 5.24 and 5.25. Based on the figures, it seems like three clusters formed. GuidedBackpropagation, Deconvolution, and Saliency maps are accurate but high-entropy responses. InputXGradients and IntegratedGradients are low-entropy but inaccurate responses. GNNExplainer and DummyExplainer are bad at both, which implies that GNNExplainer is not only bad but can also be actively worse than random explanations.

Kernel Distance

Kernel distance was almost 0 across the board. Even though it cannot be seen due to rounding, the resulting kernel distance of the masks was higher than the kernel distance within the original dataset. The reason the kernel distance of the masks was slightly higher was that the mask corresponding to their respective graph had a non-zero result.

We believe that this is because we normalized the score when using the kernel method. The differences in the graphs seem to be large by default, and the normalization makes all of the elements of the original dataset appear different. This tells us that the current implementation of Kernel distance needs to be modified to at least guarantee that the distance within the original dataset is not as significant.

Repeatability and VoE

Repeatability and variance over the Explanation (VoE) give similar results compared to our XGDAG experiments, where all the gradient-based methods provide consistent explanations.

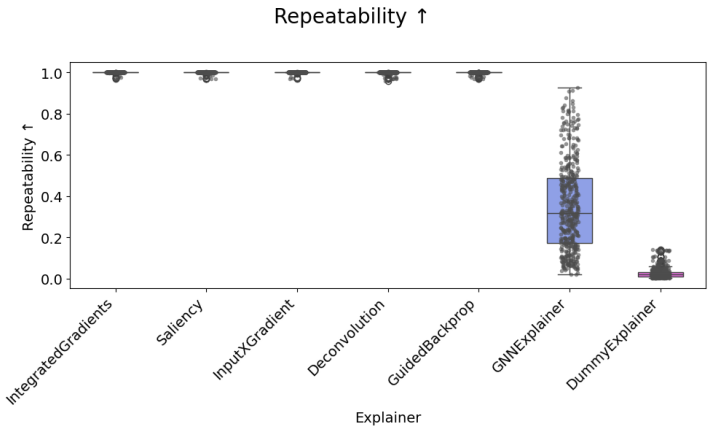

Figure 5.26: Repeatability per explainer

Even though the repeatability is 1 for most explainers in table 5.4, we can see in Figure 5.26 that the true value is not always 1. These results are not the case for VoE, where the values are truly 0 always.

We believe that this difference in the score of repeatability happens when some of the features are tied in importance. When finding the top k elements per graph, ties have an element of randomness to them; as such, the final list might have slightly differing elements. This result does imply that repeatability is a bit more sensitive than the VoE, which might be undesired.

Counterfactual

For the counterfactual values, we see that all the explainers tend to have a high counterfactual score. DummyExplainer does have worse performance, but the difference is not significant.

However, looking at the values in detail in Figure 5.27, there is no significant difference between the explainers, other than a difference in the variance. For us, this gives credence to the idea that the edges provide the most information for the models being tested. As such, the experiments were not able to provide additional insights.

5.2.2. Conclusion of Protein Dataset results

These results have highlighted one main factor: you need to fine-tune explainers. For multiple metrics, we found that not working with edges impacted the results. It seems likely that the explanations are not as good as the scores say, but are lacking in other respects. This issue with the explanations leads to the question of how to identify what to optimize for. We were not able to create good guidelines on what to optimize for, but we believe that by identifying this factor with our current tests, we have helped highlight the importance of fine-tuning the explainer to fit your needs.

Focusing on the results obtained, it seems that Saliency maps are a great all-around explainer as they are able to find a balance between concise explanations and faithful explanations. The good performance of Saliency maps aligns with the results found in GraphFramEx [2].

Comparing our results with GNNX-Bench in the same dataset [40], we find that GNNExplainer performs worse for Fidelity-, compared to the other models, where the original authors found it to be one of the best performers for the Fidelity- metric. Our set of explainers was different, with our explainers having a bigger focus on gradient-based methods.

While we did not do extensive testing with other perturbation-based models like GNNX-Bench did, our findings could indicate that gradient methods should be used when compared to perturbation methods. Taking into account results other than the Fidelity, this fact becomes clearer. The quality of gradient methods is in line with the results found by GraphFramEx [2] and BAGEL [57], where Saliency maps (called GradInput in BAGEL) would often be among the best-performing models.

Lastly, we believe that the results found per model show that specific GNN architectures are easier to explain, regardless of the underlying explainer. In cases where biological explanations might be needed,

