TECHNISCHE UNIVERSITÄT MÜNCHEN
TUM SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

Combining Symbolic and Sub-symbolic Methods for explaining Graph Neural Networks

Anna Katharina Himmelhuber

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung einer

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. Cristina Piazza

Prüfende der Dissertation:

1. Hon.-Prof. Dr.-Ing. Thomas Runkler
2. apl. Prof. Dr. Georg Groh

Die Dissertation wurde am 23.10.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 04.03.2024 angenommen.

Abstract

Graph Neural Networks (GNNs) are AI models that have become increasingly popular since many real-world data can be represented as graphs. However, like other connectionist models, GNNs lack transparency. In this thesis, we argue for the need for explanation methods of GNNs that come with certain properties including naturalness (explanation in natural language), sensitivity (matching user context), reference to graph topology (explanations using graph properties such as substructures) and fidelity (accurate correspondence with the model to be explained). While a number of sub-symbolic GNN explanation methods and model-agnostic symbolic explanation methods exist, they don't fulfill said properties simultaneously.

We introduce three different types of explanation models that combine symbolic and sub-symbolic elements, providing subgraph explanations, non-ontological and ontological explanations. We have two different use cases to evaluate the explanation models, the first dataset (molecular chemistry) is a widely used benchmark in the literature, while the second dataset was created in an industrial setting with cybersecurity applications. The method RERE outperforms the state of the art in terms of sensitivity, SUBGREX in terms of reference to graph topology and OntExplainer by satisfying all four identified properties.

From our experiments, we can draw a number of conclusions, including the necessity of a certain level of domain expertise in the explanation modelling process and the effect of integration such domain knowledge on the properties sensitivity and reference to graph topology. Furthermore, the integration of sub-symbolic and symbolic explainer methods, while increasing complexity, results in a level of explainability for GNNs, that cannot be achieved by either approach alone. Specifically regarding the properties fidelity, sensitivity and reference to graph topology. Overall, when choosing an explanation approach for Graph Neural Networks, there is no one-size-fits-all approach, but a number of different criteria that should be taken into account, such as use case risk, availability of structured domain knowledge and whether local and global explanations are needed.

Kurzfassung

Graph Neural Networks (GNN) sind KI-Modelle, die immer beliebter werden, da viele reale Daten als Graphen dargestellt werden können. Wie bei anderen konnektionistischen Modellen mangelt es jedoch auch GNNs an Transparenz. In dieser Arbeit argumentieren wir für die Notwendigkeit von Erklärungsmethoden für GNNs, die bestimmte Eigenschaften aufweisen, darunter Naturalness (Erklärungen in natürlicher Sprache), Sensitivity (Anpassung an den Benutzerkontext), Bezug auf die Graphentopologie (Erklärungen unter Verwendung von Grapheneigenschaften wie Unterstrukturen) und Fidelity (genaue Übereinstimmung mit dem zu erklärenden Modell). Es gibt zwar eine Reihe von subsymbolischen GNN-Erklärungsmethoden und modellagnostische symbolische Erklärungsmethoden, doch erfüllen sie nicht alle der genannten Eigenschaften.

Wir stellen drei verschiedene Arten von Erklärungsmodellen vor, die symbolische und subsymbolische Elemente kombinieren und subgraphische, nicht-ontologische und ontologische Erklärungen liefern. Wir haben zwei verschiedene Anwendungsfälle, um die Erklärungsmodelle zu evaluieren. Der erste Datensatz (Molekularchemie) ist ein weit verbreiteter Benchmark in der Literatur, während der zweite Datensatz in einem industriellen Umfeld mit Cybersicherheitsanwendungen erstellt wurde. Die Methode RERE übertrifft den Stand der Technik hinsichtlichs der Sensitivity, SUBGREX hinsichtlich des Bezugs auf die Graphentopologie und OntExplainer indem alle vier identifizierten Eigenschaften erfüllt werden.

Aus unseren Experimenten können wir eine Reihe von Schlussfolgerungen ziehen, darunter die Notwendigkeit eines gewissen Maßes an Domänenwissen im Erklärungsmodellierungsprozess und die Auswirkung der Integration solchen Domänenwissens auf die Eigenschaften Sensitivity und Bezug auf die Graphentopologie. Darüber hinaus führt die Integration von subsymbolischen und symbolischen Erklärungsmethoden zwar zu einer Erhöhung der Komplexität, aber auch zu einem Grad an Erklärbarkeit für GNNs, der mit keinem der beiden Ansätze allein erreicht werden kann. Dies gilt insbesondere für die Eigenschaften Fidelity, Sensitivity und Bezug auf die Graphentopologie. Insgesamt gibt es bei der Wahl

eines Erklärungsansatzes für GNNs keinen Einheitsansatz, sondern eine Reihe verschiedener Kriterien, die berücksichtigt werden sollten, wie z. B. das Risiko des Anwendungsfalls, die Verfügbarkeit von strukturiertem Domänenwissen und die Frage, ob lokale und globale Erklärungen erforderlich sind.

Contents

1 Introduction

Graph Neural Networks (GNNs) are AI models that have become increasingly popular since many real-world data can be represented as graphs, such as social networks, chemical molecules, and financial data. GNNs are a class of artificial neural networks in the area of machine learning, designed to perform predictions on data described by graphs.

An important criterion for successful AI applications is trustworthiness, which requires transparency on the underlying AI system's decision-making. The transparency requirement is one of 7 key requirements of the European Commission's guidelines for trustworthy AI. It's crucial for domain experts to trust in the AI's results.

Like other connectionist models, GNNs lack such transparency. To provide transparency in the inference process of GNNs, symbolic and sub-symbolic explanation methods have been proposed in the literature.

Sub-symbolic explanation methods find explanations by using statistical techniques, such as perturbation-based, gradient/feature-based, decomposition-based or surrogate-based methods. The output of sub-symbolic explanation methods are importance scores for individual edges, nodes, or node/edge features in the graph(s).

Symbolic explanation methods such as formal methods and programming languages, often associated with knowledge bases, such as ontologies or free form knowledge bases, come with the key characteristic of being able to explain and reason about their decision-making in a qualitative way. The output of symbolic explanation methods are qualitative explanations of predictions.

Existing symbolic and sub-symbolic approaches provide either local (explaining individual input output pairs) or global explanations (explaining the entire model), not both.

Important properties of explanation methods are naturalness (explanation in natural language), sensitivity (matching user context), fidelity (accurate correspondence with the model to be explained), and reference to graph topology (explanations using graph properties such as substructures).

Sub-symbolic explanation methods do not provide naturalness, because importance scores are not in natural language. Existing sub-symbolic explanation methods do not provide sensitivity, because they do not consider the user context.
Most symbolic explanation methods do not consider the GNN itself, which means that they are not able to explain the decision-making process of the GNN, and are therefore not able to achieve fidelity. None of the existing symbolic explanation methods refers to the graph topology.

We can categorize different types of explanations, including subgraph explanations, non-ontological explanations, and ontological explanations, as shown in Figure 1.1. Subgraph explanations correspond to sub-symbolic technologies and share their properties. Symbolic explanation methods can be further divided into non-ontological and ontological explanations. Both types come with naturalness, but non-ontological explanations lack structured background knowledge and may not always provide sensitivity. In contrast, ontological explanations can provide both local and global explainability.

Table 1.1 shows a matrix of explanation methods (including the contributions introduced in this thesis) and their properties.
The goal of this thesis is to overcome the weaknesses of symbolic and sub-symbolic explanation methods for GNNs by extending existing approaches and by combining symbolic and sub-symbolic methods. Therefore, answering the following questions is the basis of this thesis.

Research Question: How to generate explanations for Graph Neural Networks with fidelity, reference to graph topology, naturalness and sensitivity that come with local and global explainability?

The main contributions are:

- A sub-symbolic local explanation method called RERE that generates subgraph explanations, which uses reinforcement learning to enable the integration of sensitivity and which overcomes the state-of-the-art methods by providing sensitivity.

