On the Edit Path to GNN Decisions

Florian Seiffarth[1,2]

[1] University of Bonn, Bonn, Germany seiffarth@cs.uni-bonn.de
[2] Lamarr Institute for Machine Learning and Artificial Intelligence, Germany

Abstract. Graph neural networks (GNNs) base their predictions on two key factors: representations of nodes and edges via attributes, and the underlying graph structure. Understanding the relative role of these two sources of information remains difficult, especially because standard benchmark results often do not reveal whether a model relies on topology, label statistics, or both. This work-in-progress paper proposes graph edit paths as a controlled tool for probing how GNN predictions respond to structural and attribute-level edits. Edit paths define dataset-aware trajectories between real graphs, allowing us to evaluate how predictions change under elementary operations that modify graph structure or node and edge attributes. Unlike random perturbations, shortest edit paths follow minimal transformations between observed graphs and therefore provide a structured exploration of graph space. We first develop heuristics for generating edit paths and analyze the resulting intermediate graphs across molecular benchmark datasets. We then evaluate standard GNNs along these paths, quantifying their sensitivity to different types of edits. Our current results reveal architecture-specific sensitivity profiles: for example, GIN appears more responsive to structural edits and more robust to node relabeling operations. A full statistical analysis of the relationship between structural sensitivity and predictive performance remains ongoing.

Keywords: Graph Neural Networks · Graph Edit Paths · Decision Boundary

1 Introduction

Graph neural networks (GNNs) have emerged as powerful models for graph classification tasks such as molecular property prediction and social network analysis. Despite their success, it remains difficult to understand how GNNs utilize the information encoded in graph representations when making predictions. In particular, a model may rely on graph topology, node and edge attributes, or simple statistics of these attributes. Prior work has shown that, for some benchmark datasets, simple label-based baselines can be competitive with GNNs [20,1]. This motivates tools that can probe a model's sensitivity to structural edits and attribute-level edits. We study graph edit paths as such a tool, providing controlled trajectories through discrete graph space. In continuous domains, interpolation between samples provides a natural way to analyze decision

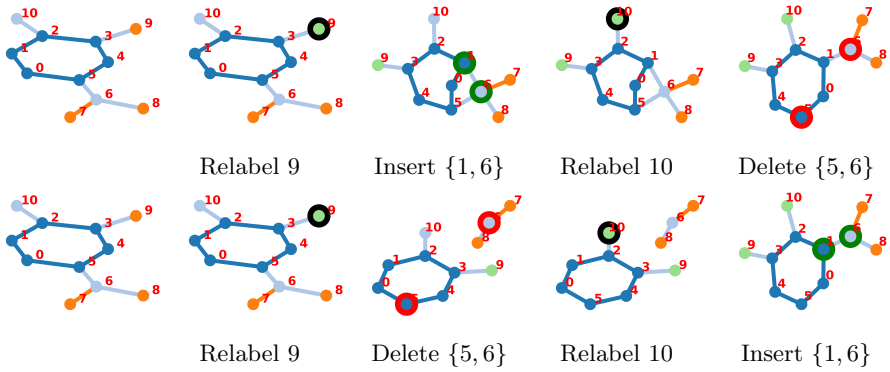


Fig. 1: From *left* to *right*: Two different shortest edit paths of length four between the same source and target molecules from the MUTAG dataset [17]. Black circles indicate the nodes that are relabeled in the respective step, green circles indicate an edge insertion, and red circles indicate an edge deletion.

boundaries. Extending such ideas to graphs is challenging due to the discrete and combinatorial nature of graph structures. To address this challenge, we propose to analyze graph edit paths, which transform one graph into another through sequences of elementary edit operations such as node insertions, deletions, and relabelings as well as edge insertions, deletions, and relabelings. Importantly, different types of edit operations affect different aspects of the graph representation. Node insertions and deletions primarily influence attribute statistics, while edge insertions and deletions modify the structural connectivity of the graph. Node and edge relabelings alter feature information without changing topology. This distinction allows us to analyze how GNN predictions react to controlled perturbations of structure and features.

Why Use Graph Edit Paths? Surprisingly, despite the long history of graph edit distances in graph theory, edit paths have rarely been combined with GNNs. Existing work focuses on data augmentation [12] rather than model analysis. The intermediate graphs along an edit path can be interpreted as synthetic samples obtained through minimal structural modifications. In contrast to random perturbations, which effectively correspond to random walks in graph space and may quickly leave the data distribution, (shortest) edit paths remain close to the original graphs by applying only the minimal sequence of edits required to transform one graph into another. As a result, the generated intermediate graphs follow trajectories that are more consistent with the underlying data distribution. Figure 1 illustrates two different shortest edit paths between the same pair of graphs. Although both paths contain the same set of edit operations, the order in which the operations are applied leads to substantially different intermediate graphs. The upper path maintains connectivity throughout the transformation, whereas the lower path contains disconnected intermediate graphs. These dif-

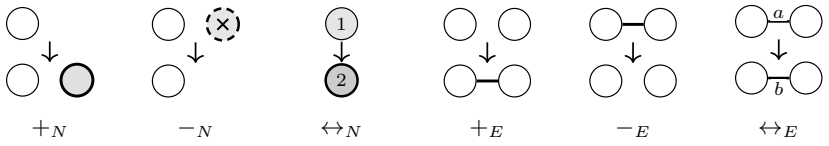


Fig. 2: Schematic overview of the basic edit operations. From left to right: node insertion ($+_N$), node deletion ($-_N$), node relabeling ($\leftrightarrow_N$), edge insertion ($+_E$), edge deletion ($-_E$), and edge relabeling ($\leftrightarrow_E$).

ferences highlight the importance of operation ordering when generating edit paths. Based on this observation, we present our main contributions:

Contribution 1: *We develop and compare heuristics for ordering graph edit operations and analyze how these choices affect the structural properties of intermediate edit-path graphs across molecular benchmark datasets.*

Contribution 2: *We use the generated edit paths to probe trained GNNs by observing their predictions along edit trajectories.*

Our experiments show that edit-path analysis exposes clear architecture-specific sensitivity profiles. These profiles provide evidence about whether a model is more responsive to topology-changing operations or to attribute-changing operations. We also observe preliminary associations between structural sensitivity and predictive performance, but a rigorous statistical correlation analysis remains part of the work-in-progress agenda.