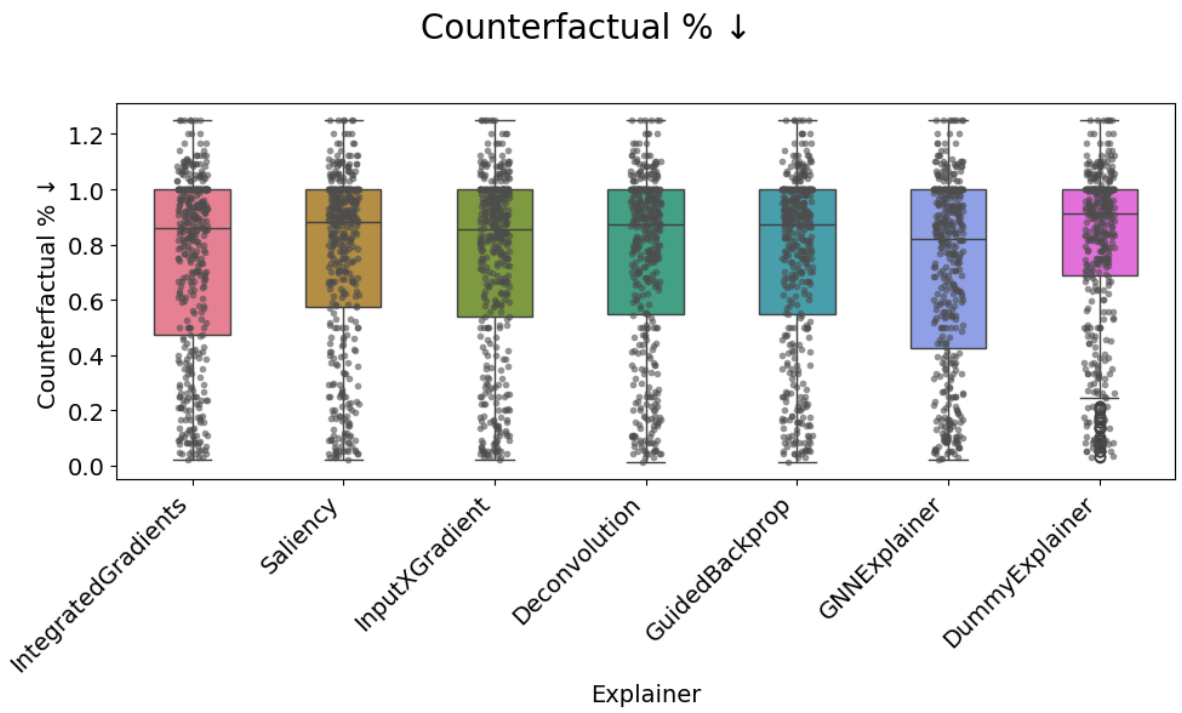

Figure 5.27: Counterfactual per explainer

GraphSAGE or GIN should be prioritized as the model architecture to guarantee that the explanation of the results is more faithful.

6

Conclusion

This thesis investigated various explainability methods for GNNs in biomedical contexts. Through experiments for graph and node classification, we have compared multiple explainers across different metrics to understand their effectiveness and limitations.

6.1. Summary of Key Findings

How can the previously explained needs of biomedicine be addressed when deciding what explainer to use?

We set an initial set of metrics and an explanation for each of the metrics in Chapter 3, based on the needs that were identified in Chapter 1. We believe our work provides key steps in making XAI with GNNs more trustworthy, which was the biggest desire for its use in biomedicine. Our metrics highlight many of the needs in biomedicine, and we provide tests on how explainers measure up to these needs. While addressing all needs is impossible, we have found that IntegratedGradients and Saliency tend to be best at addressing the needs of biomedicine.

How do dataset properties affect the explanations in biomedicine?

We found that biological datasets tend to be quite large in size. This implies that explainers that require additional training might not be viable, as the time it takes to train them can make them too unwieldy. Based on this, we believe that gradient-based methods were able to avoid these issues and were the easiest to adapt.

Class imbalance has a significant effect on every explainer, and given that class imbalance problems are most common in biology [24], it is crucial to take this into account when using explainers. In our findings, a model that is overfitted on one of the classes will not have valid explanations when dealing with other classes. The impact of this finding will depend on the user of the explainer, as a model that is overfitting on a specific class usually wants to predict that class above all.

We also found that deciding what to explain was not trivial. There were huge differences between the explainers that focused on edges, nodes and features, or only features. Deciding what aspect of the graph to explain will have a significant effect on explanation quality, and should be informed based on domain knowledge. Based on our findings, we believe that finding a good explainer for edge properties is crucial, as topological features seemed to have the most significant effect on prediction quality.

How differently do gradient-based, perturbation-based, and surrogate explainers perform between node and graph classification problems in biomedical applications?

We found issues in the trade-off between having accurate model explanations and making sure they were sparse, which provides a limitation for human users. In many cases, either an explanation is correct but hard to understand, or easy to understand but incorrect. IntegratedGradients and Saliency maps were the explainers that could best handle this trade-off, for node classification and graph clas-

sification, respectively. In case either accuracy or sparsity is more important, the specific explainer will change; detailed information about which to use can be found in Chapter 5.

We also find that gradient-based methods are consistent and fast, which are two traits desired for biomedical use. Based on the previously outlined factors, we believe that gradient-based methods are preferred over perturbation and surrogate explainers. We also found that the underlying GNN model impacts explanation fidelity. GIN and GraphSAGE models should be prioritized in case an explanation is needed, regardless of what type of explainer is used.

We find that the underlying models can be resilient to changes, putting the value of other explanations into doubt. Due to this, we believe that using random explanations as a sanity check is crucial, as this allows us to understand whether the explainer is actually explaining or whether the model is resilient to data changes. This approach is also faster to use compared to ROAR [30], and with some guidelines, it can be crucial in understanding when the explainer is providing insights into the model, or if the model is resilient to explanations.

We also provide insights for each of the individual metrics that we highlighted:

- RDT-Fidelity: This metric was able to get more nuance in the node classification task versus the graph classification task. Based on this, we believe that RDT-Fidelity is more useful as a measure of Fidelity in cases where node-level explanations are needed.
- Fidelity+ and Fidelity-: The metrics do not always correlate, so there are cases where an explanation is good at finding how to change the prediction, but is not able to keep the prediction. Based on this, domain experts must decide to focus on either sufficient or necessary explanations.
- Kernel distance: The kernel distance was already considerable when comparing the graphs of the original datasets with themselves, so an approach that can group the original graphs closer together is needed. We were not able to properly understand how good explainers are at guaranteeing that results are still biologically plausible.
- Entropy: Graph classification explanations are a lot less sparse, and might require additional restrictions to be understandable. This applies to all explainers, regardless of the type of approach.
- Counterfactual: This metric correlated with Fidelity+, and it needs to be assessed how much new information it is able to provide.
- Repeatability, VoE, and Time: Gradient-based methods significantly outperform the other explainer types in these metrics

What strategies should be used when applying general-use GNN explainers in the biomedicine domain?

