
Otto-von-Guericke University Magdeburg



Faculty of Computer Science
Digital Engineering

Master Thesis

GNN xEval
Design and Implementation of a framework for Graph
Neural Network Explainers Evaluation

Author:
Affan Ahmed

July 23, 2023

Advisers:

Supervisor
Prof. Dr.-Ing. Ernesto William De Luca

Human Centered Artificial Intelligence
Faculty of Computer Science
Otto-von-Guericke University
Universitätsplatz 2, G29-415
39106 Magdeburg, Germany

Supervisor
Erasmus Purificato

Human Centered Artificial Intelligence
Faculty of Computer Science
Otto-von-Guericke University
Universitätsplatz 2, G29-001
39106 Magdeburg, Germany

Ahmed, Affan:

GNN xEval

*Design and Implementation of a framework for Graph Neural Network Ex-
plainers Evaluation*

Master Thesis, Otto-von-Guericke University

Magdeburg, 2023.

Contents

Abstract

1 Introduction

1.1 Overview	1
1.2 Importance	2
1.2.1 Importance of Graph Neural Networks	2
1.2.2 Importance of explainability	3
1.3 Motivation	4
1.4 Aim of the thesis	5
1.5 Structure of the thesis	6

2 Background

2.1 Classification of Explainers	7
2.2 Current Research	9
2.3 Preliminaries	11
2.3.1 The simple Graph structure	11
2.3.2 Node and edges	12
2.3.3 Directed and Undirected graphs	13
2.3.4 Homogeneous and Heterogeneous graphs	14
2.3.5 Graphs in Neural Networks	16
2.3.6 Some definitions about explanations:	18

3 Adopted Methodologies

3.1 Graph Datasets	21
3.1.1 Homogeneous datasets	21
3.1.2 Heterogeneous datasets	22
3.1.3 Synthetic datasets	23
3.2 Graph Neural Network tasks	23
3.3 Graph Neural Network Architectures	27
3.4 Explainers	31
3.5 Explanation evaluation metrics	34

4 Implementation and Evaluation

4.1 Configuration	37
4.1.1 System Configuration	37

4.1.2	Languages, Tools, and Frameworks	38
4.1.3	The experiment settings	39
4.2	Implementation	41
4.2.1	User Interface	41
4.2.2	The Process	43
4.2.3	The Implementation	44
4.3	Results and discussion	46
4.3.1	The Application	46
4.3.2	Characterisation Score	47
4.3.3	Unfaithfulness	48
5	Conclusions and Future Work	
5.1	Conclusions	49
5.2	Limitations	50
5.3	Future Work	51
A	Detailed Results	
B	Abbreviations and Notations	
C	List of Figures	
D	List of Tables	
E	Bibliography	

Abstract

Graph Neural Networks have developed a sharp interest in researchers because of their applications and simplicity. Due to the black box of nature model during training in Neural Networks and Graph Neural Networks, it is important to interpret and explain them.

The explanation can provide answers to its trustworthiness, transparency, debugging, error analysis, feature importance, ethical issues, and insights into the model providing an opportunity to improve them. This leads us to the domain of Explainable AI (XAI). We have ample methods to explain the models. However, these explainer models are largely available for simple Neural Networks.

With the recent interest in GNN, there is significant interest in its interpretation and explanation. While some of the models available in Neural Networks can be modified to work for GNNs, there are several novel approaches for explainability.

Depending on their nature some of the Explainers are model agnostic while others are simply model specific. As most of this work is available in individual pieces of code having a mismatching configuration for GPU. Even though a few libraries are provided however no integrated and complete single and user friendly platform is provided for the researchers to research and experiment their research.

I present a unified platform for Graph Neural Network Explainability called "GNN xEval" that can implement explanations and provide results that can be effectively compared with the results of different explainers. The application provides a user friendly interface for the researchers that can create their explainers and use this interface in different configurations and compare their research with the existing explainers including benchmarks.

Acknowledgements

- I would like to acknowledge the efforts of my supervisor who has always facilitated me at every possible step. His friendliness and his ability to suggest solutions according to the situations, his understanding and acknowledgment of the efforts of the students and other researchers are admirable.
- Secondly, I would like to acknowledge Advaneo GmbH, who has allowed me to work as a student for the duration of my thesis.
- Thirdly, I am thankful to everyone who has directly or indirectly supported my efforts.
- Also, I highlight my friends who have helped even a bit including Hamza Zaidi who has helped me alongside this thesis having a similar topic of his thesis.
- And finally and foremost my parents and close relatives who always lend me unconditional support in every way possible.

1

Introduction

1.1 Overview

Although literature and algorithms such as backpropagation (written in 1970) existed, interest in Machine Learning and Artificial Intelligence increased among researchers in the decade before the turn of the millennium. The past decade has seen a focus on a subset of Machine Learning called Deep Learning. Deep Learning processes different input data types, resulting in a variety of applications with unique architectures to solve each problem. It is more challenging to obtain results than with traditional Machine Learning methods.

The Graph data structure is a commonly used data structure, like in social media. Especially the Knowledge Graphs(a type of heterogeneous graph) are in use for a long time. They are schema-less alternatives to databases.

Infusing the two concepts gives us the Graph Neural Networks. In other words, we can apply deep Neural Networks to Graph data structures as inputs and get some desired outputs.

The nature of Neural Networks is that we know the inputs, and get the desired or undesired outputs, however, we are unaware of the insights of the model itself. Therefore we have a different domain that can interpret and explain the insights of the model during the training. This is known as Explainable Artificial Intelligence.

The concept of explainable artificial intelligence also extends to GNNs. Although some methods of Neural Networks apply to GNNs, in general, due to the difference in the architecture of the GNNs as compared to Neural Networks, GNNs have their explainers. Some of which are specific to the model being considered while others are model agnostic.

There are several explainers researched and presented and in the case of GNNs there are no gold standards. Also, an explainer might perform better on a dataset or some configuration, but it may perform worse than others on a different dataset. Furthermore, one explainer may be good for a specific GNN task, however, another may be outperforming it in a different GNN task. Therefore it is important to know the quality of the explanations.

Thus this research work presents several datasets, GNN tasks, explainers, and explainer evaluation metrics available in the present literature. It provides a framework where the researchers can easily test several explainers, datasets, etc.

1.2 Importance

1.2.1 Importance of Graph Neural Networks

Graph Neural Networks(GNNs) are rapidly evolving and a new domain in the revolutionary field of Deep Learning. They have importance due to several reasons:

- **Versatility**

A lot of different tasks can be performed on GNNs

- **Scalability**

Millions and billions of nodes and edges can be handled.

- **Can handle Graph data structure**

- **Transductive and Inductive**

GNNs can work both transductive and inductively. It can work on predicting unseen nodes/edges and it can also work on classifying existing nodes and graphs.

- **Local and Global neighborhood**

GNNs can gather information in the local and the neighborhood

GNNs are network-based Neural Networks that can be applied in networks like

- Local Area Networks,
- Drug chemical structures,
- road networks,
- Rail networks,
- Networks of users in Social Media, etc

Where the data is structured in the form of graphs or can be mapped in the graphical data. GNNs have applications like

- Recommendation Systems,
- Drug Discovery/Invention,
- Text classification, and many more, etc.

1.2.2 Importance of explainability

GNN XAI has had ample research interest since 2019. The working of Neural Networks usually consists of several layers leading to a large network of nodes. However, during the training phase, it is hard to interpret the insights of the model. While one understands what has been inputted and what is on the output, it is not straightforward to know what is inside the layers. More recently it has become more important to know the insights, for example, it is important to know the insights as it might violate European Privacy Laws or General Data Protection Regulations(GDPR). These insights can make AI more

- trustworthy
- fair
- ethical
- faithful
- efficient
- improvement

- compliant with regulations
- responsible AI

1.3 Motivation

The field of Artificial Intelligence is captivating and constantly evolving. One of its prominent subsets is Machine Learning, which is an expansive subject that continues to be extensively researched. This dynamic nature ensures that algorithms are not confined to specific boundaries but rather evolve rapidly, introducing new and improved approaches.

As time progresses, newer algorithms emerge, often surpassing the performance of existing ones or catering to different requirements. This constant drive for innovation keeps the field vibrant and ensures that algorithms remain relevant and adaptable to the ever-changing landscape of Artificial Intelligence.

Deep Learning which in its entirety itself is a subset of Machine Learning and Data Mining is a very large subject. Due to interest in Neural Networks, more subsets have evolved such as Generative AI, diffusion models, Spiking Neural Networks Graph Neural Networks etc. While in the last decade, a sharp rise in interest in Deep Learning was witnessed, more recently the research interest in Graph Neural Network which has applications in chemical compounds like in Drug discovery and knowledge graphs have gained strength.

I am truly passionate about embracing and conquering challenges. As I delved deeper into the realm of Artificial Intelligence, what may seem like a mere buzzword to others transformed into a vast and fascinating field for those who comprehend its intricacies. The potential applications of Artificial Intelligence span across numerous domains, further solidifying my belief in its significance.

Driven by an insatiable curiosity to acquire new knowledge and accomplish meaningful milestones, coupled with my profound interest in Neural Networks, I decided to embark on a research journey and author a thesis focused on Graph Neural Networks. This endeavor allows me to explore cutting-edge developments in this field and contribute to its advancement.

1.4 Aim of the thesis

The underlying problem faced in Neural Networks including Graph Neural Networks is their interpretability, such that they are black Boxes. The input and the output are visible, however, what happens inside the process is hard to reason directly and it is important to interpret and explain them.

Explaining it lets us answer the question marks regarding the fairness, interpretability, and lack of explainability of the models(they are black boxes). Since it is important to answer these questions sometimes, this leads us to another domain called explainable AI (XAI).

Traditionally Machine, Deep Learning, Neural Networks, and Graph Neural Networks which is training the model and inferring from the trained model, however explainable AI focuses on what is going on in the layers of the deep learning and Graph Neural network model.

Some of the questions that arise and this work will try to address include:

1. What are the types of GNN datasets?
2. What are the common GNN datasets?
3. What are the statistics of the datasets?
4. What are the different GNN tasks?
5. What are the GNN models?
6. What are some of the explainers?
7. Is the explanation sufficient?
8. Is the explanation good?
9. Does the explainer work well for most of the problems?
10. What are some of the possible combinations of datasets, explainers, and metrics?
11. What are the libraries that work for GNN?