2 Preliminaries

Graphs A graph is a tuple $G = (V, E)$ where V denotes the set of nodes and $E = \{\{v, w\} : v, w \in V,\ v \neq w\}$ the set of edges. When node and edge labeling functions $l_V : V \to \mathbb{N}$ and $l_E : E \to \mathbb{N}$ are provided, the graph is called *labeled*. Typical examples include molecular graphs, where nodes represent atoms and edges represent chemical bonds. A graph representation is given by $(G = (V, E), X_V, X_E)$ where $X_V \in \mathbb{R}^{n \times d}$ and $X_E \in \mathbb{R}^{m \times d'}$ denote the node and edge feature matrices, with $n = |V|$ and $m = |E|$. For labeled graphs, the node and edge features are typically one-hot encodings of the node and edge labels.

Graph Neural Networks A graph neural network (GNN) is a learnable function that takes a graph representation as input and produces an output, e.g., a class label or a regression value. As we specifically focus on graph classification (with k classes), we consider a graph neural network to be a learnable function $f : (V, E, X_V, X_E) \to [0, 1]^k$.

Edit Operations and Edit Paths Labeled graphs can be modified through a set of elementary edit operations. In this work we consider the following six operations: *node insertions* ($+_N$), *node deletions* ($-_N$), *node relabelings* ($\leftrightarrow_N$), *edge*

insertions ($+_E$), *edge deletions* ($-_E$), and *edge relabelings* ($\leftrightarrow_E$). Figure 2 illustrates the six edit operations. Node insertions add new (initially isolated) nodes to the graph, while node deletions remove existing nodes. Note that all incident edges must be removed before a node can be deleted. Edge insertions create new edges between existing nodes that are not already connected, whereas edge deletions remove existing edges. Node and edge relabeling operations modify the corresponding labels without changing the graph topology. Each edit operation is typically associated with a cost that reflects its importance in transforming one graph into another. For simplicity, we assign unit cost to all operations, such that the cost of an edit path equals the number of operations it contains. This assumption is common in the literature, particularly for molecular graph datasets. While more domain-specific cost schemes are possible (e.g., chemically motivated edit costs for molecular graphs), we use uniform costs to obtain a neutral and dataset-independent baseline.

An *edit path* is a sequence of these operations that transforms one labeled graph into another. The graphs obtained after each operation are referred to as *edit path graphs* or *intermediate graphs*. The *graph edit distance* (GED) between two graphs is defined as the length of a shortest edit path between them. A *shortest edit path* is an edit path of length equal to the GED. In general, multiple shortest edit paths may exist between the same pair of graphs. Importantly, the intermediate graphs generated along these paths can differ substantially depending on the order in which edit operations are applied (see Figure 1). Although computing the exact GED is generally NP-hard, it is practically feasible for many standard graph learning benchmarks because they contain relatively small graphs. For larger graphs, GED approximation algorithms [2] can be used to compute edit paths, albeit not necessarily shortest ones.

3 Related Work

Understanding how graph neural networks behave under structural variations of the input graph has recently attracted growing attention. Prior work has primarily explored this question from two perspectives: explainability methods, which aim to reveal important features or structures responsible for predictions, and graph interpolation or augmentation approaches.

Explainability of Graph Neural Networks Explainability methods aim to identify the most influential subgraphs, node features, or structural patterns. Prominent examples include perturbation-based methods such as GNNExplainer [28], parameterized explainers [16], and counterfactual methods such as [15]. While they provide insights into why a particular prediction is made, they typically operate locally around a single input instance and do not capture how predictions evolve across the broader graph space. Model-level explanation approaches instead aim to characterize the global decision behavior of GNNs. Methods such as XGNN [29] and GNNInterpreter [24] generate synthetic graphs that strongly activate specific classes, thereby revealing discriminative structural

patterns learned by the model. More recently, GNNBoundary [25] proposed analyzing decision boundaries directly by generating synthetic graphs that lie close to the boundary between pairs of classes. These boundary graphs are obtained via optimization and enable the study of geometric properties such as boundary margin, thickness, and complexity. While this approach provides valuable insights into the geometry of decision regions, it relies on optimization-based generation of synthetic graphs, which may not necessarily follow dataset-aware transformation trajectories. Beyond explainability, robustness and adversarial studies also investigate decision boundaries of GNNs. Adversarial attack methods on graphs [30,8] analyze how minimal structural perturbations can induce prediction changes, implicitly probing boundary proximity. However, these approaches are typically attack-driven and do not provide a structured view of how predictions evolve under controlled graph transformations.

