Beyond Spectral Graph Theory

An Explainability-Driven Approach to Analyzing the Stability of Graph Neural Networks to Topology Perturbations

Rauno Arike[1]

Supervisor(s): Elvin Isufi[1]**, Mohammad Sabbaqi**[1]**, Maosheng Yang**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

Abstract

Graph Neural Networks (GNNs) have emerged as a powerful tool for learning from relational data. The real-world graphs such models are trained on are susceptible to changes in their topology. A growing body of work in the field of GNNs' stability to topology perturbations is trying to characterise how models respond to those changes, providing valuable insights that have enhanced the robustness of GNNs to adversarial attacks. The past work in this field, however, has only approached stability analysis using spectral graph theory, which is not applicable to all kinds of GNN models. In this paper, we aim to extend the past work in GNN stability by proposing an algorithm for analysing the stability of GNNs with model-agnostic GNN explainability tools instead of the mathematical framework of spectral graph theory. We demonstrate that the outputs of explainability tools can encode useful insights into the stability of GNNs and present a case study on using those insights to analyse node removal, edge removal, and edge weight perturbations.

1 Introduction

Graph-structured data is ubiquitous throughout numerous domains, from social to biological networks. To leverage the unique structure of this data for machine learning, practitioners often use Graph Neural Networks (GNNs), a type of machine learning model tailored to learning from relational data. GNNs can be used for various tasks:

- Node classification—for example, to classify the topic of documents based on hyperlink or citation graphs [1]
- Link prediction—to predict the side effects of drugs [2]
- Graph classification—to predict the toxicity of molecules [3]

In all of those applications, the underlying network that the GNN is trained on is assumed to be perfectly known. However, this is often not the case in real-world scenarios: networks can change, there can be estimation errors when determining the network components, and networks can be attacked by adversaries [4]. Such changes to the underlying graph are known as alterations to its topology, and the ability of a GNN to output reliable predictions under such changes is referred to as its stability to topology perturbations. The stability of a GNN is an important property for several real-world applications: for example, stabler networks are less susceptible to adversarial attacks [4] and thus more reliable.

There are several works that have analysed the stability of GNNs to topology perturbations [4]–[8] from the theoretical perspective of spectral graph theory [9]. However, the framework of spectral graph theory is not applicable to all popular GNN architectures. To describe the stability properties of those architectures, other approaches are required. One candidate for this is the growing field of GNN explainability, which attempts to explain why GNNs produce particular kinds of predictions [10]–[12]. Despite that, there has been very little work in the intersection of GNN stability and explainability. This paper aims to to demonstrate the feasibility of applying explainability tools in this context.

Towards this end, our contributions are the following:

- We present an algorithm for generating topology perturbations informed by the explanations for model predictions generated by explainability tools.
- Using this algorithm, we demonstrate correlations between the frequency at which specific nodes or edges are identified as explanations and the impact of these nodes/edges on model stability.
- We explore this link across two datasets and three types of perturbation: node removal, edge removal, and edge weight perturbations.

2 Background

2.1 The Stability of Graph Neural Networks to Topology Perturbations

One of the first works examining the stability of GNNs to topology perturbations was by Gama et al. [5]. In that paper, the authors proved that all GNN architectures which employ graph convolutions with integral Lipschitz filters are stable to small topolgy perturbations. Such architectures are known as graph convolutional neural networks (GCNNs). The result is inherited from spectral graph theory: the stability properties that hold for graph filters that satisfy the integral Lipschitz condition also hold for any GNN that uses integral Lipschitz filters in its architecture.

Subsequent works have focused on extending these results to a wider range of perturbation types, as well as on gaining a deeper theoretical understanding of the stability of graph filters. Gao et al. [7] show that GCNNs are stable to not only deterministic but also to stochastic perturbations. Kenlay et al. [8], [13] attempt to express the upper bound on the change of the filter output not only as a function of the magnitude of the perturbation, but also as a function of the structural properties of the graph and the perturbation. Such a bound is desirable as it provides more information about the sufficient conditions under which a graph filter will be stable, thus being helpful for devising strategies for defense against adversarial attacks.

This usefulness of stability analysis to the field of adversarial robustness has prompted follow-up analyses: for example, Chang et al. [4] conduct an adversarial vulnerability analysis of GNNs based on matrix perturbation theory.

A common theme among all of the aforementioned works is that they approach the stability analysis of GNNs using the tools of spectral graph theory [9]. However, not all GNN architectures are amenable to such analysis: various popular spatial GNN architectures such as Graph Attention Networks [14] and GraphSage [15] cannot be easily represented in the Fourier domain and are thus difficult to analyse. It would be desirable to develop tools that better enable the analysis of the stability of such spatial architectures. In other words, we would like to develop stability analysis tools that are *model-*

agnostic, meaning that they can be applied on any type of GNN model.

A further commonality among the aforementioned works is that they assume the graph filters used in the GNN architectures to satisfy the integral Lipschitz condition. The stability properties of GNN architectures that do not fulfill this assumption are also poorly understood.

2.2 A Taxonomy of Graph Explainability Methods

How to approach the analysis of the stability properties of GNNs in a model-agnostic way? One possible solution is to use the set of tools developed in the field of Explainable AI (XAI) [16]. Explainability methods enable the identification of the features, nodes, and edges that most strongly impact model predictions. One might expect that since the model relies heavily on those graph elements, perturbations involving those elements would have the biggest impact on stability. In other words, one might expect the explanations generated using XAI methods to also involve information about the stability properties of the graph. In this paper, we will empirically test this hypothesis.

A crucial step towards providing faithful explanations [17] for the stability properties of GNNs is choosing the correct explainability technique. To provide an overview of the considered techniques, we provide a short taxonomy of the techniques that have been considered when writing this paper. This taxonomy is based on three earlier surveys of explainable graph neural networks [10]–[12].

Following Yuan et al. [10], we decompose GNN explainability techniques into two categories: factual and counterfactual. The category of factual explanations can be further decomposed into self-interpretable factual explanations and post-hoc factual explanations. In this paper, we will focus on post-hoc factual explanations, as they allow for the explanation of already trained GNNs without requiring modifications to the model architecture or training process.

There are five classes of post-hoc factual explanation techniques:

- **Gradient-based methods** utilize the gradients of the model's output with respect to the input features to identify the importance of each feature in the model's decision-making process.
- **Perturbation-based methods** generate explanations by measuring the change in the model's output when certain input features or subgraphs are perturbed or removed.
- **Decomposition-based methods** aim to decompose the model's output into contributions from individual input features or subgraphs.
- **Surrogate methods** approximate the behavior of the original GNN model using a simpler, more interpretable model.
- **Generation-based methods** utilize generative models to generate explanations in the form of important subgraphs or counterfactual examples.

This paper will focus on the applicability of gradient-based and perturbation-based methods to GNN stability analysis. We will motivate this choice in Section 3.3.

3 Methodology

3.1 Experimental Approach

As mentioned in the Section 2, explainable AI techniques constitute a tool that may allow for a principled analysis of the stability properties of GNNs not amenable to a spectral graph theory-based analysis. To design experiments that verify this hypothesis, we have to choose among the five classes of explainability techniques explored in Section 2. Additionally, there are two ways these techniques can potentially be utilised for stability analysis:

1. One approach is to generate explanations to the original graph using the XAI tool of choice and to use those explanations to inform the perturbations that we make to the graph. Given that the explanations informing the perturbations are interpretable, we might hope that those interpretations can also be used to explain the effect of these perturbations to stability.
2. Another approach is to first perturb the original graph and and to then generate interpretations that explain the differences in the model predictions before and after the perturbations have been applied.

A challenge that both of these approaches face is that while stability is commonly measured over the entire graph, the outputs of all existing explainability tools are specific to a small number of nodes or edges. This is less problematic for the first approach: the explanations have to be generated once for the original graph and can then be used to inform various different perturbations. In comparison, the second approach would require generating explanations across the entire graph for each perturbation and stability measurement. Due to this requirement, we deem the second approach prohibitively expensive and focus on the applicability of explainability methods to the first approach. The next section will describe our methodology for this in detail.

3.2 Algorithm for Generating Perturbations

To verify our hypothesis that the stability properties of GNNs can be analysed using explainability tools, we developed an algorithm that can be used to create explainability-informed stochastic node removal, edge removal, and edge weight perturbations. We developed our algorithm with those perturbations in mind because these perturbation types are simpler than others such as edge rewiring perturbations, making them more suitable for the first experiments exploring this hypothesis. We expect it to be possible to extend this algorithm to other types of perturbation.