1.5 Structure of the thesis

1. The first chapter of this is the introduction. It discusses the motivation and what questions that would be addressed in the manuscript.
2. The Second chapter contains the background. It discusses related work . And it also provides some preliminaries that show the basics of GNN and its explainability.
3. The third chapter focuses on the literature review. Some of the explainers were tried however they had some constrictions and were not used again or dropped from the experiments described in the next chapter.
4. The fourth chapter introduces the experimental settings and the tools that were used. It provides how the experiments were done. And finally, it provides the results and a discussion.
5. The final chapter gives the conclusion, the limitations that occurred, and improvements that can be done in the future.
6. In the end all the references are provided in different sections

2

Background

2.1 Classification of Explainers

Relevant and related work is done concerning the GNN explainability of Graph Neural Networks. Starting with The Taxonomic Survey of GNN explainability (YUAN et al. (2022)), discusses multiple Research articles for GNN XAI. It categorizes the methods and provides performance metrics.

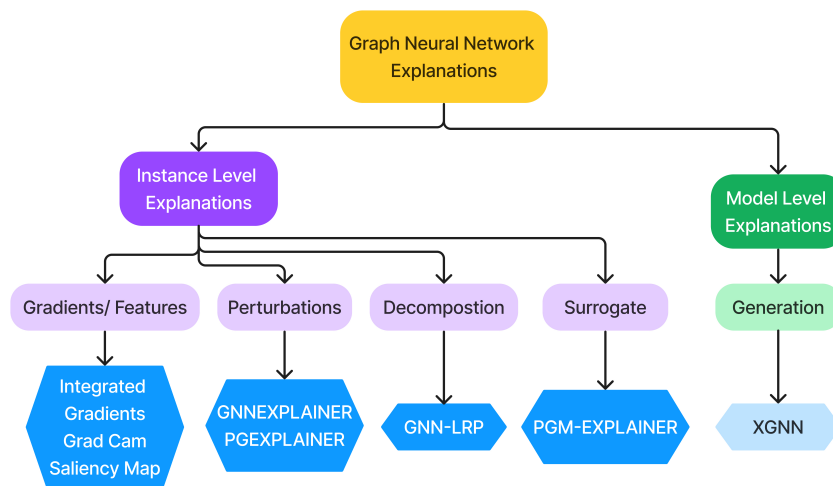


Figure 2.1: GNN Explainers classification and examples

There are several GNN Explainers, some of which are based on Neural Networks. The paper YUAN et al. (2022) has introduced a type of classification of explainers according to the approach towards GNN explainability.

- **Gradients/ Feature:** These explainers are commonly used in texts and images, they capture the gradients or hidden feature values. They get the importance scores of the input. The gradient methods capture the gradient of the predictions with respect to the input using backpropagation. While feature-based methods map the hidden features to the input. They use interpolation of the input features and measure the importance scores. In general the larger the gradient/ feature the greater the importance.

- **Perturbations**

These methods check the output variations with respect to different inputs (Perturbations). In case an output is similar despite the input variation, the important information is retained. Mostly perturbation methods use generate masks (filters) and apply them to data (like Images). For a given graph different masks are generated for example a node masks according to important input features depending on the task. The masks are then combined with the input graph. This graph has important information and is then fed to a trained GNN to evaluate and update the mask generation algorithms.

- **Surrogate Methods**

The idea is to use a simple and more interpretable model to predict a more complex model. There is an assumption that the relationships in the local neighborhood of the surrogate model are less complex than the original model. The local neighborhood of the input graph is extracted along with predictions. Then the training of the interpretable model is done and the explanations are considered explanations of the original model.

- **Decomposition Methods**

Decompose into several terms with importance scores depending on the set of rules. As the importance scores are layerwise, backpropagation is applied repeatedly. In the end, we obtain the layerwise relevance depending on the importance score of each of the layers.

- **Graph Generation Methods**

Graph Generation Methods generates Graphs and optimizes the Graph generation to get accurate predictions. Graph generation

is a reinforcement learning problem, where the generator predicts where to add an edge to obtaining feedback. Learning has several rules.

2.2 Current Research

XGNN:

(YUAN et al. (2020)) The paper is classified as Model-based methods (rather than instance-based) of which the Graph Generation methods with only example as XGNN. It uses a trained GNN and generates graphs, the purpose of which is to maximize certain predictions by the model. The XGNN generates graphs using Graph Generator using Reinforcement Learning technique. The technique uses a trained GNN model to produce predictions. The predictions are maximized by Graph Generator using GCNs. The Graph Generator decides what Edges can be created next step in a step-by-step manner for every node. The Graph Generator can be trained itself using Reinforcement learning. The XGNNs can produce a Graph with maximum probability with regards to the Ground truth however it may not be able to produce a graph that is the actual ground truth.

PGMExplainer:

(VU und THAI (2020)) This method can be classified as a Surrogate method. This is a sample/instance-based method. The method can be summarised into three parts. The first part is the data generation using perturbation on the input graph. Variable selection needs to be done for structure Learning as Complex and large graphs may contain too many variables using Pairwise dependence tests. The final phase uses BIC scores and the hill climbing method to explain the graphs.

GCAN:

(LU und LI (2020)) This method targets social media and creates a fake news detector. The purpose is to detect fake tweets. The explainer can be used to detect short fake tweets from Twitter. It can also be used in short text and similar domains like sentiment analysis.

Backdoor Attack using Explainability:

(XU et al. (2021)) This paper discusses the detection of a Backdoor attack and defense from the attack using explainability. For Explainability, it, however, used Captum AI (PyTorch-based Neural Interpretation library). It talks about using explainability creating an importance map from 0 to 1, 1 being high importance and 0 being low importance, and creating an edge mask on the important only features. It suggests that the backdoor will contain more unimportant features and thus detect and eliminate them.

Explainer for Temporal GNNs:

(HE et al. (2022)) Uses PGM explainer for TGNNs.

GNN-LRP:

(SCHNAKE et al. (2022)) This particular research is classified as the Decomposition method. The paper discusses using a recursive function (such that the Higher model is a nesting of the First order expansions) to model the explanation and using Taylor Series to find the Higher-order expansion (output layer towards input). An attribution scheme called walks is introduced (GNN prediction on the collection of edges). Moving from the top layer to the input layer, the First order explanation can be inserted into every layer and LRP is used here.

GraphFram Explainer:

(AMARA et al. (2022)) The paper discusses metrics of explainability rather than being an explainer which it argues mostly does not work on real data but synthetic data so they can have a ground Truth. This means the accuracy metric is not appropriate. It argues some explainers might be better but in other data, they might not be good. This method does not target model-based explainers like XGNN. It targets the Focus of explanation, Mask nature, and mask transformation. It introduces a new measure called the Characterization score which is derived from +Fidelity and -Fidelity.

Self Explainable GNN:

(DAI und WANG (2021)) The paper first evaluates node similarity and local structure. For this purpose, similarity scores of edges between two nodes are evaluated with provided Labels. Since nodes with similar labels are likely to be similar pair, implicit supervision (labels guiding prediction) is provided for Similarity modeling. This leads us to K nearest Labeled node for prediction and explanations. Contrastive Learning can further enhance the explanations.

GraphSVX:

(DUVAL und MALLIAROS (2021)) The paper combines all the classifications provided by the taxonomic survey to present the explainer. Such that it uses a decomposition method working on a perturbed dataset. Further, it evaluates the Shapley values. It uses GNNExplainer, Pgexplainer, GraphLime, PGM Explainer, and XGNN. All methods can input a Graph into the mask generator to create node edge and feature mask. These masks are fed into a graph generator, which converts them into input space and feeds them to GNN. The predictions of GNN are used for Mask Generator, Graph Generator, and explanation generator. The explanation generator provides an explanation.

ShapeGGen:

(AGARWAL et al. (2023)) ShapeGGen is a data generator to overcome the synthetic data unreliable ground truth problem. The paper uses eight GNN explainer. It uses GraphXAI a Python library to implement some tasks. GraphXai library makes use of the ShapeGGen dataset. It introduces several metrics. It also discusses fairness related to the ground truth.

2.3 Preliminaries

2.3.1 The simple Graph structure

To gain a solid foundation in Graph Neural Networks (GNNs), it is crucial to comprehend the fundamental structure of graphs. Graphs are com-

posed of interconnected nodes, with edges serving as the links between them. In Figure 2.2 we have 6 nodes (labeled as numbers) with 6 interconnections between them (represented by black lines).

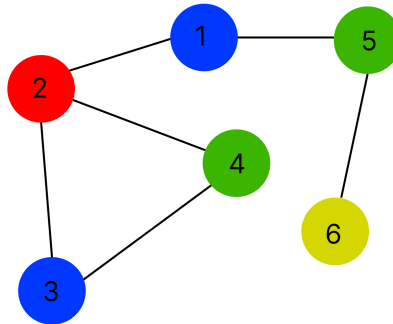


Figure 2.2: A simple Graph

2.3.2 Node and edges

Different literature and formats use different terms. For this reason, some of the terms are mentioned below for clarity.

For the Nodes, the similar terminologies include:

- Nodes
- Vertices
- Points
- Entities

For the Edges, the similar terminologies are:

- Edges
- Links
- Relationships

- Connections
- Arcs
- Interconnections

2.3.3 Directed and Undirected graphs

The graphical datasets consist of edges that can be either directed or undirected. In case a graph is directed it means, node 'X' is related to Node 'Y' where the edge from 'X' is pointing towards node 'Y'. In case Node Y is also related to Node X then another arrow can be pointed in the reverse direction. Fig 2.3 contain pointed edges is an example of a directed graph.

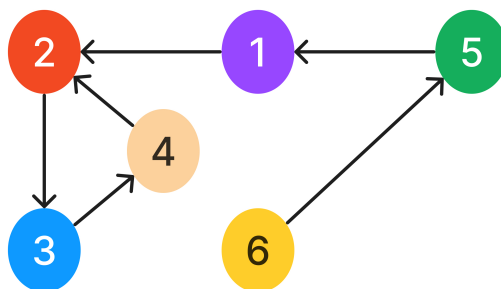


Figure 2.3: A simple directed graph

(LESKOVEC und KIPF (2021)) In the case of undirected, however, the links are not pointed and that means node X is related to node Y and vice versa is also true. Figure 2.4 is an undirected graph

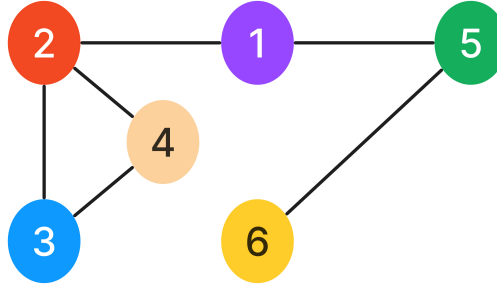


Figure 2.4: A simple undirected graph

2.3.4 Homogeneous and Heterogeneous graphs

A homogeneous graph (or Homogeneous network) where the type of the nodes and edges is the same. The node and the edge attributes have the same category. An example of a Homogeneous graph is social media websites.