The performance of the explainers varies between the two biomedical datasets. For the disease-gene association task, IntegratedGradients found a balance between fidelity and entropy, while Saliency maps performed better on the Proteins dataset. Based on this, we believe that IntegratedGradients should be used for node classification problems, while Saliency should be used for graph classification problems. Gradient explainers are preferred for both problem types.

The GNN architecture has an effect on explainer performance, and the GIN and SAGE models are generally easier to explain.

In both datasets, we found evidence suggesting that the underlying graph structure, particularly edge connections, plays a crucial role in model predictions. This was especially apparent in the Proteins dataset, where even DummyExplainer achieved reasonable RDT-Fidelity scores, hinting that topological properties may carry more predictive information than individual node features. Determining when to focus on nodes and when to focus on edges remains critical for future work.

We observed that class imbalance has a significant effect on the explainer's faithfulness in our experiments. This effect in the explainers highlights how explainer use has to take into account how the model was built. It also highlights that additional steps need to be taken in cases where there is a significant class imbalance, which is common in biomedicine [24], with rare diseases as an example.

We find that most explainers need to be tuned to fit the specific needs of the problem, either by focusing on specific aspects of the explanation or by setting the hyperparameters based on prior knowledge. We find that the importance of the topology is critical, and is not always easy to extract from given explanations.

6.2. Limitations and Future Work

Several limitations in our study point to important directions for future research.

Post-Hoc Focus: For our tests, we focus on using post-hoc explainers to remain flexible to different approaches, which does partly limit our results. Future tests could include intrinsic explanations.

Kernel Method Refinement: The Kernel Distance metric provided little to no new information. Given the difficulty of finding the distance between two graphs [41], this is not surprising. Finding an alternative for this should be explored in the future. Some ideas include not normalizing the kernels or using datasets with consistent node and feature dimensions.

Human Evaluation: Direct evaluation by domain experts would give the most critical information, as they are the ones who will be using these systems. Systematic tests involving biologists and clinicians would be essential to define the most vital factors to make explanations useful to them.

Explainer Fine-Tuning: Developing guidelines for the use of explainers in specific biomedical tasks would greatly help users. This could include automated approaches to identify what needs to be explained (nodes, edges, or features) based on preliminary data analysis, or information on how to tune the hyperparameters depending on the problem's restrictions.

In conclusion, this thesis provides an introduction to what explainable GNNs are, their use in biomedicine, important metrics for their continued use in the domain, and does an exploratory study of how explainers are affected by usage in the biomedical domain. It highlights pitfalls that affect the biomedicine field in particular and finds ways in which explainers need to be adapted when being used in the field. We find that no single explanation is perfect and that selecting which explainer to use will depend significantly on the data, problem type, and model being explained; we are able to highlight some use cases depending on explanation needs.

References

[1] Subhan Ali et al. “The enlightening role of explainable artificial intelligence in medical & healthcare domains: A systematic literature review”. In: *Computers in Biology and Medicine* 166 (2023), p. 107555. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2023.107555. URL: https://www.sciencedirect.com/science/article/pii/S001048252301020X.

[2] Kenza Amara et al. *GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks*. 2024. arXiv: 2206.09677 [cs.LG]. URL: https://arxiv.org/abs/2206.09677.

[3] Elvio Amparore, Alan Perotti, and Paolo Bajardi. “To trust or not to trust an explanation: using LEAF to evaluate local linear XAI methods”. In: *PeerJ Computer Science* 7 (2021), e479.

[4] Rachel KE Bellamy et al. “AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias”. In: *IBM Journal of Research and Development* 63.4/5 (2019), pp. 4–1.

[5] Karsten M Borgwardt et al. “Protein function prediction via graph kernels”. In: *Bioinformatics* 21.suppl_1 (2005), pp. i47–i56.

[6] Alex Broadbent and Thomas Grote. “Can robots do epidemiology? Machine learning, causal inference, and predicting the outcomes of public health interventions”. In: *Philosophy & Technology* 35.1 (2022), p. 14.

[7] Aishwarya Budhkar et al. “xSiGra: explainable model for single-cell spatial data elucidation”. In: *Briefings in Bioinformatics* 25.5 (Aug. 2024), bbae388. ISSN: 1477-4054. DOI: 10.1093/bib/bbae388. eprint: https://academic.oup.com/bib/article-pdf/25/5/bbae388/58784630/bbae388.pdf. URL: https://doi.org/10.1093/bib/bbae388.

[8] Gianluca Carloni, Andrea Berti, and Sara Colantonio. “The role of causality in explainable artificial intelligence”. In: *arXiv preprint arXiv:2309.09901* (2023).

[9] Hugh Chen et al. “True to the model or true to the data?” In: *arXiv preprint arXiv:2006.16234* (2020).

[10] Yu-Liang Chou et al. “Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications”. In: *Information Fusion* 81 (2022), pp. 59–83.

[11] Papers with code. *Datasets: Biology*. 2025. URL: https://paperswithcode.com/datasets?mod=biology (visited on 04/18/2025).

[12] European Commission et al. *Robustness and explainability of Artificial Intelligence – From technical to policy solutions*. Publications Office, 2020. DOI: doi/10.2760/57493.