Our algorithm, illustrated on Figure 1, consists of the following six steps:

1. Train the chosen model on the chosen dataset.
2. For each node in the graph, train the chosen explainability algorithm (the *explainer*) on the model prediction for that node to retrieve a small compact *explanation subgraph* that explains the model's prediction for that node.
3. For each node in the graph, calculate the total number of explanation subgraphs it appears on. Each node appears the explanation subgraph generated for itself, but many

of them are relevant to the predictions made for other nodes and thus appear on multiple subgraphs. The explanation subgraphs include both nodes and edges, but the edges should be ignored in the case of node removal perturbations. For edge removal or edge weight perturbations, ignore the nodes and calculate the same statistics for the edges instead.

4. Sort the nodes based on the frequencies with which they appeared on the explanation subgraphs and divide them into n equally-sized bins based on the frequencies. The first bin should contain the nodes with the highest frequencies, the second bin the nodes with the second-highest frequencies, etc. We refer to those bins as the *frequency groups*. In the case of perturbations involving the edges, sort the edges rather than the nodes into bins.

5. Randomly draw m samples from each bin, each sample containing half of the elements in that bin. This step makes the perturbations stochastic, enabling better uncertainty estimates for the stability values calculated for each bin.

6. For each sample, remove the elements in that sample from the graph and measure the model's stability. In the case of edge removal or edge weight perturbations, only edges are removed, while in the case of node removal perturbations, nodes are sampled and removed together with the edges that connect to them. This step produces a tensor of size $n \times m$ with the measured stability values. The model's stability is measured as follows:

$$\Delta \mathbf{y} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i)^2}} \quad (1)$$

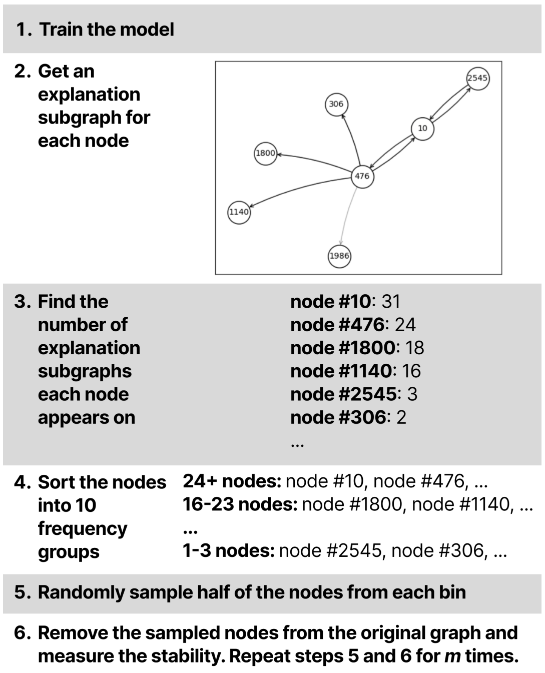
where y_i is a single element in the model's raw (i.e., pre-softmax) output vector on the original graph, $\hat{y}_i$ a single element in the model's raw output vector on the perturbed graph, and $\Delta \mathbf{y}$ the root relative mean squared error (RRMSE) between these output vectors. This is our measure of stability to topology perturbations: the smaller RRMSE, the more stable the model.

3.3 Explainability Tools

This section gives a short overview of the explainability tools used for the experiments. All explainability tools were implemented using the PyTorch Geometric library [18].

To keep the focus of our project on exploring the applicability of explainability techniques to GNN stability analysis rather than on the implementation of explainability techniques, we decided to focus our experiments on methods that are already available in open-source libraries such as PyTorch Geometric. This ruled out the use of decomposition-based methods, surrogate methods and generation-based methods. Concretely, we decided to use GNNExplainer [19] from the class of perturbation-based explainability methods and Integrated Gradients [20] from the class of gradient-based methods.

Alternative perturbation-based methods that were considered are GraphMask [21] and PGExplainer [22]. We decided against using GraphMask due to the fact that it was mainly developed for NLP-focused datasets, which were out of the scope of this project. PGExplainer was not chosen as it only supports explaining the phenomenon the model is trying to predict, rather than the model prediction itself.


1. Train the model

2. Get an explanation subgraph for each node

3. Find the number of explanation subgraphs each node appears on

node #10: 31
node #476: 24
node #1800: 18
node #1140: 16
node #2545: 3
node #306: 2
...

4. Sort the nodes into 10 frequency groups

24+ nodes: node #10, node #476, ...
16-23 nodes: node #1800, node #1140, ...
...
1-3 nodes: node #2545, node #306, ...

5. Randomly sample half of the nodes from each bin

6. Remove the sampled nodes from the original graph and measure the stability. Repeat steps 5 and 6 for *m* times.

Figure 1: This figure illustrates our algorithm for generating explainability-informed perturbations. The algorithm is illustrated using the example of node removal perturbations, but works very similarly for edge removal and edge weight perturbations. To illustrate step 2, the figure includes a compact subgraph of the Cora graph outputted by the GNNExplainer algorithm to explain the prediction that the GAT model made for vertex 10.

For the gradient-based method, we also considered using Saliency Attribution [23], [24], InputXGradient [25], Deconvolution [26], and Guided Backpropagation [24]. We decided in favour of Integrated Gradients over those approaches due to the robustness and complexity of the method. Though the rest of the mentioned gradient-based methods are useful in offering a quick, computationally simple overview of the features important for the model prediction, it has been shown that they often fail to offer a faithful overview of the computational processes that actually took place in the model to produce the output [27]. Although the rest of the gradient-based methods aside from Integrated Gradients have not been analysed in this paper, support has been implemented in the project repository to generate explanations using these methods and to obtain the graphs presented in this paper for those methods as well.

Finally, we considered using attention-based explainability methods. Those were not used due to the fact that they are not model-agnostic and that there have been concerns about the faithfulness of such explanations [28].

GNNExplainer

GNNExplainer is a method designed to provide insights into the predictions made by GNNs. It identifies a small compact subgraph structure and the node features that most influence the model's prediction for a given node. In this paper, we will only use the compact subgraphs identified by the explainer, so the rest of the section will be focused on those. Figure 1 provides an example of one such compact subgraph.

Given a trained GNN and a specific node or graph prediction, GNNExplainer optimizes to maximize the mutual information between the GNN's prediction and the distribution of possible subgraphs. This involves learning a continuous mask on the adjacency matrix $\mathbf{A}$. The loss function balances the fidelity of the explanation (how well the masked input explains the prediction) with the complexity of the explanation (favouring sparser masks).

Formally, the objective can be described as:

$$\min_{\mathbf{M}}[-H\left(P_\phi\left(Y=y \mid G=\mathbf{A}_c \odot \sigma(\mathbf{M})\right)\right)+\lambda\|\sigma(\mathbf{M})\|_1] \tag{2}$$

where $H(\cdot)$ represents the entropy, $P_\phi(\cdot)$ is the GNN's prediction probability, $\mathbf{A}_c$ is the binary adjacency matrix associated with the computation graph of the node that is being explained, $\mathbf{M}$ is the mask applied on $\mathbf{A}_c$, $\odot$ denotes element-wise multiplication, and σ is the sigmoid function mapping the mask values to the range $[0, 1]$. Large explanations are penalized through an ℓ_1 regularization term. The strength of this term is set using the hyperparameter λ.

This objective is mathematically equivalent to maximizing mutual information between the predicted label distribution Y and the explanation. By optimizing this objective, GNExplainer reveals the minimal subgraph necessary for the GNN to make a prediction.

Integrated Gradients

Integrated Gradients [20] is a gradient-based attribution method for explaining the predictions of neural networks, including GNNs. It indicates how much each feature and each edge contributes to the model's prediction for a given node by assigning importance scores, ensuring that the attributions are consistent and proportional to the actual contributions of the edges. We will again focus on the compact subgraphs found using the technique.

Given a trained GNN and a specific node or graph prediction, Integrated Gradients computes the attributions by integrating the gradients of the model's output with respect to the edges along a path from a baseline graph to the actual input graph, effectively measuring the cumulative effect of the gradients along this path. We use a baseline input where the weight of all edges is zero. The explanation subgraph is returned based on the edge attributions; the nodes that appear on the subgraph are the ones that those edges connect to.

Formally, the attribution of an edge (i, j) can be described as:

$$\mathrm{IG}_{(i,j)} = \int_{\alpha=0}^{1} \frac{\partial \mathbf{F}(\mathbf{x}, \mathbf{A}_\alpha)}{\partial w_{ij}} d\alpha \tag{3}$$

where $\mathbf{F}(\mathbf{x}, \mathbf{A})$ is the output of the GNN on input $\mathbf{x}$ given an adjacency matrix $\mathbf{A}$, $\mathbf{A}_\alpha$ denotes the adjacency matrix of the original input graph with all edge weights set to α, and w_{ij} is the weight of the edge (i, j). As direct computation of this value is intractable, it is approximated by a discrete sum.