A graph (homogeneous) is defined by the expression

$$G(N, E) \tag{2.1}$$

- N are the nodes
- E are the edges

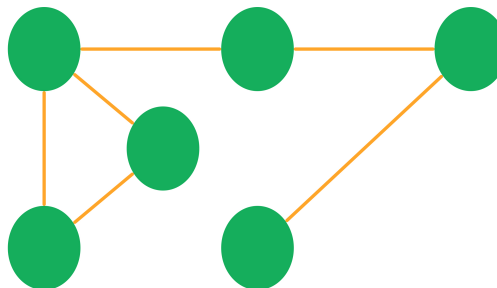


Figure 2.5: A Homogeneous Graph

On the other hand, a Heterogeneous graph (or Heterogeneous Network or multimodal network) can contain nodes and edges of different categories. The attributes of nodes and edges consist of different types. An example of a Heterogeneous Graph is Knowledge Graph Google Card that appears on Google searches in the browser.

A heterogeneous graph is defined by the equation

$$G = (V, E, R, T) \quad (2.2)$$

- V are the Vertices
- E are the edges with relation types
- T are the nodes types
- R are the Relation types

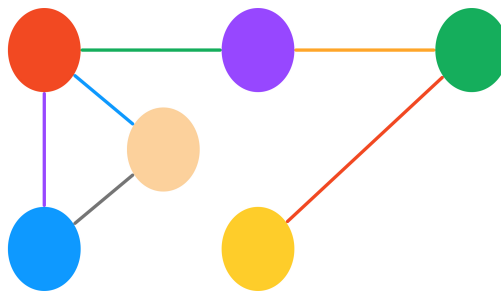


Figure 2.6: A Heterogeneous Graph

the fundamental differences between the two graphs included in tab

Table 2.1: Homogeneous graphs vs Heterogeneous graphs

	Homogeneous	Heterogeneous
type	same types of nodes and edges	different types of nodes and edges
e r type	entities and relationship types are of same kind	diverse entities and relationships
e.g.	social networks	recommendation system
e.g.	citation networks	knowledge graphs

2.3.5 Graphs in Neural Networks

Neural Networks cater data in the following:

- Image data or Computer Vision
- Text data or Natural Language Processing
- Audio data
- Structured data
- Times series data

However, when the data is in the form of graph data structure (such as Knowledge Graphs), the concepts of Neural Networks can be applied. Such that, the goal of the Neural Network is to find a function F which maps input nodes to D dimensional vectors (or embeddings), such that similar nodes are close to each other.

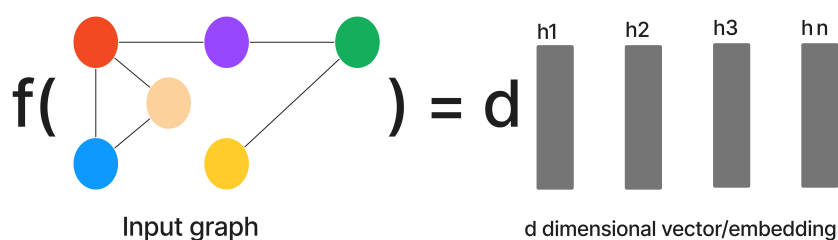


Figure 2.7: The embedding function f

The general tasks in most GNN architectures is as follows

- Aggregation
a function of aggregation is defined e.g. averaging, summing, or Max
- Loss Function
a loss function is defined e.g. cross-entropy
- Train
training is done of a set of nodes in a local neighborhood

- Generate Embeddings

Finally the embedding for the entire graph is generated

Challenges for GNN

- Arbitrary size and complex structure
- No fixed ordering
- Dynamic
- Heterogenous features

Computation Graph for GNN

In Figure 2.8 given below, an example input graph is given. We assume that node 2 is the target node.

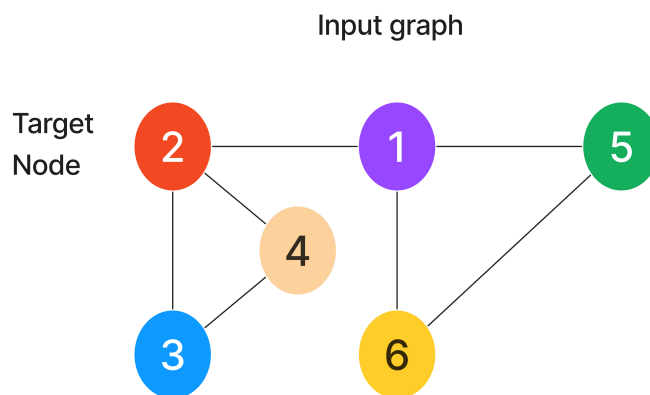


Figure 2.8: Input Graph with target node as node labelled 2

Figure given below 2.9, is the computation graph for the input graph for Figure 2.7. The target node is preceded by a Neural Network with inputs of Node 3, 1, and 4 which are located in the immediate neighborhood of Node 2(in the input graph). It is followed by another layer of the Neural Network and the nodes of the input are in the immediate neighborhood of nodes 3, 1, and 4 in the original input graph.

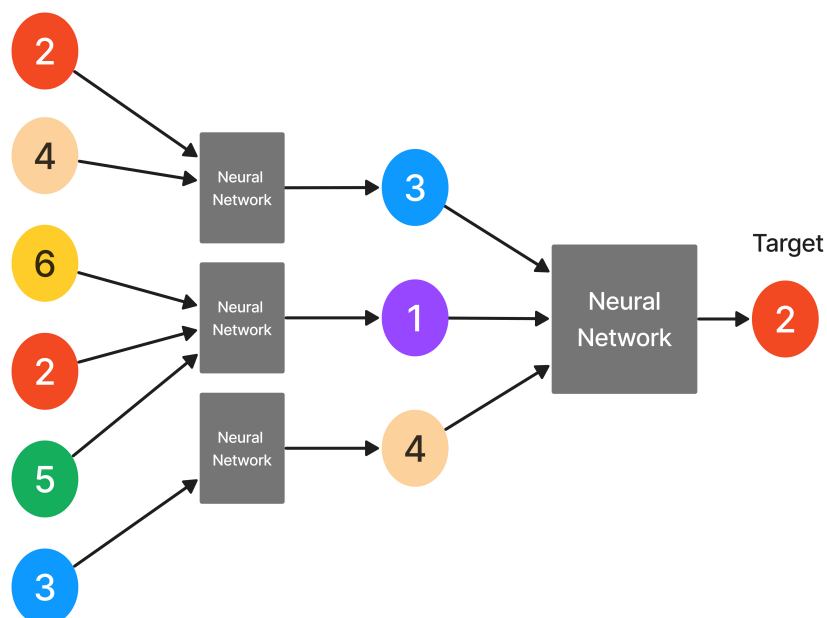


Figure 2.9: The computation graph for GNN

2.3.6 Some definitions about explanations:

- **Phenomenon:**

The ground truth of the dataset is given. The explainer is configured as 'phenomenon' such that it explains the model according to the ground truth node.

- **Model (explanation):**

The explanation focuses on the output of the GNN model. It can be handy when the ground truth labels are not included in the dataset.

- **Sufficient explanations:**

A sufficient explanation is obtained from the model's initial predictions. There can be multiple sufficient explanations of the model and different explanations can lead to the same prediction. As discussed later a negative fidelity close to 0 means explanation is sufficient.

- **Necessary explanations:**

Similar to a counterfactual explanation, A necessary explanation is one in which, if you remove the model from the initial graph, it changes the model predictions. A positive fidelity close to 1 means that the explanation is necessary.

3

Adopted Methodologies

3.1 Graph Datasets

There are numerous graph datasets publicly available. The available datasets can be categorized into distinct types.

3.1.1 Homogeneous datasets

- **Karate Club**

(ROZEMBERCZKI et al. (2020)) It is an open-source dataset based on a social network. This dataset can cater to community detection, node classification, and graph classification tasks

- **Tudatasets**

(KERSTING et al. (2016) Provided and maintained by the Technical University of Dortmund. It contains a variety of datasets. Most notable are

- MUTAG,
arguably the most common benchmark dataset representing chemical dataset. It is used mainly for classification tasks. The class labels are mutagen or non-mutagen
- ENZYMES,
another common dataset with nodes representing amino acids. It has multiple classes. They represent different functional classes
- PROTEINS,
- COLLAB,

- IMDB-BINARY
- REDDIT-BINARY.

- **Planetoid**

(YANG et al. (2016)) It contains three datasets representing citation networks. They all contain a single graph and are used for node classification. They are

- CORA,
- PUBMED,
- CiteSeer

- **Mutagenicity**

TONG et al. (2016) The dataset contains drug compounds with binary target class. It is used as a Graph Classification task.

Table 3.1: Statistics for common Homogeneous graph datasets

Name	# Graphs	# Nodes	# Edges	# features	# classes
KarateClub	1	34	156	34	4
MUTAG	188	17.9	39.6	7	2
ENZYMES	600	32.6	124.3	3	6
PROTEINS	1,113	39.1	145.6	3	2
COLLAB	5,000	74.5	4914.4	0	3
Cora	1	2,708	10,556	1,433	7
CiteSeer	1	3,327	9,104	3,703	6
PubMed	1	19,717	88,648	500	3

3.1.2 Heterogeneous datasets

- **Movielens**

(HARPER und KONSTAN (2015)) It is a database of 62k movies and 162k user with 25 million ratings. It is used for recommendation. The GNN task that can be performed is Link prediction.

- **DBP15K**

(SUN et al. (2017)) The dataset is based on cross-entity linguistic alignment. The languages Chinese, Japanese, and French are linked to English.

- **OGBN-MAG**

(HU et al. (2020)) It is an open-source heterogeneous citation network dataset. It can be used in Node classification and link prediction.

- **IMDB**

(FU et al. (2020)) A heterogeneous dataset for movies from a subset of the IMDB database. The knowledge graph contains three types of entities, Movies with 4278 nodes, Actors with 5257 nodes, and directors with 2081 nodes. The class represents the genre of the movie. There are three possible classes i.e. Action, comedy, and drama.

3.1.3 Synthetic datasets

- **BA2-Motifs**

(LUO et al. (2020)) used for graph classification task, the ba2-motifs contains 1000 graphs. The target class is either house or a 5 node cycle. Features are initialized as all 1s vector.

- **BA-Shapes**

(LUO et al. (2020)) A single graph BA based graph synthetic dataset. Classes are defined according to the position of the node in the graph such that they are labelled 1,2 and 3 if they are either at top or middle or bottom of the house motif accordingly. The base of the BA graph is labelled 0,

- **BA-MultiShapes**

(AZZOLIN et al. (2023)) The synthetic graph is used for Graph classification task. It contains 1000 Babarasi -Albert graph. The class 0 can contain empty set or a house or a grid or a wheel or the three motifs together. However class 1 contains the combinations of either house and grid, or house and wheel or wheel and grid.