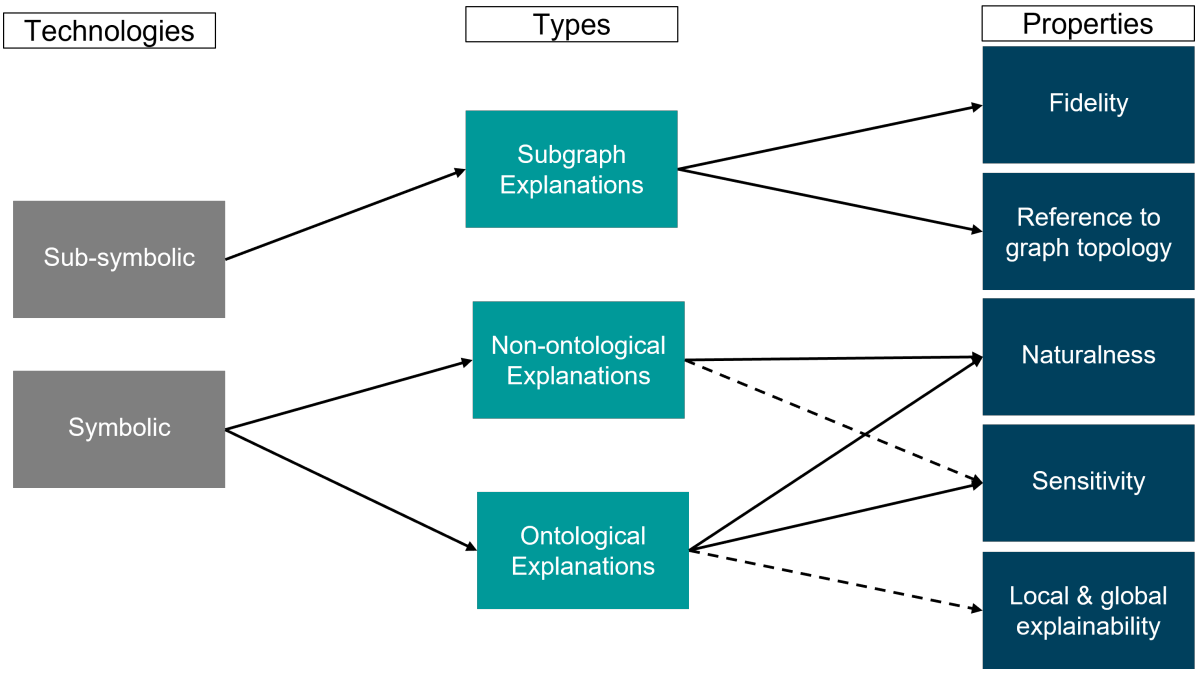


Figure 1.1: Technologies, types, and properties for Graph Neural Network Explainability

- A global explanation method for free form knowledge bases that generates non-ontological explanations, called SUBGREX [27], which combines symbolic and sub-symbolic approaches using a post-processing rule-based companion to a sub-symbolic explanation method, to complement sub-symbolic local explanations with global rules, and which satisfies the three properties naturalness, sensitivity, and reference to graph topology, but not fidelity.
- A global and local explanation method for ontology knowledge bases that generates ontological explanations, called OntExplainer [28]. It combines symbolic and sub-symbolic approaches by extracting global and local symbolic explanations while taking into account domain-specific ontologies, and which satisfies all four properties (naturalness, sensitivity, fidelity, and reference to graph topology)
- The successful application and experimental evaluation of all three proposed explanation methods RERE, SUBGREX, and OntExplainer for two use cases (molecular chemistry and cybersecurity), in comparison with the relevant state-of-the-art methods.

The following publications in peer-reviewed international conferences and patents have been achieved:

- Himmelhuber, A., Grimm, S., Zillner, S., Runkler, T., Ontology-Based Skill Description Learning for Flexible Production Systems, IEEE International Conference on Emerging Technologies and Factory Automation 2020,
- Himmelhuber, A., Ringsquandl, M., Joblin, M., Runkler, T., Demystifying Graph Neural Network Explanations & Joint European Conference on Machine Learning and Knowledge Discovery in Databases, ECML-PKDD 2021,
- Himmelhuber, A., Grimm, S., Zillner, S., Ringsquandl, M., Joblin, M., Runkler, T., Combining Sub-Symbolic Explanations with Semantic Web Technologies & International Joint Conference on Rules and Reasoning 2021,
- Himmelhuber, A., Grimm, S., Zillner, S., Ringsquandl, M., Joblin, M., Runkler, T., A New Concept for Explaining Graph Neural Networks & International Workshop on Neural-Symbolic Learning and Reasoning 2021,
- Himmelhuber, A., Ringsquandl, M., Joblin, M., Runkler, T., Receptive Field Reducer for Explaining Graph Neural Networks, IJCAI XAI Workshop 2023,
- Himmelhuber, A., Dold, D., Grimm, S., Zillner, S., Runkler, T., Sub-symbolic explainer and symbolic methods for Cybersecurity, IEEE Symposium Series On Computational Intelligence,
- Demir C., Himmelhuber A., Liu Y., Bigerl A., Mousallem D., Ngomo A., Rapid Explainability for Skill Description Learning & International Semantic Web Conference Industry 2022,
- Himmelhuber A., Grimm S., Zillner S., Joblin M., Runkler T., Combining Symbolic and Sub-symbolic Methods for Explainable AI, Chapter in Compendium of Neuro-Symbolic Artificial Intelligence, Series: Frontiers in Artificial Intelligence and Applications
- Himmelhuber A., Zillner S., Grimm S. (2021). METHOD AND SYSTEM FOR SEMI-AUTOMATED GENERATION OF MACHINE-READABLE SKILL DESCRIPTIONS OF PRODUCTION MODULES. (U.S. Patent No. 2021/0334670). U.S. Patent and Trademark Office.
- Himmelhuber A., Joblin M., Ringsquandl M. (2022). COMPUTER IMPLEMENTED METHOD FOR CONTROLLING A CLASSIFICATION MADE BY A GRAPH NEURAL NETWORK, CONTROL UNIT AND COMPUTER PROGRAM PRODUCT. (WO Patent No. 2022/258292). World Intellectual Property Organization.

- Liu Y., Himmelhuber A., Joblin M., Ringsquandl M., Grimm S. (2022). METHOD AND SYSTEM FOR MAINTAINING A PLANT. (EP Patent No. 4006674). European Patent Office.
- Himmelhuber A., Joblin M., Ringsquandl M., Grimm S., Zillner S., Runkler T. (2022). METHOD AND CONTROLLER FOR GENERATING A PREDICTIVE MAINTENANCE ALERT. (WO Patent No. 2022/218685). World Intellectual Property Organization.
- Himmelhuber A., Liu Y. (2022). METHOD AND SYSTEM FOR PROVIDING RECOMMENDATIONS CONCERNING A PROJECT CONFIGURATION TO CONFIGURE AN INDUSTRIAL SYSTEM. (EP Patent No. 4086825). European Patent Office.

This thesis is structured as follows: Chapter 2. provides the reader with the necessary background on Graph Neural Networks, methods of symbolic AI and explainable AI. Chapters 3, 4, and 5 introduce the three proposed methods RERE (subgraph explanations), SUBGREX (non-ontological explanations), and OntExplainer (ontological explanations). Chapter 6 evaluates the introduced methods on a molecular chemistry and cybersecurity use case. Chapter 7 summarizes this thesis, draws the main conclusions from our experiments, and closes with an outlook.

Table 1.1: Comparison of different GNN explainer methods with regard to the requirements of naturalness, sensitivity, fidelity, graph topology; whether they are local or global and whether or which knowledge base (free form, ontology, or knowledge graph (KG)) they use.