Graph Edit Paths and Graph Interpolation Graph edit distance (GED) is a principled similarity measure for graphs [21,6,4]. with recent advances in exact GED computation, including improved A^*-based search [7] and stronger ILP formulations [9]. Neural approaches learn GED or reconstruct edit paths from matching matrices [23,18]; these models are themselves interesting subjects for our edit-path analysis, which we leave for future work. In EPIC [12] edit paths have been leveraged beyond similarity computation for learning-related tasks. They construct intermediate graphs along edit paths between graph pairs and use them as augmentation samples during training. By learning context-sensitive edit costs, EPIC performs interpolation in discrete graph space to improve model generalization. Other augmentation strategies for graph data include random edge perturbations [19]. However, these methods primarily aim at improving predictive performance rather than analyzing the behavior of trained models.

Our Contribution Instead of generating graphs through optimization (as in model-level explainability) or using edit paths for training augmentation (EPIC), we study the predictions of trained GNNs along shortest edit paths and analyze how predictions change along these controlled trajectories. This provides a dataset-aware exploration of decision boundary crossings and quantifies sensitivity to specific edit operations (insertion, deletion, relabeling). In this sense, our method complements approaches such as GNNBoundary and EPIC by offering a trajectory-based perspective on decision boundaries in discrete graph space.

4 Generating Graph Edit Paths

For computation of exact pair-wise GED as well as approximations we use the C++ library *gedlib* [3]. Given two graphs, an (optimal) node matching is returned. That is, for every pair of graphs (G, G'), the solver outputs a mapping $m_{G,G'} : V(G) \to V(G') \cup \{\perp\}$ that assigns each node in G to a node in G' or to $\perp$ if the node is deleted. However, to the best of our knowledge, recent tools do not generate the intermediate graphs along an edit path between two input graphs.

This limitation motivates our first contribution called *gedpaths*[3], an extension of *gedlib* that generates ordered sequences of edit operations (using several heuristics) and returns the intermediate graphs for given pairs of input graphs. Additionally, it provides direct export to the PyTorch Geometric [10] graph format, enabling straightforward integration into GNN workflows and facilitating the analysis of model behavior along edit paths.

Given an (optimal) node matching $m_{G,G'}$ between the graphs G and G', we follow the standard algorithm to derive the unordered set of basic edit operations required to transform G into G'. The operations are determined directly from the node matching as follows: *Node deletion operations* correspond to nodes $v \in V(G)$ with $m_{G,G'}(v) = \perp$. *Node insertion operations* correspond to nodes $v' \in V(G')$ for which $m^{-1}_{G,G'}(v') = \emptyset$. *Node relabel operations* occur when a node $v \in V(G)$ satisfies the inequality $l(v) \neq l'(m_{G,G'}(v))$. Edge operations are derived analogously. *Edge deletion operations* correspond to edges $\{u, v\} \in E(G)$ if $\{m_{G,G'}(u), m_{G,G'}(v)\} \notin E(G')$. *Edge insertion operations* correspond to edges $\{u', v'\} \in E(G')$ for which $\{m^{-1}_{G,G'}(u'), m^{-1}_{G,G'}(v')\} \notin E(G)$. Finally, *edge relabel operations* occur when an edge $\{u, v\} \in E(G)$ exists such that $\{m_{G,G'}(u), m_{G,G'}(v)\} \in E(G')$ and $l(\{u, v\}) \neq l(\{m_{G,G'}(u), m_{G,G'}(v)\})$.

The resulting set of operations uniquely specifies the modifications required to transform G into G' but not their application order. However, in principle, multiple optimal node matchings exist for a graph pair. In this work, we use the optimal matching returned by the GED solver, leaving the analysis of multiple optimal matchings as future work. Note that a node can only be deleted after all its incident edges have been removed, and an edge can only be inserted after both of its endpoints exist. An order that respects these dependencies is called a *valid order*. Each valid order corresponds to an edit path. Different orderings may produce substantially different intermediate graphs, although the length of the shortest edit path is fixed. To characterize these extreme cases, we formalize the following optimization problems.

Let $\mathcal{S} := \mathcal{S}_{G,G'}$ denote the set of all valid orders, i.e., shortest edit paths between the graphs G and G'. For $s \in \mathcal{S}$, let $\mathcal{G} := (G_1, \ldots, G_n)$ denote the sequence of intermediate graphs generated by applying the operations. We then consider the following six optimization problems over $\mathcal{S}$: (Min/Max Edge Path) Find $s \in \mathcal{S}$ that minimizes or maximizes $\sum_{i=1}^{n} |E(G_i)|$, (Min/Max Node Path) Find $s \in \mathcal{S}$ that minimizes or maximizes $\sum_{i=1}^{n} |V(G_i)|$ and (Min/Max Disconnected Path) Find $s \in \mathcal{S}$ that minimizes or maximizes $\sum_{i=1}^{n} \mathbb{1}(G_i)$, $\mathbb{1}$ being the indicator function on disconnected graphs.

A detailed elaboration of these problems is left to future work. Motivated by these objectives, we design the following simple heuristics that influence selected structural properties of the intermediate graphs: *Random Baseline Heuristic* (Rnd) where all operations are applied in a uniformly random valid order, and *Edge-First Heuristics ($+_E/-_E$)* where we prioritize edge insertions/deletions to keep the total number of edges high/low in the edit path. Edge operations are applied first, followed by the remaining operations. Relabel operations are in-

[3] See `https://github.com/mlai-bonn/gedpaths` for the code.

serted at random valid positions since they do not affect structural dependencies. Node deletions associated with isolated nodes are applied immediately to reduce the occurrence of isolated nodes in intermediate graphs.

Figure 1 illustrates the effect of choosing different heuristics. The top path corresponds to the $+_E$ heuristic, maintaining graph connectivity throughout the transformation, whereas the bottom path corresponds to the $-_E$ heuristic. This example shows that $+_E$ tends to maximize the total number of edges along the path and minimizes the disconnected graphs whereas $-_E$ minimizes the total number of edges along the path and maximizes the disconnected graphs.