The complete formulation of Integrated Gradients is more complicated, but since we use a baseline where all edges are set to zero, the equation simplifies to the aforementioned one. Such a baseline is used for two reasons. First, it ensures that all contributions are measured relative to a neutral starting point, which allows for a clear interpretation of the attributions since any non-zero weight directly indicates a contribution to the prediction. Second, it minimizes the computational overhead.

3.4 Model Architectures

This section gives a short overview of the architectures used for the experiments. All architectures were implemented using the PyTorch Geometric library. We perform the experiments on Graph Convolutional Networks [1] and Graph Attention Networks [14]—two of the most popular GNN architectures.

Graph Convolutional Networks

Graph Convolutional Networks (GCNs) use graph convolutions to aggregate node features from their neighbors, providing a localized first-order approximation of spectral graph convolutions. Though GCNs normally operate in the spatial domain, the convolutional architecture enables analysing them in the spectral domain.

GCNs perform a weighted aggregation of a node's neighbors' features, followed by a linear transformation and non-linear activation. Given a set of node features, $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$, where $\vec{x}_i$ is the feature vector of node i and N is the number of nodes, the core operation in GCNs is defined as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{4}$$

where $\mathbf{H}^{(l)}$ is the matrix of node features at the l-th layer (with $\mathbf{H}^{(0)} = \mathbf{X}$), $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}$ the diagonal degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{W}^{(l)}$ a learnable weight matrix for the l^{th} layer, and σ a non-linear activation function (in our paper, the ReLU).

In addition to GCNs, we attempted to perform the experiments using Topology Adaptive GCNs (TAGCNs) [29], which generalize the GCN architecture by providing a k^{th}-order rather than only a first-order approximation of spectral graph convolutions. However, we failed in getting the explainability techniques to converge to small compact subgraphs. This failure is further explained in Appendix D.

Graph Attention Networks

Graph Attention Networks (GATs) introduce an attention mechanism to the message propagation step in GNNs. The key feature of GATs is that they compute the hidden representations of each node by attending over its neighbors, thus assigning different importance to different nodes in a neighborhood without requiring any knowledge of the graph structure upfront.

Given a set of node features, $\mathbf{H} = \{\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_N}\}$, where $\mathbf{h_i}$ is the feature vector of node i and N is the number of nodes, the attention mechanism of GATs can be described as follows:

$$\mathbf{h_i'} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\mathbf{h_j}\right) \tag{5}$$

where $\mathcal{N}_i$ is the set of neighbors of node i, $\mathbf{W}$ is a learnable weight matrix, σ denotes a non-linear activation function (in our paper, the ELU [30]), and α_{ij} are the attention coefficients computed as:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}h_i\|\mathbf{W}h_j]))}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}h_i\|\mathbf{W}h_k]))} \tag{6}$$

Here, $\mathbf{a}$ is a learnable vector and $\|$ denotes concatenation.

4 Experiments

4.1 Datasets

We performed our experiments on two datasets: Cora [31] and DBLP [32]. Only a subset of the DBLP citation network was used. Cora was chosen for its popularity and simplicity, while DBLP was chosen thanks to the fact that it is a heterogeneous graph well-suited for being transformed into a homogeneous weighted graph. Such dataset choice was necessary as we did not find a weighted homogeneous node classification dataset of similar size to Cora. An overview of the properties of the datasets is given in Appendix A.

To turn the heterogeneous graph featured in the DBLP dataset into a homogeneous one, we followed three steps:

1. Isolate the nodes with the ’Author’ node type from the rest of the graph.
2. Based on edges between ’Author’ nodes and ’Paper’ nodes, identify which authors have coauthored papers with each other and create undirected edges between those authors.
3. Based on the number of times each author has coauthored papers with each of their coauthors, set their corresponding edge weights.

Using the edge weights retrieved through this process enhances the accuracy of both of our models, though the difference was at best 0.4%pt (the exact model accuracies are presented in Appendix A). Though the improvements were small, this indicates to us that our synthetically generated edge weights contain meaningful information about the graph that can be leveraged for better predictions by the models.

4.2 Stability to Explainability-Informed Node and Edge Removal Perturbations

To test our algorithm, we first applied it to generate node and edge removal perturbations for models trained to perform node classification on the Cora dataset. We started by training the chosen model on this task. We then trained the chosen explainer to output an explanation subgraph for the model’s predictions for each node. For each such subgraph, the nodes and edges that appeared on the subgraph were saved into a dictionary. The hyperparameters used for training the models were chosen based on the papers introducing our models [1], [14]. Those values, alongside the hyperparameters used for the explainers, can be found in Appendix B.

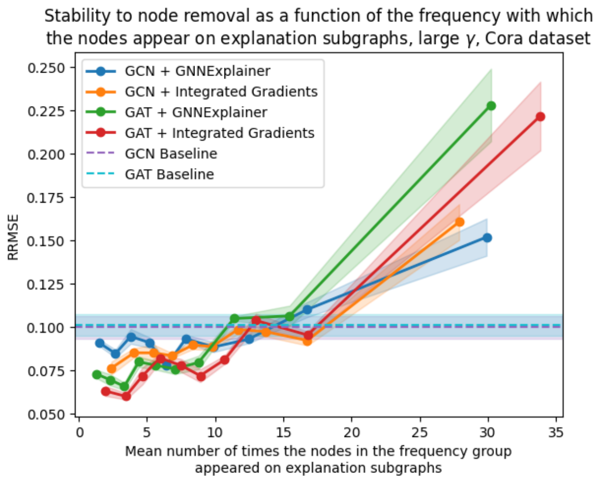


Figure 2: This plot displays the stability to node removal perturbations as a function of the mean number of times the nodes in the frequency group that the removed nodes were sampled from appeared on the explainability subgraphs. The y-axis displays the root relative mean squared error between the model’s output vectors on the original and perturbed graphs. Since the perturbed graph has fewer nodes, only the predictions for the nodes that appear on both graphs are considered when calculating this distance. The shaded areas represent the 95% confidence intervals for the calculated stability values and are calculated based on the 10 samples drawn for each frequency group. The dashed horizontal lines indicate the baseline stability of the models. The significant edge mask value is $\gamma = 0.3$ for GNNExplainer and $\gamma = 0.003$ for Integrated Gradients.

Once the training processes were finished, we aggregated the statistics for the frequencies with which each node and edge appeared on the explanation subgraphs generated for each model. We then divided the sorted nodes/edges into 10 equally sized bins. We chose to use 10 as the number of bins to remove 5% of the nodes/edges on the graph with each perturbation as a reasonable trade-off between having larger perturbations and more data points.

For the next step, we drew 10 random samples from each bin. This number of samples was chosen empirically to retrieve reasonable uncertainty estimates for the stability values calculated for the bins. We found the differences between the stability values measured for different samples to be fairly small, which enabled clear trends to emerge between the stability values for different frequency groups.

This procedure was repeated for each of the three models with which the experiments were performed, considering both node and edge removal. The experiments were performed with two different values for the significant edge mask value, which we will refer to as γ in this paper. This value determines the threshold that the importance weights assigned to edges by the explainer have to exceed in order to be included on an explainability subgraph. The values used for γ, as well as other GNNExplainer hyperparameters, can be seen in Table 4. As the optimal value of γ is discussed by neither Ying et al. [19] nor by Sundararajan et al. [20], we empirically chose two values that both produce reasonably-sized, but substantially different explanation subgraphs.

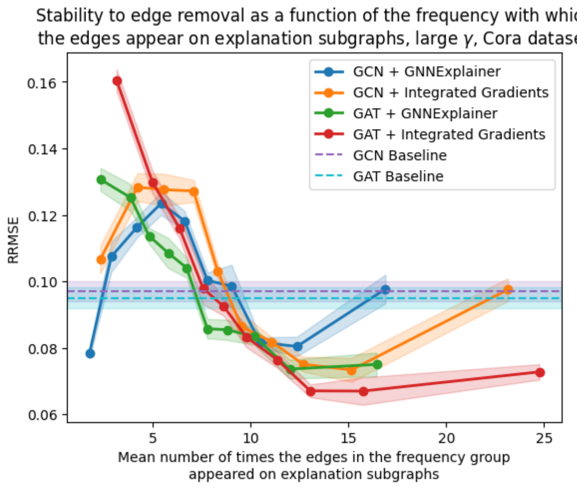


Figure 3: The results shown on this plot were generated following the same experimental design as for Figure 2, applying it to edge removal perturbations instead of node removal.