[13] Flavia Costi et al. “Predictive Modeling for Diabetes Using GraphLIME”. In: *medRxiv* (2024), pp. 2024–03.

[14] Enyan Dai et al. “A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability”. In: *Machine Intelligence Research* (2024), pp. 1–51.

[15] Alex J DeGrave, Joseph D Janizek, and Su-In Lee. “AI for radiographic COVID-19 detection selects shortcuts over signal”. In: *Nature Machine Intelligence* 3.7 (2021), pp. 610–619.

[16] Paul D Dobson and Andrew J Doig. “Distinguishing enzyme structures from non-enzymes without alignments”. In: *Journal of molecular biology* 330.4 (2003), pp. 771–783.

[17] Juan Manuel Durán and Karin Rolanda Jongsma. “Who is afraid of black box algorithms? On the epistemological and ethical basis of trust in medical AI”. In: *Journal of Medical Ethics* 47.5 (2021), pp. 329–335.

[18] Alexandre Duval and Fragkiskos D Malliaros. “Graphsvx: Shapley value explanations for graph neural networks”. In: *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer. 2021, pp. 302–318.

[19] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *nature* 542.7639 (2017), pp. 115–118.

[20] Matthias Fey and Jan E. Lenssen. “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[21] Kunihiko Fukushima. “Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements”. In: *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pp. 322–333. DOI: 10.1109/TSSC.1969.300225.

[22] Thorben Funke, Megha Khosla, and Avishek Anand. “Zorro: Valid, Sparse, and Stable Explanations in Graph Neural Networks”. In: *CoRR* abs/2105.08621 (2021). arXiv: 2105.08621. URL: https://arxiv.org/abs/2105.08621.

[23] Susan Dina Ghiassian, Jörg Menche, and Albert-László Barabási. “A DIseAse MOdule Detection (DIAMOnD) algorithm derived from a systematic analysis of connectivity patterns of disease proteins in the human interactome”. In: *PLoS computational biology* 11.4 (2015), e1004120.

[24] Guo Haixiang et al. “Learning from class-imbalanced data: Review of methods and applications”. In: *Expert systems with applications* 73 (2017), pp. 220–239.

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[26] Ronan Hamon et al. “Bridging the gap between AI and explainability in the GDPR: towards trustworthiness-by-design in automated decision-making”. In: *IEEE Computational Intelligence Magazine* 17.1 (2022), pp. 72–85.

[27] Henry Han and Xiangrong Liu. “The challenges of explainable AI in biomedical data science”. In: *BMC bioinformatics* 22.Suppl 12 (2022), p. 443.

[28] Andreas Holzinger, André Carrington, and Heimo Müller. “Measuring the quality of explanations: the system causability scale (SCS) comparing human and machine explanations”. In: *KI-Künstliche Intelligenz* 34.2 (2020), pp. 193–198.

[29] Andreas Holzinger et al. “Causability and explainability of artificial intelligence in medicine”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.4 (2019), e1312.

[30] Sara Hooker et al. “A benchmark for interpretability methods in deep neural networks”. In: *Advances in neural information processing systems* 32 (2019).

[31] Qiang Huang et al. “Graphlime: Local interpretable model explanations for graph neural networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.7 (2022), pp. 6968–6972.

[32] Yu Huang et al. “A scoping review of fair machine learning techniques when using real-world data”. In: *Journal of Biomedical Informatics* 151 (2024), p. 104622. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2024.104622. URL: https://www.sciencedirect.com/science/article/pii/S1532046424000406.

[33] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[34] Jeya Vikranth Jeyakumar et al. “How can i explain this to you? an empirical study of deep neural network explanation methods”. In: *Advances in neural information processing systems* 33 (2020), pp. 4211–4222.

[35] Jaykumar Kakkad et al. *A Survey on Explainability of Graph Neural Networks*. 2023. arXiv: 2306.01958 [cs.LG].

[36] Md. Rezaul Karim et al. *Explainable AI for Bioinformatics: Methods, Tools, and Applications*. 2023. arXiv: 2212.13261 [q-bio.QM].

[37] Steven Kearnes et al. “Molecular graph convolutions: moving beyond fingerprints”. In: *Journal of computer-aided molecular design* 30 (2016), pp. 595–608.

[38] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG]. URL: https://arxiv.org/abs/1609.02907.

[39] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. "Inherent Trade-Offs in the Fair Determination of Risk Scores". In: *CoRR* abs/1609.05807 (2016). arXiv: 1609.05807. URL: http://arxiv.org/abs/1609.05807.

[40] Mert Kosan et al. "Gnnx-bench: Unravelling the utility of perturbation-based gnn explainers through in-depth benchmarking". In: *arXiv preprint arXiv:2310.01794* (2023).

[41] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. "A survey on graph kernels". In: *Applied Network Science* 5 (2020), pp. 1–42.

[42] Bo Li et al. "Trustworthy AI: From Principles to Practices". In: *ACM Comput. Surv.* 55.9 (Jan. 2023). ISSN: 0360-0300. DOI: 10.1145/3555803. URL: https://doi.org/10.1145/3555803.

[43] Han Li et al. "Improving molecular property prediction through a task similarity enhanced transfer learning strategy". In: *Iscience* 25.10 (2022).

[44] Yang Liu et al. "Synthetic benchmarks for scientific research in explainable machine learning". In: *arXiv preprint arXiv:2106.12543* (2021).

[45] Jörn Lötsch, Dario Kringel, and Alfred Ultsch. "Explainable Artificial Intelligence (XAI) in Biomedicine: Making AI Decisions Trustworthy for Physicians and Patients". In: *BioMedInformatics* 2.1 (2022), pp. 1–17. ISSN: 2673-7426. DOI: 10.3390/biomedinformatics2010001. URL: https://www.mdpi.com/2673-7426/2/1/1.