We next analyze path graph properties for standard molecule datasets regarding unique graphs, number of nodes and edges and graph connectivity.

Datasets For evaluation, we compute shortest edit paths for 5000 random graph pairs for six widely used molecule datasets: MUTAG, PTC_FM, DHFR, NCI1, NCI109 and Mutagenicity [17]. They are: (1) used in many GNN benchmarks and (2) previous studies have shown that datasets such as MUTAG and PTC_FM are not substantially better solved by GNNs than by simple baselines [20,1], making them particularly suitable for studying the relative importance of structural information and node/edge attributes in model predictions. In contrast, datasets such as NCI1, NCI109, DHFR, and Mutagenicity are known to benefit more strongly from graph-based models. This diversity allows us to analyze GNN behavior across benchmarks with different levels of structural relevance.

Path Generation We use the $F2$ algorithm [14] in *gedlib* with a timeout of 180 seconds per graph pair across all datasets to compute 5000 optimal node matchings per dataset. Pairs exceeding this limit are skipped. The proportion of skipped pairs varies across datasets: for smaller datasets the skip rate ranges from 0.75% (PTC_FM) to 1% (MUTAG), whereas for larger datasets it increases to 18.3% (Mutagenicity) and up to 52.7% (NCI1), see Table 3 for the details. Table 4 summarizes the heuristic-independent statistics of the resulting edit paths. While timeouts introduce a potential bias toward graph pairs with smaller edit distances, most graphs still appear as endpoints in the sampled paths, suggesting that the generated paths remain broadly representative of the datasets.

Statistics of Generated Edit Paths Generating intermediate graphs along edit paths produces a substantial number of additional non-isomorphic graphs. For example, the procedure increases the effective dataset size by a factor of approximately 512 for MUTAG and about 50 for Mutagenicity. The average edit path length, i.e., number of edit operations per path, varies considerably across datasets. For instance, MUTAG has an average path length of 18.25, whereas Mutagenicity exhibits an average of 41.41, more than twice as long. Interestingly, the distribution of operation types is highly uneven. In particular, there are only a few node relabelings, and extremely few edge relabelings, compared to all other edit operations. This observation suggests that structural changes dominate the transformations between graphs in these benchmarks.

Dataset	Unique Graphs			#Nodes			#Edges			#Discon. Graphs		
	Rnd	$+_E$	$-_E$	Rnd	$+_E$	$-_E$	Rnd	$+_E$	$-_E$	Rnd	$+_E$	$-_E$
MUTAG	76 141	+2.6%	−3.7%	18.83	+0.3%	−0.4%	19.07	+6.5%	−5.0%	10.94	−25.0%	+15.3%
PTC_FM	114 929	+1.6%	−0.2%	17.98	−0.6%	0.0%	15.86	+8.8%	−6.2%	18.75	−20.1%	+10.0%
DHFR	132 537	+0.8%	+0.7%	41.05	−0.6%	−0.8%	40.21	+5.6%	−4.8%	23.26	−29.1%	+7.9%
NCI1	171 124	+0.4%	+0.6%	30.16	−1.3%	−0.6%	27.84	+7.8%	−5.4%	29.38	−15.4%	+5.7%
NCI109	168 598	+0.5%	+0.6%	30.17	−1.3%	−0.6%	27.88	+7.8%	−5.4%	28.85	−15.6%	+5.8%
Mutagenicity	204 550	+0.2%	+0.2%	33.58	−1.8%	−0.5%	29.28	+9.4%	−5.7%	34.65	−20.5%	+7.0%

Table 1: Results across path heuristics, showing absolute random-baseline values and relative heuristic changes for (*Unique Graphs*), i.e., a lower bound of non-isomorphic graphs generated (using the Weisfeiler-Lehman test [26]), the average number of nodes, edges, and disconnected graphs over all edit path graphs from 5000 paths per dataset. The Rnd column reports the baseline values, while $+_E$ and $-_E$ show relative (positive, negative) changes with respect to Rnd.

Heuristic Comparison Table 1 presents the characteristics of edit path graphs for our heuristics, including: non-isomorphic edit path graphs (Weisfeiler-Lehman test [26]), average number of nodes and edges, and disconnected graphs across all six datasets. To facilitate comparison, results are reported relative (increase, decrease) to the random baseline heuristic (Rnd). The results reveal several consistent patterns across datasets: (1) The $+_E$ heuristic (edge insertions first) produces more edges and fewer disconnected graphs than Rnd. (2) The $-_E$ heuristic tends to generate fewer edges and significantly more disconnected graphs. (3) Unique graph generation varies across datasets: for smaller datasets, $+_E$ tends to produce slightly more unique graphs, whereas for larger datasets the differences are small relative to the total number of generated graphs. In summary, the ordering of edit operations strongly influences structural characteristics of intermediate graphs and may therefore affect downstream GNN analysis.

Having characterized the structural properties of edit path graphs for different heuristics, we next use path graphs to analyze pre-trained GNNs. Specifically, we investigate how model predictions evolve along edit paths and which types of edit operations are most likely to trigger decision changes.

5 Graph Neural Network Decisions

To analyze the decision behavior of graph neural networks (GNNs), we proceed in two steps: (1) we generate edit paths between pairs of graphs in the benchmark datasets as described in Section 4 and (2) we train five standard architectures, GCN [13], GraphSAGE [11], GIN [27], GAT [22], and GATv2 [5], on the original datasets and evaluate their predictions on the intermediate graphs generated along the edit paths[4] Importantly, this analysis does not attempt to generate

[4] The code for training and evaluation can be found under the following link `https://github.com/fseiffarth/GNNGEDAnalysis` using the python library `https://github.com/fseiffarth/SimpleGNN` for training the GNNs.