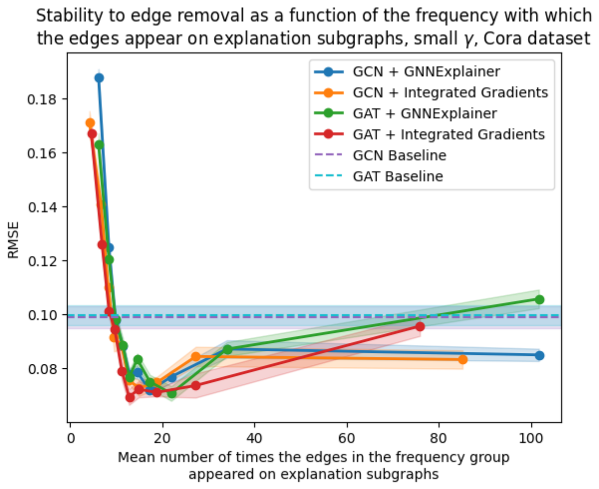


Figure 4: The results shown on this plot were generated following the same experimental design as for Figure 3, but using a significant edge mask value of $\gamma = 0.1$ for GNNExplainer and $\gamma = 0.0001$ for Integrated Gradients.

As the value of γ didn't strongly impact the node removal results, we present the results only for experiments using the larger values of γ. Results for experiments using the smaller values of γ can be found in Appendix C. For edge removal, the impact of this hyperparameter was significant and both results are presented.

The results for node removal perturbations are displayed on Figure 2 and the results for edge removal perturbations on Figure 4 and Figure 3.

The results of the node removal experiments exhibited a clear trend: perturbations including nodes that appear on explainability subgraphs with higher frequencies have a stronger effect on the stability of the networks. Perhaps counterintuitively, however, the results for edge removal perturbations followed the opposite trend. As this is an unexpected result, the reasons behind it are discussed in depth in Section 5.2.

4.3 Stability to Explainability-Informed Edge Weight Perturbations

In order to perform edge weight perturbations, we first had to select a weighted dataset. We chose the heterogeneous DBLP dataset and converted it into a homogeneous weighted dataset, as explained in Section 4.1.

We then followed a similar procedure to the one described in Section 4.2. To corroborate our results, we first performed the same kinds of node and edge removal perturbations for the modified DBLP graph that we performed for Cora. Since our modified version of the DBLP dataset has a similar size to Cora—4057 nodes—, we used the same models and same explainers with the same hyperparameters. The most significant difference compared to Cora was that the GAT model, which was less stable than the GCN model to perturbations on Cora, was significantly more stable to edge removal perturbations than the GCN model across all frequency groups. Otherwise, the results we obtained were very similar to the ones presented in Section 4.2. For this reason, we leave a more detailed overview of them to Appendix C.

Next, we performed edge weight perturbations. To make the setup of the experiment as similar as possible to the rest of the experiments in this paper, we again randomly chose half of the edges from each frequency group to perturb for each sample, meaning that the weights of 5% of all the edges in the graph were perturbed in each experiment. We chose the magnitude of each perturbation to be sampled from a uniform distribution $U(\text{edge\_weight} - 10, \text{edge\_weight} + 10)$. To avoid the creation of edges with negative weights, the absolute value of each sampled value was taken and assigned to the edge as its new weight. We observed that different choices for the magnitude of the perturbation did not qualitatively influence the results: though the absolute impact on stability changed, the relative impact of the perturbations on different frequency groups remained the same. As the final step, we normalized the new weights of the perturbed edges to have the same norm as the old weights of those edges. This was done to ensure that the results are not confounded by changes in the norm of the edge weight vector.

The most surprising observation of the edge weight perturbation experiments was that our GAT model was almost completely unaffected by the perturbations. Another important observation is that unlike most of the other experiments, the results changed significantly when the significant edge mask value was changed. Figure 5 provides a detailed overview of those observations.

5 Discussion

5.1 Stability to Node Removal Perturbations

Across all the experiments on node removal perturbations, we observed a similar trend: perturbations involving the nodes that appear on a higher number of explainability subgraphs have a stronger effect on the stability of our models. This result follows our expectations. Since the explainability sub-

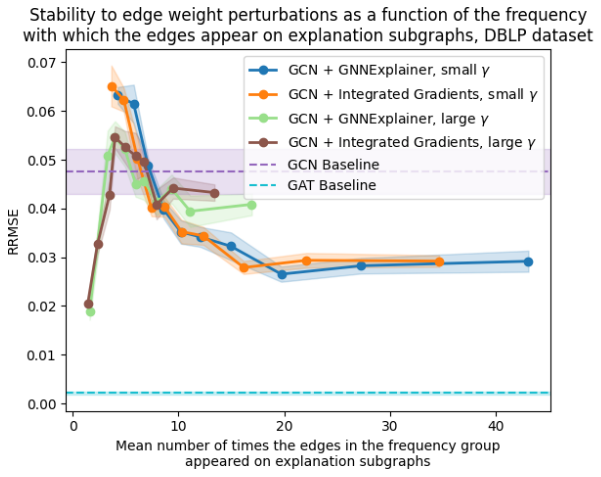


Figure 5: The results shown on this plot were generated following the same experimental design as for Figure 4, applying it on edge weight rather than edge removal perturbations. As the effect of the perturbations was negligible on the GAT model, as shown with the GAT baseline line on the plot, we only present the results for the GCN model, displaying them for both values of γ. The detailed results for the GAT model are shown in Appendix C.

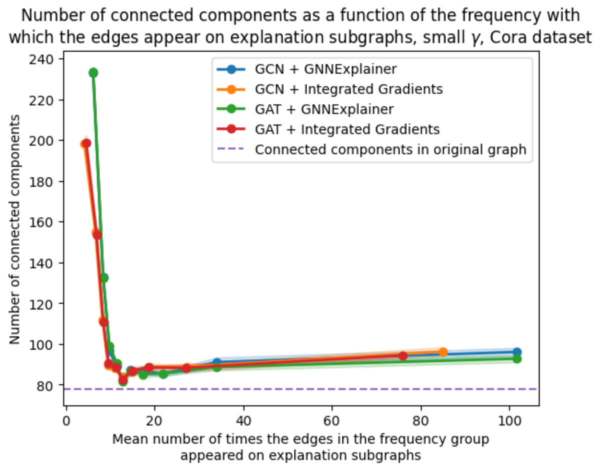


Figure 6: The number of connected components in the graph after each perturbation we made during the edge removal experiments with smaller values of γ on Cora, averaged over the samples taken for each frequency group.

graphs found by explainers should include nodes/edges that are the most influential for the model's predictions, it should follow that if a node appears on a lot of such subgraphs, it has a strong influence on the model's predictions for many different nodes. Removing it from the graph should make it more difficult for the model to generate accurate predictions for all those other nodes, and thus, its removal should be expected to have a relatively high impact on the model's stability. Our results confirm this intuitive expectation.

To verify that the trends we observed were statistically significant, we performed a linear fit on each of our results and calculated the corresponding p-values. This analysis confirmed to us that the observed trends are highly unlikely to be random: e.g., the p-value for the line fitted to the node removal result attained for GAT and GNNExplainer on the Cora dataset with a small value of γ was 2.5×10^{-7}! All of the 16 node removal trend lines had a p-value smaller than or equal to 0.012. The precise p-values are presented in Appendix C.

5.2 Stability to Edge Removal Perturbations

In stark contrast to the experiments on node removal, perturbations involving edges that appear on a higher number of explainability subgraphs were observed to have a weaker effect on the stability of our models. Why did we observe such counterintuitive results?

We initially hypothesized that this behavior can be attributed to the the fact that many of the edges that appear on only one or two of the explanation subgraphs are connected to degree-1 or degree-2 nodes. We expected that removing these edges deprives the model of the necessary graph signals to base its predictions for these low-degree nodes, thus strongly influencing stability. This is not an issue for node removal perturbations: edges connected to low-degree nodes are removed only if the node itself is also removed from the graph.

However, we did not find strong evidence for this hypothesis. Empirically, it appears that choosing a large significant edge mask value tends to eliminate most of the edges connected to low-degree nodes from explainability subgraphs (the figures that illustrate this are presented in Appendix C). Nevertheless, as one can see on Figure 3, the same trend still holds for edge removal perturbations when using a large value for γ. Furthermore, we performed an experiment where we limited the allowed perturbations only to those that don't leave any nodes isolated, but still saw very similar results. The results of this experiment are presented in Appendix C.

