Causal Path Analysis for Explainable Graph Neural Networks: An Interactive Framework

Thesis submitted by

TC Singh
2023EET2657

under the guidance of

Prof. Sougata Mukherjea, Indian Institute of Technology Delhi

in partial fulfilment of the requirements
for the award of the degree of

Master of Technology

Department Of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI
June 2025

THESIS CERTIFICATE

This is to certify that the thesis titled **Causal Path Analysis for Explainable Graph Neural Networks: An Interactive Framework** , submitted by **TC Singh (2023EET2657)**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

[illegible]

Sougata Mukherjea
Professor
Dept. of Electrical Engineering
IIT-Delhi, 110016

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to IIT Delhi and the Department of Electrical Engineering for granting me this invaluable opportunity to work on this project. Their forward-thinking approach in promoting project-based learning has provided an exceptional platform for my personal and professional growth. I am truly grateful to my project advisor, Prof. Sougata Mukherjea, for his invaluable guidance and support throughout the project. His expertise has enabled me to develop a profound understanding of the key concepts and fundamentals involved. I would also like to thank our course coordinator Prof. Tanmay Chakraborty for his timely guidance throughout the course duration.

Lastly, but certainly not least, I would like to express my heartfelt thanks to my wife, family members and friends for their constant inspiration and unwavering support during times of need.

ABSTRACT

Graph Neural Networks (GNNs) have become a cornerstone of modern machine learning, demonstrating exceptional performance on tasks involving relational data structures. Their ability to learn from the intricate connections within graphs has led to state-of-the-art results in domains such as social network analysis, bioinformatics, and recommender systems. Despite this success, the inherent complexity of GNNs poses a significant challenge to their adoption in critical applications. The iterative message-passing mechanism creates an opaque "black box," making it difficult to trust, debug, and ensure the fairness of GNN-driven decisions.

The field of Explainable AI (XAI) seeks to address this, but the landscape of interactive tools reveals a critical research gap. Existing visual analytics systems are powerful for inspection and debugging but primarily offer correlational views and lack mechanisms for direct, user-driven experimentation on the graph structure. This thesis confronts this challenge by creating a framework centered on user-driven intervention and integrated causal reasoning.

My primary scientific contribution is a novel explanation technique, Causal Path Analysis Interventional-variant (CPA-IV). This method moves beyond standard correlational techniques by operating on the principle of causal intervention. It identifies influential reasoning pathways within the GNN's computation graph by simulating their removal and measuring the impact on the model's prediction, providing a deeper, more mechanistic understanding.

The core practical contribution is a comprehensive, interactive web-based dashboard that serves as an experimental workbench. The tool provides a user-centric interface to visualize the graph, run inference, and generate explanations. Crucially, it integrates three complementary paradigms: intrinsic GAT Attention, the widely-adopted GNNExplainer, and my proposed CPA-IV method. Most importantly, it empowers users to perform "what-if" analyses by directly editing the graph structure and observing the real-time consequences on model predictions and their explanations.

Through a series of case studies, I demonstrate that this interactive, multi-faceted approach provides a holistic understanding of a GNN's behavior. This research thus delivers two principal outcomes: CPA-IV, a new method for causal GNN explanation, and a practical software framework that enables a deeper, more trustworthy, and more rigorous analysis of GNN models by putting the human user in the loop.

Contents

List of Tables

List of Figures

ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CPA-IV	Causal Path Analysis Interventional Variant
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
HCI	Human-Computer Interaction
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Background: The Rise of Graph Machine Learning

In the modern digital era, data is being generated at an unprecedented scale. While much of this data is structured in tabular formats or as unstructured text and images, a vast and increasingly important portion is inherently relational. From the intricate web of social connections on platforms like Facebook and Twitter, to the complex interactions between proteins in a biological network, to the citation patterns linking academic papers, the world is fundamentally a network of interconnected entities. This relational data is most naturally represented using graphs, a mathematical structure composed of nodes (or vertices) that represent entities and edges (or links) that signify the relationships between them.

For many years, traditional machine learning techniques struggled to effectively leverage the rich structural information embedded within graphs. Methods designed for grid-like data, such as Convolutional Neural Networks (CNNs) for images, or sequential data, like Recurrent Neural Networks (RNNs) for text, could not be directly applied to the arbitrary and complex topology of graphs. Early approaches often relied on hand-crafted features derived from graph metrics like node degree or centrality, a process that was both labor-intensive and often failed to capture the full complexity of the underlying structure.

The last decade has witnessed a paradigm shift with the advent of Graph Representation Learning [Ham20] and, more specifically, Graph Neural Networks (GNNs). GNNs have revolutionized the field of machine learning on graphs by providing a powerful and principled framework for learning directly from graph-structured data. Inspired by the success of deep learning in other domains, GNNs operate through a message passing scheme, where nodes iteratively aggregate information from their local neighborhoods [KW16]. Through this process, each node learns a dense, low-dimensional vector representation, or "embedding," that encodes not only its own features but also the structural information of its surrounding context within the graph. These learned embeddings can then be used for various downstream tasks, such as node classification, link prediction, and graph classification, achieving state-of-the-art performance across numerous domains.

1.2 Problem Statement: The "Black Box" Nature of Graph Neural Networks

Despite their remarkable success, the widespread adoption of GNNs is hindered by a significant challenge: their inherent lack of transparency. Like many deep learning models, GNNs often function as "black boxes." The multi-layer, non-linear transformations and complex aggregation functions make it exceedingly difficult for human operators to understand the reasoning behind a model's prediction [YYGJ22]. When a GNN classifies a node in a citation network as belonging to the "Neural Networks" category, what specific features of the paper or which of its neighbors influenced this decision? Was it a key phrase in its abstract, a connection to a seminal paper in the field, or a combination of many weaker signals?

This opacity presents a major barrier in high-stakes application areas where accountability, trust, and fairness are not just desirable, but mandatory. For example:

- In finance, if a GNN-based system flags a transaction as fraudulent, regulators and customers will demand a clear explanation for the decision.
- In drug discovery, if a GNN predicts that a molecule has high efficacy against a disease, researchers need to understand the structural components (subgraphs) responsible for this prediction to validate the finding and guide further research.
- In content moderation, relying on an unexplainable model to flag content or users can lead to accusations of bias and censorship, eroding user trust.

Without the ability to "look inside the box," we are left with a model that we cannot fully trust, debug, or improve. If the model makes an error, we have no clear path to diagnose the cause. Furthermore, we cannot be certain that the model has learned genuine, generalizable patterns from the data, rather than exploiting spurious correlations or biases present in the training set.

1.3 Motivation: The Need for Explainability and Trust in GNNs

The "black box" problem directly motivates the burgeoning field of Explainable Artificial Intelligence (XAI), and its application to GNNs [YYGJ22]. The primary goal of XAI for GNNs is to develop methods that can provide faithful, human-understandable explanations for a model's predictions. An explanation can take many forms, such as identifying a critical subgraph of nodes and edges [YBY+19], highlighting the most salient input features, or providing a counterfactual example (e.g., "the prediction would have been different if this edge did not exist") [LOG+22].

This thesis is motivated by the critical need for tools that bridge the gap between the power of GNNs and the human need for understanding. An effective GNN explainer can:

- **Build Trust:** By revealing the model's reasoning, it allows stakeholders to verify that the model is making decisions based on sensible criteria.
- **Facilitate Debugging:** When a model makes a mistake, explanations can pinpoint whether the error stems from faulty data, a flawed model architecture, or an incomplete feature set.
- **Enable Scientific Discovery:** In scientific applications, explanations can uncover novel patterns and hypotheses. For instance, an explanation for a protein classification might reveal a previously unknown functional motif.
- **Ensure Fairness and Compliance:** Explanations are essential for auditing models for biases and ensuring they comply with regulatory standards.

While several algorithms for GNN explanation have been proposed, the landscape of interactive tools for this purpose reveals a critical research gap. Existing visual analytics systems like GNNLens [KLM$^+$22] and CorGIE [HTQ$^+$21] are powerful but serve different primary goals. GNNLens focuses on developer-centric debugging by comparing GNN layer representations and performance metrics. CorgIE excels at helping users understand the relationship between a graph's topology and the learned embedding space. While both are interactive, their interactivity is primarily for exploration and inspection. A user can select nodes and see their properties, but they cannot easily perform experiments on the graph structure itself.

Crucially, these tools are built around correlational explanation methods. This project is motivated by the hypothesis that a deeper understanding requires moving beyond passive exploration and correlational views. True insight comes from asking "what if?" questions. This necessitates an interactive environment where a user can not only view explanations but also directly manipulate the graph and see the immediate impact on the model's predictions. Furthermore, it requires integrating explicitly causal explanation methods [JWZ$^+$23] to complement existing techniques. This project aims to fill this gap by creating a framework centered on user-driven intervention and integrated causal reasoning.

1.4 Project Aims and Objectives

The central aim of this thesis is to design, implement, and evaluate an interactive web-based tool for the visual explanation of Graph Neural Network predictions. To achieve this aim, the following objectives were established:

- **Implement and Train Standard GNN Architectures:** To build a foundation for explanation, train two widely-used GNN models, a Graph Convolutional Network

(GCN) [KW16] and a Graph Attention Network (GAT) [VCC+17], on standard benchmark citation network datasets (Cora and CiteSeer).

- **Integrate a Correlational Explanation Method:** Implement a well-established explainability method, GNNExplainer [YBY+19], to identify and visualize the most influential node features and subgraph structures that correlate with a given prediction.
- **Implement a Causal Explanation Method:** Go beyond correlation by developing a module for Causal Path Analysis (CPA-IV). This objective involves implementing an algorithm to identify and score causal pathways within the GNN's computation graph that are responsible for a node's classification.
- **Develop an Interactive Visualization Dashboard:** Build a user-friendly, web-based dashboard using Python's Dash framework. The dashboard must provide multiple, synchronized views, including the main graph, node embeddings, and panels for detailed explanation output.
- **Enable Dynamic Graph Manipulation:** Empower users to perform "what-if" analyses by implementing features to dynamically add or remove nodes and edges from the graph, with the system providing real-time feedback on how these perturbations affect model outputs and explanations.

1.5 Core Contributions

This thesis makes several key contributions to the field of Explainable AI for Graph Neural Networks:

- **A Novel Interactive Explainer Dashboard:** The primary contribution is a self-contained, interactive application that integrates multiple explanation paradigms into a single, cohesive user interface. Unlike static explanation reports, this tool provides a dynamic environment for hands-on exploration.
- **The Integration of Causal Path Analysis:** A significant contribution is the implementation and integration of the CPA-IV method as an explanation modality. This provides users with a causal perspective on the model's reasoning [JWZ+23], complementing the correlational insights from methods like GNNExplainer [YBY+19] and GAT attention. It allows users to understand the "flow of influence" through the graph that leads to a prediction.
- **A Multi-Faceted Explanation Framework:** The tool does not rely on a single explanation method. It synthesizes insights from GNNExplainer (feature and edge importance), GAT attention weights (for GAT models), and Causal Path Analysis. This multi-faceted approach gives the user a more holistic and robust understanding of the GNN's behavior.
- **A Platform for Live "What-If" Scenarios:** The ability to modify the graph structure and immediately observe the cascading effects on predictions and explanations

is a key contribution. This interactive feature transforms the user from a passive observer into an active participant in the discovery process, fostering deeper intuition and facilitating model debugging [WPH+19].