	Naturalness	Sensitivity	Fidelity	Graph Topology	Local	Global	Knowledge Base
Sub-symbolic - Subgraph Explanations							
GNNExplainer [1]	✗	✗	✓	✓	✓	✓	NA
PGExplainer [2]	✗	✗	✓	✓	✓	✗	NA
SubgraphX [3]	✗	✗	✓	✓	✓	✗	NA
GraphMask [4]	✗	✗	✓	✓	✓	✗	NA
Cf-GNNExplainer [5]	✗	✗	✓	✓	✓	✗	NA
ZORRO [6]	✗	✗	✓	✓	✓	✗	NA
Pope et al. [7]	✗	✗	✓	✗	✓	✗	NA
Baldassarre et al. [8]	✗	✗	✓	✗	✓	✗	NA
GNN-LRP [9]	✗	✗	✓	✓	✓	✗	NA
Xiong et al. [10]	✗	✗	✓	✓	✓	✗	NA
Relex [11]	✗	✗	✓	✓	✓	✗	NA
PGM-Explainer [12]	✗	✗	✓	✓	✓	✗	NA
GraphLIME [13]	✗	✗	✓	✓	✓	✗	NA
XGNN [14]	✗	✗	✗	✓	✗	✓	NA
RERE	✗	✓	✓	✓	✓	✗	NA
Symbolic - Non-ontological Explanations							
NIWT [15]	✓	✓	✗	✗	✓	✗	Free form
TCAV [16]	✓	✓	✗	✗	✓	✗	Free form
TREPAN [17]	✓	✗	✓	✗	✗	✓	NA
Symbolic - Ontological Explanations							
TREPAN Rel.[18]	✓	✓	✓	✗	✗	✓	Ontology
Sarker et al. [19]	✓	✓	✗	✗	✗	✓	Ontology
Geng et al. [20]	✓	✓	✗	✗	✗	✓	Ontology
X-NeSyL [21]	✗	✓	✗	✗	✓	✗	KG
Widmer et al. [22]	✓	✓	✗	✗	✗	✓	Ontology
Chen et al. [23]	✓	✓	✗	✗	✓	✓	Ontology
Labaf et al. [24]	✓	✓	✗	✗	✗	✓	Ontology
KDTCN [25]	✓	✓	✗	✗	✓	✗	KG
Doctor XAI [26]	✓	✓	✓	✗	✓	✗	Ontology
Symbolic + Sub-symbolic							
SUBGREX [27] (Non-ontological Explanations)	✓	✓	✓	✓	✗	✓	Free form
OntExplainer [28] (Ontological Explanations)	✓	✓	✓	✓	✓	✓	Ontology

2 Background and Challenges

In this chapter, the background information required for the rest of the thesis is presented. We first introduce Graph Deep Learning, including the notation used in Table 2.1. Then, we explore Graph Neural Networks (Section 2.1.2) and its variants (Section 2.1.3). In Section 2.2, we provide background information on symbolic methods in artificial intelligence and Semantic Web Technologies (Section 2.2). A toy example that is used throughout the thesis is introduced in Section 2.3. General explainable AI is introduced in Section 2.4, with a focus on XAI for GNNs in Section 2.4.1. Sub-symbolic explanation methods for GNNs and symbolic explanation methods are discussed in Sections 2.4.2 and 2.4.3.2. Finally, Section 2.4.4 introduces evaluation metrics for explanations.

2.1 Graph Deep Learning

Many important real-world data sets come in the form of graphs or networks, including social networks [29] [30], knowledge graphs [31], protein-interaction networks [32], physical systems [33], the World Wide Web and many more [34]. Due to their great expressive power, analyzing graphs with machine learning has been become increasingly popular [35]. Graphs are a kind of data structure which models a set of objects and their relationships, or in other words, nodes and edges. Existing machine learning algorithms, such as Convolutional Neural Networks (CNNs) popular in the image domain, have difficulties handling the complexity of this non-Euclidean data. As graphs do not have specific ordering, and their nodes may have a differing number of neighbors, operations like convolutions are difficult to apply in the graph domain. Furthermore, the core assumption of many machine learning algorithms, that instances are independent of each other, doesn't hold for graph data because each node is related to others by links of various types [36].
GNNs are deep learning based methods that operate on the graph domain, capturing the dependency structure between the nodes of graphs and ignoring the node order on the input. Unlike standard neural networks, GNNs retain a state that can represent information from its neighborhood with arbitrary depth, as well as incorporate node feature information [37]. The

transformation function to update each node's features might take only the node itself, or the node and its neighbors, or the complete graph topology into account. Due to its convincing performance, GNNs have been widely applied in graph analysis methods recently.

2.1.1 Definitions

A graph G is a tuple $G = (V, E)$, with V being the set of nodes (vertices) and E being the set of edges. We denote a single node as $v \in V$, using subscripts to distinguish between nodes when necessary. Edges are pair-wise connections between nodes; we refer to an edge from node v_i to v_j as $e_{ij} = (v_i, v_j) \in E$. Nodes that are connected by an edge are also called adjacent. The neighborhood of a node v is defined as $N(v) = \{u \in V | (u, v) \in E\}$. The adjacency matrix A is a $N \times N$ matrix with $A_{ij} = 1$ if $eij \in E$ and $A_{ij} = 0$ if $eij \notin E$. A graph may have node attributes X, where $X \in R^{N \times d}$. X is a node feature matrix with $x_i \in R^d$ representing the feature vector of a node i. We use the k-hop neighborhood $N_k(v)$ of a node v, which is the set of all nodes connected to v by a path of length k. A directed graph is a graph with all edges directed from one node to another, whereas an undirected graph is considered as a special case of directed graphs where there is a pair of edges with inverse directions if two nodes are connected.

Most commonly, a graph is represented by either an adjacency matrix or an adjacency list. Both rely on defining a fixed but arbitrary node order, with the adjacency list being a list of length $|E|$, containing an element (i, j), if an edge between the i-th and j-th node exists. Edge features can be easily represented by using a separate edge feature matrix $|E| \times |E|$, whose i-th entry corresponds to the i-th entry in the adjacency list. For graph representation learning, we use h_v and o_v as the hidden state and output vector of node v. Unless otherwise specified, the notations used in this thesis are illustrated in Table 2.1. Motifs or network motifs refer to characteristic patterns and substructures in graphs, such as cycles or triangles. Each motif $Mi = (V_i, E_i)$ is a subgraph of G. They can be found in biochemical, neurobiological, ecological, engineering and social networks [38]. One example from the chemical domain is the functional group hydroxide OH^-, which typically implies high water solubility [39]. A variety of algorithmic procedures exist to count or detect network motifs in order to identify functional properties of a network. The frequency or existence of certain motifs can indicate how nodes behave in the network.

Table 2.1: Symbols used in this thesis.

G	Graph
V	Set of Nodes
E	Set of Edges
v_i, v_j	i-th and j-th node in the population
e_{ij}	Edge between i-th and j-th node
$\mathcal{N}(v)$, $\mathcal{N}_k(v_i)$	Single-hop and k-hop neighbourhood of node v_i
$\mathcal{V}_i$, $\mathcal{E}_{ij}$	Features of node v_i, e_{ij}
$\mathbf{A}$	Adjacency matrix
$\mathbf{X}$	Node Feature matrix
N	Total number of nodes
x_i	Feature vector belonging to node v_i
$deg(v)$	Degree of a node v
W	Weight matrix representing the weights of each edge
w_{ij}	Element of W representing edge weight between node i and j
k	k nearest neighbors
$\mathbf{I_N}$	Identity matrix of dimension N
$g_w \star x$	Convolution of g_w and x
$\odot$	Element-wise multiplication
$\oplus$	Aggregation function
Θ	Diagonal weight matrix which has one parameter per eigenvector
M_i	Motif
M_E	Edge Mask
M_X	Node Feature Mask

2.1.2 Graph Neural Networks

Graph neural networks are useful tools on non-Euclidean structures, and there are various methods proposed in the literature trying to improve the model's capability. We are describing the general graph neural network pipeline and then introduce several variants, which utilize different propagation functions and advanced training methods, following [37]. A typical GNN is usually built by combining the following computation models, shown in Figure 2.1. The sampling module is needed to conduct propagation on graphs when these are large. It tends to be combined with the propagation module, which is used to propagate information between nodes. Here, the convolutions and recurrent operators aggregate information from neighbors to capture feature and topological information. The skip connection operation gathers historical information, and all these modules are used to propagate information in each layer, which are stacked for better representations. The pooling operator extract information from nodes for high-level information. This architecture applies to most GNNs, with some exceptions. The most commonly used propagation operators for GNNs are convolution