Our experiments on edge weight perturbations also confirmed that it is insufficient to explain the results on edge removal perturbations solely by alluding to the creation of isolated nodes. Edge weight perturbations leave all connections on the graph intact, only changing the properties of some connections. Thus, if the difference between the node removal and edge removal results was only caused by the creation of isolated nodes, we would expect the impact of edge weight perturbations to be very similar to the impact of node removal perturbations. However, the results of the edge weight perturbation experiments were much more similar to the results of edge removal experiments (Figure 5).

Our next hypothesis was that the observed trends may be linked to the number of connected components in the graph. Since the edges that appear on a small number of explanation subgraphs are connected to many degree-1 and degree-2 nodes, removing those edges is expected to significantly increase the number of connected components in the perturbed graph. We found this to indeed be the case: Figure 6 displays an incredibly similar trend to Figure 4. We found the same correlation between the number of connected components in the perturbed graph and the stability to edge removal perturbations in experiments using larger values of γ (Appendix C).

If the observed trend is linked to the number of connected components in the perturbed graph, what is the mechanism through which the number of connected components influences stability? One might argue that when the graph gets more fragmented and the number of connected components increases, the nodes that end up in smaller, isolated components no longer receive global information from the larger graph. However, given that we used fairly shallow 2-layer GNNs in our experiments which can only aggregate information from 1-hop and 2-hop neighbours for each node, this explanation does not appear to be likely.

A more likely explanation is that the fragmentation disrupts the flow of information across sparser regions in the graph. In a well-connected graph, information can propagate through multiple paths, ensuring redundancy and robustness. In a fragmented graph, small local neighbourhoods are formed which are deprived of crucial signals from adjacent neighbourhoods, leading to poorer node embeddings in those neighbourhoods. This degradation in node embeddings leads to changes in the output vectors of the GNN and thus impacts stability. Such degradation can also occur for edge weight perturbations, though to a smaller extent: while edge weight perturbations modify the strength of connections, they do not sever them completely. This allows some information to still traverse between the sparser regions, albeit less effectively. This explains both the similarities and the differences between our edge removal and edge weight perturbation experiments. Nevertheless, a rigorous exploration of the correctness of this hypothesis was beyond the scope of this project. Performing this rigorous exploration would be an interesting direction for future works.

Finally, we also performed a p-value analysis for edge removal and edge weight perturbations. The obtained p-values were far less convincing than the ones obtained for node removal perturbations: out of the 16 edge removal trend lines, 7 had a p-value smaller than 0.05, and out of the 8 edge weight perturbation trend lines, 2 had a p-value smaller than 0.05. We expect that the main reason behind this is that the results are not as well described by a linear fit as the node removal results, rather than that the results do not display any meaningful trends: e.g., Figure 4 exhibits an L-shaped rather than a linear trend.

5.3 Are Explainability Tools Useful for Stability Research?

As there has been very little prior work in the intersection of the field of GNN stability and the field of explainable AI, it is important to ask whether this paper gained anything from combining those two fields and whether research in this direction should continue.

The results of the explainability-guided experiments on edge removal perturbations were certainly surprising, and we believe our investigation into the reasons behind this provided a deeper understanding into what makes a model stable through the identified correlation of our results with the number of connected components in the perturbed graph. Our results on node removal perturbations, on the other hand, show that in at least some situations, the intuitions provided by explainability tools can be trusted to guide our understanding of which model predictions are more or less stable to perturbations. Furthermore, though GNNExplainer and Integrated Gradients are very different explainability techniques, the former being perturbation-based and the latter gradient-based, they seem to provide similar explanations for stability. This indicates that explainability tools reliably capture similar aspects of model stability.

However, this paper only studied the applicability of explainability techniques to small networks which contain up to 4057 nodes. Furthermore, we only explored the node classification task. Scaling our methodology to large graphs more relevant to real-world applications may prove to be computationally challenging, as it requires generating a separate explanation for each node in the graph. One way to circumvent this issue is to apply our method only on the most important subsets of real-world graphs, but we nevertheless view scalability as an important limitation of our work that should be addressed by future research. Work on explainability techniques that can provide global explanations for the model's predictions across the entire graph would make it significantly easier to scale our methodology to larger models and is thus a relevant direction to explore.

6 Conclusions and Future Work

In this paper, we presented a novel method for characterising the stability properties of GNNs. We first proposed an algorithm that leverages tools developed in the field of Explainable AI for stability analysis, creating a model-agnostic approach for finding the nodes and edges in a graph which, when perturbed, have the biggest impact on model stability. Then, we demonstrated the results of applying this algorithm on node removal, edge removal, and edge weight perturbations, presenting them across two models, explainability algorithms and datasets. Finally, we discussed the observed trends, as well as the implications of our results.

As there has been almost no prior research in the intersection of GNN stability analysis and Explainable AI, several questions about potential benefits and drawbacks of combining those fields remain unanswered. For example, it was outside the scope of the project to investigate edge and node addition and edge rewiring perturbations. Recent work by Brown [33] provides a strong foundation for this by comparing the impact of different types of perturbation on stability. Our work could also be combined with recent work by Rullens [34] to extend our results to other graph learning tasks: link prediction and graph classification. For the types of perturbation that we investigated—node removal, edge removal and edge weight perturbations—, future research could expand the range of models and hyperparameters our algorithm is tested on. Additionally, future research could investigate possibilities for improving upon the computational complexity of our algorithm, extending it to heterogeneous graphs. Finally, the reasons behind the correlation between our results for edge removal perturbations and the number of connected components in the perturbed graphs should be rigorously explored to provide new insights into the implications of the number of connected components in a graph for stability. Correlations between our results and other graph properties such as assortativity, centrality, and connectivity could also be explored, drawing from the work of Colakoglu [35].

7 Responsible Research

All of the code used for the experiments presented in this paper, as well as CSV files with the results based on which the presented plots were generated, can be found in the project repository.[1] The repository's README contains step-by-step instructions for regenerating all of the presented plots, as well as for reproducing the experiments from scratch.

Two secondary sources of data are used in this paper: the Cora dataset and the DBLP dataset. As both of these are well-known citation networks containing only objective metadata about scientific papers, we do not anticipate them containing any hidden sources of bias.

The research presented in this paper contributes to the field of adversarial robustness. More specifically, by researching the types of perturbations that GNNs are the most and least stable to, we elucidate the kinds of adversarial attacks to the graph topology that these models are the most vulnerable to. Ethically, the importance of adversarial robustness lies in its capacity to protect against malicious attacks that can exacerbate biases, compromise data privacy, or lead to incorrect decisions with potentially harmful consequences. By advancing our understanding and techniques in this area, we contribute to building safer, more trustworthy AI systems.

Though it's possible in principle that the insights presented in this paper can also be used to craft better adversarial attacks in addition to being used for adversarial defense, we find it more important that people working on adversarial robustness are aware of the potential vulnerabilities, as this awareness fosters the development of more robust protective measures. By revealing vulnerabilities, we enable the community to devise effective countermeasures, thereby enhancing the overall resilience of systems. Furthermore, the perturbations studied in this paper cannot directly be used as the basis of new adversarial attacks: we only study small networks, while real-world systems of interest are usually complex networks with many millions of nodes.

References

[1] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2017. arXiv: 1609.02907 `[cs.LG]`.

[2] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bty294.

[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry*, 2017. arXiv: 1704.01212 `[cs.LG]`.

[4] H. Chang, Y. Rong, T. Xu, Y. Bian, S. Zhou, X. Wang, *et al.*, "Not all low-pass filters are robust in graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25 058–25 071, 2021.

[5] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.

[6] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5680–5695, 2020.

[7] Z. Gao, E. Isufi, and A. Ribeiro, "Stability of graph convolutional neural networks to stochastic perturbations," *Signal Processing*, vol. 188, p. 108 216, 2021.

[8] H. Kenlay, D. Thanou, and X. Dong, "Interpretable stability bounds for spectral graph filters," in *International Conference on Machine Learning*, PMLR, 2021, pp. 5388–5397.

[9] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[10] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," 2022. arXiv: 2012.15445 `[cs.LG]`.

[11] J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, and S. Medya, *A survey on explainability of graph neural networks*, 2023. arXiv: 2306.01958 `[cs.LG]`.

[12] Y. Li, J. Zhou, S. Verma, and F. Chen, *A survey of explainable graph neural networks: Taxonomy and evaluation metrics*, 2023. arXiv: 2207.12599 `[cs.LG]`.