3.2 Graph Neural Network tasks

Node classification:

The task at hand involves determining the labels of samples represented as nodes by examining the labels of their neighboring nodes. In this setting, the objective is to leverage the labeled nodes to infer the labels of the unlabeled nodes through the connectivity and relationships within the graph.

By utilizing the graph structure and the labeled nodes as guidance, the aim is to generalize the labels to the entire graph, making predictions for the unlabeled nodes based on the information propagated from their labeled neighbors.

$$Z_i = f(h_i) \quad (3.1)$$

- where Z_i is the predicted label of the target node
- and h_i represents the updated node to be predicted
- f represents the classifier function

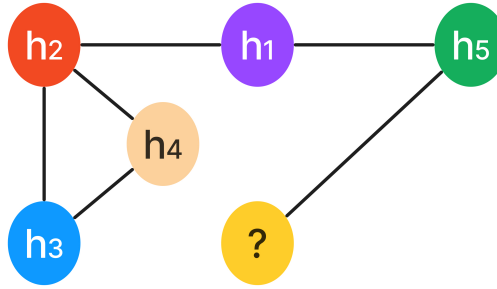


Figure 3.1: Node classification

Link prediction:

In this context, the algorithm's primary objective is to unravel the relationships between entities within graphs and accurately predict the presence or absence of connections between them. This capability holds significant importance in various domains, particularly within social networks. One such application lies in the knowledge graphs.

$$Z_{16} = f(h_1, h_6, e_{16}) \quad (3.2)$$

- where Z_{16} represent the score of the predicted link from node 1 to node 6. A higher score means more probability of the existence of a link
- h_1 and h_6 represent the updated node representation of node 1 and node 6 respectively
- f represents the link prediction function
- e_{16} represents the feature representation edge between nodes 1 and 6

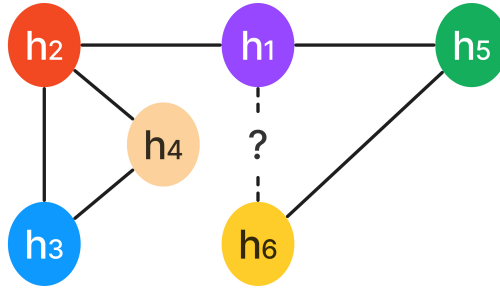


Figure 3.2: Link Prediction

Graph classification:

In this particular task, the entire graph is divided into labels rather than individual nodes or edges. The labels are assigned according to the structures. It is trained. The targets are accordingly to the application of the graphs such as social media and recommendation system.

$$Z_G = f(\sum h_i) \quad (3.3)$$

- where Z_G represents the label over the entire graph link
- h_i represent the learned node representation node N_i
- $\sum h_i$ represents the summation of node representation in the entire graph G

- f represents the graph classification function

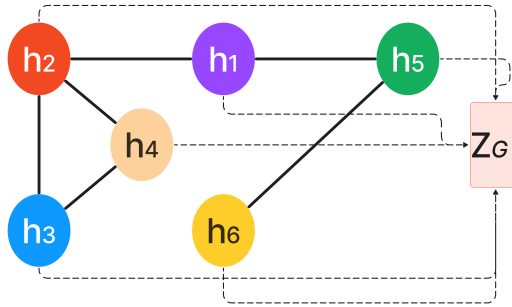


Figure 3.3: Graph Classification

Graph clustering/ Community Detection/ Graph Partitioning:

This refers to two tasks, Vertex clustering and Graph Clustering. In vertex clustering the nodes are clustered according to the edge weight or edge distances. Clusters formed are groups of similar nodes in terms of connectivity. In Graph Clustering, however, clusters are formed based on the similarity in terms of structure, topological structure, etc.

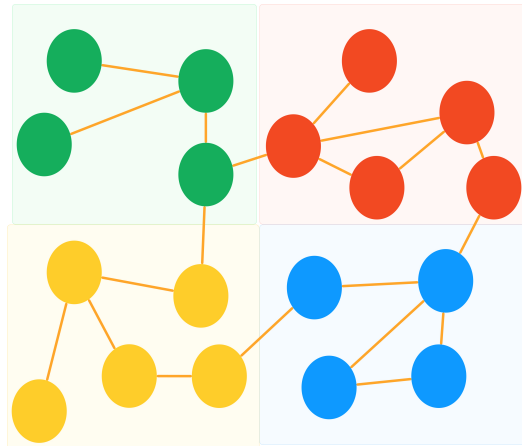


Figure 3.4: Community Detection

Graph generation:

Used to generate/synthesize new Graph data. It has applications in Drug discovery. Also, it is used to create synthetic datasets that can be used as benchmarks.

3.3 Graph Neural Network Architectures

There are loads of models and algorithms used. A few of the commonly adapted ones include:

- **Graph Convolution Network (or GCN)**

(KIPF und WELLING (2016))The concept of GCN is similar to the concept of Convolutional Neural Networks in Artificial Neural Networks such that it uses layer-wise propagation of a first-order approximation of spectral graph convolutions. This is the most commonly used method in the GNN domain.

$$H^{(l+1)} = \sigma(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^{(l)} W^{(l)}) \quad (3.4)$$

- where $H^{(l+1)}$ represents the updated node representation of layer l+1.
- σ represents the sigmoidal or the Relu activation function.
- D is the diagonal matrix degree matrix
- $D^{-\frac{1}{2}}$ is the element-wise inverse square root of D
- where $H^{(l)}$ represents the updated node representation of layer l.
- $W^{(l)}$ Represents the weight matrix for layer l

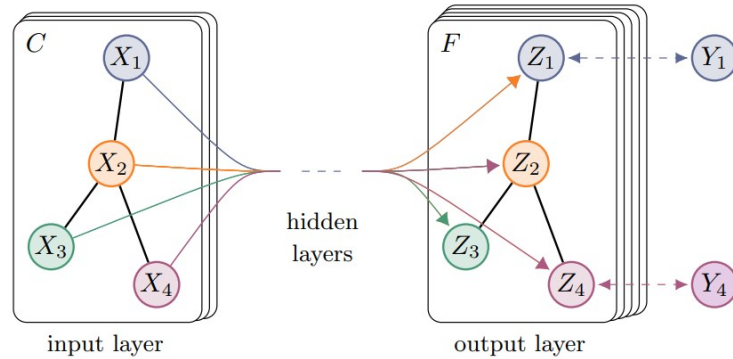


Figure 3.5: Graph Convolutional Network

GCN involves two steps of typical GNNs

- Aggregation:
an aggregation function such as mean/pooling/RNNs can be used in the local neighborhood
- Neural Network
then the multiplication by the matrix is performed along with applying activation functions.

In summary a node embedding is an average of the neighbourhood passed over a Neural Network

- **Message Passing Neural Network (or MPNN)**

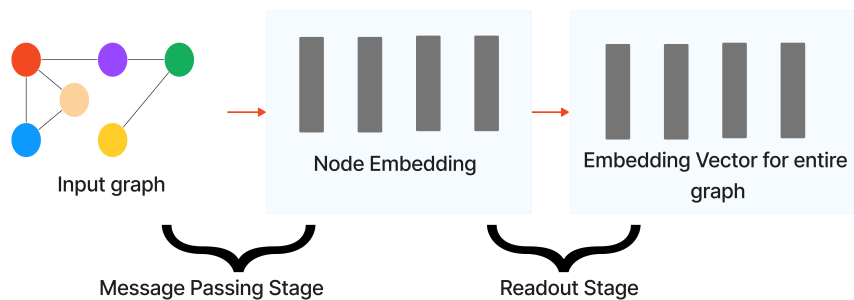


Figure 3.6: A message passing network

(GILMER et al. (2017)) Message Passing Neural Networks can generate node as well as graph embeddings. It consists of two stages:

- Message Passing stage
During this phase the message m for a node v . Then the Node embedding h is updated.
- Readout Stage
In this phase an embedding h is produced for the entire graph using the node embedding from the previous stage. The output is an embedding vector for the entire graph.

MPNNs are computationally expensive and thus used for small datasets or a small subset.

- **Graph Attention (or GAT)**

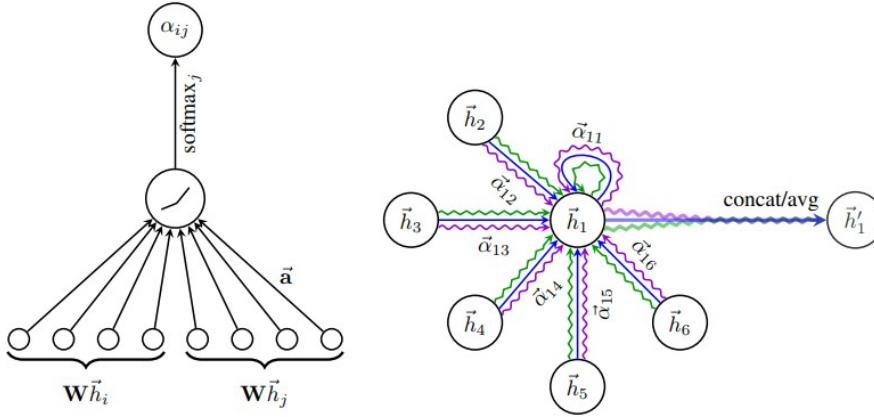


Figure 3.7: Graph Attention Network

(VELIČKOVIĆ et al. (2018)) Similar to GCN however we have an additional function that can calculate weights. The assigned weights mean that the attention function allows more attention to certain nodes(they are important) than other(less important) nodes. It is a combination of attention mechanisms and GCNs. Contrary to GCN the importance of the nodes is not equal. It performs better than Message Passing Networks(MPNs) but not better than GCN in terms of computation power.

$$e_{ij} = \text{LeakyReLU}(a^T [Wh_i || Wh_j]) \quad (3.5)$$

$$a_{ij} = \text{Softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum \exp(e_{ij})} \quad (3.6)$$

- where e_{ij} represents the attention weight between the nodes i and j .
- (LeakyRelu) represents the Leaky Relu activation function for nonlinearity.
- a^T represents the activation weights which are learnable parameters.
- $[Wh_i || Wh_j]$ it represents the concatenations transformed node i and j
- h_i and h_j represents the original node representations for the node i and node j respectively
- $\exp(e_{ij})$ Represents the exponential function multiplying elementwise to the attention scores.
- $\sum \exp(e_{ij})$ Summation of the exponential attention scores
- (Softmax) Softmax layer applied so that the attention weights sum up to one

- **GraphSage**

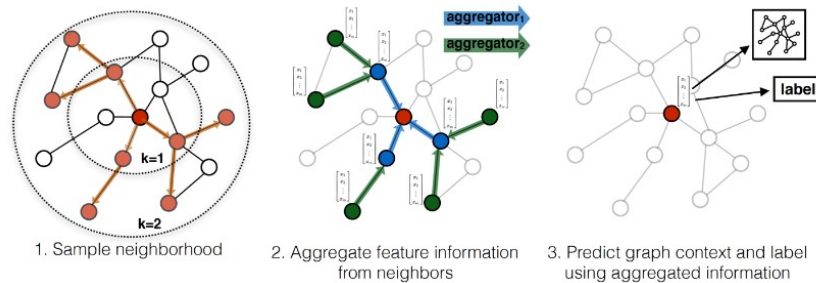


Figure 3.8: GraphSage

(HAMILTON et al. (2017)) GraphSage learns embeddings by sampling and aggregating information local neighborhood of the graph. It uses a mini-batch such that it samples a fixed set of neighborhoods for every node and aggregates their features. Thus it allows for capturing information from the immediate neighborhood and generating node embeddings.