Dataset	GCN	GraphSAGE	GIN	GAT	GATv2
MUTAG	$\underline{85.0 \pm 8.7}$	80.0 ± 11.5	$\mathbf{92.2 \pm 6.5}$	83.3 ± 10.5	83.9 ± 8.5
PTC_FM	$\underline{65.6 \pm 7.0}$	$\mathbf{65.9 \pm 8.4}$	64.2 ± 4.6	64.5 ± 3.8	64.8 ± 2.2
DHFR	81.4 ± 3.6	80.0 ± 4.9	81.6 ± 3.0	$\mathbf{84.1 \pm 4.2}$	$\underline{82.3 \pm 3.8}$
NCI1	76.7 ± 1.3	73.5 ± 2.1	$\mathbf{81.6 \pm 1.9}$	$\underline{80.3 \pm 2.5}$	79.2 ± 1.3
NCI109	75.9 ± 1.7	72.1 ± 2.5	$\mathbf{80.9 \pm 1.1}$	$\underline{78.7 \pm 1.5}$	77.7 ± 2.4
Mutagenicity	78.8 ± 2.7	80.2 ± 2.3	$\mathbf{83.3 \pm 2.6}$	$\underline{82.7 \pm 1.8}$	82.2 ± 2.9

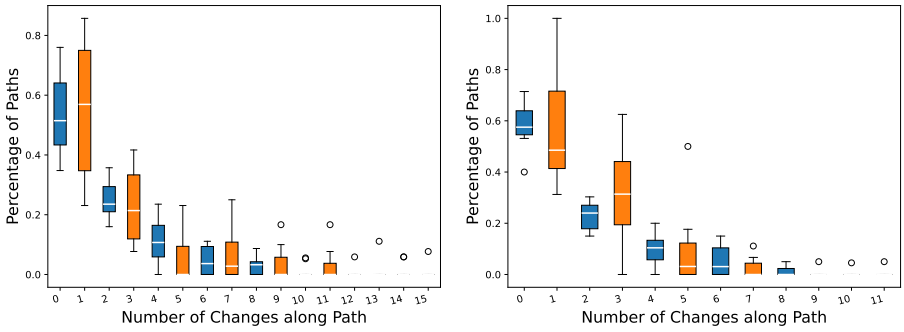
Table 2: Model performance (mean ± std) on the benchmark datasets. Best results are shown in bold and second-best results are underlined.

adversarial examples or optimize graphs toward the decision boundary. Instead, edit paths provide dataset-aware trajectories between real graphs, allowing us to study how predictions evolve under minimal transformations.

All models are trained with the same hyperparameters (see Section A.2). The resulting classification accuracies are shown in Table 2. For the edit-path evaluation, we used the trained models to predict labels for all intermediate graphs generated along the edit paths. Depending on the dataset, this results in up to approximately 212k evaluations on intermediate graphs per heuristic. To ensure that decision changes reflect meaningful model behavior, we restrict our analysis to edit paths whose endpoints are correctly classified. This restriction ensures that an observed decision change reflects a boundary crossing along the path rather than an already-misclassified endpoint; characterizing how misclassified graphs sit relative to the boundary in edit-distance terms is left for future work. Our analysis of GNNs with graph edit paths focuses on three key questions:

Q1 *How often does a model's prediction change along an edit path?*
Q2 *Which edit operations are most likely to trigger a decision change?*
Q3 *Are Q1 and Q2 affected by the path heuristic?*

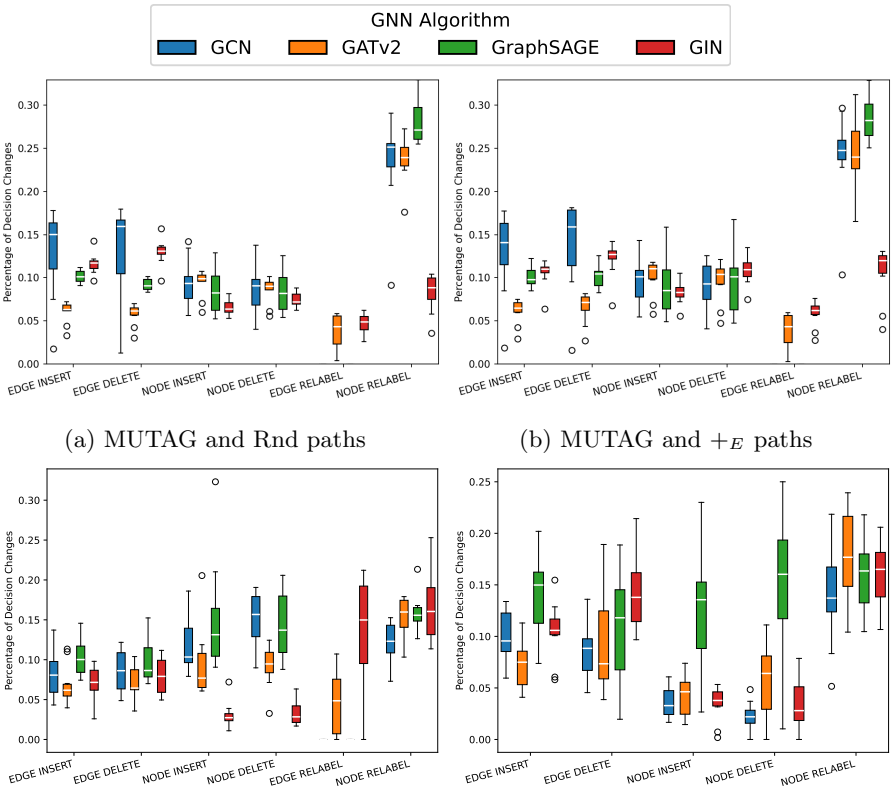
Decision Changes along Edit Paths (Q1) This experiment evaluates both the robustness of the models and the validity of edit paths as meaningful transformation trajectories. We stratify every path by whether its two endpoints carry the same class label or different labels, and report the two groups separately throughout. Ideally, paths whose endpoints share the same label exhibit no decision change, whereas paths whose endpoints have different labels should contain at least one decision change. Figure 3 illustrates the distribution of decision changes for $+_E$ paths (same/different endpoint labels in blue/orange), for GCN and GATv2 trained on the DHFR dataset. Across datasets and models, most paths exhibit either no prediction change (same endpoint labels) or exactly one change (different endpoint labels), confirming the expected pattern that same-label paths tend to stay on one side of the boundary while different-label paths cross it mostly once. Both cases account for roughly $50 - 60\%$ of all paths. Nevertheless, the full distribution of prediction changes varies considerably across architectures (cf. (a) and (b)) and datasets.