[13] H. Kenlay, D. Thanou, and X. Dong, "On the stability of graph convolutional neural networks under edge rewiring," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 8513–8517.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2018. arXiv: 1710.10903 `[stat.ML]`.

[15] W. L. Hamilton, R. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, 2018. arXiv: 1706.02216 `[cs.SI]`.

[16] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson, "Explainable artificial intelligence: An analytical review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 5, e1424, 2021.

[17] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Faithful and customizable explanations of black box models," Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3306618.3314229.

[18] M. Fey and J. E. Lenssen, *Fast graph representation learning with PyTorch Geometric*, https://github.com/rusty1s/pytorch_geometric, Accessed: 2024-04-23, 2019.

[19] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, *Gnnexplainer: Generating explanations for graph neural networks*, 2019. arXiv: 1903.03894 `[cs.LG]`.

[20] M. Sundararajan, A. Taly, and Q. Yan, *Axiomatic attribution for deep networks*, 2017. arXiv: 1703.01365 `[cs.LG]`.

[1]The project repository can be found at https://github.com/RaunoArike/gnn-topology-perturbations.

[21] M. S. Schlichtkrull, N. D. Cao, and I. Titov, *Interpreting graph neural networks for nlp with differentiable edge masking*, 2022. arXiv: 2010.00577 [cs.CL].

[22] M. N. Vu and M. T. Thai, *Pgm-explainer: Probabilistic graphical model explanations for graph neural networks*, 2020. arXiv: 2010.05788 [cs.LG].

[23] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, 2014. arXiv: 1312.6034 [cs.CV].

[24] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," *CoRR*, 2019. arXiv: 1905.13686.

[25] A. Shrikumar, P. Greenside, and A. Kundaje, *Learning important features through propagating activation differences*, 2019. arXiv: 1704.02685.

[26] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, 2013. arXiv: 1311.2901.

[27] L. Sixt, M. Granz, and T. Landgraf, *When explanations lie: Why many modified bp attributions fail*, 2024. arXiv: 1912.09818 [cs.LG].

[28] S. Jain and B. C. Wallace, "Attention is not Explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, pp. 3543–3556. [Online]. Available: https://aclanthology.org/N19-1357.

[29] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, *Topology adaptive graph convolutional networks*, 2018. arXiv: 1710.10370 [cs.LG].

[30] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus)*, 2016. arXiv: 1511.07289 [cs.LG].

[31] A. K. McCallum, K. Nigam, J. Rennie, *et al.*, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000.

[32] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proceedings of The Web Conference 2020*, ACM, 2020. [Online]. Available: http://dx.doi.org/10.1145/3366423.3380297.

[33] A. Brown, *Stability of graph neural network with respect to different types of topological perturbations*, unpublished, 2024.

[34] V. Rullens, *The impact of graph neural network task types on the stability of graph neural networks in face of perturbations: A coded experiment on gnn stability*, unpublished, 2024.

[35] Y. Colakoglu, *An experimental look at the stability of graph neural networks against topological perturbations: The relationship between graph properties and stability*, unpublished, 2024.

A Overview of Datasets

Table 1: Overview of the datasets used in the experiments.

Statistic	**Cora**	**Modified DBLP**
Number of Vertices	2708	4057
Number of Edges	5429	3528
Edge Type	Undirected	Undirected
Edge Weights	Unweighted	Weighted
Feature Vector Dimension	1433	344
Number of Classes	7	4

Table 2: Overview of model accuracies after being trained on the datasets for 100 epochs. The random seed used when training all of the models was 1234567. Though the modified DBLP graph with unweighted edges was not used in our experiments, we also give an overview of the model accuracies on that graph to present the improvement in accuracies brought by the introduction of edge weights created using our methodology that was detailed in Section 4.1.

Model	**Cora**	**Modified DBLP**	**Modified DBLP with unweighted edges**
GCN	81.10%	81.12%	80.72%
GAT	81.00%	81.49%	81.12%

B Overview of Hyperparameters

Table 3: Overview of model hyperparameters.

Hyperparameter	**GAT**	**GCN**
Learning Rate	0.005	0.01
Epochs	100	100
Layers	2	2
Hidden Channels	8 heads, 8 channels	16
Dropout Rate	0.6	0.5
Weight Decay	5×10^{-4}	5×10^{-4}
Optimizer	Adam	Adam

Table 4: Overview of explainer hyperparameters.

Hyperparameter	**GNNExplainer**	**Integrated Gradients**
Epochs	100	N/A
Node Mask Type	Object	Attributes
Edge Mask Type	Object	Object
Used γ Values	0.1, 0.3	1×10^{-4}, 3×10^{-3}

C Additional Results

This appendix provides an overview of additional results that were not presented in the main part of the paper. First, we expand upon the results in Section 4.2 with Figure 7, which presents the results for node removal perturbations performed using a small value of γ. After those figures, we present further results for the experiments described in Section 4.3. Figure 8 and Figure 9 show the results for node removal perturbations, while Figure 10 and Figure 11 present the results for edge removal perturbations on the DBLP graph. Figure 12 and Figure 14 expand upon the edge weight perturbation results, also showing the curves for the GAT model which was almost completely unaffected by the perturbations.

Finally, we expand upon Section 5. First, we present the p-values calculated for each of our results in Table 5. An example of the linear fits used for calculating those values is shown on Figure 15. Then, Figure 16 and Figure 17 show that while a modification to our algorithm that prohibits perturbations which leave nodes isolated softens the trend observed for edge removal perturbations (Figure 4 and Figure 3), it doesn't reverse it to match the trend that we observe for nodes. Figure 18 and Figure 19 confirm the claim we made in Section 5.2 that increasing the value of γ mostly removes low-degree nodes from the explanation subgraphs. Finally, Figure 20 confirms our claim in Section 5.2 that there is a very strong correlation between the number of connected components in the perturbed graph and the stability to edge removal perturbations for experiments using larger values of γ (for a quick overview, compare Figure 20 and Figure 3).

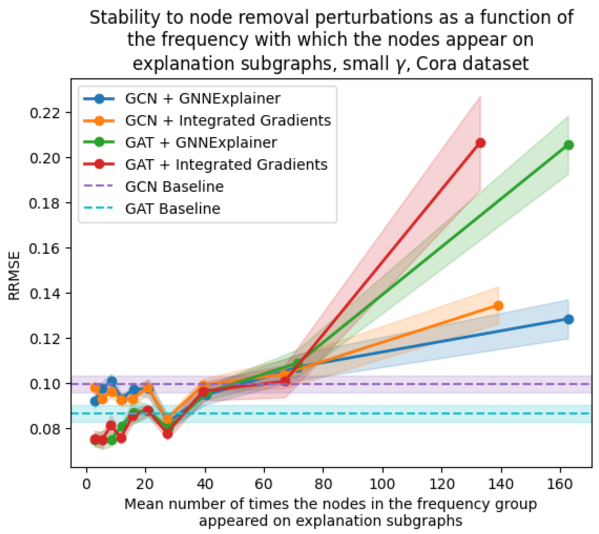


Figure 7: The results shown on this plot were generated following the same experimental design as for Figure 2, but using a significant edge mask value of $\gamma = 0.1$ for GNNExplainer and $\gamma = 0.0001$ for Integrated Gradients.

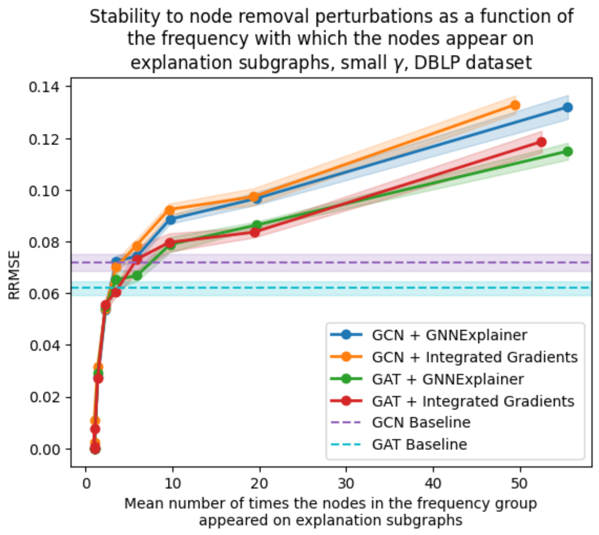


Figure 8: The results shown on this plot were generated following the same experimental design as for Figure 7, applying it on the DBLP dataset instead of Cora.