1.6 Thesis Outline

The remainder of this thesis is structured as follows:

- **Chapter 2: Background and Related Work** provides a detailed review of the foundational concepts, including graph theory, GNN architectures (GCN and GAT), and the landscape of Explainable AI, with a specific focus on existing methods for GNN explanation.
- **Chapter 3: System Design and Methodology** describes the overall architecture of the system. It details the datasets used, the implementation of the GNN models, the training process, and the algorithmic details of the GNNExplainer and Causal Path Analysis modules.
- **Chapter 4: Implementation of the Interactive Explainer** delves into the technical implementation of the web-based dashboard, discussing the technology stack (Dash, Plotly, Cytoscape) and the design of the user interface and its interactive components.
- **Chapter 5: Results and Case Studies** demonstrates the capabilities of the tool through a series of case studies on the Cora and CiteSeer datasets. It showcases how the different explanation methods provide complementary insights and highlights the value of the interactive features.
- **Chapter 6: Conclusion and Future Work** summarizes the key findings and contributions of this project, discusses its limitations, and proposes promising directions for future research.

Chapter 2

Background and Related Work

This chapter provides the theoretical underpinning for the thesis. It begins by establishing the fundamental concepts of graph theory and the evolution of learning on graphs, leading to the development of Graph Neural Networks (GNNs). It then delves into the specific architectures used in this project—Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT). Subsequently, the chapter shifts focus to the domain of Explainable Artificial Intelligence (XAI), outlining its core principles and surveying the landscape of GNN explanation methods. A critical distinction is drawn between correlational and causal explanations, and finally, the chapter reviews the state of interactive visualization tools, setting the stage for the methodology presented in Chapter 3.

2.1 Fundamentals of Graph Theory

A graph is a mathematical structure used to model pairwise relations between objects. Formally, a graph G is defined as an ordered pair $G = (V, E)$, where V is a set of nodes (or vertices) representing the objects, and E is a set of edges (or links) representing the relationships between these nodes. An edge $e = (u, v)$ connects node u to node v.

Graphs can be categorized based on the nature of their edges:

- **Undirected vs. Directed:** In an undirected graph, edges have no orientation; the relationship (u, v) is identical to (v, u). In a directed graph (or digraph), edges have a direction, meaning (u, v) is distinct from (v, u).
- **Unweighted vs. Weighted:** In an unweighted graph, all edges are considered equal. In a weighted graph, each edge is assigned a numerical weight, representing the strength, cost, or capacity of the relationship.

In many real-world applications, nodes also possess their own attributes or features. A graph where nodes have features is often denoted as $G = (V, E, X)$, where X is a matrix of node features.

A key concept for computation is the adjacency matrix $\mathbf{A}$ of a graph, a square matrix where the entry A_{ij} is non-zero if an edge exists from node i to node j, and zero otherwise. For an unweighted graph, $A_{ij} = 1$ if an edge exists. The degree of a node is the number of edges connected to it.

2.2 Graph Representation Learning

The primary challenge in applying machine learning to graphs is that they do not have a regular, grid-like structure like images or a simple sequential order like text. To bridge this gap, Graph Representation Learning aims to embed the components of a graph (typically nodes) into a low-dimensional vector space [Ham20]. The goal is to create node embeddings—dense vector representations—such that the geometric relationships between these vectors in the embedding space reflect the structural relationships in the original graph. For instance, nodes that are close in the graph (e.g., connected by an edge or a short path) should have similar vector representations.

Early methods for graph representation learning, such as `DeepWalk` and `node2vec`, were largely inspired by techniques from natural language processing like `Word2Vec`. They operate by generating random walks on the graph—sequences of nodes—and then using these walks as "sentences" to learn node embeddings. While effective, these methods are often transductive (unable to generate embeddings for unseen nodes without retraining) and do not directly incorporate node features.

2.3 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) have become the dominant and most powerful approach to graph representation learning [Ham20]. They address the limitations of earlier methods by providing an end-to-end framework that learns node representations by directly operating on the graph structure and incorporating node features.

2.3.1 Message Passing Neural Networks

Most modern GNNs can be understood through the lens of the Message Passing Neural Network (MPNN) framework. This framework describes a general process where nodes iteratively update their vector representations by exchanging information with their neighbors. A GNN model is typically composed of multiple message-passing layers. In each layer k, every node $u \in V$ performs two main operations:

- **AGGREGATE:** The node u collects feature vectors (or "messages") from its immediate neighbors, denoted by the set $\mathcal{N}(u)$. These messages are aggregated using a permutation-invariant function (e.g., sum, mean, or max) to create a single neighborhood vector, $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$.
- **UPDATE:** The node u updates its own representation from the previous layer, $\mathbf{h}_u^{(k-1)}$, by combining it with the aggregated neighborhood vector $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$. This combination is typically done using a neural network layer.

This process can be formalized with the following equations:

$$\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_v^{(k-1)} : v \in \mathcal{N}(u)\right\}\right) \tag{2.1}$$

$$\mathbf{h}_u^{(k)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right) \tag{2.2}$$

After K layers of message passing, the final node representation $\mathbf{h}_u^{(K)}$ captures structural information from its K-hop neighborhood. The initial representation $\mathbf{h}_u^{(0)}$ is simply the input feature vector of node u. The specific mathematical forms of the AGGREGATE and UPDATE functions define the particular GNN architecture.

2.4 GNN Architectures

This project focuses on two of the most influential GNN architectures: GCN and GAT.

2.4.1 Graph Convolutional Networks (GCN)

The Graph Convolutional Network (GCN) layer is a specific and efficient implementation of the message passing idea [KW16]. It simplifies the process by performing a weighted average of the feature vectors of a node and its neighbors. The layer-wise propagation rule for a GCN can be expressed as:

$$\mathbf{H}^{(k+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(k)}\mathbf{W}^{(k)}\right) \tag{2.3}$$

Where:

- $\mathbf{H}^{(k)}$ is the matrix of node representations at layer k.
- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops added (so a node includes its own features in the aggregation).
- $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$. The term $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ represents a symmetrically normalized adjacency matrix, which helps stabilize the learning process.
- $\mathbf{W}^{(k)}$ is a trainable weight matrix for layer k.
- σ is a non-linear activation function, such as ReLU.

In essence, a GCN layer updates a node's representation by taking the mean of its neighbors' representations from the previous layer (including itself), and then passing this result through a standard neural network layer. Its simplicity and strong performance have made it a popular baseline model.

2.4.2 Graph Attention Networks (GAT)

A key limitation of GCNs is that they assign equal importance to all neighbors during the aggregation step. The Graph Attention Network (GAT) addresses this by incorporating an attention mechanism, allowing nodes to learn to weigh the importance of their neighbors' messages [VCC+17].

In a GAT layer, the importance of a neighbor node j's message to a target node i is calculated by an attention coefficient, α_{ij}. These coefficients are computed for every pair of connected nodes using a shared, learnable attention mechanism. First, an un-normalized attention score e_{ij} is computed:

$$e_{ij} = a(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j) \tag{2.4}$$

Here, a is a small feed-forward neural network that computes the attention score. These scores are then normalized across all neighbors of node i using the softmax function:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(\text{LeakyReLU}(e_{ij}))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(e_{ik}))} \tag{2.5}$$

The final updated representation for node i, $\mathbf{h'}_i$, is a weighted sum of its neighbors' transformed features, where the weights are the learned attention coefficients:

$$\mathbf{h'}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right) \tag{2.6}$$

To stabilize the learning process and increase model capacity, GATs employ multi-head attention, where multiple independent attention mechanisms are run in parallel, and their resulting embeddings are concatenated or averaged. This ability to inspect the learned attention weights (α_{ij}) makes GATs partially interpretable by design.

2.5 Explainable Artificial Intelligence (XAI)

Explainable AI (XAI) is a subfield of artificial intelligence focused on developing methods and models that can explain their decisions to human users [HKPC18]. The need for XAI is driven by the increasing complexity of models and their deployment in critical systems. Key concepts in XAI include:

- **Post-hoc vs. Ante-hoc:** Post-hoc methods explain a model after it has been trained (e.g., `GNNExplainer`), while ante-hoc (or interpretable-by-design) models have a struc-

© 2025, *Indian Institute of Technology Delhi*

ture that is inherently understandable (e.g., linear regression, or GAT to some extent via its attention weights).

- **Local vs. Global:** Local explanations focus on a single prediction for a specific instance (e.g., "Why was this node classified this way?"). Global explanations aim to describe the overall behavior of the entire model. This thesis is focused on local explanations.

2.6 Explainability Methods for GNNs

The principles of XAI are actively being adapted for GNNs to address their "black box" nature [YYGJ22].

2.6.1 Instance-Level Explanation Methods

Instance-level (or local) methods are the most common form of GNN explanation. They aim to identify the parts of the input—specifically, a subset of nodes and edges forming a computational subgraph, and a subset of the input node features—that were most influential for a particular node's prediction.

2.6.2 A Review of GNNExplainer

`GNNExplainer` is a pioneering and influential post-hoc, instance-level explanation method [YBY+19]. Its core objective is to identify a compact and connected subgraph and a small subset of node features that are most critical to the model's prediction for a given node.

It frames this as an optimization problem. The explainer learns a "soft mask" over the graph's edges and node features. The goal is to learn the mask that maximizes the mutual information between the GNN's prediction on the original graph and its prediction on the masked (explained) graph. In simpler terms, it seeks to remove as many edges and features as possible while ensuring the GNN's prediction remains largely unchanged. The elements (edges and features) that are assigned high values in the learned mask are considered the most important parts of the explanation.

2.6.3 Causal vs. Correlational Explanations

A critical distinction in the pursuit of faithful explanations is between correlation and causation.

- **Correlational Explanations:** Most current XAI methods, including `GNNExplainer` [YBY+19] and GAT's attention mechanism [VCC+17], are correlational. They identify which input features are statistically associated with or "activated for" a given prediction. They answer the question: "What parts of the input did the model look at?" For instance, GAT attention shows which neighbors a node paid attention to, but not necessarily if that attention caused the final decision.
- **Causal Explanations:** A more rigorous form of explanation seeks to identify the true causal drivers of a prediction. Causal methods aim to answer the question: "What parts of the input, if they were different, would have changed the prediction?" [TLW+22]. This involves performing interventions on the input or the model and observing the outcome.

To illustrate, consider a GNN that has learned to identify spam accounts in a social network. It may learn that spam accounts often use a specific profile picture (e.g., a cartoon avatar) and are connected to other known spam accounts. A correlational explainer like `GNNExplainer` might highlight both the avatar feature and the connections as important. However, a causal explainer could reveal, through intervention, that removing the connections to other spam accounts would flip the prediction to "not spam," while changing the profile picture would not. In this case, the connections are the causal drivers, while the avatar is merely a correlated feature.