- Sampling

A fixed number of neighbors are sampled for every node in a mini-batch

- Aggregating

In this stage an aggregation can be applied on the sampled node such as Mean/Max or LSTM

In this way, GraphSage can capture local as well as global information from other nodes.

- **Graph Isomerism Network**

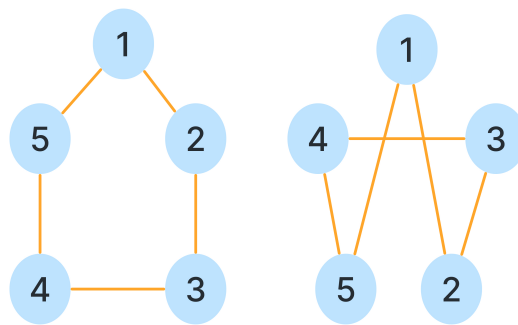


Figure 3.9: Two isomeric graphs

(XU et al. (2019)) The concept of Graph isomerism is based on the Weisfeiler-Lehman (WL)(HUANG und VILLAR (2021)) graph isomorphism test. An isomeric graph means that the two graphs have the same structure and identical connections but there is a permutation of nodes. GIN aggregates and updates node features iteratively. All the nodes begin with the same label but are hashed in repeatedly until the iterations stop when the labels don't change any more. The test can decide whether the graphs are isomeric or not. The resultant graph is very powerful and can perform different GNN tasks like Node classification, link prediction, graph classification, etc.

3.4 Explainers

- **GNNexplainer**

(YING et al. (2019)) One of the most common benchmark explainers, GNNexplainer is a model Agnostic explainer. It provides insights

about the predictions of the training. It produces nodes and/or edge masks highlighting their importance. It can cater to Node classification, Link prediction as well as Graph classification. It involves a few steps

- Training of the model is first done and the predictions are generated
- GNNexplainer takes a sub-graph and the predictions to give explanations
- The results are visualized for example in a heat map.

• GraphMask Explainer

(SCHLICHTKRULL et al. (2022)) The GraphMask Explainer gives us insights by removing the edges with lower importance scores and retaining the edges with higher importance scores. It is similar to sparsity where the goal is to keep the output intact ignoring as many edges as possible.

• PGexplainer

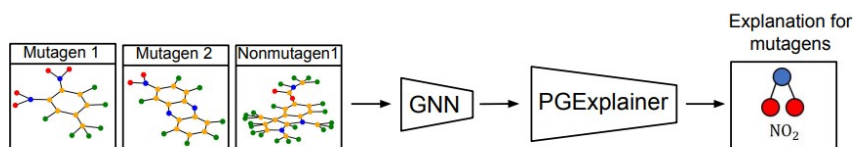


Figure 3.10: PGExplainer

(LUO et al. (2020)) It is another commonly used explainer, used as a benchmark. It is also model agnostic. It uses a Neural Network to parameterize the process of explanation generation. Compared to GNN explainer which can capture a local subset, the nature of PGExplainer means it targets not just the subset but the entire graph. Its performance is better than GNNexplainer in terms of AUC-ROC.

• Captum Explainers

(ADEBAYO et al. (2020)) The Captum explainer is a set of explainers developed for Neural Networks. They can be used in Graph Neural Networks as well. Some of the explainers belonging to gradient-based methods include:

- **Integrated Gradients**

A baseline input is randomly selected or set to zero. Then it computes the gradients using backpropagation of the output with respect to inputs. Finally, the gradients are integrated with the actual input from the baseline input. These Integrated gradients are the importance scores. It highlights the overall input features importance.

- **Saliency**

Another gradient-based approach. Input is assigned from the data that needs analysis. The gradients are computed at the output with respect to the input using backpropagation. The gradients are again the importance scores. It highlights the individual input features that contribute to the output.

Some of the Perturbations methods include:

- **Occlusion**

Given the perturbation nature, occlusion takes input data points or a set of data points and masks different parts of the data. It uses the original data to get the baseline predictions. Different strategies can be used like sliding windows or grid-based masking. The masking strategy is applied to the data to get the predictions of the occluded version. Finally, it is compared with the baseline to measure the importance of the features.

- **PGMExplainer**

(VU und THAI (2020)) Another model agnostic explainer. It constructs a graph. The probabilities are calculated. The unimportant features (with low probabilities) are eliminated. Finally, the explanation is generated with the remaining important features.

- **Build own Explainers**

The PyTorch Geometric Library is underdeveloped and allows the researchers to use their explainers and built them to the PyTorch format so it can work like any other explainer that is already part of the library.

3.5 Explanation evaluation metrics

Several Research papers propose several evaluation metrics.

- **Positive(+) Fidelity**

(AMARA et al. (2022)) The metric captures positive features or elements that to GNN predictions. A positive fidelity close to one means that the positive instances of the model are well aligned with the explanations. A positive fidelity close to one means the explanation is necessary.

$$\textit{Phenomenon}: F_{\text{id}^+} = \frac{1}{N} \sum_{i=1}^N \left(1(\hat{y}_i = y_i) - 1(\hat{y}_i^{\text{Gc}\setminus S} = y_i) \right) \quad (3.7)$$

$$\textit{Model}: F_{\text{id}^+} = 1 - \frac{1}{N} \sum_{i=1}^N \left(1(\hat{y}_i^{\text{Gc}\setminus S} = \hat{y}_i) \right) \quad (3.8)$$

- N represents the number of samples
- \hat{y}_i represents the predicted label for sample number i
- y_i it is ground truth of sample i
- $\hat{y}_i^{\text{Gc}\setminus S}$ represents the predicted label when the sample i is removed from the graph

- **Negative(-) Fidelity**

(AMARA et al. (2022)) Important components of the graph are altered(e.g. node removed) and how stable the prediction is and its consistent. It measures how well negative instances of the model align with the explanations. A negative fidelity close to 1 means that the explanation is sufficient.

$$\textit{Phenomenon}: F_{\text{id}^-} = \frac{1}{N} \sum_{i=1}^N \left(1(\hat{y}_i = y_i) - 1(\hat{y}_i^{\text{Gs}} = y_i) \right) \quad (3.9)$$

$$\textit{Model}: F_{\text{id}^-} = 1 - \frac{1}{N} \sum_{i=1}^N 1(\hat{y}_i^{\text{Gs}} = \hat{y}_i) \quad (3.10)$$

- N represents the number of samples
- \hat{y}_i represents the predicted label for sample number i
- y_i it is ground truth of sample i
- \hat{y}_i^{Gs} represents the predicted label when the sample i is removed from the graph

- **Characterisation Score**

$$\text{charact} = \frac{w_+ + w_-}{\frac{w_+}{f_{id+}} + \frac{w_-}{1-f_{id-}}} = \frac{(w_+ + w_-) \times f_{id+} \times (1 - f_{id-})}{w_+ \times (1 - f_{id-}) + w_- \times f_{id+}} \quad (3.11)$$

- w_+ represents the positive weights
- w_- represents the negative weights
- f_{id-} represents the obtained negative fidelity scores
- f_{id+} represents the obtained positive fidelity scores

(AMARA et al. (2022)) It is a harmonic mean between the positive and the negative fidelity. It is similar to the F1 score in the confusion matrix calculated from Precision and Recall. A higher characterization score signifies that the model is accurate and the explanations capture the important factors towards the predictions. It balances both positive and negative fidelity, which means the characterization of the explanation is both necessary and sufficient.

- **Fidelity curve auc**

(AMARA et al. (2022)) It is a common evaluation metric. It shows the trade-off between the true positive rate and the false positive rate. A higher (ideally .0.5) depicts that the model can well distinguish between the false and the true instances.

- **Unfaithfulness**

$$\text{GEF}(y, \hat{y}) = 1 - \exp(-\text{KL}(y||\hat{y})) \quad (3.12)$$

- y represents the ground truth distribution
- \hat{y} represents the predicted label distribution
- $\text{KL}(y||\hat{y})$ it represents Kullback-Leibler divergence between the ground truth and the prediction distributions

(AGARWAL et al. (2023)) The probability vector of the original graph and the masked subgraph is obtained. The Kullback-Leibler (KL) divergence score is calculated between the masked and the original outputs. Probabilities variables or vectors are obtained for both.

- **Accuracy**

A basic metric to know the number of correct predictions. Explanations are compared with the Ground Truth. This is the same as the one derived from the confusion matrix.

- **Sparsity**

Only important features are used for explanations and unimportant features are not included in the explanations so the Graph is smaller than the original.

4

Implementation and Evaluation

4.1 Configuration

4.1.1 System Configuration

A virtual machine that can be accessed remotely was obtained from a high-performance PC of the university. The main purpose of HPC use to train the models and the explainers. The training takes a huge chunk of time and resources. The GPUs are well known to perform well in Deep Learning. (The final application does not require GPU and HPC). The following are the CPU specifications:

Table 4.1: System configuration

Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s):	128
Thread(s) per core:	2
Core(s) per socket:	32
Socket(s):	2
Model name:	AMD EPYC 7532 32-Core Processor
Frequency boost:	enabled
CPU MHz:	1500.000
CPU max MHz:	2400.0000
CPU min MHz:	1500.0000
Virtualization:	AMD-V

4.1.2 Languages, Tools, and Frameworks

- **Python** language is implied in this thesis. It is the most widely used language in AI. It is open source. PyPI is a repository where the community shares libraries. Hence strong support is available.
- **Visual Studio Code** is used as a compiler. It is very versatile as it allows extensions of other tools to be downloaded. In this project, it provided multiple functions:
 - Remote desktop connection to the Virtual machine provided
 - it allows file explorer extensions in SSH remote desktop
 - Not only can it include the Python compiler but it can work like a notebook incorporating the Ipython kernel separately
 - A local virtual environment for the workspace, keeping it separate from other environments
 - As described later, it also contains its terminal and can be used to run the application server

- **PyTorch**

(PASZKE et al. (2019)) PyTorch is a deep learning framework for Python. It is the preferred platform of the researchers. The alternative to PyTorch is Tensorflow (ABADI et al. (2016)) is better for the non - research applications. For the GNN the following PyTorch-based frameworks were tried and tested:

- **Dive into Graphs (or DIG)**
(LIU et al. (2021)) available at (DIVE INTO GRAPHS (2023))
- **Deep Graph Library (or DGL)**
(WANG et al. (2019)) available at (DEEP GRAPH LIBRARY (2023))
- **PyTorch Geometric (PyG)**
(FEY und LENSSEN (2019)) available at (PYTORCH GEOMETRIC (2023))

We will however use simply PyG because each library uses object-oriented frameworks and using the frameworks simultaneously is not possible even though we require the capability and different implementation of every library. Because PyG is from the developers of

PyTorch itself, there is a general expectation that in the future it will gain superiority over the others. Otherwise, in the nascent domain of GNN, considering all three libraries, one is better in some areas but lacks the other ones in other areas, and so on.