[46] Dongsheng Luo et al. "Parameterized Explainer for Graph Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19620–19631. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/e37b08dd3015330dcbb5d6663667b8b8-Paper.pdf.

[47] Andrea Mastropietro, Gianluca De Carlo, and Aris Anagnostopoulos. "XGDAG: explainable gene–disease associations via graph neural networks". In: *Bioinformatics* 39.8 (2023), btad482.

[48] Ninareh Mehrabi et al. "A survey on bias and fairness in machine learning". In: *ACM computing surveys (CSUR)* 54.6 (2021), pp. 1–35.

[49] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[50] Christopher Morris et al. *TUDataset: A collection of benchmark datasets for learning with graphs*. 2020. arXiv: 2007.08663 [cs.LG]. URL: https://arxiv.org/abs/2007.08663.

[51] Marion Neumann et al. "Propagation kernels: efficient graph kernels from propagated information". In: *Machine learning* 102 (2016), pp. 209–245.

[52] Janet Piñero et al. "The DisGeNET knowledge platform for disease genomics: 2019 update". In: *Nucleic acids research* 48.D1 (2020), pp. D845–D855.

[53] Nicoletta Prentzas, Antonis Kakas, and Constantinos S. Pattichis. *Explainable AI applications in the Medical Domain: a systematic review*. 2023. arXiv: 2308.05411 [cs.AI]. URL: https://arxiv.org/abs/2308.05411.

[54] Daniel L. Purich. "Enzyme catalysis: a new definition accounting for noncovalent substrate- and product-like states". In: *Trends in Biochemical Sciences* 26.7 (2001), pp. 417–421. ISSN: 0968-0004. DOI: https://doi.org/10.1016/S0968-0004(01)01880-1. URL: https://www.sciencedirect.com/science/article/pii/S0968000401018801.

[55] Luyu Qiu et al. "Generating perturbation-based explanations with robustness to out-of-distribution data". In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 3594–3605.

[56] Yanou Ramon et al. "A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C". In: *Advances in Data Analysis and Classification* 14 (2020), pp. 801–819.

[57] Mandeep Rathee et al. *BAGEL: A Benchmark for Assessing Graph Neural Network Explanations*. 2022. arXiv: 2206.13983 [cs.LG].

[58] Franco Scarselli et al. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[59] Avanti Shrikumar et al. “Not Just a Black Box: Learning Important Features Through Propagating Activation Differences”. In: *CoRR* abs/1605.01713 (2016). arXiv: 1605.01713. URL: http://arxiv.org/abs/1605.01713.

[60] Giannis Siglidis et al. “GraKeL: A Graph Kernel Library in Python”. In: *Journal of Machine Learning Research* 21.54 (2020), pp. 1–5.

[61] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034 [cs.CV]. URL: https://arxiv.org/abs/1312.6034.

[62] Jost Tobias Springenberg et al. “Striving for simplicity: The all convolutional net”. In: *arXiv preprint arXiv:1412.6806* (2014).

[63] Chris Stark et al. “BioGRID: a general repository for interaction datasets”. In: *Nucleic acids research* 34.suppl_1 (2006), pp. D535–D539.

[64] Paola Stolfi et al. “NIAPU: network-informed adaptive positive-unlabeled learning for disease gene identification”. In: *Bioinformatics* 39.2 (2023), btac848.

[65] Peng Gang Sun, Lin Gao, and Shan Han. “Prediction of human disease-related gene clusters by clustering analysis”. In: *International journal of biological sciences* 7.1 (2011), p. 61.

[66] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328.

[67] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

[68] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. In: *Harv. JL & Tech.* 31 (2017), p. 841.

[69] Qianwen Wang et al. “Extending the nested model for user-centric XAI: A design study on GNN-based drug repurposing”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2022), pp. 1266–1276.

[70] Ziquan Wei et al. *NeuroPath: A Neural Pathway Transformer for Joining the Dots of Human Connectomes*. 2024. arXiv: 2409.17510 [q-bio.NC]. URL: https://arxiv.org/abs/2409.17510.

[71] Barry Payne Welford. “Note on a method for calculating corrected sums of squares and products”. In: *Technometrics* 4.3 (1962), pp. 419–420.

[72] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG]. URL: https://arxiv.org/abs/1810.00826.

[73] Ziduo Yang et al. “MGraphDTA: deep multiscale graph neural network for explainable drug–target binding affinity prediction”. In: *Chemical science* 13.3 (2022), pp. 816–833.

[74] Zhitao Ying et al. “Gnnexplainer: Generating explanations for graph neural networks”. In: *Advances in neural information processing systems* 32 (2019).

[75] Hao Yuan et al. “Explainability in Graph Neural Networks: A Taxonomic Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2023), pp. 5782–5799. DOI: 10.1109/TPAMI.2022.3204236.

[76] Hao Yuan et al. “On explainability of graph neural networks via subgraph explorations”. In: *International conference on machine learning*. PMLR. 2021, pp. 12241–12252.

[77] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.

[78] Jiaxing Zhang, Dongsheng Luo, and Hua Wei. “Mixupexplainer: Generalizing explanations for graph neural networks with data augmentation”. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 3286–3296.

[79] Zhongheng Zhang et al. “Opening the black box of neural networks: methods for interpreting neural network models in clinical applications”. In: *Annals of translational medicine* 6.11 (2018).

[80] Xu Zheng et al. “Towards robust fidelity for evaluating explainability of graph neural networks”. In: *arXiv preprint arXiv:2310.01820* (2023).