The pursuit of causal explanations is a significant frontier in XAI, as they provide a more robust and reliable understanding of a model's behavior [JWZ+23]. This distinction directly motivates the inclusion of Causal Path Analysis in this thesis, as a method to move beyond the purely correlational insights offered by `GNNExplainer`.

2.7 Interactive Visualization for GNN Explainability

The final pillar of related work concerns the use of interactive visualization as a medium for explainability. While an explanation algorithm produces a result (e.g., a subgraph or a feature list), an interactive system is required to present this result to a human user in an effective and comprehensible manner [HKPC18].

2.7.1 A Review of Existing Interactive Tools

The field of visual analytics for GNNs has produced several notable systems, each with a different focus. Reviewing two of the most prominent, GNNLens and Corgi, helps to contextualize the specific contributions of this thesis.

- **GNNLens:** This is a powerful visual analytics system designed primarily for GNN developers and researchers [KLM+22]. Its main strength lies in model debugging and error diagnosis. GNNLens provides a suite of linked views that allow a user to inspect and compare the internal representations (embeddings) of different GNN layers. For example, a developer can use it to see how a node's representation evolves as it passes through the network or to compare the GNN's final embedding with that of a simpler model (like an MLP) to understand if the graph structure is helping or hurting performance. Its interactivity is centered on this deep, multi-layered inspection.

- **Corgi:** This tool focuses on a different but related problem: understanding the relationship between the graph's original topological structure and the high-dimensional geometry of the learned embedding space [HTQ$^+$21]. Corgi provides linked views of a node's local neighborhood in the graph and its position relative to its neighbors in a 2D projection of the embedding space. Its primary goal is to help a user evaluate the quality of the learned representations—for instance, by seeing if nodes that have similar structural roles in the graph are also clustered together in the embedding space.

2.7.2 Identifying the Research Gap

While GNNLens and Corgi are excellent tools for their respective purposes, their analysis reveals a critical research gap that this project aims to fill.

1. **Focus on Observation, Not Intervention:** The interactivity in these existing systems is primarily observational or exploratory. A user can select, filter, and inspect different parts of the model and data. However, they are not designed for direct, experimental intervention. A user cannot easily ask "what if?" by modifying the graph structure (e.g., deleting a suspected spurious edge) and immediately observing the consequences on the model's prediction and explanation [WPH$^+$19].

2. **Reliance on Correlational Explanations:** The explanations provided by or integrated into these tools are fundamentally correlational (e.g., GAT attention, feature importance). They show what the model is looking at but do not provide a deeper, causal narrative about the flow of influence.

This thesis addresses this gap directly. My interactive dashboard is built on the principle of **interventional analysis**, making graph editing a core feature of the user experience. This transforms the user from a passive observer into an active experimenter. Furthermore, by integrating my novel **Causal Path Analysis (CPA-IV)** method alongside the correlational GNNExplainer, my system provides a more holistic, multi-faceted explanatory view than is available in existing tools. It is this unique combination of user-driven intervention and integrated causal reasoning that defines the primary contribution of my work.

Chapter 3

System Design and Methodology

This chapter details the technical foundation of the project, outlining the systematic approach I took to build and evaluate the interactive GNN explanation framework. It begins with a high-level overview of the system's architecture, followed by a description of the datasets and the GNN models implemented. The core of the chapter focuses on the implementation details of the explanation frameworks, providing a thorough walkthrough of the established correlational methods and the novel Causal Path Analysis (CPA-IV) algorithm introduced in this work.

3.1 System Architecture: An Overview

The system is designed as a modular pipeline that separates the concerns of model training, data serving, and interactive explanation. This architecture ensures that the computationally intensive task of training is performed offline, while the interactive application remains lightweight and responsive. The overall workflow, illustrated conceptually in Figure 3.1, consists of two main phases: an offline training phase and an online interactive explanation phase.

3.1.1 The Two-Stage Workflow: Offline Training and Online Explanation

Offline Training Phase:

- **Data Preprocessing:** Standard benchmark datasets (Cora, CiteSeer) are loaded using the `torch_geometric.datasets.Planetoid` library.
- **Model Training:** As detailed in the `train_models.py` script, GCN and GAT models are trained on the preprocessed datasets. The script employs an early stopping mechanism based on validation accuracy to prevent overfitting.
- **Model Serialization:** Upon completion of training, the best-performing model's state dictionary (`state_dict`) and its initialization arguments (e.g., number of hidden channels, attention heads) are packaged together and serialized to a `.pkl` file using Python's `pickle` library. This creates a self-contained, pre-trained model artifact for each model-dataset pair (e.g., `GCN_Cora.pkl`).

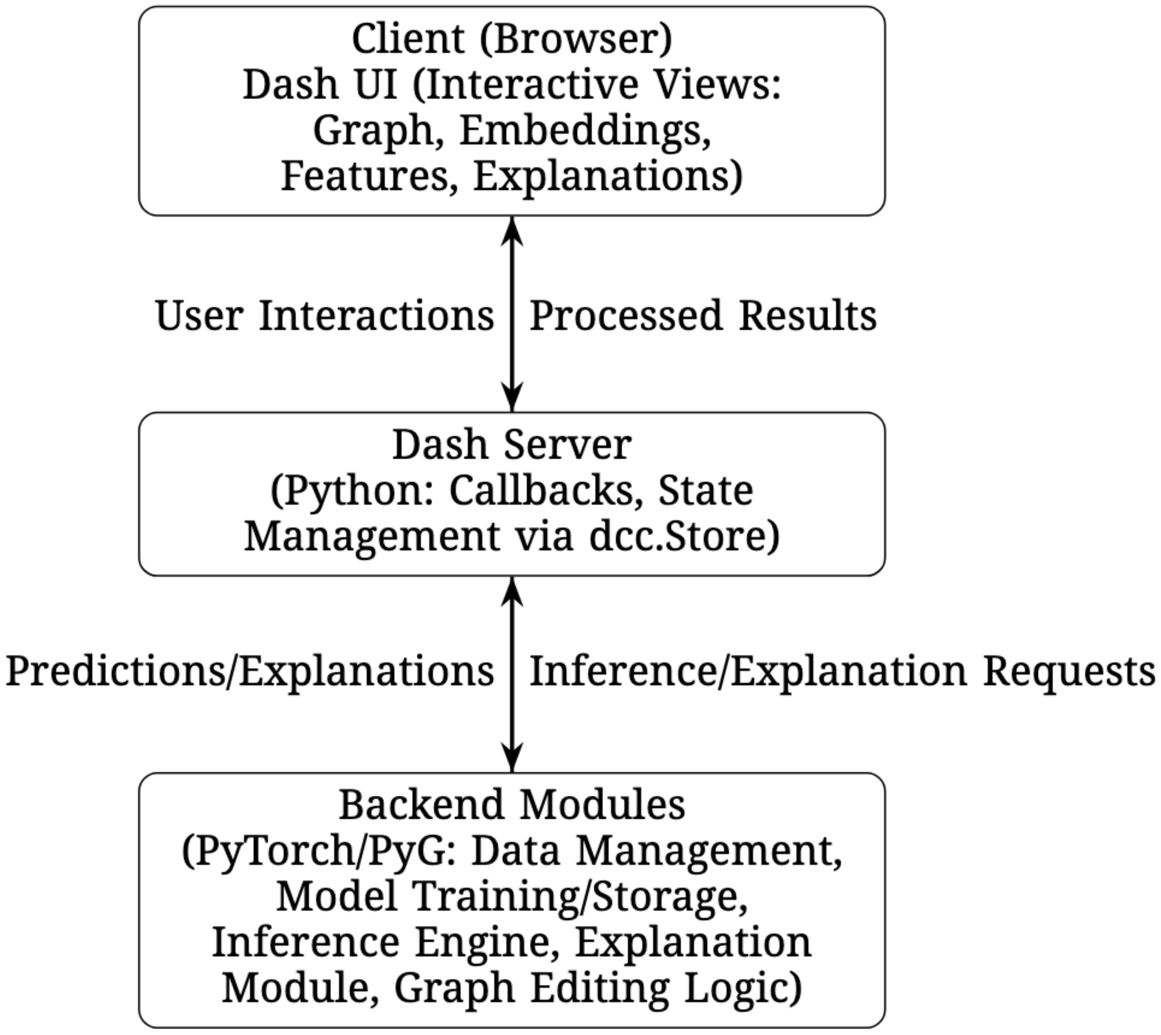

Client (Browser)
Dash UI (Interactive Views: Graph, Embeddings, Features, Explanations)
User Interactions
Processed Results
Dash Server
(Python: Callbacks, State Management via dcc.Store)
Predictions/Explanations
Inference/Explanation Requests
Backend Modules
(PyTorch/PyG: Data Management, Model Training/Storage, Inference Engine, Explanation Module, Graph Editing Logic)

Figure 3.1: A high-level overview of the two-stage system architecture.

Online Interactive Phase (The Dash Application):

- **User Request:** The user selects a dataset and a model from the dashboard's UI.
- **Model Loading:** The application backend (`app.py`) loads the corresponding serialized `.pkl` file. It reconstructs the model architecture using the saved initialization arguments and loads the trained weights from the state dictionary.
- **Inference and Visualization:** The loaded model performs inference on the full graph to generate predictions and node embeddings. These are used to render the initial graph view and embeddings plot.
- **Explanation Generation:** When a user selects a node, the backend triggers the explanation modules (`GNNExplainer`, CPA-IV, GAT attention) for that specific node.
- **Interactive Feedback Loop:** The results of the explanation are sent to the frontend for visualization. Any user-driven modifications to the graph (e.g., adding/removing an edge) trigger a re-inference and explanation cycle, providing real-time feedback.

3.1.2 Technology Stack

The implementation of this system relies on a carefully selected stack of open-source Python libraries:

- **PyTorch & PyTorch Geometric (PyG):** The core deep learning framework and its graph-specific extension, used for model definition, training, and running explanation algorithms.
- **Dash by Plotly:** The Python framework used to build the entire interactive web application without writing HTML or JavaScript.
- **Plotly:** The visualization library used for creating the interactive scatter plots and bar charts in the dashboard.
- **Dash Cytoscape:** A specialized Dash component for rendering and interacting with the main graph visualization.
- **NetworkX:** A standard library for graph analysis, used within the CPA-IV algorithm for its efficient path-finding capabilities.

3.2 GNN Model Implementation (`train_models.py`)

3.2.1 Graph Convolutional Network (GCNNet): Architecture and Layer Definition

The `GCNNet` is a classic two-layer Graph Convolutional Network [KW16]. The first `GCNConv` layer maps the input features to a hidden dimension, followed by a `ReLU` activation and dropout. The second `GCNConv` layer maps the hidden representations to the number of output classes.

3.2.2 Graph Attention Network (GATNet): Architecture and Attention Heads

The `GATNet` also uses a two-layer structure but replaces the GCN layers with the more expressive `GATConv` layers [VCC+17]. The first layer employs multi-head attention (8 heads), and its outputs are passed through an `ELU` activation. The second layer uses a single attention head to compute the final class logits.