- **Streamlit**

(STREAMLIT) Streamlit is an easy and simple application platform. It has good integration with Machine learning implementations. It is used for fast prototyping and works on .py extensions rather than huge frontends and backends.

The following languages and their versions are mentioned. The versions are consistent for every part of the thesis.

Table 4.2: Languages and Frameworks Version

Python	3.10.11
PyTorch	2.0.1(py3.10 cuda 11.8 cudnn 8.7.0)
pyg	2.3.0
tqdm	4.65.0
scikit-learn	1.2.2
scipy	1.10.1
streamlit	1.24.1
matplotlib	3.7.1
torchmetric	0.11.2
torchvision	0.15.2
torchaudio	2.0.2
ipykernel	6.15.0
ipython	8.13.2
ipywidgets	8.0.4

4.1.3 The experiment settings

Input configuration

For simplicity, we select the following configuration:

- Task: Node Classification
- Datasets: Cora, Citeseer, and Pubmed

these datasets contain single homogeneous graph citation networks suitable for Node classification

- Models: four possible GNN architectures GCN, GAT, GIN, and GraphSage are used
- Explainer use includes GNNExplainer and GraphMask.
- To maintain standardization, we select the node index as 10. This means for every setting we use the same node with regards to the selected dataset like Cora. So each time we use for example Cora we use the same Node (no. 10).

Table 4.3: Node Classification combinations

	Dataset	Explainer	Model
1	Cora	GNNExplainer	GCN
2	Cora	GNNExplainer	GAT
3	Cora	GNNExplainer	GIN
4	Cora	GNNExplainer	GraphSage
5	Cora	GraphMask	GCN
6	Cora	GraphMask	GAT
7	Cora	GraphMask	GIN
8	Cora	GraphMask	GraphSage
9	Citeseer	GNNExplainer	GCN
10	Citeseer	GNNExplainer	GAT
11	Citeseer	GNNExplainer	GIN
12	Citeseer	GNNExplainer	GraphSage
13	Citeseer	GraphMask	GCN
14	Citeseer	GraphMask	GAT
15	Citeseer	GraphMask	GIN
16	Citeseer	GraphMask	GraphSage
17	Pubmed	GNNExplainer	GCN
18	Pubmed	GNNExplainer	GAT
19	Pubmed	GNNExplainer	GIN
20	Pubmed	GNNExplainer	GraphSage
21	Pubmed	GraphMask	GCN
22	Pubmed	GraphMask	GAT
23	Pubmed	GraphMask	GIN
24	Pubmed	GraphMask	GraphSage

the combinations are all available in PyTorch Geometric Library. This is closely associated with PyTorch itself which is Deep Learning framework. Currently, it is still under development. Even though it also currently has limited implementations and lack of helping resources, it is still important because, it provides classes where you can implement your datasets, your GNN explainers. This means that not only the library will incorporate more research papers into its platforms but also gives researchers the ability to use their own explainers and datasets.

Evaluation metrics

Upon training the models and the Explainers in the GNN xEval application based on Streamlit, we will be able to get the performance metrics of the explainers. The performance metrics chosen include:

- Fidelity(positive and negative)
- Characterisation Score
- Unfaithfulness

4.2 Implementation

4.2.1 User Interface

Figure 4.1 shows the simple interface of the Streamlit application. It contains the three possible options with drop-down menus. The application takes a bit of time when initialized by the terminal in VS code. The options are :

- Dataset
- explainer
- Architecture

GNN xEval

Select a dataset

CORA

Select an Explainer

GNNExplainer

Select an GNN architecture

GCN

Explain

Figure 4.1: GNN xEval application

As shown in 4.2, The xEval application dorp down menus. The train does not start until the explain button is clicked.

GNN xEval

Select a dataset

CORA

Select an Explainer

GNNExplainer

Select an GNN architecture

GCN

GCN

GAT

GIN

GraphSage

Figure 4.2: GNN xEval choosing the options

In 4.3 at the top right we can see the running sign. This occurs after the explain button is picked. The entire process of training the GNN and its explainer along with the evaluation metrics is being worked on.

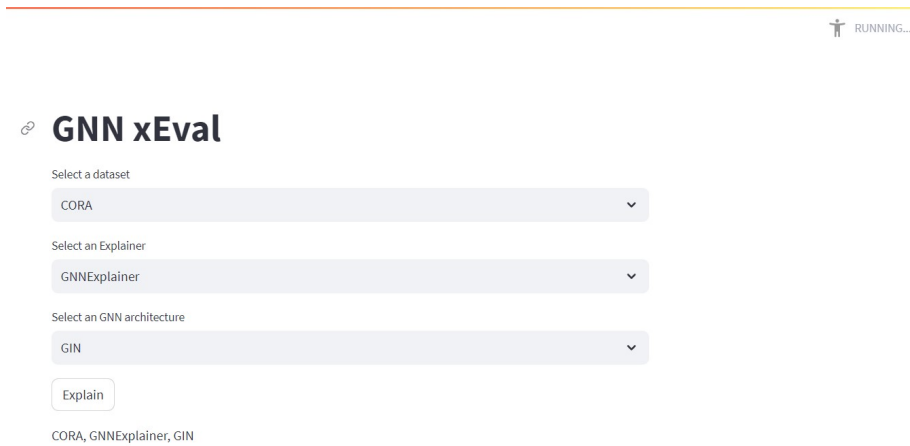


Figure 4.3: GNN xEval in training (shows running at top right for the duration of training)

Finally in 4.4 we can see the results. The training phase takes about one minute depending on the configuration.

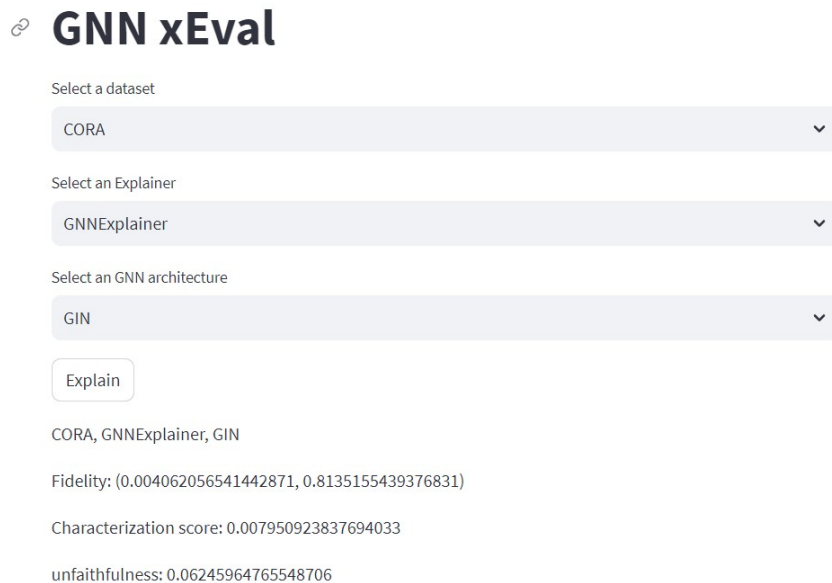


Figure 4.4: Application displaying results of the evaluation

4.2.2 The Process

The steps for the entire process are as follows:

1. load the dataset
2. Define the GNN architecture
3. Train the GNN model
4. Select the explainer
5. Set the relevant configuration
6. train on the node to explain
7. Display the relevant performance metrics

4.2.3 The Implementation

The thesis implementation can be divided into phases.

Firstly, in VS Code, Python, and Ipython kernel need to be added along with SSH client. The SSH settings and credentials are provided by the University. Alternatively, the virtual environment can also be from Anaconda (miniconda)(ANACONDA (2023)). The virtual environment is necessary, otherwise, the libraries will be installed on the root of the operating system.

In the next phase the libraries following libraries will be added to the virtual environment.

1. PyTorch
2. PyTorch Geometric
3. Matplotlib (for visualisations)

This order is important otherwise it would create problems.

There is a bifurcation here.

- Running and testing GNN and its explainer individually
- The same environment can be replicated to be used for Streamlit Application. (Also adding Streamlit libraries and other dependencies to the environment.)

For the first direction, individual codes were written for the following:

1. Load the datasets

Here I ran and tested a lot of datasets for the research. These datasets are loaded into PyG explainer class. The statistics of the data were also printed in the cell output of the notebook.

2. GNN models

Then I trained them on four different GNN models, GCN, GAT, GIN, GraphSage separately.

3. Explainers

Then I implemented the codes for different explainers including GNN explainer and tested them separately. Many of them face some library-related issues. Some of them worked correctly after debugging and correcting, whereas others still faced problems with the PyG library. Explainer training usually takes more time to train than the model training of GNN.

4. Evaluation Metrics

I wrote the codes and investigated different metrics including Fidelity. I faced some with regard to the fact that some metrics require the Ground Truth to provide the confusion metrics while other metrics did not relate to the ground truth. While some datasets including the synthetics provide the ground truth others don't. However, it is not easy to find their documentation whether they include the ground truth or not. In the end, I had to drop many unimportant metrics, that had library-related issues and those based on the ground truth (as they were only relevant to individual cases). For the experiments I selected:

- +Fidelity
- - Fidelity
- Characterisation Score
- Unfaithfulness

GNN xEval Streamlit application For the second part, once I got the working results, it's time to implement them on a separate Streamlit application(a bifurcation of the virtual environment discussed earlier). The basic structure is created and discussed in the previous section. The Streamlit application requires a working .py extension rather than the iPython kernel. Hence a separate file was created. The running codes were extracted from the earlier implementations. The structure of the Python file is different and requires some changes in the code. Furthermore, setting it in a way that Streamlit can compile and process its required changes(It would be a running application server).