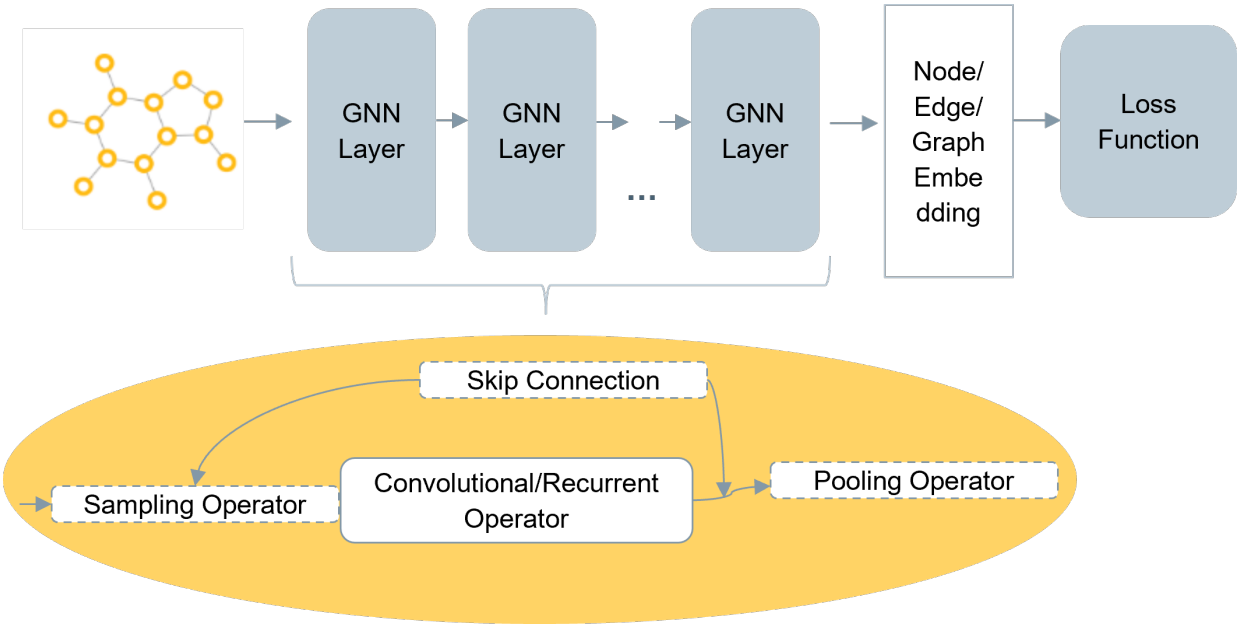


Figure 2.1: General design pipeline for a Graph Neural Network

operators, which can be categorized as spectral approaches and spatial approaches.

2.1.2.1 Spectral Approaches

Spectral approaches use spectral graph theory to define convolutional layers, which transform node features to a final representation. Their mathematical foundation is in graph signal processing [40]. Here, a graph signal $\mathbf{x}$ is transformed to the spectral domain by the graph Fourier transform $F(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$, where then the convolution operation is conducted. The resulting signal is transformed back using inverse graph Fourier transform $F^-1(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ represents the resulted signal. $\mathbf{U}$ is the matrix of eigenvectors of the normalized graph Laplacian $\mathbf{L} = \mathbf{I_N} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}$, where $\mathbf{D}$ is the degree and $\mathbf{A}$ the adjacency matrix. The normalized graph Laplacian matrix being real symmetric positive semi-definite can be factored as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda} = \mathbf{U}^T$ with $\mathbf{\Lambda}$ being the diagonal matrix of eigenvalues (spectrum). The convolution operation is defined as:

$$g \star \mathbf{x} = \mathbf{U}g_w\mathbf{U}^T\mathbf{x} \tag{2.1}$$

with the learnable diagonal matrix g_w being the filter in the spectral domain.

2.1.2.2 Spatial Approaches

Spatial approaches define convolutions directly on the graph based on the graph topology. Similarly to CNNs, spatial Graph Convolutional methods define graph convolutions based on spatial relations existing between nodes. The challenge is to define these operations with neighborhoods differing in size and maintaining the local invariance of CNNs [37]. The main implementations are expressed as message passing approaches, which use only the immediate neighborhood of a node to compute that node's new features. A message function M creates messages $m_{ij} = M(v_i, v_j, e_{ij})$ for node v_i based on its neighboring nodes v_j and edges e_{ij}. Each node's new features are then computed by a node update function U_v

$$\mathcal{V}_i' = U_v(\mathcal{V}_i, \bigoplus_{v_j \in \mathcal{N}(v_i)} m_{ij}), \tag{2.2}$$

with collation function $\bigoplus$, which has to be transitive and associative to deal with the node order. The most common implementation is a sum, followed by maximum and mean operations. While the computational complexity is lower for spatial approaches compared to spectral ones, spectral approaches are better suited for modelling of global graph structures.

2.1.3 Variants of Graph Neural Networks

There are a number of different variants of GNNS, that utilize different aggregators to gather information from each node's neighbors and specific updaters to update nodes' hidden states. Some of these variants, namely Graph Convolutional Network (GCN), Graph Attention Networks and GraphSAGE are described as spatial approaches below. The activation functions are omitted in the formalization.

Graph Convolutional Networks Graph Convolutional Networks (GCNs) are an extreme form of approximating the spectral convolution using only a first-order Chebyshev polynomial approximation [41]. It can also be expressed as a message passing approach. A GCN layer is based on a weighted sum of neighboring node features, which are then transformed by a learned weight matrix. It can be expressed as a message passing approach, with message function M creating messages m_{ij} for node v_i based on its neighboring nodes v_j and the edge e_{ij}:

$$M(v_i, v_j, e_{ij}) = (deg(v_i)deg(v_j))^{1/2}\mathbf{A}_{ij}\mathcal{V}_j, \tag{2.3}$$

therefore depending only on the degrees of the corresponding nodes and the hidden features of the neighbors, with collation function being a sum, $\bigoplus = \sum$, and each node's new features are then computed by node update function U_v

$$U_v(v_i) = \Theta \sum_{v_j \in \mathcal{N}^+(v_i)} m_{ij}. \tag{2.4}$$

Graph Attention Networks Graph Convolutional Networks (GCNs) treat all neighbors equally and generate a final output by combining their features in an equal manner. In contrast, Graph Attention Networks (GATs) [42] compute attention weights for each neighboring node based on its features. Specifically, GATs augment the original node update function with edge-specific attention weights α_{ij}, which allow for more informative and selective feature aggregation:

$$U_v(vi) = \Theta \sum_{v_j \in \mathcal{N}^+(v_i)} \alpha_{ij} m_{ij} \tag{2.5}$$

α is computed using the agreement between transformed source and target node features, normalized using the soft max operator:

$$\alpha_{ij} = softmax_{\mathcal{N}^+(v_i)}(\sigma(\Theta\mathcal{V}_i || \Theta\mathcal{V}_j)).$$

GraphSAGE GraphSAGE [43] is a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data and is based on a localized sampling of the neighborhood. Instead of training individual embeddings for each node, a function is learned that generates embeddings by sampling and aggregating features from a node's local neighborhood. Each node is updated based on the *k*-neighborhood:

$$\mathcal{V}_i^{'} = \Theta(\mathcal{V}_i || \bigoplus_{\mathcal{V}_j \in \mathcal{N}_k(node_i)} (v_j)) \tag{2.6}$$

Three different aggregation functions are explored. The first aggregation function takes the element-wise mean of the vectors, the second applies learned Long Short-Term Memory (LSTM) units on a random permutation of the neighbors, and the third applies maximum pooling on independently transformed node features.

Learning Tasks on Graphs The goal in graph-based machine learning tasks can be sorted into one of three categories: Node-level task, edge-level tasks and graph-level tasks.
Node-Level Task:. Node-level prediction produces outputs per node. Trained on a set of

labelled nodes, the goal is to predict the output values for each node on unseen graphs.

Edge-Level Tasks: Edge-level tasks relate to the edge classification and link prediction tasks. With two nodes' hidden representations from GNNs as inputs, a similarity function or a neural network can be utilized to predict the label or whether a link exists.