[81] Zhongliang Zhou et al. “Xai meets biology: A comprehensive review of explainable ai in bioinformatics applications”. In: *arXiv preprint arXiv:2312.06082* (2023).

A

Extra figures

Explainer	Sparsity ↓	Counterfactual ↓
Deconvolution	5.54921 ± 1.25711	0.4717 ± 1.82768
DummyExplainer	11.09628 ± 0.00098	1158.86321 ± 2496.39053
GNNExplainer	9.26114 ± 0.63465	7.53302 ± 48.02521
GraphLIME	0.45557 ± 0.2553	0.67453 ± 2.02185
GuidedBackprop	5.54921 ± 1.25711	0.4717 ± 1.82768
InputXGradient	4.19479 ± 1.30109	0.33491 ± 1.53509
IntegratedGradients	4.19575 ± 1.15522	0.41981 ± 1.6886
PGExplainer	1.82045 ± 0.88433	52269.25 ± 95429.22773
Saliency	5.54921 ± 1.25711	0.16038 ± 1.58869

Table A.1: Original Sparsity and Counterfactual XGDAG

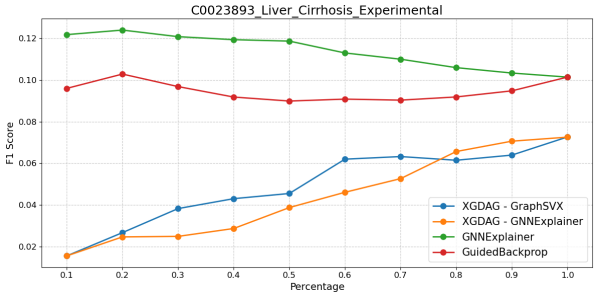

Figure A.1: Liver Cirrhosis XGDAG vs new methods

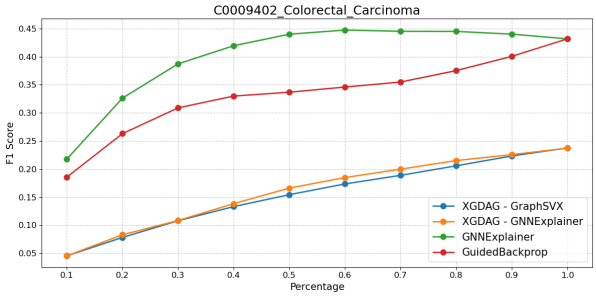

Figure A.2: Colorectal Carcinoma XGDAG vs new methods

Model/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
IntegratedGradients	0.117966	0.202863	0.282659	0.356696	0.418394	0.476966	0.539158	0.617802	0.706811	0.823462
Saliency	0.123560	0.221948	0.320171	0.411320	0.495722	0.569102	0.636723	0.702369	0.763574	0.823462
InputXGradient	0.109246	0.198091	0.282165	0.356203	0.419710	0.482396	0.542613	0.618625	0.708950	0.823462
Deconvolution	0.106449	0.203685	0.289240	0.360480	0.428266	0.487002	0.550839	0.624054	0.713557	0.823462
GuidedBackprop	0.106449	0.203685	0.289240	0.360480	0.428266	0.487002	0.550839	0.624054	0.713557	0.823462
GNNExplainer	0.133432	0.234288	0.330536	0.414610	0.493912	0.572721	0.639191	0.704508	0.765712	0.823462
DummyExplainer	0.085884	0.166009	0.251069	0.332511	0.410826	0.493748	0.576834	0.660744	0.740375	0.823462
PGExplainer	0.121751	0.220796	0.320665	0.414446	0.500823	0.581606	0.649391	0.725568	0.781836	0.823462

Table A.2: Recall per Explainer XGDAG C0006142. Pct columns represent percentages from 0.1 to 1.0.

Model/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
IntegratedGradients	0.538288	0.462664	0.429715	0.406754	0.381660	0.362556	0.351308	0.352218	0.358179	0.375553
Saliency	0.563814	0.506191	0.486743	0.469043	0.452199	0.432591	0.414880	0.400431	0.386943	0.375553
InputXGradient	0.498498	0.451782	0.428964	0.406191	0.382861	0.366683	0.353559	0.352687	0.359263	0.375553
Deconvolution	0.485736	0.464540	0.439720	0.411069	0.390665	0.370185	0.358919	0.355783	0.361597	0.375553
GuidedBackprop	0.485736	0.464540	0.439720	0.411069	0.390665	0.370185	0.358919	0.355783	0.361597	0.375553
GNNExplainer	0.608859	0.534334	0.502501	0.472795	0.450548	0.435343	0.416488	0.401651	0.388027	0.375553
DummyExplainer	0.391892	0.378612	0.381691	0.379174	0.374756	0.375313	0.375858	0.376700	0.375188	0.375553
PGExplainer	0.555556	0.503565	0.487494	0.472608	0.456851	0.442096	0.423135	0.413657	0.396198	0.375553

Table A.3: Precision per Explainer XGDAG C0006142

Model/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
IntegratedGradients	0.193522	0.282054	0.341008	0.380084	0.399184	0.411965	0.425419	0.448653	0.475432	0.515846
Saliency	0.202699	0.308590	0.386264	0.438289	0.472961	0.491545	0.502402	0.510066	0.513612	0.515846
InputXGradient	0.179217	0.275420	0.340413	0.379558	0.400440	0.416655	0.428145	0.449250	0.476870	0.515846
Deconvolution	0.174629	0.283198	0.348948	0.384116	0.408602	0.420634	0.434636	0.453193	0.479969	0.515846
GuidedBackprop	0.174629	0.283198	0.348948	0.384116	0.408602	0.420634	0.434636	0.453193	0.479969	0.515846
GNNExplainer	0.218893	0.325746	0.398769	0.441795	0.471235	0.494671	0.504349	0.511620	0.515051	0.515846
DummyExplainer	0.140891	0.230813	0.302898	0.354313	0.391963	0.426460	0.455147	0.479838	0.498008	0.515846
PGExplainer	0.199730	0.306988	0.386860	0.441620	0.477827	0.502345	0.512398	0.526913	0.525896	0.515846