3.3 Dataset and Preprocessing

3.3.1 Benchmark Datasets: Cora and CiteSeer

This project utilizes two standard citation network datasets to demonstrate and evaluate the explanation methods. Their well-understood structure and community properties provide an intuitive basis for evaluating graph-based explanations.

- **Cora:** This dataset consists of 2,708 publications classified into one of seven classes, with 5,429 citation links.
- **CiteSeer:** This dataset is larger and more sparsely connected, containing 3,327 publications, 4,732 citation links, and six class labels.

3.3.2 Data Loading and Preprocessing Pipeline

The data loading logic is encapsulated in the `preprocess_datasets.py` script. It uses the `torch_geometric.datasets.Planetoid` class, which automatically handles the downloading and processing of these datasets. The script processes each dataset once and saves the resulting PyG `Data` object, containing features, edge index, labels, and training masks, to a serialized `.pt` file for fast, repeated access.

3.4 Model Training and Serialization

3.4.1 The Training Loop: Optimization and Loss Function (NLL-Loss)

The training loop in `train_models.py` follows a standard supervised protocol. It uses the `Adam` optimizer and the Negative Log Likelihood Loss (`NLLLoss`), which is appropriate for the `log_softmax` output of the models. The loss is computed only on the nodes specified by the training mask.

3.4.2 Hyperparameter Tuning and Early Stopping

To ensure robust performance, hyperparameters such as learning rate and weight decay were optimized for each model-dataset pair. An early stopping mechanism with a patience of 20 epochs was used to prevent overfitting, saving the model state that achieved the highest accuracy on the validation set.

3.4.3 Saving Models for the Interactive Application (`.pkl` format)

The final step of the offline phase is serialization. The best model's learned weights (`state_dict`) and its architectural hyperparameters (`init_args`) are packaged into a tuple and saved to a `.pkl` file using Python's `pickle` library.

3.5 Implementation of Explanation Methods

3.5.1 Integrating GNNExplainer: The `run_gnn_explainer_callback`

The project leverages the `torch_geometric.explain.Explainer` class with the `GNNExplainer` algorithm [YBY+19]. When a node is selected, this callback configures the explainer for the node classification task and learns the edge and feature masks that are most important for the model's prediction. These masks are then used to power the visualizations in the dashboard.

3.5.2 Developing the Causal Path Analysis (CPA-IV) Method

This novel method is grounded in the principle of causal intervention. It measures the importance of a path by simulating its removal from the computation graph and quantifying the negative impact on the model's prediction confidence. This approach aligns with research into counterfactual and interventional explanations [GAH+23, WPH+19]. The `run_cpa_iv` function, detailed in Algorithm 1, executes this logic step-by-step.

Algorithm Step-by-Step: Subgraph Extraction, Path Enumeration, Perturbation, and Scoring

The algorithm executes a precise sequence of operations:

1. **Subgraph Extraction:** It first isolates the k-hop neighborhood of the target node to create a localized computation graph.
2. **Baseline Score:** It computes the model's prediction probability for the original class on this unaltered subgraph. This score serves as the baseline for measuring causal effects.
3. **Path Enumeration:** It then identifies all simple paths up to a specified length that terminate at the target node within the subgraph.
4. **Perturbation and Scoring:** In the core loop, it iterates through each path. For every path, it creates a perturbed subgraph by removing the path's edges. It re-runs

the model on this perturbed graph and calculates the causal effect score as the drop in the baseline probability.

5. **Ranking:** Finally, it ranks all enumerated paths by their calculated causal effect score and returns the top paths as the explanation.

Algorithm 1 Causal Path Analysis - Interventional Variant (CPA-IV)

```
1: Input:
2:     $M$: A trained GNN model.
3:     $G = (V, E, X)$: The full input graph with nodes, edges, and features.
4:     $n_{target}$: The target node for which to generate an explanation.
5:     $C_{true}$: The predicted class of $n_{target}$ for which to explain.
6:     $k$: The neighborhood size for subgraph extraction (e.g., 2 for a 2-hop neighborhood).
7:     $L$: The maximum length of paths to consider (e.g., 2).
8:     $T$: The number of top paths to return (e.g., 3).

9: Output:
10:    $TopPaths$: A list of the top $T$ most influential causal paths.

11: procedure RunCPAIV($M, G, n_{target}, C_{true}, k, L, T$)
12:     $CausalScores \leftarrow []$                         ▷ Initialize an empty list to store path scores
                                                             ▷ Step 1: Subgraph Extraction
13:     $V_{sub}, E_{sub} \leftarrow$ k_hop_subgraph($n_{target}, k, G$)
14:     $G_{sub} \leftarrow (V_{sub}, E_{sub}, X_{sub})$     ▷ Create the computation subgraph
                                                             ▷ Step 2: Baseline Score Calculation
15:     $P_{original} \leftarrow M(G_{sub})$                 ▷ Run model on the original subgraph
16:     $S_{baseline} \leftarrow P_{original}[n_{target}, C_{true}]$   ▷ Get probability for the target class
                                                             ▷ Step 3: Path Enumeration
17:     $AllPaths \leftarrow$ all_simple_paths($G_{sub}$, source=any, target=$n_{target}$, max_length=$L$)
                                                             ▷ Step 4: Perturbation and Scoring Loop
18:     for each $path$ in $AllPaths$ do
19:         $E_{perturbed} \leftarrow E_{sub} \setminus$ edges_in_path($path$)   ▷ Remove path's edges
20:         $G_{perturbed} \leftarrow (V_{sub}, E_{perturbed}, X_{sub})$        ▷ Create perturbed subgraph

21:         $P_{perturbed} \leftarrow M(G_{perturbed})$     ▷ Re-run inference on perturbed graph
22:         $S_{perturbed} \leftarrow P_{perturbed}[n_{target}, C_{true}]$      ▷ Get new probability

23:         $score_{causal} \leftarrow S_{baseline} - S_{perturbed}$           ▷ Calculate the causal effect
24:         add ($path, score_{causal}$) to $CausalScores$
25:     end for
                                                             ▷ Step 5: Ranking and Selection
26:     Sort $CausalScores$ in descending order by $score_{causal}$
27:     $TopPaths \leftarrow$ first $T$ elements of $CausalScores$

28:     return $TopPaths$
29: end procedure
```

Chapter 4

Implementation of the Interactive Explainer

While Chapter 3 laid out the theoretical methodology, this chapter delves into the engineering and implementation details of the project's centerpiece: the interactive explainer dashboard. The successful realization of this project required not only the implementation of GNN models and explanation algorithms but also the creation of a robust, intuitive, and highly interactive user interface to make these complex concepts accessible. This chapter details the technology stack chosen to meet these demands, describes the architecture and design philosophy of the user interface in comparison to existing tools, and provides a deep dive into the core components and the reactive callback system that brings the application to life. The implementation is primarily captured in the `app.py` script.

4.1 Technology Stack

The selection of an appropriate technology stack was critical to balancing the need for powerful backend computation with the demand for a responsive and feature-rich frontend. A Python-centric stack was chosen to maintain a seamless environment between the machine learning research and the web application development.

4.1.1 Backend: PyTorch Geometric, Dash

The backend is responsible for all heavy-lifting tasks: loading data, running GNN model inference, executing explanation algorithms, and managing the application state.

- **PyTorch Geometric (PyG):** As the GNN models and explanation methods are fundamentally rooted in graph structures, `PyTorch Geometric` was the natural choice. It provides optimized GNN layers (`GCNConv`, `GATConv`), a powerful `Explainer` framework that includes `GNNExplainer` [YBY$^+$19], and crucial graph utilities like `k_hop_subgraph`.
- **Dash by Plotly:** `Dash` was selected as the core web framework. It allows for the construction of complex, interactive web applications purely in Python, enabling a tight integration with the `PyTorch Geometric` backend without requiring a context switch to other languages.

4.1.2 Frontend: Plotly, Dash Cytoscape

The frontend is what the user sees and interacts with. It is built using a collection of `Dash` components.

- **Plotly:** This scientific graphing library powers the `dcc.Graph` component and was used for all standard charts, such as the node embeddings scatter plot and the feature importance bar charts.
- **Dash Cytoscape:** This specialized wrapper for `Cytoscape.js` is the core of the Graph View. Its rich styling and event-handling capabilities are essential for visualizing the graph and capturing user interactions.

4.2 User Interface and Dashboard Layout

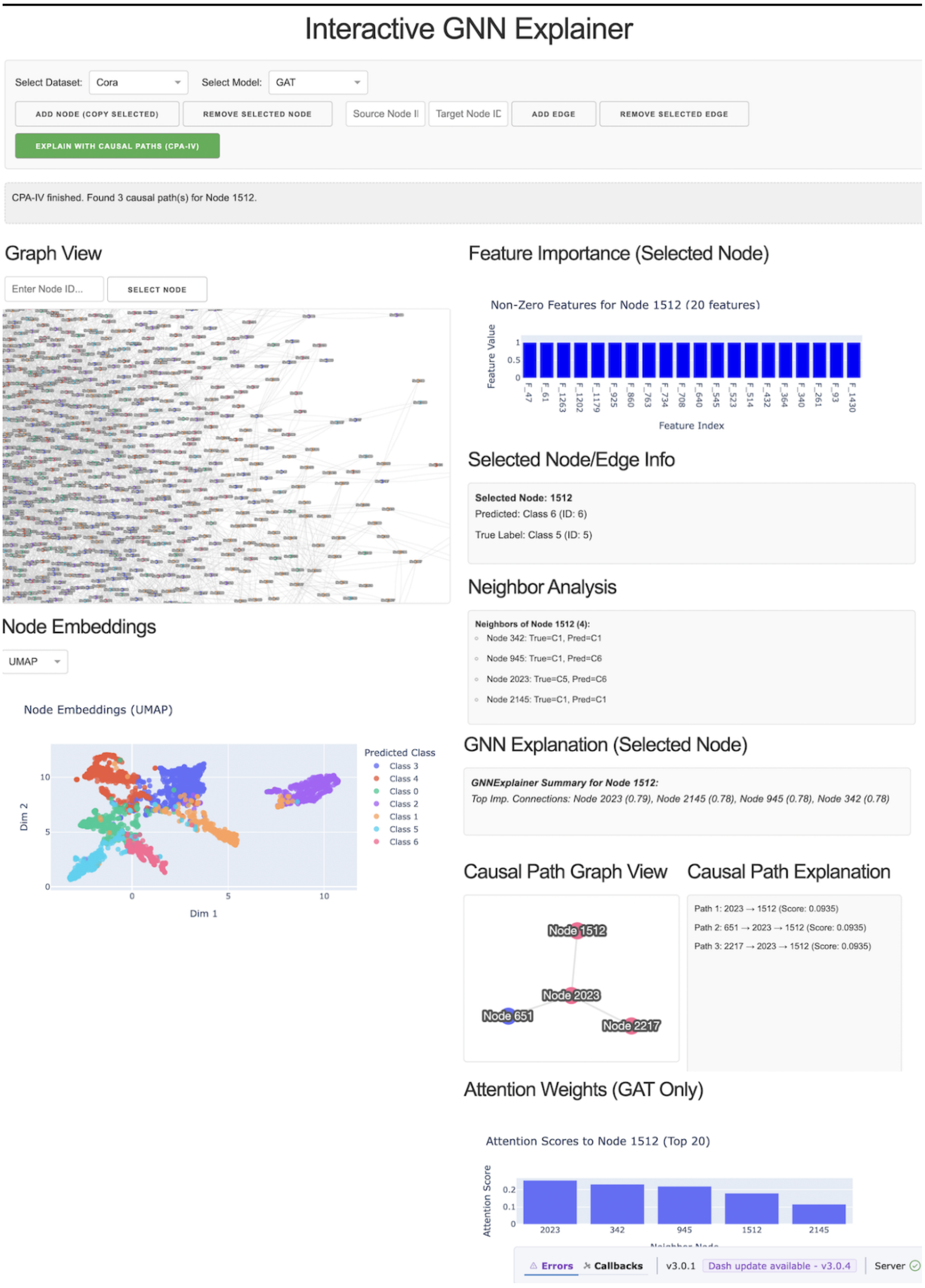
The user interface was designed to present a high density of interconnected information in a clear and logical manner. The layout is structured to facilitate a natural workflow, guiding the user from high-level exploration to detailed, node-specific explanation.