Graph-Level Tasks: Graph-level outputs relate to the graph classification task. To obtain a compact representation on the graph level, GNNs are often combined with pooling and readout operations. A classical example for this task is molecule property prediction, where a molecule is modelled as a graph with atoms as nodes and chemical bonds as edges and attributes such as mutagenicity [44] are predicted.

2.2 Symbolic AI Methods and Semantic Web Technologies

Methods in AI can be categorized into sub-symbolic or symbolic methods, neuro-symbolic methods refer to the integration of both. Sub-symbolic methods establish correlations between input and output variables, that come with high complexity, and are often formalized by functions that map the input to the output data or the target variables [45]. Sub-symbolic AI includes statistical learning methods, such as Bayesian learning, deep learning and backpropagation, with GNNs being a sub-symbolic method. These methods are able to solve complex tasks over unstructured data using supervised or unsupervised learning, including problems which cannot reasonably be done by humans [46]. Their characteristics include robustness again noisy data, high computing performance and scalability. Therefore, they are suitable for big datasets and large knowledge graphs.

Symbolic AI refers to AI approaches that are based on explicit symbol manipulation which can include term rewriting or natural language question answering, but tends to refer more narrowly to methods based on formal logic, such as knowledge representation and reasoning [47]. As these are defined by explicit symbolic methods, such as formal methods and programming languages, they are usually used for deductive knowledge [48]. They consist of first order logic rules, ontologies, decision trees, planning and reasoning [45] and are often associated with knowledge bases and expert systems [49], with one of their key characteristic being their ability to explain and reason about their decision-making. Further characteristics are rule modularity, as rules are discrete and autonomous knowledge units that can easily be inserted or removed from a knowledge base [50]. Examples for application fields of symbolic systems include agent planning, constraint solving, data management and integration and querying [46].

In terms of representation of data, information and knowledge, sub-symbolic and symbolic AI differ fundamentally. Symbolic systems' representation of knowledge is explicit and human-understandable, e.g. through rules. The representations in sub-symbolic systems tend to be through weighted connections between neurons and corresponding activations of a number of neurons, which isn't usually human-understandable. These distributed representations which are learned during training are also known as embeddings. These vectors of real numbers are in terms of entities within a high-dimensional and continuous, differentiable vector space, whereas symbolic representations are discrete [47]. Neuro-symbolic AI focuses on finding ways to combine sub-symbolic learning algorithms with symbolic reasoning techniques, to get the best of both worlds. While symbolic systems are susceptible to noisy data or flaws in the knowledge base, sub-symbolic systems show more robustness. Symbolic systems come

with inherent explainability and reasoning capabilities, whereas sub-symbolic systems tend to be black-boxes [46]. Examples of neuro-symbolic integration include the translation of symbolic knowledge into the network, the learning of addition knowledge from examples, executing the network and symbolic knowledge extraction from the network, which provides explanations [51]. Furthermore, integration can lead to utilization of background knowledge, given as knowledge graphs or ontologies in deep learning applications [46] with the aim of making them more transparent, understandable, verifiable, and trustworthy. This is however no trivial problem, since the methodologies of distinct areas have to be conciliated — namely predominantly statistics and logic — in order to combine the respective advantages and circumvent the shortcomings and limitations.

Semantic Web Technologies Through the creation of the World Wide Web, content can be created and made available online [52]. The Semantic Web was developed to unify such content by tagging it with unique identifiers representing definitions from taxonomies and ontologies, making it possible to reach semantic understanding of digital content and enable data sharing [53]. To make the content explainable to a wider range of users, methods have been developed in order to include provenance in the semantic representations [54] and to enable reasoning mechanisms [55]. These are called Semantic Web technologies and can provide formal description and semantic processing of data, therefore making the data interpretable with regard to its content and meaning — making it a symbolic method. Definitions of frameworks and tools related to the Semantic Web used in this thesis can be found in Table 2.2. The explicit knowledge representation of the Semantic Web includes modeling of knowledge and application of formal logics over the knowledge base. One such knowledge base is a knowledge graph, where information is encoded in the form of a directed labeled graph, with nodes representing entities and edges representing different types of possible relationships between entities. Another approach are ontologies, which enable the modeling of information and consist of classes, relations, and instances [56]. We follow [19], in differentiating knowledge graphs (KGs) and ontologies insofar as KGs are usually a set of triples most often expressed using the Resource Description Framework (RDF) while ontologies additionally posses logics and are regularly expressed using Web Ontology Language (OWL).

The basic constituents for representing knowledge in OWL are individuals, classes, and properties. They are used to forming axioms, i.e., statements within the target domain, and an

ontology $\mathcal{O}$ is a set of axioms to describe what holds true in this domain. The most relevant axioms for our work are class assertions $\tau(\sigma)$ assigning an individual σ to a class τ, property assertions $\rho(\sigma_1, \sigma_2)$ connecting two individuals σ_1, σ_2 by property ρ, and subclass axioms $\tau_1 \sqsubseteq \tau_2$ expressing that class τ_1 is a subclass of class τ_2 [1]. Classes can be either atomic class names, such as '`Individual`' or '`hasFriend`', or they can be composed by means of complex class expressions. An example of a complex class expression noted in Manchester syntax is '`Individual and hasGender some female`', which refers to all individuals having gender female. For details about all types of axioms and the way complex concepts are constructed, we refer to [57]. The Manchester OWL syntax is a user-friendly syntax for OWL Description Logics, fundamentally based on collecting all information about a particular class, property, or individual into a single construct [58].
The basic constituents for representing knowledge in OWL are individuals, classes, and properties. They are used to forming axioms, i.e., statements within the target domain, and an ontology.
The following definitions are related to the concepts of explainability used in this thesis:

Definition 1 (Entailment) *Given ontology $\mathcal{O}$, if axiom α logically follows from $\mathcal{O}$, as can be derived by a standard OWL reasoner, then we call α an* entailment *of $\mathcal{O}$ and write $\mathcal{O} \models \alpha$* [2].

Definition 2 (Inductive Logic Learning (ILL)) *Given an ontology $\mathcal{O}$, a set of positive instances E^+ and a set of negative instances E^-, a resulting target predicate class expression ε is constructed such that $\mathcal{O} \models \varepsilon(\sigma)$ holds for all individuals $\sigma \in E^+$ and does not hold for individuals $\sigma \in E^-$* [3].

In the context of OWL ontologies, ILL attempts to construct class expressions from an ontology $\mathcal{O}$ and two sets E^+, E^- of individuals that act as positive and negative examples for being instances of the target class, respectively. In this thesis, the DL-Learner [60] is used as the key tool to derive OWL class expressions.

Definition 3 (Justification) *Given an ontology $\mathcal{O}$ and an entailment α, the* justification $\mathcal{J}(\mathcal{O}, \alpha)$ *for α in $\mathcal{O}$ is a set $\mathcal{J} \subseteq \mathcal{O}$, such that $\mathcal{J} \models \alpha$ and $\mathcal{J}' \not\models \alpha$ for all proper subsets $\mathcal{J}' \subset \mathcal{J}$.*

Justifications are a type of explanation for entailments in ontologies, with a justification being the minimal subset of the ontology being sufficient for the respective entailment to hold.

[1] For better readability we will denominate variables represented in ontology form in Greek letters and sub-symbolic graph representations in Latin letters.
[2] As defined in [59]
[3] As defined in [60]

Table 2.2: Semantic Web Technology Definitions

Description Logic	A knowledge representation formalism based on a subset of first-order predicate logic that is a declarative formalism for the representation and expression of knowledge and sound, tractable reasoning methods founded on a theoretical logical basis [61].
Ontology	An ontology models the vocabulary and meaning of domains of interest. The objects in domains, the relationships among those things, the properties, functions, and processes involving those things as well as constrains on and rules about those things [61].
Protégé	Protégé is an ontology management tool developed and maintained by the Medical Informatics Laboratory at Stanford University and is recognized as an exemplary tool for managing ontology [61].
Resource Definition Framework (RDF)	Language that expresses local semantic relations phrased in terms of a triple: <subject, verb object>, i.e., <object1, relation1, object2> [61].
Resource Definition Framework Schema (RDFS)	Language that expresses class-level semantic relations describing acceptable local relations [61].
OWL	Web Ontology Language is part of W3C's Semantic Web Technology Stack and is designed to represent rich and complex knowledge about things, groups of things, and relations between things [61].
DL-Learner	DL-Learner is a tool for learning concepts in DLs from user-provided examples. It can also be used to learn classes in OWL ontologies from selected objects [60].
LD2NL	Verbalization framework for the three key languages of the Semantic Web, i.e., RDF, OWL, and SPARQL. Based on a bottom-up approach, resulting in verbalization that are close to natural languages [62].