Table 5: P-values for Node and Edge Removal and Edge Weights (Cora and DBLP datasets)

P-value	**GCN+GNNExpl.**	**GCN+IG**	**GAT+GNNExpl.**	**GAT+IG**
		Cora		
Node removal, small γ	7.9×10^{-4}	3.6×10^{-4}	2.5×10^{-7}	1.4×10^{-5}
Node removal, large γ	4.2×10^{-4}	3.5×10^{-4}	3.8×10^{-5}	1.2×10^{-5}
Edge removal, small γ	0.45942	0.33475	0.86755	0.50497
Edge removal, large γ	0.30010	0.07490	4.4×10^{-4}	0.00810
Edge weights, small γ	0.02943	0.01891	0.07372	0.39081
Edge weights, large γ	0.69072	0.40876	0.14761	0.10252
		DBLP		
Node removal, small γ	0.0072	0.0052	0.01031	0.00940
Node removal, large γ	9.2×10^{-4}	0.01202	1.4×10^{-3}	1.1×10^{-3}
Edge removal, small γ	0.03076	1.7×10^{-4}	0.02165	0.03092
Edge removal, large γ	0.60158	0.99714	0.00561	0.06550

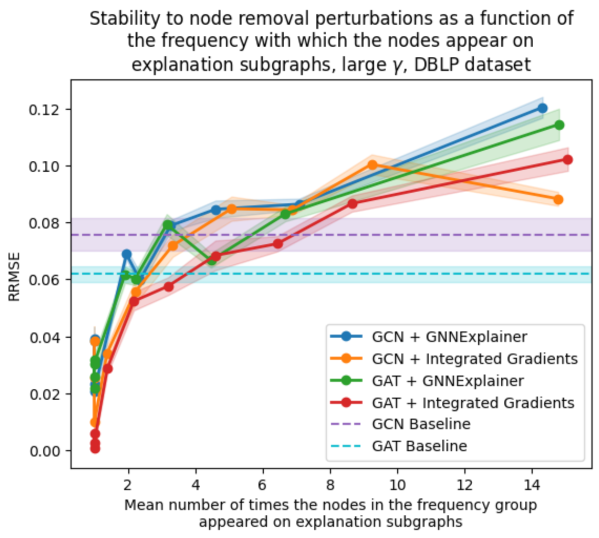


Figure 9: The results shown on this plot were generated following the same experimental design as for Figure 2, applying it on the DBLP dataset instead of Cora.

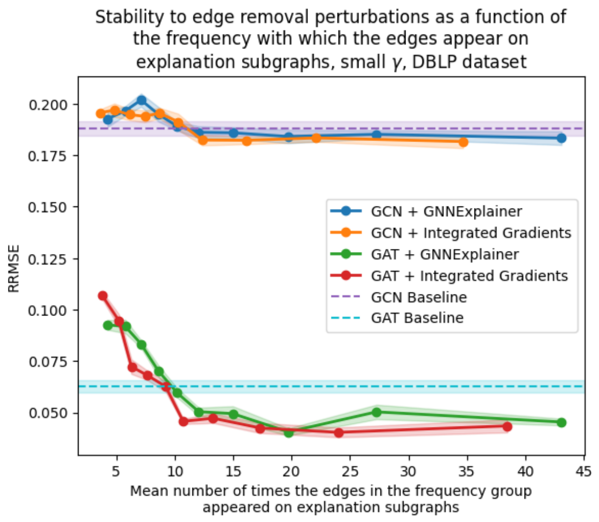


Figure 10: The results shown on this plot were generated following the same experimental design as for Figure 4, applying it on the DBLP dataset instead of Cora.

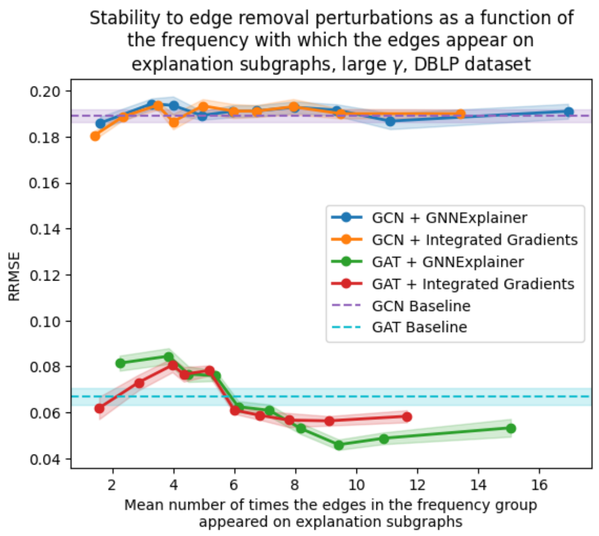


Figure 11: The results shown on this plot were generated following the same experimental design as for Figure 3, applying it on the DBLP dataset instead of Cora.

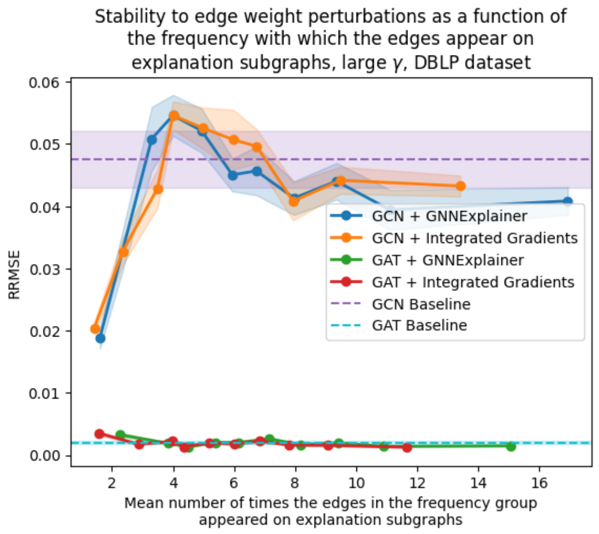


Figure 13: The results shown on this plot were generated following the same experimental design as for Figure 11, applying it on edge weight rather than edge removal perturbations.

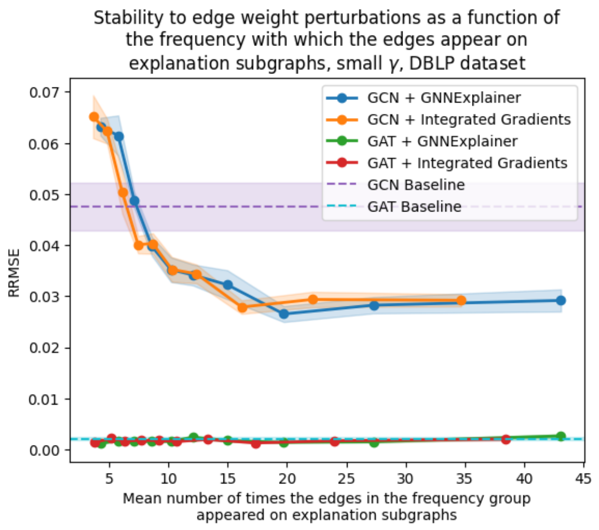


Figure 12: The results shown on this plot were generated following the same experimental design as for Figure 10, applying it on edge weight rather than edge removal perturbations.

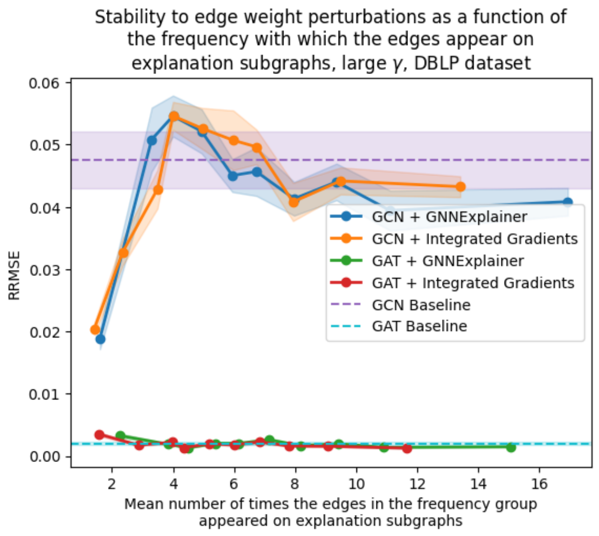


Figure 14: The results shown on this plot were generated following the same experimental design as for Figure 11, applying it on edge weight rather than edge removal perturbations.