The overall structure, defined within the `app.layout` object and shown in Figure 4.1, is divided into three main areas:

1. **Top Control Panel:** Contains the global controls for selecting the `Dataset` and `Model`, as well as buttons for direct graph manipulation.
2. **Status Message Area:** A dedicated `html.Div` provides real-time feedback to the user (e.g., "Loading dataset...", "Inference complete.").
3. **Main Two-Column View:** The core layout, designed to facilitate a comparative workflow.
    - *Left Column:* Dedicated to holistic, visual representations of the graph. It contains the primary Graph View and the Node Embeddings View.
    - *Right Column:* Dedicated to displaying detailed, node-specific information. It is a scrollable area containing panels for selected node info, feature importance, neighborhood analysis, and the outputs of the different explanation methods.

4.2.1 Design Philosophy and Comparison to Existing Tools

The design of this dashboard is deliberately different from existing visual analytics tools like GNNLens [KLM^{+}22] and Corgi [HTQ^{+}21]. While those systems are powerful, they are primarily designed for *observational analysis*. GNNLens excels at allowing developers to inspect and debug layer-wise representations, and Corgi is designed to help users understand the mapping between graph topology and the embedding space. Their interactivity is geared towards selection and inspection.



Interactive GNN Explainer

Select Dataset: Cora
Select Model: GAT

ADD NODE (COPY SELECTED) | REMOVE SELECTED NODE | Source Node II | Target Node IC | ADD EDGE | REMOVE SELECTED EDGE

EXPLAIN WITH CAUSAL PATHS (CPA-IV)

CPA-IV finished. Found 3 causal path(s) for Node 1512.

Graph View

Enter Node ID... | SELECT NODE

Feature Importance (Selected Node)

Selected Node/Edge Info

Selected Node: 1512
Predicted: Class 6 (ID: 6)
True Label: Class 5 (ID: 5)

Node Embeddings

UMAP

Neighbor Analysis

Neighbors of Node 1512 (4):
- Node 342: True=C1, Pred=C1
- Node 945: True=C1, Pred=C6
- Node 2023: True=C5, Pred=C6
- Node 2145: True=C1, Pred=C1

GNN Explanation (Selected Node)

GNNExplainer Summary for Node 1512:
Top Imp. Connections: Node 2023 (0.79), Node 2145 (0.78), Node 945 (0.78), Node 342 (0.78)

Causal Path Graph View

Causal Path Explanation

Path 1: 2023 → 1512 (Score: 0.0935)
Path 2: 651 → 2023 → 1512 (Score: 0.0935)
Path 3: 2217 → 2023 → 1512 (Score: 0.0935)

Attention Weights (GAT Only)

Figure 4.1: The main user interface of the interactive explainer.

My design philosophy is centered on enabling *interventional analysis* [WPH+19]. The two-column layout is not just for organization; it is specifically structured to support an "observe-hypothesize-intervene" workflow. A user can observe the graph and explanations on both sides, form a hypothesis about a specific node or edge, use the control panel to perform an intervention (e.g., delete the edge), and immediately see the consequences reflected in both the graph visualizations on the left and the detailed outputs on the right. This focus on a direct and responsive feedback loop for user-driven experiments is the key differentiator of my system.

Furthermore, by integrating CPA-IV, my tool provides a causal dimension of explanation not present in these other systems [JWZ+23]. The dashboard is therefore not just a tool for seeing what a model *did*, but a workbench for understanding *how* its reasoning is structured and *what would happen* if its inputs were to change.

4.3 Core Components

This section details the implementation of the three most critical visual components of the dashboard.

4.3.1 The Graph View

The Graph View, an instance of `cyto.Cytoscape`, serves as the user's primary window into the dataset. A key aspect of its implementation is the use of a shared `SHARED_STYLESHEET` to visually encode information:

- **Node Color** is mapped to the model's predicted class.
- **Selection and Neighborhood** are shown with a prominent orange color and dashed blue borders, respectively.
- **Explanation Visualization:** Important edges from `GNNExplainer` are colored green, while causal paths from `CPA-IV` are rendered with distinct colors and shapes to clearly visualize the flow of influence.

The layout is managed by the `cose` (Compound Spring Embedder) algorithm, which is well-suited for revealing the underlying community structure of the graph.

4.3.2 The Node Embeddings View

This `dcc.Graph` component provides a view into the latent space learned by the GNN. The implementation allows the user to switch between `t-SNE`, `PCA`, and `UMAP` for dimensionality reduction. The plot is bidirectionally linked with the rest of the app; clicking a point selects the node throughout the dashboard, and vice-versa.

4.3.3 The Explanation and Information Panels

The right-hand column provides a multi-faceted explanation for a selected node through several components, including a feature histogram that highlights important features in red, a neighborhood analysis panel, and dedicated output areas for GAT Attention, `GNNExplainer`, and `CPA-IV`.

4.4 Interactive Functionality

The dashboard's interactivity is orchestrated by a network of `Dash` callbacks.

4.4.1 Node and Edge Selection

A "multi-trigger" callback listens for tap events from the graph and embedding views. It identifies the selected node/edge and stores its ID in a central `dcc.Store` component. This use of a central store is a key design choice that decouples components and allows multiple other callbacks to react to a change in selection state without creating complex dependencies.

4.4.2 Dynamic Graph Manipulation

The "what-if" analysis is implemented via callbacks tied to the "Add/Remove" buttons. When a user removes an edge, the callback modifies the graph data in `current-graph-store`. This update has a cascading effect, automatically triggering the inference callback, which in turn updates the model output store, causing the entire visualization to re-render with new predictions and explanations. This provides the immediate visual feedback that is central to the tool's design philosophy and aligns with work on interventional and counterfactual explanation systems [WPH+19, GAH+23].

4.4.3 Backend Callback Architecture for Real-Time Updates

The application's reactive nature is built upon a carefully designed chain of callbacks mediated by `dcc.Store` components. This stateful, multi-stage architecture ensures that expensive computations (like model inference or explanation generation) are only triggered when their specific inputs change, enabling a seamless and interactive user experience. It is this architecture that transforms a series of static Python scripts into a dynamic and powerful tool for research and analysis.

Chapter 5

Results and Case Studies

This chapter presents the empirical heart of my research, transitioning from the theoretical and implementation details of the preceding chapters to a practical demonstration of the framework's analytical power. The primary objective here is not to push the boundaries of predictive accuracy but to use the interactive dashboard as an analytical workbench to dissect the internal logic of trained GNNs. My goal is to demonstrate the qualitative insights offered by each explanation method and to underscore the unique value of an interactive, intervention-based approach to understanding model behavior.

The chapter begins by outlining the experimental setup. It then proceeds through a series of detailed case studies designed to simulate a methodical, inquiry-driven workflow within the dashboard. These case studies will explore GCN and GAT models, providing a deep, comparative analysis of GNNExplainer, my CPA-IV method, and intrinsic GAT attention. Finally, and most critically, I showcase the power of direct interaction by performing a "what-if" scenario, manipulating the graph structure to validate an explanatory hypothesis and debug a model prediction.

5.1 Experimental Setup

All experiments described in this chapter were conducted using the models and dashboard implementation detailed in Chapters 3 and 4. I selected the Cora and CiteSeer datasets to provide a diverse set of analytical challenges.

- **Datasets:** My analysis relies on two canonical citation network benchmarks:
  - *Cora:* This well-studied dataset contains 2,708 scientific publications (nodes) classified into one of seven subjects. The 5,429 citation links (edges) form the relational structure. Each node is accompanied by a 1,433-dimensional binary feature vector representing its vocabulary.
  - *CiteSeer:* A larger and often more challenging dataset, CiteSeer consists of 3,327 publications and 4,732 citations, with nodes categorized into one of six classes. Each publication has a sparse 3,703-dimensional feature vector.
- **Trained Model Performance:** A prerequisite for any meaningful explanation is a model that has learned effectively. Explaining a random or poorly performing model yields little insight. I therefore ensured my models were well-trained, achieving performance consistent with published literature. The final test accuracies on the standard data splits are shown in Table 5.1.

Table 5.1: Final test accuracies of the GNN models analyzed in this chapter.

Dataset	Model	Test Accuracy
Cora	GCN	81 %
CiteSeer	GAT	70 %

The strong performance of these models confirms they have learned complex patterns from the data, making them worthy and interesting subjects for an in-depth explanatory analysis.

5.2 Qualitative Analysis through Case Studies

The following sections present a series of case studies conducted using the interactive dashboard. The goal of these studies is to demonstrate how the features of my system enable a unique form of analysis that complements existing tools.