(a) DHFR (GCN) on $+_E$ paths (b) DHFR (GATv2) on $+_E$ paths

Fig. 3: Decision changes per path, split by whether the path endpoints have the **same** or **different** class labels, for GCN and GATv2 on the DHFR dataset.

Influence of Edit Operations (Q2) We investigate which edit operations are most likely to alter a model's prediction. Node insertions and deletions operate on isolated nodes and therefore have *no* direct influence on message passing, though they can shift mean-pooled graph representations. Edge insertions and deletions change the structure of the graph. Thus, models that are primarily sensitive to edge operations are likely to rely more on structural information. Node and edge relabeling operations change the input features without modifying the graph structure. Models that are more sensitive to relabeling operations are likely to rely more on node and edge label statistics. Figure 4 reports for each GNN, the proportion of operations per edit type that trigger a decision change for selected datasets. The results reveal substantial differences between architectures, e.g., on the MUTAG dataset, only about 10% of node relabeling operations change the prediction of the GIN model, whereas more than 25% affect the predictions of GraphSAGE. This coincides with the observation that GIN achieves the highest predictive performance on this dataset, while GraphSAGE performs relatively poorly. This is consistent with GIN relying less on node-label perturbations than other models on MUTAG, and suggests that structural information may contribute to its stronger performance on this dataset. Interestingly, the variance across training folds is low, indicating that sensitivity patterns are largely independent of random initialization. Additionally, we analyzed the mean change in class zero prediction for each operation type for PTC_FM on $+_E$ paths (Figure 5). We observe three main patterns: (i) operations that cause a decision change (b) induce a much stronger change in class prediction than arbitrary operations (a); (ii) node relabelings (right column) induce a much stronger change in class prediction than all other operations. The change is more than twice as high as for all other operations, across all models (a) and more than 50% higher than that of other operations regarding (b), i.e. decision changing operations only. (iii) GIN is the most sensitive model to all operations, except for node insertions and deletions, where GATv2 is more sensitive.



(a) MUTAG and Rnd paths

(b) MUTAG and $+_E$ paths

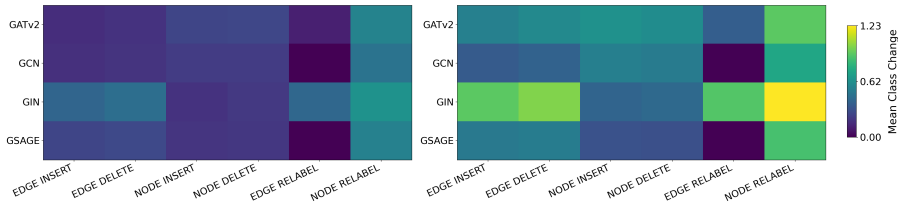
(c) Mutagenicity and $+_E$ paths

(d) NCI1 and $+_E$ paths

Fig. 4: Percentage of edit operations that change the decision for different GNN architectures for MUTAG (a,b), Mutagenicity (c) and NCI1 (d).

Influence of Path Heuristic (Q3) Our experiments reveal that the edit path heuristics have only a minor effect on the relative importance of different edit operations. Comparing $+_E$ and Rnd in Figure 4 (a,b) and Rnd and $-_E$ in Figure 6 shows that the operations which are more likely to cause a decision change, are largely consistent across heuristics. The observed sensitivity patterns seem to be robust to the specific trajectory taken through graph space, and the relative importance of different edit operations is an intrinsic property of the model's decision behavior rather than an artifact of the path generation process.

Summary and Work-In-Progress Our experiments demonstrate that edit-path analysis provides a unified framework for comparing GNNs and benchmark datasets through operation-level sensitivity. Across architectures and datasets, clear behavioral differences emerge. The per operation analysis (Figure 4) suggests that GraphSAGE is more sensitive to node-label statistics, while GIN is comparatively robust to this form of feature-level perturbation. In contrast, GIN



(a) all operations (b) flip operations

Fig. 5: Mean change of class 0 predicted probability per edit type for PTC_FM on $+_E$-paths for all operations (a) and decision change (flip) operations (b).

and GCN respond more strongly to edge insertions and deletions in several settings, suggesting greater sensitivity to structural connectivity. Architectures with stronger structural sensitivity, such as GIN and GATv2, also achieve high predictive performance in our experiments (Table 2). However, this relationship is currently an empirical observation rather than a statistically verified conclusion. The work-in-progress part of this paper is to test whether structural sensitivity correlates with predictive performance across datasets, architectures, folds, and path heuristics. Overall, edit-path-based analysis appears to be a promising tool for studying how GNNs balance structural and attribute-level information.