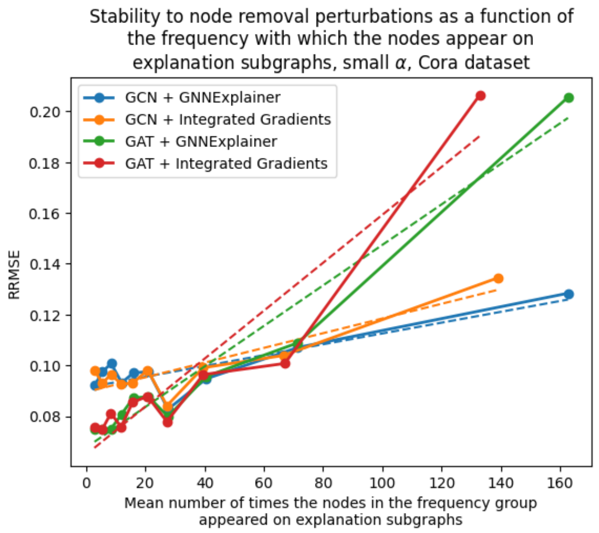


Figure 15: This figure presents an example of finding the best linear fits, denoted by the dashed lines, to our results.

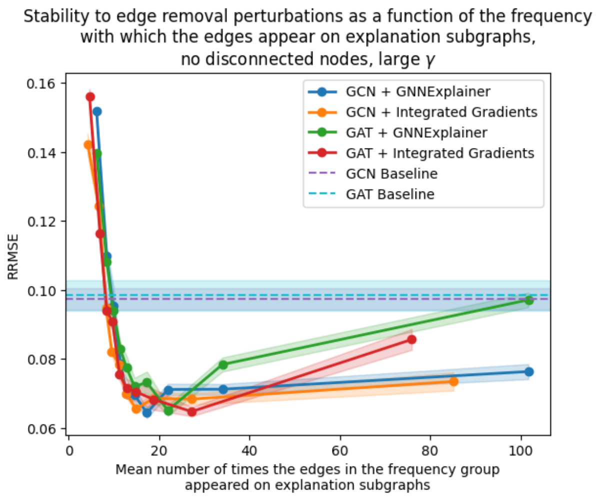


Figure 17: The results shown on this plot were generated following the same experimental design as for Figure 3, but using a variation of the algorithm which prohibits perturbations which leave a node disconnected from the rest of the graph.

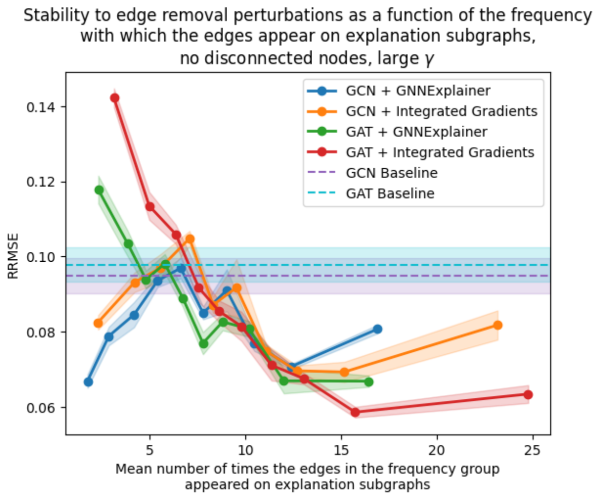


Figure 16: The results shown on this plot were generated following the same experimental design as for Figure 4, but using a variation of the algorithm which prohibits perturbations which leave a node disconnected from the rest of the graph.

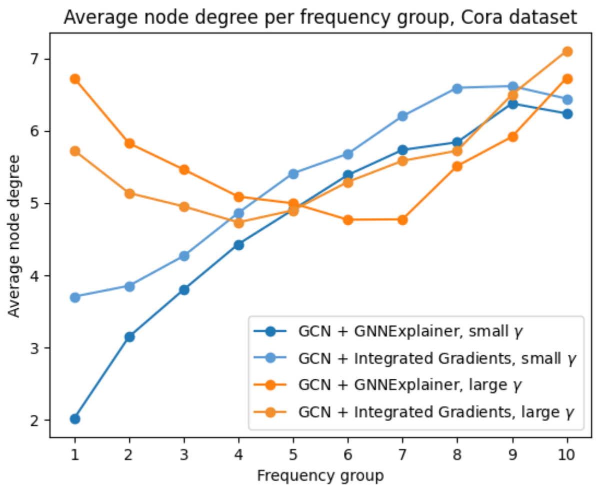


Figure 18: This plot displays the average node degree per frequency group for the GCN model, using the frequency groups generated for the edge removal experiments on Cora.

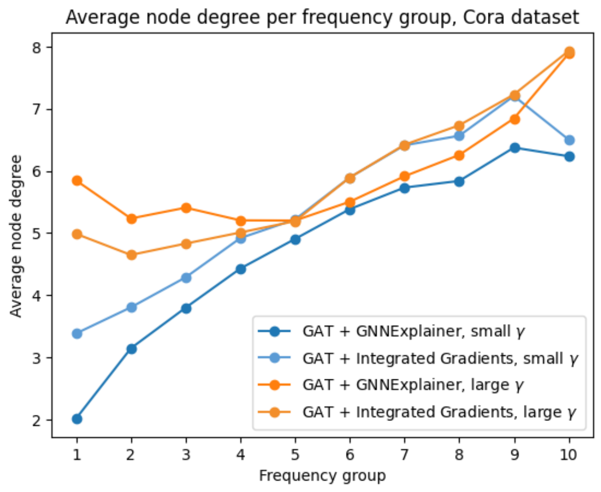


Figure 19: This plot displays the average node degree per frequency group for the GAT model, using the frequency groups generated for the edge removal experiments on Cora.

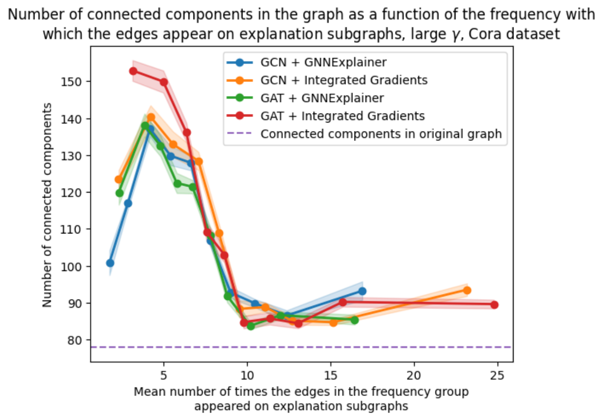


Figure 20: The number of connected components in the graph after each perturbation we made during the edge removal experiments with larger values of γ on Cora, averaged over the samples taken for each frequency group.

D Failed Experiments with TAGCN

As mentioned in Section 3, we also attempted to test our experimental pipeline using the TAGCN architecture, but were unsuccessful. The reason was that we were unable to generate small and compact explanations for the TAGCN model unless we set the parameter K in the convolutional layers to 1, in which case the TAGCN model is functionally equivalent to the GCN model that we used in this paper. The bigger the value of K, the more pronounced this issue was. It occurred for both GNNExplainer and Integrated Gradients. An example of an unsuccessfully generated explanation subgraph for TAGCN is presented on Figure 21.

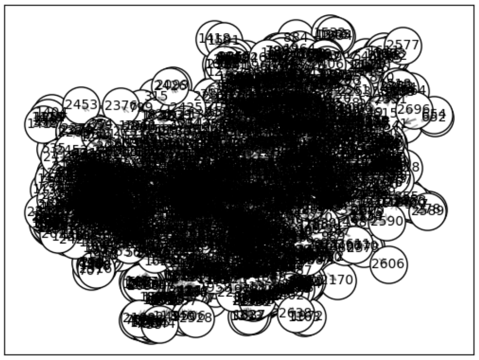


Figure 21: This figure displays an explanation subgraph outputted by the GNNExplainer explainer for a TAGCN model with $K = 3$. As can be seen from the figure, the subgraph contains hundreds of nodes and isn't thus useful for explaining the model prediction.

We expect the reason behind this to be that the TAGCN architecture, by design, aggregates information from a larger neighbourhood (determined by the K parameter) within a single convolutional layer. When generating explanations, our explainers aim to identify the most relevant subgraph that contributes to the model's prediction. However, due to the multi-hop aggregation in TAGCN, the explanations tend to include a larger portion of the graph, resulting in dense and less compact subgraphs.

In contrast, the GCN and GAT architectures only aggregate information from immediate neighbours (i.e., 1-hop neighbours) in each convolutional layer. This local aggregation allows the explainers to generate more focused and compact explanations.

We expect that this issue could be mitigated by exploring alternative explainability tools, as some of them are more suitable for multi-hop aggregation within a single convolutional layer than others. We leave it to future works to determine the explainability tools that can solve this issue.