2.3 Toy Example: SportsNetwork

Introducing SportsNetwork, a toy example to showcase the methods used in this thesis. SportsNetwork is a social network that comprises individuals or organizations connected by social interactions. Nodes in the network represent individuals, while edges depict the connections between them. Graph-based methods can be used to analyze the structure of the entire social entity and understand observed patterns [63]. Figure 2.2 illustrates a common motif in which an individual's connections are also connected to each other, which has significance in both social networks and epidemiological contact networks [64]. This concept dates back to the late 1970s, when 3-node motifs or triadic structures (as shown in Figure 2.3) were first identified in small-scale social structures [65].

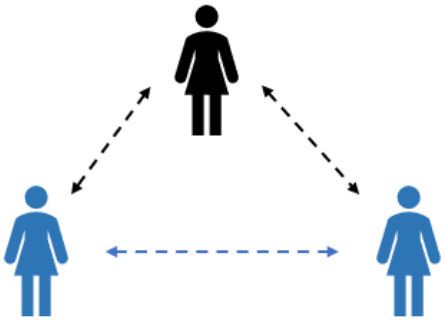


Figure 2.2: Social clustering network motif

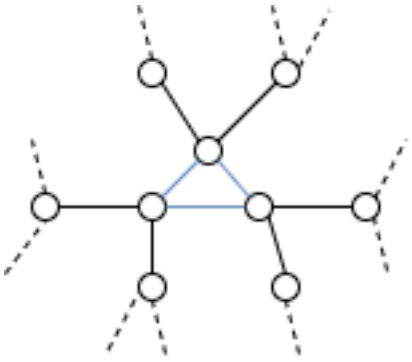


Figure 2.3: Clustering triadic motif

SportsNetwork is a toy example of a social network, featuring athletes as nodes and connections between them as edges. The node features include details about the athletes such as gender, home country, and age. A glimpse of the network is shown in Figure 2.4. A

hypothetical node classification task, where a 2-layer Graph Neural Network (GNN) predicts the athlete's sport (e.g. equestrian or football player), is also demonstrated. The graph also showcases triadic structures.

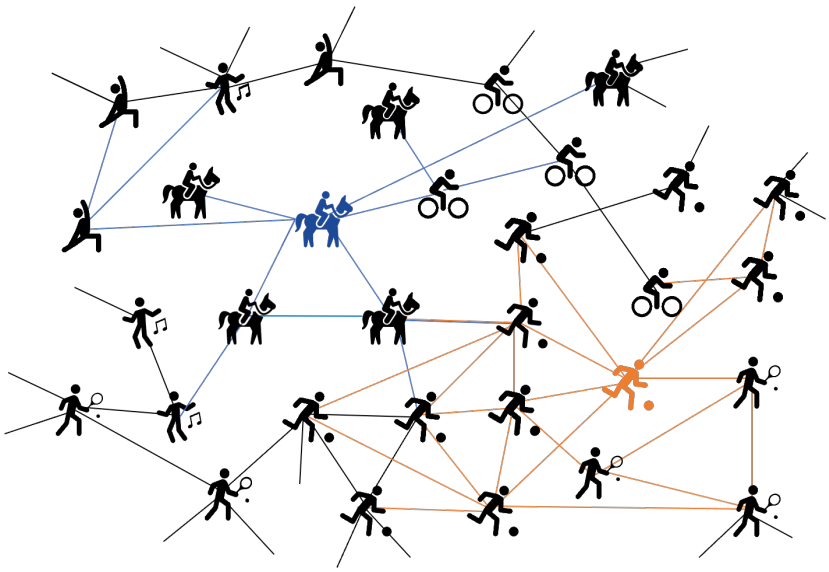


Figure 2.4: SportsNetwork excerpt showing the node classification for an equestrian node (blue) and a football player node (orange) by a 2-layer GNN with their respective receptive fields.

Table 5.1 shows an excerpt of the SportsNetwork Ontology $\mathcal{O}^{SportsNetwork}$ pertaining to the toy example. The ontology includes some hierarchies, e.g., '`Football`' being a subclass of '`TeamSports`'. Furthermore, structures known in social networks are included such as '`FootballTriadicMotif`', which is a subclass of '`SocialStructure`'. Property assertions are shown, such as '`fromCountry(Individual_3, Germany)`', detailing the origin of individual '`Individual_3`'.

Table 2.3: Example excerpt of $\mathcal{O}^{SportsNetwork}$.

(1)	`Football` $\sqsubseteq$ `TeamSports`	football is a team sport
(2)	`FootballTriadicMotif` $\sqsubseteq$ `ClusteringTriadicMotif` $\sqsubseteq$ `SocialStructure`	football triadic motifs are clustering triadic motifs, which are social structures
(3)	`FootballPlayer(Individual_2)`	Individual_2 is a football player
(4)	`hasGender(Individual_2, male)`	Individual_2 has gender male
(5)	`fromCountry(Individual_3, Germany)`	Individual_3 is from Germany
(6)	`hasFriend(Individual_2, Individual_1)`	Individual_2 has friend Individual_1

2.4 Explainable AI

The development of black-box machine learning models such as deep neural networks have revolutionized the field of artificial intelligence. As these models have achieved promising performance in many research tasks, they are increasingly being employed to make important predictions in critical contexts [66]. Very little human intervention is required for their design and deployment. While the very first AI systems or “good old-fashioned AI" are more easily interpretable, the last decade has witnessed the rise of opaque decision systems such as Deep Neural Networks (DNNs), which combine efficient learning algorithms and a huge parametric space comprised of hundreds of layers and millions of parameters [67]. When critical decisions are derived from such black-box systems which ultimately affect humans' lives, e.g., in medicine, law or defense, the demand for transparency is increasing [68]. Without understanding the underlying mechanisms behind the predictions, black-box models cannot be fully trusted, leading to problems regarding fairness, privacy, and safety [69]. Companies increasingly offer services and products by embedding sophisticated machine learning models trained on massive datasets. These offers are often in safety-critical industries such as robotics, medicine or self-driving cars [70]. Inherent and systematic bias in the training data that leads to spurious correlation is one risk of using black-box models. Therefore, the demand for explainable and accountable AI grows as tasks with higher sensitivity, safety and social impact are increasingly decided by AI systems. Explainability is needed in holding organizations responsible and accountable for their products, services, and communication of information [71].
The legal right to explanations has been established in the European Union General Data Protection Regulation (GDPR) commission. This right holds for all individuals to obtain “meaningful explanations of the logic involved” when automated decision-making takes place [70]. Furthermore, the European Commission's High-Level Expert Group on AI recently presented the “Ethics Guidelines for Trustworthy Artificial Intelligence". The guidelines put forward a set of 7 key requirements that AI systems should meet in order to be deemed trustworthy, including a transparency requirement [72]:

“*The data, system and AI business models should be transparent. Traceability mechanisms can help achieving this. Moreover,* ***AI systems and their decisions should be explained in a manner adapted to the stakeholder concerned****. Humans need to be aware that they are interacting with an AI system, and must be informed of the system's capabilities and limitations.*”

How the terms explainability and interpretability are used varies across AI communities, and can reach from interpretations of how an AI system works to explanations mapping inputs of a particular example to an output. As these terms are often interchangeably used in literature, we will follow the definitions of [73], where a model that can provide human-understandable interpretations by itself is defined as interpretable. A black-box model whose predictions are explained by post hoc explanation techniques is explainable.