5.2.1 Framing the Analysis in Context of GNNLens and Corgi

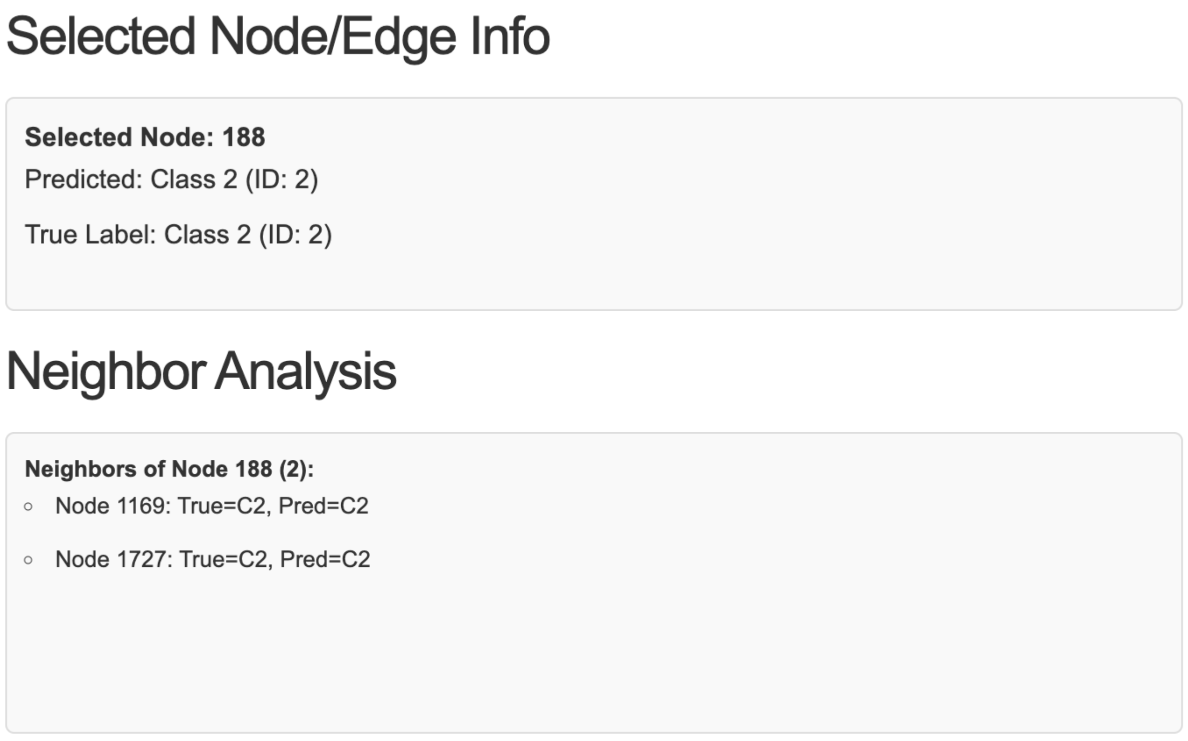
As discussed in Chapter 2, visual analytics tools like GNNLens [KLM+22] and Corgi [HTQ+21] are powerful platforms for GNN analysis. GNNLens provides deep insights into the internal layer-by-layer workings of a model, making it invaluable for developer-centric debugging. Corgi excels at helping users understand the mapping between graph topology and the learned embedding space.

The case studies presented here are designed to showcase a different, yet equally critical, mode of analysis: user-driven interventional debugging and comparative causal reasoning. While GNNLens and Corgi are primarily observational tools, my framework is an experimental workbench. The following scenarios will focus specifically on how the ability to (1) directly manipulate the graph and (2) compare correlational (GNNExplainer) and causal (CPA-IV) explanations allows a user to answer questions that are difficult to address with existing systems.

5.2.2 Case Study 1: Explaining a GCN Prediction on the Cora Dataset

My first case study focuses on a standard scenario: using the explanation tools to understand why a GCN model made a specific, correct prediction. The objective is to compare the surface-level correlational story provided by `GNNExplainer` with the deeper, structural narrative uncovered by my Causal Path Analysis (CPA-IV) method, even in a seemingly simple case.

Scenario: I begin by loading the GCN model trained on the Cora dataset. For my analysis, I select a target node, **Node 188**, which the model has correctly classified as belonging to **Class 2**. The "Neighbor Analysis" panel reveals that Node 188 has a simple and strongly homophilous neighborhood, consisting of only two neighbors: **Node 1169** and **Node 1727**, both of which are also correctly classified as Class 2. This appears to be a straightforward case, making it a perfect test for the nuanced insights our tool can provide.


Selected Node/Edge Info

Selected Node: 188
Predicted: Class 2 (ID: 2)
True Label: Class 2 (ID: 2)

Neighbor Analysis

Neighbors of Node 188 (2):
- Node 1169: True=C2, Pred=C2
- Node 1727: True=C2, Pred=C2

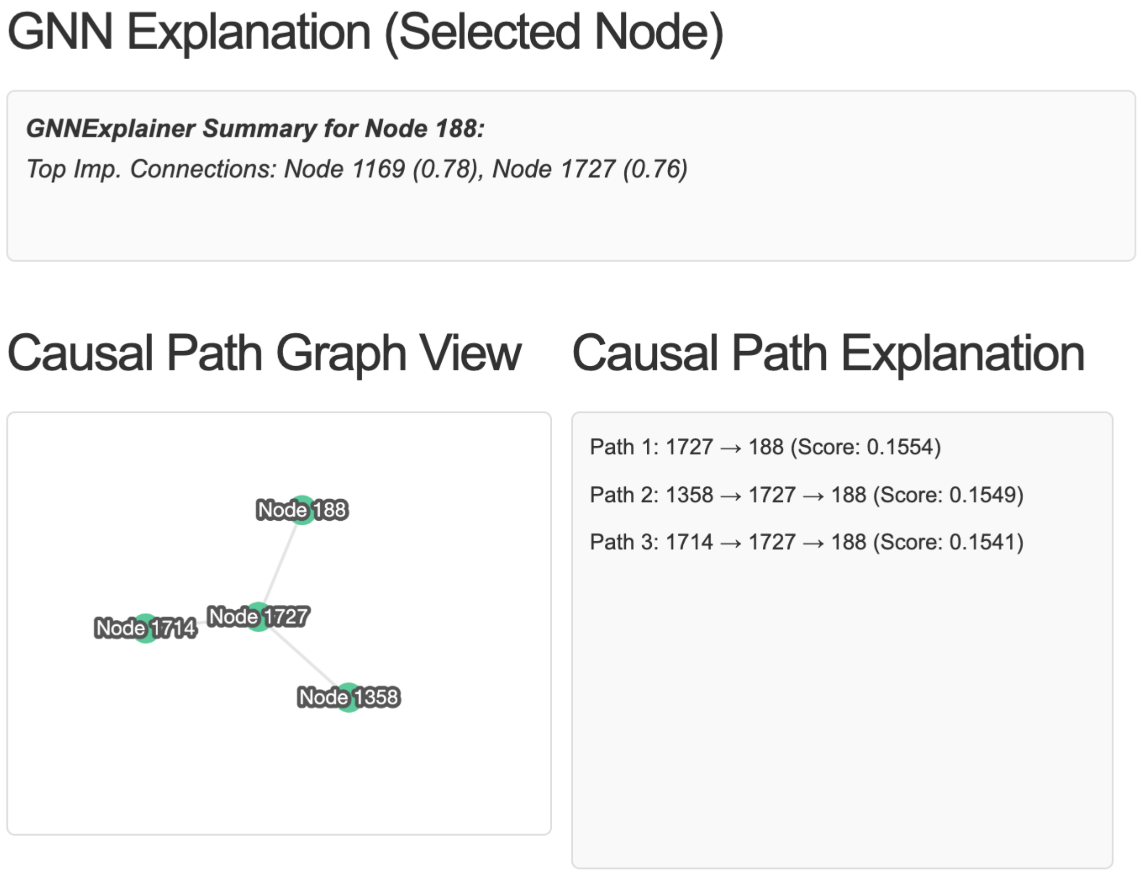
Figure 5.1: Initial state for Node 188, along with its two neighbors, both of the same class. The 'Selected Node Info' panel confirms the correct prediction.

GNNExplainer Results

My first action is to generate a baseline explanation using `GNNExplainer` [YBY+19]. This method identifies the most influential components of the computation subgraph that are correlated with the prediction. The dashboard shows the following:

- **Subgraph Importance:** `GNNExplainer` assigns high and nearly equal importance scores to both connections: the edge to Node 1169 (score: 0.78) and the edge to Node 1727 (score: 0.76).

Interpretation: `GNNExplainer` provides a clear, correlational narrative. It suggests the model's decision is based on the strong, unambiguous evidence from its two neighbors, which both belong to the correct class. This explanation is plausible and confirms the model has learned to leverage homophily. However, it treats the influence of both neighbors as roughly equivalent, providing little insight into the structural dynamics of the information flow. It answers "what" the model looked at (the neighbors), but not "how" it processed their influence.


GNN Explanation (Selected Node)

GNNExplainer Summary for Node 188:
Top Imp. Connections: Node 1169 (0.78), Node 1727 (0.76)

Causal Path Graph View

Causal Path Explanation

Path 1: 1727 → 188 (Score: 0.1554)
Path 2: 1358 → 1727 → 188 (Score: 0.1549)
Path 3: 1714 → 1727 → 188 (Score: 0.1541)

Figure 5.2: Highlighting the GNN Explanation results for Node 188.

Causal Path Analysis Results

To probe deeper into the structural reasoning, I engage my CPA-IV method. After the analysis completes, the "Causal Path Explanation" panel displays a more nuanced ranking of influential pathways:

1. Path 1: Node 1727 → Node 188 (Score: 0.1554)
2. Path 2: Node 1358 → Node 1727 → Node 188 (Score: 0.1549)
3. Path 3: Node 1714 → Node 1727 → Node 188 (Score: 0.1541)

Interpretation: This causal analysis provides a much more sophisticated explanation. While `GNNExplainer` saw both neighbors as equally important, CPA-IV reveals a clear hierarchy of influence. **Node 1727 acts as the primary causal gateway.**

The top-ranked causal path is the direct 1-hop connection from Node 1727. More importantly, the second and third most influential paths are 2-hop paths that *also* channel their influence through Node 1727. This indicates that the model's reasoning is heavily dependent not just on Node 1727 itself, but on the information it relays from other nodes in the graph (like 1358 and 1714).

In contrast, the other neighbor, Node 1169, does not appear in any of the top causal paths. This suggests that while its presence is correlated with the correct prediction, its structural role in the flow of causal influence is minimal compared to Node 1727. CPA-IV has successfully distinguished between two seemingly identical neighbors, identifying one as a simple supporter and the other as a crucial information hub. This uncovers a structural preference in the GCN's reasoning that a purely correlational method would miss.

5.2.3 Case Study 2: Explaining a GAT Misclassification on the CiteSeer Dataset

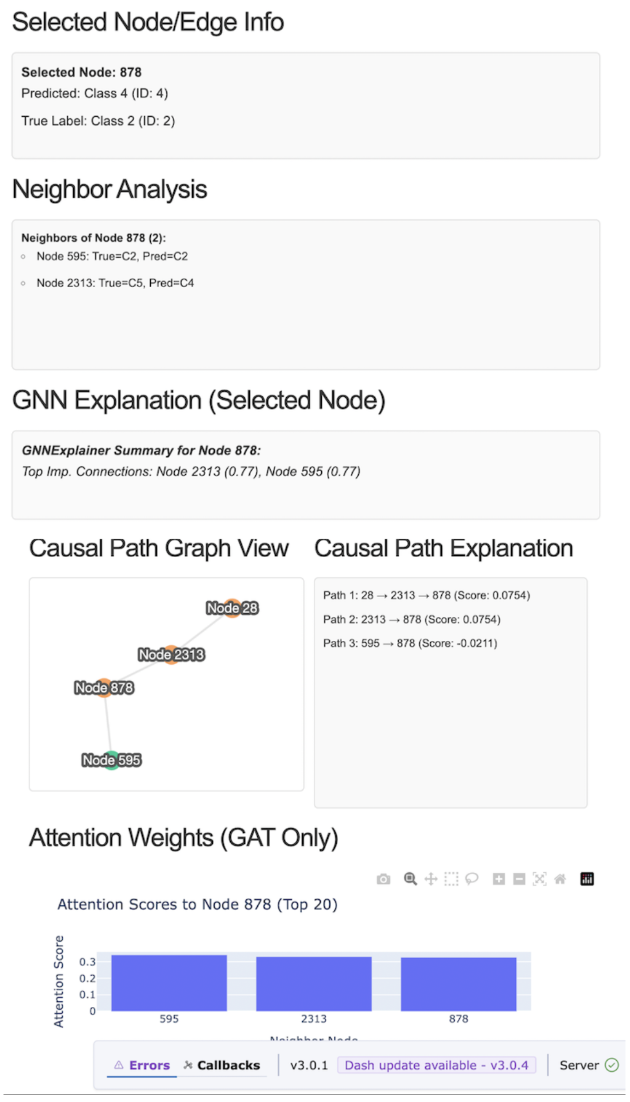
In this second case study, I turn my attention to the GAT model trained on the more challenging CiteSeer dataset. The goal here is to perform a three-way comparison between the explanatory methods to understand why the model made a specific prediction error.