Finally the combinations that are pointed out in the earlier section are tried and tested on the GNN xEval and the detailed results are mentioned in the references in the end A.1.

4.3 Results and discussion

4.3.1 The Application

It is simple and user-friendly. The following diagram shows a glimpse of the application and its capabilities:



GNN xEval

Select a dataset
CORA

Select an Explainer
GNNExplainer

Select an GNN architecture
GCN

Explain

Figure 4.5: GNN xEval application

The application has an easy-to-use interface. We can select several options from the drop-down options and then press Explain button to start the training. There is a bar (figure given in the detailed results) at the top right that says running when it's training. Usually, the training takes not more than one minute. The results are finally displayed in terms of metrics.

4.3.2 Characterisation Score

Instead of comparing the Fidelity score positive and negative, the characterization score is the harmonic mean of both the Fidelities. Hence we directly discuss the Charact. A better characterization score means the explanations are sufficient and necessary.

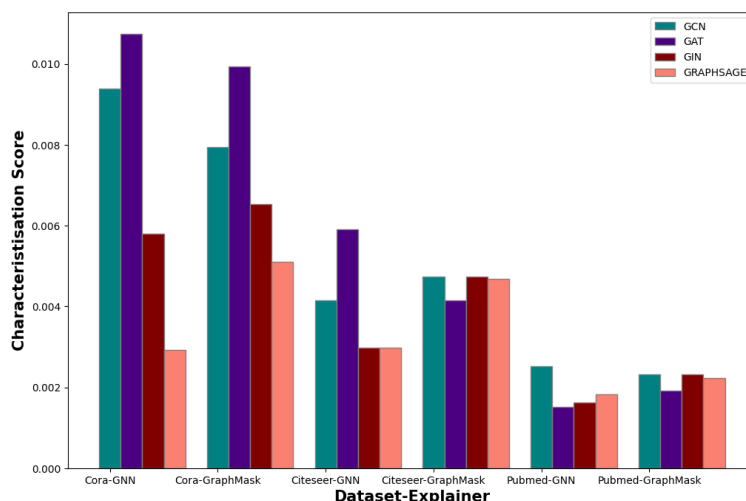


Figure 4.6: Characterisation Score

- The comparison of characterization score shows that applying GAT on the Cora dataset using a GNN explainer has the highest score showing that it has a more sufficient explanation and necessary explanation than other configurations.
- The CORA dataset has been in general better explanations than others in terms of the Characterisation score.
- GraphSage generally does not fare very well
- However GAT explanations generally do exceptionally high or fare comparable to others. It does not usually fare less

4.3.3 Unfaithfulness

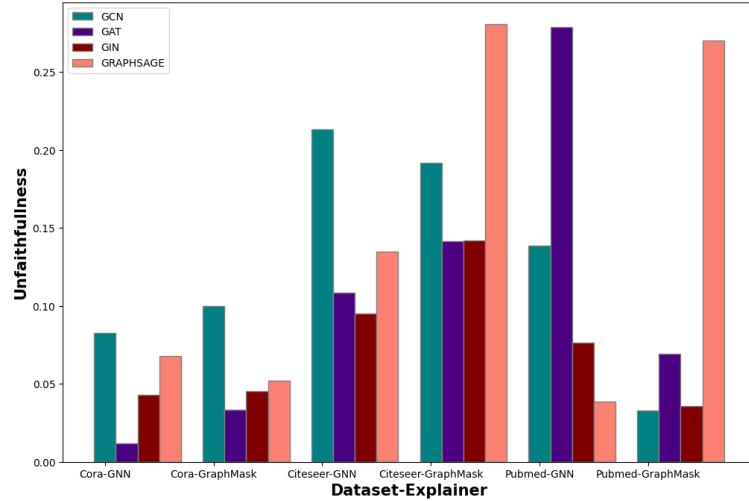


Figure 4.7: Unfaithfulness

Then the higher the values of Unfaithfulness mean the higher the degree of Unfaithfulness.

- The comparison of unfaithfulness shows that explanations are more unfaithful when using GraphSage when used with GraphMask explainer except in the Pubmed dataset.
- Explanation with GCN has been generally more unfaithful than other cases except when used in Pubmed with GraphMask.
- GAT only has been less Unfaithful when used in the Cora dataset than others.

Hence the researchers can get the results from the application. With the PyG library, they can include their dataset and explainer and use them in this application.

5

Conclusions and Future Work

5.1 Conclusions

The thesis aimed to show the concept that a unified application platform that has possible different settings. It can have different configurations with variations in datasets, GNN models, Explainers, and explainer evaluation metrics. The application can produce results that can be compared with other results. A user simply selects an explainer, dataset, and the GNN model.

Some research questions were introduced in the introduction chapter1. An understanding of sufficient and necessary explanations is provided and in the results, we can see some sufficient and necessary explanations.

The thesis presented an overview of the dataset types like homogeneous and heterogeneous, and some of the common datasets used that are in the libraries. A number of statistics are provided to have a comparison and determine which graph is suitable for which GNN task.

As far as the GNN model architectures are concerned, four relevant tasks including GCN have been discussed and implemented. They are all part of the GNN xEval application.

The GNN tasks are presented however due to standardization of the results, only Node Classification was selected. The explainers mentioned (like GraphMask) were tested, however, some of them dropped due to various reasons including bugs (like PGMexplainer). Hence GNN explainer and GraphMask were used. The combination provided in the table 4.3 (similar to A.1) works without trouble.

The explainer evaluations also suffer from library problems. Only a handful of them are provided in the PyG. Further, the existing libraries for GNN are shown. All the datasets, explainers, GNN tasks, as well as explainer evaluation metrics, are selected.

The unified user-friendly Streamlit application named GNN xEval is presented. The GNN xEval in a very simple interface allows the user to select the relevant configuration and allows the explainer to train. The results of the evaluation metric are then displayed in text form. This application takes a small chunk of around a minute on a CPU device. The results of a few metrics are then displayed. Hence the user can then use the outputs to compare the results with the ones they acquired previously.

5.2 Limitations

During the progression of the thesis several constraints were faced. Some of the limitations were overcome while others require a change of approach. Some limitations meant that the entire part was dropped. Some of the limitations are mentioned below

1. During the literature review, several research papers had no associated or restricted code provided. The ones that provided code rarely worked, as they have very specific hardware configurations (for PyTorch GPU binaries)
2. Although included in the library PyG, the PG explainer, Captum explainer even the PGM explainer gave errors upon training for an explanation, whereas the GNN explainer and the Graph Mask explainer worked just fine. The unusual data type error means the library is still buggy.
3. The Streamlit application cannot run Matplotlib for plotting graphs perfectly, hence feature visualization and importance graphs were dropped
4. The GPU available is accessible remotely, however, it has to be either accessed from University premises or via VPN as part of the university. However, there are connectivity issues and upon some event occurrence, the entire training(which takes a long time) phase is lost.

5. The general problem with GNN and its explainability is that the resources are scattered, and different platforms, provide different resources. For example, Sparsity and Fidelity metrics are available in DGL but sparsity is not available in another library. The other library has unfaithfulness and accuracy but the first one does not.
6. The GNN xEval application is currently based on CPU. To include more explainers, and some particular evaluation metrics like fidelity vs time and identity Area under curve (receiver operator characteristics) it requires GPU. Otherwise, it takes either a lot of time or some GPU-intensive tasks are simply dropped.
7. Limited literature exists on this matter. For instance, merely examining the data statistics does not allow us to distinguish whether it is utilized for tasks like node classification, link prediction, or graph classification.

5.3 Future Work

Some of the future work can include:

1. More evaluation metrics can be manually written as a function. Also, with PyTorch Geometric under development, we would see more and more stuff incorporated into the library. All of this can be mirrored and reflected in the Graph xEval Application.
2. Also, more explainers can be added to the platform either using the library's custom explainer or waiting for the developer to include more in the future.
3. The GNN xEval application can provide two or more explanations simultaneously and provide a comparison within the application.
4. Because of the different nature of tackling the explanation problem. The explainers usually do not follow some general metrics. Expanding the current number of explainers and metrics in GNN xEval can help researchers compare more explainers.
5. Apart from Node Classification; Link prediction, and Graph Classification can be targeted.

6. Wider range of datasets can be included.
7. More options can be included for the configuration like explainer type "phenomenon" or "model".
8. A time-based analysis can be included such that GIN always performs faster than other models. This information can be a helpful metric for the researchers.
9. A complex interface can be introduced and certain checks, e.g. Attention explainer cannot run on GCN or PG explainer only runs on the phenomenon.
10. More visualization can be incorporated into the application. There would be some workarounds in the Streamlit application to include the visualization including the one specific to GNN like using the library Networkx(HAGBERG et al. (2008)).

A

Detailed Results

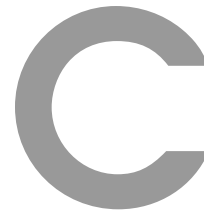
Table A.1: Results for explainer evaluation using certain metrics

	Dataset	Explainer	Model	Fidelity (+,-)	Charact score	Unfaithfulness
1	Cora	GNNExplainer	GCN	0.0048006, 0.7821270	0.0093942	0.0829030
2	Cora	GNNExplainer	GAT	0.0055391, 0.8220088	0.0107438	0.0117922
3	Cora	GNNExplainer	GIN	0.0029541, 0.8415805	0.0058002	0.0428004
4	Cora	GNNExplainer	GraphSage	0.0014771, 0.8770310	0.0029191	0.0680301
5	Cora	GraphMask	GCN	0.0040620, 0.8268094	0.0079379	0.1001273
6	Cora	GraphMask	GAT	0.0051698, 0.8689069	0.0099474	0.0334298
7	Cora	GraphMask	GIN	0.0033234, 0.8205317	0.0065261	0.0451869
8	Cora	GraphMask	GraphSage	0.0025849, 0.8194239	0.0050969	0.0522613
9	Citeseer	GNNExplainer	GCN	0.0021039, 0.8569281	0.0041469	0.2135248
10	Citeseer	GNNExplainer	GAT	0.0030056, 0.8331830	0.0059049	0.1083208
11	Citeseer	GNNExplainer	GIN	0.0015028, 0.8232641	0.0029803	0.0950056
12	Citeseer	GNNExplainer	GraphSage	0.0015028, 0.8256687	0.0029800	0.1346806
13	Citeseer	GraphMask	GCN	0.0024045, 0.8334836	0.0047406	0.1916656
14	Citeseer	GraphMask	GAT	0.0021039, 0.8154493	0.0041605	0.1416324
15	Citeseer	GraphMask	GIN	0.0024045, 0.8412984	0.0047373	0.1419266
16	Citeseer	GraphMask	GraphSage	0.0024045, 0.9161406	0.0046750	0.2807985
17	Pubmed	GNNExplainer	GCN	0.0012679, 0.5880712	0.0025281	0.1387156
18	Pubmed	GNNExplainer	GAT	0.0007607, 0.5649946	0.0015189	0.2790257
19	Pubmed	GNNExplainer	GIN	0.0008114, 0.7092356	0.0016183	0.0764944
20	Pubmed	GNNExplainer	GraphSage	0.0009129, 0.6379773	0.0018212	0.0384463
21	Pubmed	GraphMask	GCN	0.0011665, 0.5644367	0.00232681	0.0327048
22	Pubmed	GraphMask	GAT	0.0009636, 0.6277831	0.0019222	0.0694633
23	Pubmed	GraphMask	GIN	0.0011665, 0.6263123	0.0023257	0.0358163
24	Pubmed	GraphMask	GraphSage	0.0011157, 0.5686970	0.0022258	0.2701116