6 Conclusion and Future Work

We introduced a work-in-progress framework for analyzing GNN decision behavior through graph edit paths. Compared to random perturbations, edit paths follow minimal transformations between real graphs and therefore provide a structured way to probe model behavior in discrete graph space. By evaluating predictions along shortest edit paths, we can identify which elementary operations trigger prediction changes and obtain evidence about whether a model is more sensitive to structural modifications or attribute-level perturbations. Our current experiments show that different GNNs exhibit distinct operation-level sensitivity profiles. They also suggest a possible relationship between structural sensitivity and predictive performance, but this relationship still requires a dedicated statistical analysis. Next steps include testing this correlation more rigorously across datasets, folds, architectures, and path-generation heuristics. Moreover, we will apply the framework to synthetic datasets, larger graph datasets, heterophilic and non-molecular benchmarks where the structure/attribute interplay is less well understood, and non-message-passing paradigms such as diffusion- and energy-based GNNs. Additionally, we would like to analyze edit operations at a more semantic level, for example by studying whether edge insertions or deletions create or destroy cycles, bridges, or other relevant substructures. Finally, we plan to investigate how edit-path analysis may inform training objectives or regularization strategies that explicitly control a model's sensitivity to structural and feature-level perturbations.

References

1. Bechler-Speicher, M., Amos, I., Gilad-Bachrach, R., Globerson, A.: Graph neural networks use graphs when they shouldn't. In: ICML 2024. OpenReview.net (2024), https://openreview.net/forum?id=fSNHK7mu3j
2. Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L.: Comparing heuristics for graph edit distance computation. VLDB J. **29**(1), 419–458 (2020)
3. Blumenthal, D.B., Bougleux, S., Gamper, J., Brun, L.: GEDLIB: A C++ library for graph edit distance computation. In: 12th IAPR-TC-15 International Workshop, GbRPR. LNCS, vol. 11510, pp. 14–24. Springer (2019)
4. Blumenthal, D.B., Gamper, J.: On the exact computation of the graph edit distance. Pattern Recognit. Lett. **134**, 46–57 (2020)
5. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: ICLR (2022), https://openreview.net/forum?id=F72ximsx7C1
6. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognit. Lett. **18**(8), 689–694 (1997)
7. Chang, L., Feng, X., Yao, K., Qin, L., Zhang, W.: Accelerating graph similarity search via efficient GED computation. IEEE Trans. Knowl. Data Eng. **35**(5), 4485–4498 (2023). https://doi.org/10.1109/TKDE.2022.3153523, https://doi.org/10.1109/TKDE.2022.3153523
8. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial attack on graph structured data. In: ICML 2018. Proceedings of Machine Learning Research, vol. 80, pp. 1123–1132. PMLR (2018)
9. D'Ascenzo, A., Meffert, J., Mutzel, P., Rossi, F.: Enhancing graph edit distance computation: Stronger and orientation-based ilp formulations. Proc. VLDB Endow. **18**(11), 4737–4749 (Jul 2025). https://doi.org/10.14778/3749646.3749726, https://doi.org/10.14778/3749646.3749726
10. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric (2019), http://arxiv.org/abs/1903.02428
11. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS. pp. 1024–1034 (2017)
12. Heo, J., Lee, S., Ahn, S., Kim, D.: EPIC: graph augmentation with edit path interpolation via learnable cost. In: IJCAI (2024)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017), https://openreview.net/forum?id=SJU4ayYgl
14. Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Héroux, P., Adam, S.: New binary linear programming formulation to compute the graph edit distance. Pattern Recognit. **72**, 254–265 (2017). https://doi.org/10.1016/J.PATCOG.2017.07.029
15. Lucic, A., ter Hoeve, M.A., Tolomei, G., de Rijke, M., Silvestri, F.: Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In: AISTATS 2022. Proceedings of Machine Learning Research, vol. 151, pp. 4499–4511. PMLR (2022)
16. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. In: NeurIPS 2020 (2020)
17. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tu-dataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop (GRL+ 2020) (2020), www.graphlearning.io
18. Piao, C., Xu, T., Sun, X., Rong, Y., Zhao, K., Cheng, H.: Computing graph edit distance via neural graph matching. Proc. VLDB Endow. **16**(8), 1817–1829 (2023). https://doi.org/10.14778/3594512.3594514, https://www.vldb.org/pvldb/vol16/p1817-cheng.pdf

19. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. In: ICLR 2020. OpenReview.net (2020), https://openreview.net/forum?id=Hkx1qkrKPr
20. Schulz, T.H., Welke, P.: On the necessity of graph kernel baselines (2019), https://api.semanticscholar.org/CorpusID:210171276
21. Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23**(1), 31–42 (1976). https://doi.org/10.1145/321921.321925
22. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2018), https://openreview.net/forum?id=rJXMpikCZ
23. Wang, R., Zhang, T., Yu, T., Yan, J., Yang, X.: Combinatorial learning of graph edit distance via dynamic embedding. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021. pp. 5241–5250. Computer Vision Foundation / IEEE (2021). https://doi.org/10.1109/CVPR46437.2021.00520, https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Combinatorial_Learning_of_Graph_Edit_Distance_via_Dynamic_Embedding_CVPR_2021_paper.html
24. Wang, X., Shen, H.: Gnninterpreter: A probabilistic generative model-level explanation for graph neural networks. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/forum?id=rqq6Dh8t4d
25. Wang, X., Shen, H.: Gnnboundary: Towards explaining graph neural networks through the lens of decision boundaries. In: ICLR 2024. OpenReview.net (2024), https://openreview.net/forum?id=WIzzXCVYiH
26. Weisfeiler, B., Lehman, A.A.: A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction. Nauchno-Tech Inform **Ser. 2**(9) (1968)
27. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019), https://openreview.net/forum?id=ryGs6iA5Km
28. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. In: NeurIPS 2019. pp. 9240–9251 (2019)
29. Yuan, H., Tang, J., Hu, X., Ji, S.: XGNN: towards model-level explanations of graph neural networks. In: KDD 2020. pp. 430–438. ACM (2020)
30. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: KDD 2018. pp. 2847–2856. ACM (2018)