XAI Users and Goals Explainability is a very broadly defined concept, with different scientific communities addressing the problem from a different perspective and providing a different meaning to explanation [70]. The machine learning community is mostly focused on designing new interpretable models and explain black-box models with ad-hoc explainers, while the visual analytics community prioritize tools and methods for data and domain experts to visualize complex black-box models and study interactions to manipulate machine learning models [71]. While the increasing level of research activities regarding explainable AI is a positive development, explanatory models are often built for AI experts. These are machine learning scientists and engineers who design machine learning algorithms and interpretability techniques for XAI systems. However, explanations should be targeted for domain experts when necessary, since XAI and with it the widespread application of AI models are more likely to succeed if the evaluation of these models is focused more on the user's needs [74]. In this algorithmic-centered discourse, understanding who interacts with the black-box AI and what their distinct goals are, is of equal importance [75]. Domain experts use machine learning for analysis, decision-making, or research in specialized forms and domains such as cybersecurity [76] [77], medicine [78] [79], text analysis [80] [81], and satellite image analysis [82]. While these users are experts of certain domain areas, they tend to lack expertise in the technical specifics of the machine learning algorithms [71]. Further user groups of XAI include business owners, decision-makers, who are direct users of AI decision support applications, impacted groups and regulatory bodies [83].

There are a number of different goals the research communities around XAI have so far exposed, dependent on the respective users' needs. XAI Goals for domain experts include:

- **User Trust and Reliance.** Through providing explanations, the user can improve their trust in the AI, as they can assess the system reliability and calibrate their perception of system accuracy [71]. Domain experts can therefore benefit from machine learning interpretability to inspect model uncertainty [84].

- **Model Visualization and Inspection.** Domain experts can identify and analyze failure cases of machine learning models and systems [85]. Here, explanations help to visualize models [86] and enable the inspection for problems like biased models [87].
- **Algorithmic Transparency and Model Interpretability.** Domain users' mental models of the underlying AI can be improved by providing explanations [88]. Furthermore, through better understanding of the model output, user experience and interactions can be enhanced [89].

Other explainability goals for AI experts include detecting dataset bias [90], adversarial attack detection [91], model debugging and tuning [92] [93]. Furthermore, XAI is used for privacy awareness [67] and bias mitigation [94] [95], that could result in discrimination in algorithmic decision-making.

Global and Local Explanations Explanations can be classified into global and local explanations, with the first category describing the entire machine learning model, whereas the second category explains an individual input instance. Global explanations, also called model-level explanations are, for example, model visualizations [96], decision rules [97] or interpretable approximations of more complex models such as tree regularization [71] [98]. Local or instance-level explanations are focusing on explaining the results for an individual prediction requested by the user. Local explanations are thought to be more easily understandable. Examples of local explanations are saliency methods [92] [1] that show what features in the input strongly influence the model's prediction or local approximations representing the underlying model's behavior [93] [71].
For local methods, more human supervisions are needed since experts need to explore the explanations for different input graphs. It is left to the user to construct a global understanding of the model's decision-making process. It has been shown, that users have difficulties forming a global understanding and assessing how representative instance explanations were for the overall model behavior [99]. For global methods, since the explanations are high-level, less human supervision is needed. However, the explanations for global methods may not be human-intelligible. Especially for GNNs, the obtained graph patterns may not even exist in the real world [100].

Type of Explanations There is no agreement on what constitutes an explanation, including which shape it is provided in, nor which properties it has to exhibit [70]. This thesis focuses on post-hoc explainability as opposed to interpretable models such as linear regression models

or decision trees. Complex models including Graph Neural Networks, that come with high performance and robustness in real-world applications are not interpretable by human users due to their large variable space [71]. Therefore, post-hoc explainability uses a number of different means to enhance their interpretability, such as visual explanations, explanations by simplification and feature relevance explanations techniques amongst others. These can be model-agnostic, meaning they are designed to be plugged to any model with the intent of extracting some information from its prediction procedure [66] or model-specific, only applying to a specific type of model.

- **Visual explanations.** These explanation techniques aim at visualizing the model's behavior. These methods often come with dimensionality reduction techniques that allow for a human interpretable simple visualization. Such visualizations can be coupled with other techniques to improve their understanding, usually with feature relevance techniques which provide the information the visualization is based on [66]. The inner structure of the machine learning model to be explained is disregarded.

- **Explanations by simplification.** These explanation techniques rebuild new and simplified models based on the respective trained models to be explained. The complexity of the student model (new model) is reduced compared to the teacher model (i.e. trained model to be explained) while their resemblance is optimized.

- **Explanations by example.** These explanation techniques work by extracting representative examples that capture the internal relationships and correlations of the model [101].

- **Feature relevance explanations.** These explanation techniques provide importance scores for the variables of the model to be explained, which quantify the importance a feature has upon the output of the model. The functioning of the model is explained by ranking or measuring the influence, relevance, or importance each feature has in the prediction output [66].

- **Text explanations.** These explanations techniques output text explanations, including methods that generate symbols representing the logic of the model through appropriate semantic mapping [101].

2.4.1 Explainability for GNNs

Similarly to other black-box machine learning methods, Graph Neural Networks require explanations. This is exacerbated through their growing popularity and usage in various domains, coming with increasingly complex models and growing complexity of the underlying data they work on [102]. These data domains can consist of various types of graphs and heterogeneous data, making many explanations meaningful only in a specific domain. While most GNNs can be categorized as message-passing, recently there has been an increase in the use of abstract constructs i.e., from graph theory such as motifs [103] [104], often requiring not only data but also model-specific explanations. Further differences in explainability techniques are regarding the learning tasks, namely node-level, edge-level and graph-level tasks, which can affect its structure. While task-agnostic explanation methods exists, other explainability techniques are tailored to one task. Although these differences in explanation methods exist, the explanations are usually expressed as graphs with either nodes, edges, or features that have been identified as important, highlighted in some way [105].

Properties of Explanations for GNNs There is a various number of explanation properties that have been deemed desirable by different research communities. This thesis centers around the challenge of making Graph Neural Networks explainable for domain experts, an objective that comes with its own set of properties that should be fulfilled, which are laid out in this section. While the emergence of explainable AI is positive, most explainer models for GNNs are built for AI experts rather than the intended user of the AI, such as domain experts. In order for the explainer models to succeed, properties that are focused more on the user needs should be considered [74]. No conclusive evidence has been reached, that feature-based explanations such as importance scores have a meaningful impact on user decision accuracy [106]. Even data scientist have been shown to over-trust XAI, not being able to accurately describe visual feature-based explanations provided [107]. Inspiration from experiments on human psychology or cognitive science, e.g., dual-process theories [108] [109] provides an approach to reduce subjectivity of understanding. Dual-process theory divides the information processing of humans into two systems, the intuitive (System 1) and analytical thinking (System 2). System 2 relies on careful, cognitively demanding reasoning, making humans resort to System 1. Since mental shortcuts are used here, this system is susceptible to cognitive biases [83]. These biases can also be present when interpreting XAI, where users are unlikely to carefully consider every bit of the explanation [110] [83]. E.g., it has been shown that users are vulnerable to anchoring bias, mentally sticking to model behaviors that

were observed early on [111] and associating a numerical presentation of explanations with intelligence and algorithmic thinking [112]. This is more likely to happen, when the user lacks the ability to understand the explanation in-depth [109]. In [113] it is shown that non-experts gain less from XAI compared to AI experts. Leveraging textual explanation can be used to reduce the cognitive workload and therefore be more understandable [113]. Therefore, one of the desired properties is **naturalness**, entailing the use of natural language [114]. For generating coherent natural language explanations, standard patterns of discourse employed by humans should be followed.