Scenario: I select Node 878, which the GAT model has incorrectly classified as Class 4, when its true label is Class 2. The neighborhood provides the first clue: it contains a correctly classified neighbor from the true class (Node 595, True/Pred: C2) and another neighbor that was also misclassified into the same wrong class (Node 2313, True: C5, Pred: C4).

Comparing GNNExplainer, Attention, and Causal Paths

I generate the explanations to build a comprehensive picture of the GAT's reasoning failure.

- **GNNExplainer & Attention:** My first stop is the GNNExplainer summary, which acts as a proxy for the model's attention. The result immediately highlights the model's central conflict: it assigns an identical importance score (0.77) to both the correctly classified neighbor (Node 595) and the misclassified neighbor (Node 2313). This shows the model was unable to distinguish between the "good" and "bad" influence, paying equal attention to both.
- **Causal Path Analysis (CPA-IV):** Next, I run CPA-IV to understand the direction and nature of these influences. The causal scores tell the full story of the error:
  - Path 2: 2313 $\rightarrow$ 878 (Score: 0.0754): This path, originating from the misclassified neighbor, has a strong positive score. This is the smoking gun: it shows the error



Selected Node/Edge Info

Selected Node: 878
Predicted: Class 4 (ID: 4)
True Label: Class 2 (ID: 2)

Neighbor Analysis

Neighbors of Node 878 (2):
- Node 595: True=C2, Pred=C2
- Node 2313: True=C5, Pred=C4

GNN Explanation (Selected Node)

GNNExplainer Summary for Node 878:
Top Imp. Connections: Node 2313 (0.77), Node 595 (0.77)

Causal Path Graph View

Causal Path Explanation

Path 1: 28 → 2313 → 878 (Score: 0.0754)
Path 2: 2313 → 878 (Score: 0.0754)
Path 3: 595 → 878 (Score: -0.0211)

Attention Weights (GAT Only)

Figure 5.3: Dashboard showing the state for Node 878. The GAT model has misclassified the node, and the "Neighbor Analysis" reveals a conflict between its neighbors.

from Node 2313 was actively and positively propagated, pushing Node 878 toward the wrong prediction (Class 4).

– Path 3: 595 → 878 (Score: -0.0211): Conversely, the path from the correctly classified neighbor has a negative score. This indicates that the signal from the one neighbor that could have led to the right answer was ignored, suppressed, or counted against the final prediction.

Synthesized Interpretation: The three methods together weave a powerful and cohesive narrative of the GAT model's failure.

1. **GNNExplainer/Attention** [YBY+19, VCC+17] answers the question, "Which neighbor did the model focus on?" It reveals the model's critical indecision, as it focused equally on the "good" neighbor and the "bad" one.

2. **Neighbor Analysis** adds the context that the "bad" neighbor wasn't just from the wrong class, but was itself a prediction error, setting the stage for an error cascade.

3. **CPA-IV** provides the final, crucial piece of the puzzle, answering "How did the model use these conflicting signals?" The causal analysis shows the model actively embraced the influence from the misclassified node (positive score) while suppressing the influence from the correctly classified one (negative score).

This three-part explanation reveals that the GAT model's error was not random but a direct result of error propagation, where a mistake on one node (2313) directly and positively caused a subsequent mistake on the target node (878).

5.2.4 Demonstrating Interactivity: "What-If" Scenarios

The preceding case studies demonstrate the analytical power of the dashboard for static explanation. However, its most unique contribution—and its key advantage over tools like GNNLens and Corgi—is enabling dynamic, interactive experimentation. In this final section, I will walk through a "what-if" scenario to demonstrate how I can use the tool to test a hypothesis and actively debug a model's prediction.

Scenario: I revert to the GAT model on the Cora dataset and identify an incorrect prediction. The model classifies Node 2129 as Class 4, when its True Label is Class 3. The explanations (as detailed in the prompt logs) suggest this error is primarily caused by a strong, misleading influence from a single neighbor, Node 1111, which belongs to Class 4. My hypothesis is that this single edge is poisoning the prediction.

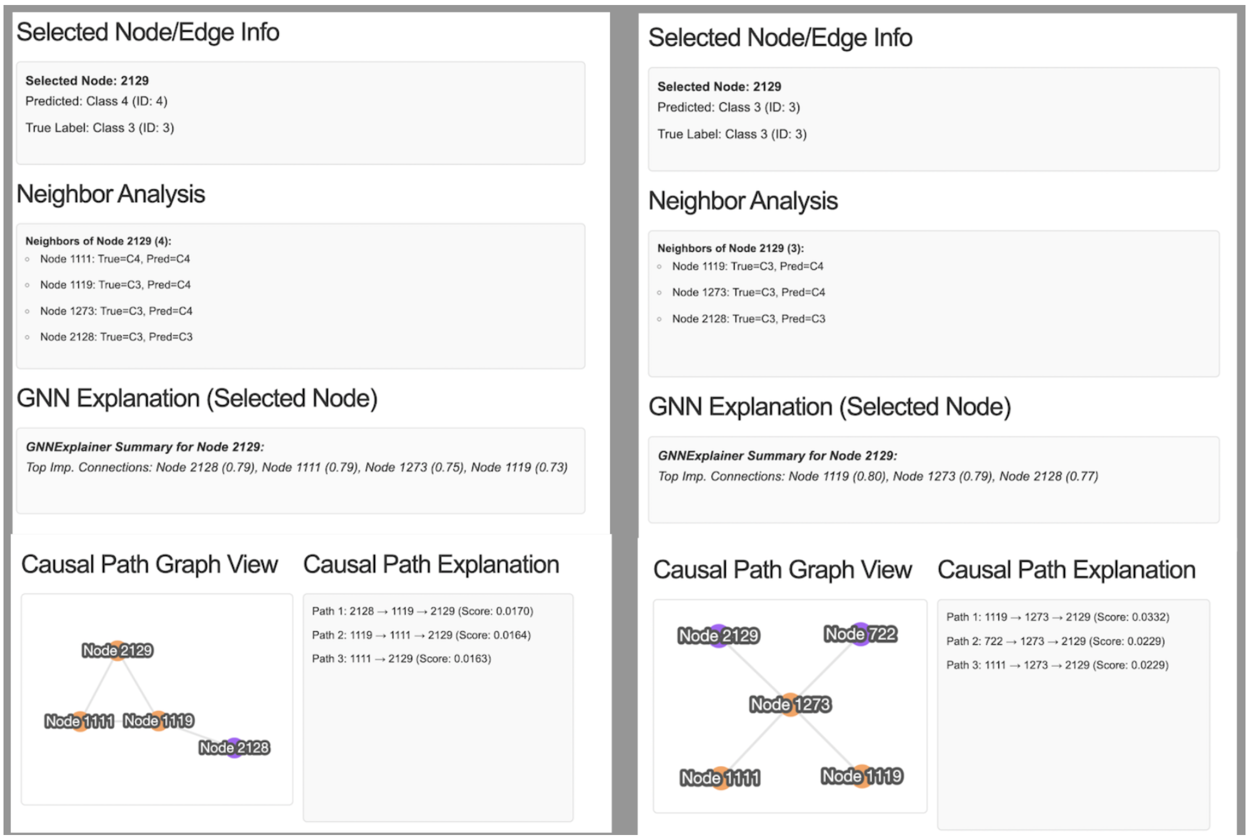
Impact of Removing a Critical Edge

I will now use the dashboard to perform a direct intervention to test my hypothesis, following a workflow inspired by interventional tools [WPH$^+$19].

1. **Hypothesis:** The edge 2129-1111 is the primary cause of the misclassification of Node 2129. Removing it should correct the prediction.
2. **Action:** In the dashboard's control panel, I enter "2129" and "1111" into the edge modification inputs and click the "Remove Edge" button.
3. **Observation:** The system processes this intervention and the UI updates in real-time. The visual feedback is immediate and unambiguous:
   - The edge connecting Node 2129 and Node 1111 disappears from the Graph View.
   - Most importantly, the "Selected Node Info" panel refreshes, and the predicted class for Node 2129 has flipped from Class 4 to Class 3. The prediction is now correct.

This observation provides powerful evidence supporting my hypothesis. The removal of a single, targeted edge was sufficient to completely change the model's output from incorrect to correct. This demonstrates a direct causal link between that edge and the original error, an example of a counterfactual explanation derived through direct manipulation [GAH+23]. To complete the analysis, I re-run the explanation tools on this new, modified graph. The new explanations show that the model now relies on reasoning paths through its remaining neighbors, which all belong to the correct class. The intervention has successfully "healed" the model's reasoning process for this specific node.

This interactive validation is the capstone of the analytical workflow. It transforms the process of explainability from a passive act of generating reports into an active, experimental science of probing, testing, and understanding the complex inner workings of a GNN.


Selected Node/Edge Info

Selected Node: 2129
Predicted: Class 4 (ID: 4)
True Label: Class 3 (ID: 3)

Neighbor Analysis

Neighbors of Node 2129 (4):
- Node 1111: True=C4, Pred=C4
- Node 1119: True=C3, Pred=C4
- Node 1273: True=C3, Pred=C4
- Node 2128: True=C3, Pred=C3

GNN Explanation (Selected Node)

GNNExplainer Summary for Node 2129:
Top Imp. Connections: Node 2128 (0.79), Node 1111 (0.79), Node 1273 (0.75), Node 1119 (0.73)

Causal Path Graph View

Causal Path Explanation

Path 1: 2128 → 1119 → 2129 (Score: 0.0170)
Path 2: 1119 → 1111 → 2129 (Score: 0.0164)
Path 3: 1111 → 2129 (Score: 0.0163)

Selected Node/Edge Info

Selected Node: 2129
Predicted: Class 3 (ID: 3)
True Label: Class 3 (ID: 3)

Neighbor Analysis

Neighbors of Node 2129 (3):
- Node 1119: True=C3, Pred=C4
- Node 1273: True=C3, Pred=C4
- Node 2128: True=C3, Pred=C3

GNN Explanation (Selected Node)

GNNExplainer Summary for Node 2129:
Top Imp. Connections: Node 1119 (0.80), Node 1273 (0.79), Node 2128 (0.77)

Causal Path Graph View

Causal Path Explanation

Path 1: 1119 → 1273 → 2129 (Score: 0.0332)
Path 2: 722 → 1273 → 2129 (Score: 0.0229)
Path 3: 1111 → 1273 → 2129 (Score: 0.0229)

Figure 5.4: A composite image showing the "before" and "after" state of the intervention. The left side shows the initial incorrect prediction for Node 2129 with the critical edge present. The right side shows the corrected prediction after the edge has been removed.