A Appendix

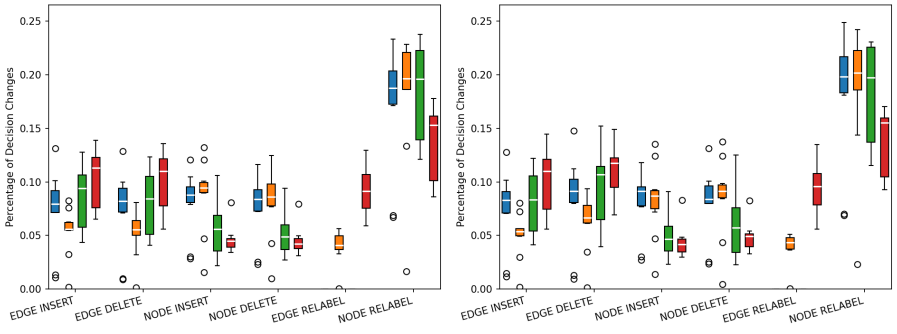
A.1 Additional Figures and Tables

Dataset	Total	Valid	Invalid	%	Avg N		Avg E		Avg ΔN		Avg ΔE		Size N		Size E	
					✓	×	✓	×	✓	×	✓	×	✓ (max)	× (min)	✓ (max)	× (min)
MUTAG	5 077	5 026	51	99.00	17.87	19.75	19.69	21.98	5.33	5.22	6.59	6.35	28	11	33	11
PTC_FM	5 055	5 017	38	99.25	14.03	28.21	14.39	29.96	8.63	14.05	9.72	15.76	64	7	71	7
NCI1	10 587	5 005	5 582	47.27	24.89	36.18	26.69	39.26	12.62	16.02	14.22	17.44	111	11	119	11
DHFR	9 310	5 012	4 298	53.83	39.59	44.92	41.63	47.18	9.79	9.92	10.11	9.94	71	20	73	21
NCI109	10 599	5 009	5 590	47.26	24.84	36.17	26.65	39.33	12.36	16.13	13.92	17.66	106	13	107	10
Mutagenicity	6 132	5 009	1 123	81.69	26.89	44.72	27.55	44.86	15.10	23.82	16.11	20.96	107	12	108	12

Table 3: Additional mapping statistics for method F2 across datasets, using a 180-second timeout, including the total number of valid/invalid mappings generated, their average, maximum (valid) and minimum (invalid) nodes and edges along the edit paths. The final two columns show the minimum nodes and edges observed in the split into valid and invalid mapping subsets.

Dataset	# Graphs	# Path Graphs	$\rightsquigarrow$	$+_N$	$-_N$	$\leftrightarrow_N$	$+_E$	$-_E$	$\leftrightarrow_E$
MUTAG	188(100%)	96 239	18.25	2.65	2.69	1.47	4.75	4.80	1.88
PTC_FM	349(100%)	132 491	25.50	4.30	4.33	2.65	6.23	6.27	1.72
DHFR	756(100%)	147 065	28.42	7.14	2.66	3.22	9.97	5.44	0.00
NCI1	3421(83.2%)	179 823	34.96	8.29	4.33	3.71	11.61	7.02	0.00
NCI109	3430(83.1%)	177 245	34.46	8.01	4.35	3.71	11.31	7.07	0.00
Mutagenicity	3720(85.8%)	212 028	41.41	8.54	6.56	4.25	11.52	9.72	0.80

Table 4: Statistics for 5000 random shortest edit paths per dataset, showing the number and percentage of original graphs occurring as endpoints in the paths (# Graphs), all generated edit path graphs (#Path Graphs), the average path lengths ($\rightsquigarrow$) and average node insertions ($+_N$), deletions ($-_N$), relabelings ($\leftrightarrow_N$), edge insertions ($+_E$), deletions ($-_E$), and relabelings ($\leftrightarrow_E$) per path.



(a) PTC_FM and Rnd paths

(b) PTC_FM and $-_E$ paths

Fig. 6: Percentage of edit operations that change the decision for different GNN architectures on PTC_FM for Rnd paths (a) and $-_E$ paths (b).

A.2 Training Configuration

Specifically, we use the Adam optimizer with cross-entropy loss, a batch size of 32, a learning rate of 0.001, and train each model for 500 epochs. The architecture consists of two message-passing layers with 64 hidden dimensions, followed by mean pooling and an MLP classifier with dimensions $(64 - 32 - 2)$. The classifier uses LeakyReLU activations, batch normalization, and dropout with probability 0.2. The attention-based architectures GAT and GATv2 additionally employ four attention heads. Node features are given as one-hot encodings of node labels, while edge features are used when available (e.g., for MUTAG, PTC, and Mutagenicity). Across datasets, we follow the standard evaluation protocol proposed by [27], performing 10-fold cross-validation with a 90/10 train/test split per fold. Model performance is reported for the epoch achieving the highest average test accuracy across folds.