Table A.4: F1 per Explainer XGDAG C0006142

Model/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
IntegratedGradients	0.403331	0.403331	0.403331	0.403331	0.403331	0.403331	0.403331	0.403331	0.403331	0.403331
Saliency	0.448407	0.448407	0.448407	0.448407	0.448407	0.448407	0.448407	0.448407	0.448407	0.448407
InputXGradient	0.396458	0.396458	0.396458	0.396458	0.396458	0.396458	0.396458	0.396458	0.396458	0.396458
Deconvolution	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980
GuidedBackprop	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980	0.398980
GNNExplainer	0.461794	0.461794	0.461794	0.461794	0.461794	0.461794	0.461794	0.461794	0.461794	0.461794
DummyExplainer	0.371633	0.371633	0.371633	0.371633	0.371633	0.371633	0.371633	0.371633	0.371633	0.371633
PGExplainer	0.451298	0.451298	0.451298	0.451298	0.451298	0.451298	0.451298	0.451298	0.451298	0.451298

Table A.5: AUC per Explainer XGDAG C0006142

Explainer	Time seconds ↓	RDT Fidelity ↑	Fidelity- ↓	Fidelity+ ↑	Mask Kernel Distance ↑
Deconvolution	0.01 ± 0.0	0.83 ± 0.3	0.12 ± 0.33	0.64 ± 0.48	0.0006 ± 0.0
DummyExplainer	0.0 ± 0.0	0.79 ± 0.34	0.25 ± 0.43	0.29 ± 0.45	0.0005 ± 0.0
GNNExplainer	1.11 ± 0.02	0.8 ± 0.33	0.26 ± 0.44	0.0 ± 0.0	0.0006 ± 0.0
GuidedBackprop	0.01 ± 0.0	0.83 ± 0.3	0.12 ± 0.33	0.64 ± 0.48	0.0006 ± 0.0
InputXGradient	0.01 ± 0.0	0.82 ± 0.31	0.39 ± 0.49	0.32 ± 0.47	0.0006 ± 0.0
IntegratedGradients	0.44 ± 0.01	0.81 ± 0.33	0.42 ± 0.5	0.27 ± 0.44	0.0006 ± 0.0
Saliency	0.01 ± 0.0	0.74 ± 0.39	0.02 ± 0.13	0.64 ± 0.48	0.0006 ± 0.0

Table A.6: Proteins GAT Fidelity metrics

Explainer	Entropy % ↓	Repeatability ↑	Total Variance ↓	Counterfactual % ↓
Deconvolution	0.78 ± 0.05	1.0 ± 0.0	-0.0 ± 0.0	0.72 ± 0.33
DummyExplainer	0.97 ± 0.0	0.02 ± 0.03	120.45 ± 119.9	0.87 ± 0.25
GNNExplainer	0.95 ± 0.03	0.25 ± 0.18	15.66 ± 26.74	0.72 ± 0.35
GuidedBackprop	0.78 ± 0.05	1.0 ± 0.0	0.0 ± 0.0	0.72 ± 0.33
InputXGradient	0.61 ± 0.07	1.0 ± 0.0	0.0 ± 0.0	0.73 ± 0.34
IntegratedGradients	0.66 ± 0.05	1.0 ± 0.0	-0.0 ± 0.0	0.77 ± 0.34
Saliency	0.78 ± 0.05	1.0 ± 0.0	0.0 ± 0.0	0.73 ± 0.34

Table A.7: Proteins GAT Other metrics

Explainer	Time seconds ↓	RDT Fidelity ↑	Fidelity- ↓	Fidelity+ ↑	Mask Kernel Distance ↑
Deconvolution	0.01 ± 0.0	0.85 ± 0.28	0.17 ± 0.38	0.66 ± 0.48	0.0007 ± 0.0001
DummyExplainer	0.0 ± 0.0	0.85 ± 0.3	0.24 ± 0.43	0.19 ± 0.39	0.0005 ± 0.0
GNNExplainer	0.8 ± 0.01	0.85 ± 0.29	0.35 ± 0.48	0.0 ± 0.0	0.0007 ± 0.0001
GuidedBackprop	0.01 ± 0.0	0.85 ± 0.28	0.17 ± 0.38	0.66 ± 0.48	0.0007 ± 0.0001
InputXGradient	0.01 ± 0.0	0.82 ± 0.35	0.43 ± 0.5	0.49 ± 0.5	0.0007 ± 0.0001
IntegratedGradients	0.31 ± 0.02	0.82 ± 0.36	0.33 ± 0.47	0.47 ± 0.5	0.0007 ± 0.0001
Saliency	0.01 ± 0.0	0.81 ± 0.34	0.04 ± 0.21	0.71 ± 0.45	0.0007 ± 0.0001