Chapter 6

Conclusion and Future Work

This thesis embarked on a mission to address the critical challenge of explainability in Graph Neural Networks. Confronted with the "black box" nature of these powerful models, I have sought to move beyond static, correlational explanations towards a more dynamic, causal, and interactive paradigm of understanding. This concluding chapter summarizes the core contributions of my work, discusses the key findings gleaned from the case studies, acknowledges the limitations inherent in this project, and proposes several exciting avenues for future research that can build upon the foundation I have established.

6.1 Summary of Contributions

This research has produced two principal contributions, one methodological and one practical, which work in concert to advance the state of GNN explainability.

1. **A Novel Causal Explanation Method (CPA-IV):** My primary methodological contribution is the design, formalization, and implementation of Causal Path Analysis Interventional Variant (CPA-IV). Recognizing the limitations of existing methods that primarily identify feature importance or influential 1-hop neighbors, I developed CPA-IV to uncover the multi-hop structural pathways that form the causal backbone of a GNN's reasoning process. By enumerating potential reasoning paths and then quantifying their importance through direct intervention—simulating their removal and measuring the impact on the model's output—CPA-IV provides a more profound, causal explanation of a model's behavior. As demonstrated in my case studies, this method can reveal critical "gateway" nodes and structural dependencies that are invisible to other explainers.

2. **An Interactive Framework for Interventional Analysis:** My core practical contribution is the creation of a fully functional, interactive web-based dashboard that serves as an integrated workbench for GNN analysis. This tool is not merely a passive visualization platform like GNNLens [KLM+22] or Corgi [HTQ+21]; it is an experimental environment. I designed it to bring together multiple explanation paradigms—the intrinsic GAT Attention [VCC+17], the popular GNNExplainer [YBY+19], and my novel CPA-IV—into a single, coherent interface. Crucially, I imbued the framework with interactive capabilities, allowing a user to directly manipulate the graph structure by adding or removing nodes and edges and observe the downstream consequences on predictions and explanations in real-time. This transforms explainability from a static reporting activity into a dynamic process of hypothesis testing and discovery [WPH+19].

Together, these contributions offer a new lens through which to view GNN behavior and a practical tool to facilitate that analysis, aiming to make these complex models more transparent, trustworthy, and debuggable.

6.2 Discussion of Findings

The experimental results presented in Chapter 5 led to several key findings that underscore the value of my approach, particularly when contextualized against existing tools.

First, the case studies consistently demonstrated that **different explanation methods provide complementary, multi-resolution insights**. No single explainer tells the whole story. GAT attention [VCC+17] provides a quantitative, model-intrinsic view of 1-hop neighbor importance. GNNExplainer [YBY+19] offers an excellent high-level correlational summary. CPA-IV, in turn, delves deeper to reveal the causal mechanics of multi-hop information flow. The most comprehensive understanding was always achieved when I synthesized these findings, as each illuminated a different facet of the model's logic.

Second, my work highlights the practical difference between **correlational and structural-causal explanations**. In several instances, GNNExplainer identified a set of influential neighbors, but it was CPA-IV that revealed that the causal influence was not evenly distributed, but rather channeled through a single "gateway" node. This suggests that to truly understand GNNs, we must analyze the structure of the information flow, not just the presence of influential neighbors.

Finally, and most importantly, the "what-if" scenarios proved that **interactivity for intervention is a transformative feature**. This is the key differentiator from observational tools like GNNLens and Corgi. While those systems allow for deep inspection, my framework allows for experimentation. By performing a targeted intervention—removing a single edge that I hypothesized was causing a prediction error—and observing the model instantly correct itself, I moved from conjecture to causal validation. This ability to probe, test, and verify hypotheses in real-time provides a level of understanding that an observational tool cannot. It is this interactive loop of observation, hypothesis, intervention, and validation that represents the most powerful application of the framework I have built.

To further contextualize these findings and clearly delineate the contributions of this work, Table 6.1 provides a direct comparison against the state-of-the-art visual analytics tools, GNNLens and CorGIE, summarizing the key differentiators discussed throughout this thesis.

Table 6.1: A comparative analysis of my framework against state-of-the-art visual analytics tools, highlighting its unique focus on interventional and causal analysis.

Feature / Aspect	GNNLens	CorGIE	My Work (CausaGNN)
Primary Goal	Model debugging & performance diagnosis.	Understanding graph-to-embedding mapping.	To provide deep, actionable insights into model reasoning by synthesizing correlational and causal explanations.
Core Interaction	**Observational:** Inspecting internal layer representations.	**Observational:** Exploring visual correspondence.	**Experimental & Interventional:** Actively probing the model by manipulating the graph and observing real-time consequences.
Explanation Paradigm	Correlational (feature importance, embedding similarity).	Correlational (visual correspondence).	**Holistic & Multi-faceted:** Uniquely combines correlational methods (GNNExplainer) with a novel structural-causal method (CPA-IV).
"What-If" Analysis	Not a core feature.	Not a core feature.	**Central Design Principle:** The entire framework is built to enable live, user-driven "what-if" scenarios for hypothesis validation.
Explicit Causal Methods	No.	No.	**Yes (Key Differentiator):** Provides true causal insights via CPA-IV, moving beyond simple correlation to identify structural drivers of a prediction.

6.3 Limitations of the Current Work

In the spirit of academic rigor, it is essential to acknowledge the limitations of this project. While the framework has proven effective in its designed context, several constraints define the boundaries of its current capabilities.

- **Scalability of CPA-IV:** The current implementation of CPA-IV relies on enumerating all simple paths up to a certain length. While effective on the sparse citation networks used in this study, this approach does not scale well to very large or dense graphs, where the number of paths could become computationally intractable.
- **Focus on Node Classification:** The entire framework was designed and tested exclusively for the task of transductive node classification. Its applicability and effec-

tiveness for other critical GNN tasks, such as link prediction or graph classification, have not been explored.

- **Limited Scope of Models and Datasets:** The experiments were confined to two GNN architectures (GCN and GAT) and two well-known academic datasets (Cora and CiteSeer). The findings cannot be assumed to generalize to all GNN architectures or to different types of graphs without further study.
- **Simplistic Intervention Model:** The interventions performed by CPA-IV are currently limited to edge removal. The analysis does not account for feature-based interventions or more complex structural interventions.

6.4 Avenues for Future Research

The limitations of the current work naturally give rise to a rich set of possibilities for future research. I see several promising directions to build upon this foundation.

6.4.1 Extending to Other GNN Models and Datasets

A logical first step would be to broaden the scope of the framework. This would involve integrating a wider variety of GNN architectures (e.g., GraphSAGE, GIN) and testing the system on larger, more diverse datasets to assess the scalability and generalizability of the methods.

6.4.2 Incorporating Additional XAI Techniques

The current dashboard compares three types of explainers. A significant extension would be to incorporate other classes of explanation, most notably **counterfactual explainers**. A counterfactual explainer answers the question, "What is the minimal change to the graph that would have flipped the model's prediction?" [GAH+23, LOG+22]. Integrating such a method would provide yet another complementary view, focusing on the decision boundary of the model.

6.4.3 Formal User Study and Evaluation

While I have demonstrated the utility of the dashboard through my own analytical case studies, the ultimate validation of any XAI tool must come from its intended users. The most critical next step is to conduct a formal user study. This would involve designing tasks (e.g., "identify the cause of this model error," "decide if you trust this prediction") and having participants use the dashboard to complete them. By measuring metrics such as task completion time, accuracy, and self-reported trust, I could quantitatively evaluate the effectiveness of an interventional and causal approach compared to purely observational tools. This empirical validation, following principles of human-centered evaluation in visual analytics [HKPC18], is essential for moving the project from a research prototype to a proven analytical tool.

Bibliography

[GAH+23] Tom Geffner, Mehdi Azabou, Peter Henderson, Jacob D Boucher, William L Hamilton, and Joelle Pineau. Graph edits for counterfactual explanations: A comparative study. 2023.

[Ham20] William L Hamilton. *Graph representation learning.* Morgan & Claypool Publishers, 2020.

[HKPC18] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693, 2018.

[HTQ+21] Zilong He, Hansen Tong, Zai Qu, Yuting Ma, Yixuan Wang, and Kwan-Liu Ma. Visualizing graph neural networks with corgie: Corresponding a graph to its embedding. In *2021 IEEE Visualization Conference (VIS)*, pages 156–160. IEEE, 2021.

[HWWZ22] Qizhang Huang, Meiyi Wu, Chuang Wu, and Jian Zhao. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[HZY+24] Zilong He, Zepeng Zheng, Zao Yang, Jian Xu, and Kwan-Liu Ma. Gnn 101: Visual learning of graph neural networks in your web browser. *arXiv preprint arXiv:2402.01032*, 2024.

[JWZ+23] Yuhang Jie, Yujie Wang, Yichen Zhang, Shurui Gui, Zaiyu Lu, and Jing Liu. Exploring causal learning through graph neural networks: An in-depth review. *arXiv preprint arXiv:2306.01257*, 2023.

[KLM+22] Zhihang Khoo, Yao Li, Yuting Ma, Xidao Wang, Yu-Shun Wang, Laurens Van Der Maaten, and Yixuan Wang. Gnnlens: A visual analytics approach for prediction error diagnosis of graph neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):13–23, 2022.

[KW16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[LOG+22] Ana Lucic, L-P O'Bray, Patrick Gunn, Mihai Lupu, Barry O'Sullivan, Barry Marquis, and Nhien-An Le-Khac. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, pages 233–241, 2022.

[TLW+22] Jialu Tan, Siqi Le, Peilin Wang, Meng Wang, Ting Liu, and Y-Lan Chang. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM Web Conference 2022*, pages 1787–1797, 2022.

[VCC+17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[WPH+19] James Wexler, Mahima Pushkarna, Tolga Hemsley, Fernanda Viegas, Martin Wattenberg, and Jascha Lahore. The what-if tool: interactive probing of machine learning models. In *2019 IEEE Visualization Conference (VIS)*, pages 56–65. IEEE, 2019.

[YBY+19] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, volume 32, 2019.

[YYGJ22] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4501–4521, 2022.

© 2025, *Indian Institute of Technology Delhi*

.1 APPENDIX

All the project files are available at the following GitHub repository: Major Project

© 2025, *Indian Institute of Technology Delhi*