B

Abbreviations and Notations

Acronym	Meaning
AI	Artificial Intelligence
AUC	Area under curve
BA	Barabasi-Albert (BA) base graph
BIC	Bayesian Information Criterion
Charact	Characterisation Score
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GIN	Graph Isomerism Network
GNN	Graph Neural Network
GPU	Graphic Processing Unit
HPC	High Performance computer
MPNN	Message Passing Neural Network
NLP	Natural Language Processing
NN	Neural Network
RNN	Recurrent Neural Network
ROC	Receiver operator characteristics
SSH	Secure Shell
TGNN	Temporal Graph Neural Network
VPN	Virtual Private Network
VSCoDe	Visual Studio Code
WL	Weisfeiler-Lehman graph isomorphism test
X	dataset
XAI	Explainable Artificial Intelligence



List of Figures

2.1 GNN Explainers classification and examples	7
2.2 A simple Graph	12
2.3 A simple directed graph	13
2.4 A simple undirected graph	14
2.5 A Homogeneous Graph	14
2.6 A Heterogeneous Graph	15
2.7 The embedding function f	16
2.8 Input Graph with target node as node labelled 2	17
2.9 The computation graph for GNN	18
3.1 Node classification	24
3.2 Link Prediction	25
3.3 Graph Classification	26
3.4 Community Detection	26
3.5 Graph Convolutional Network	28
3.6 A message passing network	28
3.7 Graph Attention Network	29
3.8 GraphSage	30
3.9 Two isomeric graphs	31
3.10 PGExplainer	32
4.1 GNN xEval application	42

4.2 GNN xEval choosing the options	42
4.3 GNN xEval in training (shows running at top right for the duration of training)	43
4.4 Application displaying results of the evaluation	43
4.5 GNN xEval application	46
4.6 Characterisation Score	47
4.7 Unfaithfulness	48

D

List of Tables

2.1 Homogeneous graphs vs Heterogeneous graphs	15
3.1 Statistics for common Homogeneous graph datasets	22
4.1 System configuration	37
4.2 Languages and Frameworks Version	39
4.3 Node Classification combinations	40
A.1 Results for explainer evaluation using certain metrics	53

E

Bibliography

- [ABADI et al.] **TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems**. <https://www.tensorflow.org>.
- [ADEBAYO et al. 2020] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim, P. Medina, A. Qin und S. Zheng. **Captum: A Unified and Generic Model Interpretability Library for PyTorch**. 2020. GitHub repository.
- [AGARWAL et al. 2023] C. Agarwal, O. Queen, H. Lakkaraju und M. Zitnik. 2023, **Evaluating Explainability for Graph Neural Networks**.
- [AMARA et al. 2022] K. Amara, R. Ying, Z. Zhang, Z. Han, Y. Shan, U. Brandes, S. Schemm und C. Zhang. 2022, **GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks**.
- [ANACONDA] **miniconda**. <https://docs.conda.io/en/latest/miniconda.html>. Accessed: 20 07, 2023.
- [AZZOLIN et al. 2023] S. Azzolin, A. Longa, P. Barbiero, P. Liò und A. Passerini. **Global Explainability of GNNs via Logic Combination of Learned Concepts**. 2023.
- [DAI und WANG 2021] E. Dai und S. Wang. **Towards Self-Explainable Graph Neural Network**. 2021.
- [DEEP GRAPH LIBRARY] **Deep Graph Library**. <https://www.dgl.ai/>. Accessed: 20 07, 2023.

- [DIVE INTO GRAPHS] **Dive into Graphs Documentation.** <https://diveintographs.readthedocs.io/en/latest/>. Accessed: 20 07, 2023.
- [DUVAL und MALLIAROS 2021] A. Duval und F. D. Malliaros. **GraphSVX: Shapley Value Explanations for Graph Neural Networks.** 2021.
- [FEY und LENSSEN 2019] M. Fey und J. E. Lenssen. **Fast Graph Representation Learning with PyTorch Geometric.** Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.
- [FU et al. 2020] X. Fu, J. Zhang, Z. Meng und I. King. **MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding.** In: Proceedings of The Web Conference 2020. 2020, ACM.
- [GILMER et al. 2017] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals und G. E. Dahl. **Neural Message Passing for Quantum Chemistry.** In: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1263–1272.
- [HAGBERG et al. 2008] A. A. Hagberg, D. A. Schult und P. J. Swart. **Exploring network structure, dynamics, and function using NetworkX.** Proceedings of the 7th Python in Science Conference (SciPy2008), pp. 11–15, 2008.
- [HAMILTON et al. 2017] W. Hamilton, R. Ying und J. Leskovec. **Inductive Representation Learning on Large Graphs.** In: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
- [HARPER und KONSTAN 2015] F. M. Harper und J. A. Konstan. **The MovieLens Datasets: History and Context.** ACM Transactions on Interactive Intelligent Systems, Vol. 5(4), 2015.
- [HE et al. 2022] W. He, M. N. Vu, Z. Jiang und M. T. Thai. **An Explainer for Temporal Graph Neural Networks.** 2022.
- [HU et al. 2020] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta und J. Leskovec. **Open Graph Benchmark: Datasets for Machine Learning on Graphs.** arXiv preprint arXiv:2005.00687, 2020.
- [HUANG und VILLAR 2021] N. T. Huang und S. Villar. **A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants.** In: ICASSP 2021 - 2021

-
- IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2021, IEEE.
- [KERSTING et al. 2016] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel und M. Neumann. **Benchmark Data Sets for Graph Kernels**. 2016.
- [KIPF und WELLING 2016] T. N. Kipf und M. Welling. **Semi-Supervised Classification with Graph Convolutional Networks**. arXiv preprint arXiv:1609.02907, 2016.
- [LESKOVEC und KIPF 2021] J. Leskovec und T. Kipf. **CS224W: Machine Learning with Graphs**. 2021. Lecture slides.
- [LIU et al. 2021] M. Liu, Y. Luo, L. Wang, Y. Xie, H. Yuan, S. Gui, H. Yu, Z. Xu, J. Zhang, Y. Liu, K. Yan, H. Liu, C. Fu, B. M. Oztekin, X. Zhang und S. Ji. **DIG: A Turnkey Library for Diving into Graph Deep Learning Research**. Journal of Machine Learning Research, Vol. 22(240):1–9, 2021.
- [LU und LI 2020] Y.-J. Lu und C.-T. Li. **GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media**. 2020.
- [LUO et al. 2020] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen und X. Zhang. 2020, **Parameterized Explainer for Graph Neural Network**.
- [PASZKE et al.] **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. <https://pytorch.org>.
- [PYTORCH GEOMETRIC] **PyTorch Geometric**. <https://pytorch-geometric.readthedocs.io/en/latest/index.html>. Accessed: 20 07, 2023.
- [ROZEMBERCZKI et al. 2020] B. Rozemberczki, O. Kiss und R. Sarkar. **Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs**. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), p. 3125–3132. 2020, ACM.
- [SCHLICHTKRULL et al. 2022] M. S. Schlichtkrull, N. D. Cao und I. Titov. **Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking**. 2022.

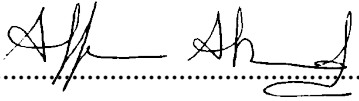
- [SCHNAKE et al. 2022] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schutt, K.-R. Muller und G. Montavon. **Higher-Order Explanations of Graph Neural Networks via Relevant Walks**. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 44(11):7581–7596, 2022.
- [STREAMLIT] **Streamlit**. <https://streamlit.io/>. Accessed: 20 07, 2023.
- [SUN et al. 2017] H. Sun, J. Hu, Y. Li, M. Qu, J. Tang, X. Huang und J. Han. **Cross-lingual entity alignment via joint attribute-preserving embedding**. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), 2017, pp. 1511–1517.
- [TONG et al. 2016] W. Tong, H. Hong, H. Fang, G. Liu, Y. Li, Y. Shi und J. Li. **Prediction of Mutagenicity of Chemical Compounds Using Ensemble Learning Methods**. Molecules, Vol. 21(7), 2016.
- [VELIČKOVIĆ et al. 2018] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò und Y. Bengio. **Graph Attention Networks**. 2018.
- [VU und THAI 2020] M. N. Vu und M. T. Thai. 2020, **PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks**.
- [WANG et al. 2019] M. Wang, D. Zheng, S. Zhang, J. Zhou, C. Ma, J. Shao, Z. Ye, Q. Guo, H. Xiong und Z. Zhang. **Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks**. 2019, In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.
- [XU et al. 2021] J. Xu, Minhui, Xue und S. Picek. **Explainability-based Backdoor Attacks Against Graph Neural Networks**. 2021.
- [XU et al. 2019] K. Xu, W. Hu, J. Leskovec und S. Jegelka. **How Powerful are Graph Neural Networks?** 2019.
- [YANG et al. 2016] Z. Yang, W. W. Cohen und R. Salakhutdinov. 2016, **Revisiting Semi-Supervised Learning with Graph Embeddings**.
- [YING et al. 2019] R. Ying, D. Bourgeois, J. You, M. Zitnik und J. Leskovec. **GNNExplainer: Generating Explanations for Graph Neural Networks**. 2019.

[YUAN et al. 2020] H. Yuan, J. Tang, X. Hu und S. Ji. **XGNN: Towards Model-Level Explanations of Graph Neural Networks**. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020, ACM.

[YUAN et al. 2022] H. Yuan, H. Yu, S. Gui und S. Ji. **Explainability in Graph Neural Networks: A Taxonomic Survey**. 2022.

Declaration of Academic Integrity

I hereby declare that I have written the present work myself and did not use any sources or tools other than the ones indicated.

Datum: 23.07.2023.....
(Signature)