Table A.8: Proteins GCN Fidelity metrics

Explainer	Entropy % ↓	Repeatability ↑	Total Variance ↓	Counterfactual % ↓
Deconvolution	0.87 ± 0.04	1.0 ± 0.01	-0.0 ± 0.0	0.75 ± 0.29
DummyExplainer	0.97 ± 0.0	0.02 ± 0.03	120.42 ± 119.68	0.79 ± 0.27
GNNExplainer	0.96 ± 0.03	0.34 ± 0.2	8.91 ± 16.13	0.68 ± 0.31
GuidedBackprop	0.87 ± 0.04	1.0 ± 0.01	-0.0 ± 0.0	0.75 ± 0.29
InputXGradient	0.73 ± 0.06	1.0 ± 0.0	-0.0 ± 0.0	0.75 ± 0.31
IntegratedGradients	0.73 ± 0.06	1.0 ± 0.0	-0.0 ± 0.0	0.69 ± 0.3
Saliency	0.87 ± 0.04	1.0 ± 0.01	-0.0 ± 0.0	0.75 ± 0.3

Table A.9: Proteins GCN Other metrics

Explainer	Time seconds ↓	RDT Fidelity ↑	Fidelity- ↓	Fidelity+ ↑	Mask Kernel Distance ↑
Deconvolution	0.01 ± 0.0	0.85 ± 0.23	0.0 ± 0.0	0.71 ± 0.45	0.0007 ± 0.0001
DummyExplainer	0.0 ± 0.0	0.88 ± 0.2	0.17 ± 0.38	0.15 ± 0.36	0.0005 ± 0.0
GNNExplainer	0.81 ± 0.02	0.9 ± 0.19	0.38 ± 0.49	0.0 ± 0.0	0.0006 ± 0.0001
GuidedBackprop	0.01 ± 0.0	0.85 ± 0.23	0.0 ± 0.0	0.71 ± 0.45	0.0007 ± 0.0001
InputXGradient	0.01 ± 0.0	0.85 ± 0.23	0.45 ± 0.5	0.54 ± 0.5	0.0007 ± 0.0001
IntegratedGradients	0.26 ± 0.02	0.89 ± 0.21	0.37 ± 0.48	0.46 ± 0.5	0.0007 ± 0.0001
Saliency	0.01 ± 0.0	0.65 ± 0.33	0.0 ± 0.0	0.71 ± 0.45	0.0007 ± 0.0001

Table A.10: Proteins GIN Fidelity metrics

Explainer	Entropy % ↓	Repeatability ↑	Total Variance ↓	Counterfactual % ↓
Deconvolution	0.9 ± 0.03	1.0 ± 0.0	-0.0 ± 0.0	0.78 ± 0.3
DummyExplainer	0.97 ± 0.0	0.02 ± 0.03	120.49 ± 119.77	0.81 ± 0.26
GNNExplainer	0.95 ± 0.03	0.47 ± 0.19	6.49 ± 11.48	0.74 ± 0.32
GuidedBackprop	0.9 ± 0.03	1.0 ± 0.0	-0.0 ± 0.0	0.78 ± 0.3
InputXGradient	0.75 ± 0.05	1.0 ± 0.0	0.0 ± 0.0	0.77 ± 0.3
IntegratedGradients	0.76 ± 0.06	1.0 ± 0.0	-0.0 ± 0.0	0.75 ± 0.32
Saliency	0.9 ± 0.03	1.0 ± 0.0	-0.0 ± 0.0	0.78 ± 0.3

Table A.11: Proteins GIN Other metrics

Explainer	Time seconds ↓	RDT Fidelity ↑	Fidelity- ↓	Fidelity+ ↑	Mask Kernel Distance ↑
Deconvolution	0.01 ± 0.0	0.93 ± 0.19	0.03 ± 0.16	0.74 ± 0.44	0.0007 ± 0.0001
DummyExplainer	0.0 ± 0.0	0.92 ± 0.2	0.18 ± 0.38	0.12 ± 0.33	0.0005 ± 0.0
GNNExplainer	0.69 ± 0.02	0.91 ± 0.22	0.43 ± 0.5	0.0 ± 0.0	0.0006 ± 0.0001
GuidedBackprop	0.01 ± 0.0	0.93 ± 0.19	0.03 ± 0.16	0.74 ± 0.44	0.0007 ± 0.0001
InputXGradient	0.01 ± 0.0	0.95 ± 0.17	0.38 ± 0.49	0.15 ± 0.36	0.0007 ± 0.0001
IntegratedGradients	0.23 ± 0.02	0.95 ± 0.16	0.46 ± 0.5	0.1 ± 0.3	0.0007 ± 0.0001
Saliency	0.01 ± 0.0	0.89 ± 0.25	0.0 ± 0.0	0.74 ± 0.44	0.0007 ± 0.0001

Table A.12: Proteins SAGE Fidelity metrics

Explainer	Entropy % ↓	Repeatability ↑	Total Variance ↓	Counterfactual % ↓
Deconvolution	0.88 ± 0.04	1.0 ± 0.0	-0.0 ± 0.0	0.74 ± 0.37
DummyExplainer	0.97 ± 0.0	0.02 ± 0.03	120.4 ± 119.82	0.75 ± 0.36
GNNExplainer	0.95 ± 0.03	0.34 ± 0.22	12.3 ± 23.09	0.67 ± 0.39
GuidedBackprop	0.88 ± 0.04	1.0 ± 0.0	-0.0 ± 0.0	0.74 ± 0.37
InputXGradient	0.7 ± 0.06	1.0 ± 0.0	-0.0 ± 0.0	0.71 ± 0.38
IntegratedGradients	0.71 ± 0.05	1.0 ± 0.0	0.0 ± 0.0	0.73 ± 0.38
Saliency	0.88 ± 0.04	1.0 ± 0.0	-0.0 ± 0.0	0.75 ± 0.37

Table A.13: Proteins SAGE Other metrics