> "Person A is classified as football player because they are friends with many other football players"

It can also happen that explanations create information overload and distract people from forming a useful mental model of how a system operates [115]. Using explanations that incorporate reasoning leads to better user satisfaction and insight [116]. Equally, [117] argues that reasoning is a necessary component of true explainability, as leaving explanation generation to human analysts based on importance scores or visualizations can be dangerous. As the user depends on their background knowledge about the domain, the explanations about the decision-making of the model might differ dependent on the user. Or in other words, such explainability techniques that don't include reasoning thus enable explanations of decisions, but do not yield explanations themselves. In order to enable reasoning to achieve actionable understanding beyond algorithmic explanations and to fill potential knowledge gaps of the user, providing domain knowledge is necessary [83]. For actionable decision-making, domain experts need appropriate reliance, an understanding about when to trust the AI's prediction and when to be cautious. Therefore, the domain knowledge should be adjusted to the user's knowledge, goal and context, or in other words, explanation **sensitivity** [114] should be incorporated. This is particularly important when attempting to adjust the complexity of an explanation, its novelty and coherence [118].

> Domain knowledge: Football is a team sport $\rightarrow$ Football player have a higher connectedness on average.

For actionable decision-making, domain experts need appropriate reliance, an understanding about when to trust the AI's prediction and when to be cautious. The properties naturalness

and sensitivity enable explanations to be human-centric, ensuring a domain expert can understand them. While human-centricity is essential, objective metrics for evaluation should be considered equally. Relying on user studies for evaluating explanations might suffer from confirmation bias [119]. The limitations of the AI and equally the explainer techniques should be transparently communicated to the user [83]. In order to evaluate how faithful an explanation is, **fidelity** is a fundamental property, representing the closeness between model and explanations. Fidelity is a measure of the accuracy of the explanations in relation to the underlying ML model, which can be phrased as a measure of accuracy of the student model w.r.t the teacher model. The student model being the explainability technique and the teacher model being the GNN [120]. Without high fidelity, an apparently good and comprehensible explanation can be simply wrong, which has been shown for popular XAI methods such as LIME [121]. It has also been shown, that if an explanation is added to an AI system, high fidelity is crucial for ensuring user trust [122].

Since GNNs are based on graphs, which contain topology information, existing explainability techniques might not be suitable [69]. Graphs are represented as feature and adjacency matrices, which only contain discrete values. These cannot be optimized in the same manner as image classifiers, where input are treated as trainable variables and optimized via back-propagation obtaining abstract images as explanation [123], [124]. Other explainability techniques learn soft masks to capture important image regions [125] [126], which would also not work with the discreteness property of adjacency matrices. Since graphs can represent such complex data as molecules or social networks, the semantic meaning of the input data of GNNs can be more difficult to understand for the user compared to images or texts [3]. Such structural information is more important with respect to graph data as graph substructures, such as network motifs, can be highly related to their functionalities, e.g. in biochemistry or engineering domains [127]. Furthermore, since the nodes in the graphs can be unlabeled and graph labels may be determined solely by graph structures, studying individual nodes is meaningless as they don't come with semantic information. Likewise, network motifs, which are recurrent and statistically significant subgraphs or patterns of a larger graph, should be considered by in the explanation. Therefore, in order to provide an explanation for the decision-making of a GNN, it is essential to take into account graph-specific properties, in other words, to provide **reference to graph topology**. Generally, graph data is less intuitive than image and text data, where the semantic meaning of the input data is more straightforward, whereas complex graph data such as chemical graphs or citation networks are more

challenging to understand for the user [69].

> "Person C is classified as football player because they are part of a triad cluster with two other football players."

2.4.2 Sub-symbolic explanation methods: Subgraph Explanations

With the growing popularity of Graph Neural Networks and their usage in various domains which require explanations for scientific or ethical reasons, explainability techniques for GNNs have recently received a lot of attention [102]. As GNNs can carry out different tasks, including node-level, edge-level and graph-level tasks, generating explanations for different tasks can influence the explanation method, with some of them only being applicable only to specific tasks. The majority of explainability techniques for GNNs produce a subgraph of important features, nodes and edges, as explanation for a GNN model prediction, which are referred to as explainer subgraph [128]. The subgraph is based on importance scores for the individual edges, nodes, or features. The methods generating these importance scores can be categorized into perturbation-based, gradient/feature-based, decomposition-based and surrogate-based methods. Perturbation-based methods monitor the change of prediction with respect to different input perturbations, gradient/feature-based methods employ the gradients or the feature values to indicate the importance of different input features, decomposition methods decompose the prediction scores to the neurons in the last hidden layer and back propagate these scores layer by layer until the input space while surrogate-based methods sample a dataset from the neighbors of the given example and fit a simple and interpretable model, such as the decision tree, to the sampled dataset [69].

Initial works towards explainable GNNs attempts to convert gradient/feature-based and decomposition-based approaches initially designed for Convolutional Neural Networks (CNNs) into graph domain [7] [8]. The gradient-feature based methods include contrastive gradient-based saliency maps, describing the extent to which variations in the input would produce a change in the output; Class Activation Mapping (CAM) identifying important, class-specific features at the last convolutional layer as opposed to the input space; Grad-CAM, improving upon CAM by relaxing the architectural restriction that the penultimate layer must be a convolutional and Sensitivity Analysis (SA), producing local explanations for the prediction of a differentiable function using the squared norm of its gradient w.r.t. the input. Decomposition methods contain Excitation Backpropagation (EB), generating heat-maps that

Table 2.4: Comparison of different GNN explanation methods with regard to the requirements of naturalness, sensitivity, fidelity, graph topology; whether they are local or global and whether or which knowledge base they use.

	Natural-ness	Sensi-tivity	Fidelity	Graph Topology	Local	Global
Perturbation						
GNNExplainer [1]	✗	✗	✓	✓	✓	✓
PGExplainer [2]	✗	✗	✓	✓	✓	✗
SubgraphX [3]	✗	✗	✓	✓	✓	✗
GraphMask [4]	✗	✗	✓	✓	✓	✗
Cf-GNNExplainer [5]	✗	✗	✓	✓	✓	✗
ZORRO [6]	✗	✗	✓	✓	✓	✗
Gradient/Feature						
Pope et al. [7]	✗	✗	✓	✗	✓	✗
Baldassarre et al. [8]	✗	✗	✓	✗	✓	✗
Decomposition						
GNN-LRP [9]	✗	✗	✓	✓	✓	✗
Xiong et al. [10]	✗	✗	✓	✓	✓	✗
RelEx [11]	✗	✗	✓	✓	✓	✗
PGM-Explainer [12]	✗	✗	✓	✓	✓	✗
Surrogate						
GraphLIME [13]	✗	✗	✓	✓	✓	✗
Generation						
XGNN [14]	✗	✗	✗	✓	✗	✓

contain evidence for or against a network predicting any particular class and layer-wise Relevance Propagation producing relevance maps by decomposing the output signal of every transformation into a combination of its inputs.

The drawback of reusing explanation methods previously applied to CNNs are their inability to incorporate graph-specific data such as the edge structure. To overcome these problems, [1] created the model-agnostic perturbation-based approach GNNExplainer, that finds a subgraph of input data which influence GNNs predictions in the most significant way by maximizing the subgraph's mutual information with the model's prediction. Here, we briefly introduce this method by taking a graph classification task as an example. We assume a trained GNN model f_θ employed as a label function to produce the predicted label $\hat{y}$ of the input graph $G = \mathbf{A}, \mathbf{X}$, whose ground truth label is y. The GNNExplainer's goal is to find a subgraph $G_s = \mathbf{A_s}, \mathbf{X_s} \subset G$. This subgraph then acts as the explanation for prediction

$\hat{y} = f_\theta(G)$ by maximizing the mutual information (MI) between G_s and y:

$$\max_{G_s} MI(y, G_s)$$

GNNExplainer then proposes to learn an edge mask $M \in \mathbb{R}^{|V| \times |V|}$, with $|V|$ being the number of nodes in G and the explanation G_s is calculated as follows:

$$G_s = \{\mathbf{A_s}, \mathbf{X_s}\} = \{\mathbf{A} \odot \omega(M), \mathbf{X}\},$$

where ω denotes the sigmoid function.

Further perturbation-based explainer models, that obtain masks to indicate important input features, include PGExplainer [2], which trains a parameterized mask predictor to predict edge masks, which also comes with inductive properties. Given an input graph, it first obtains the embeddings for each edge by concatenating node embeddings. Then the predictor uses the edge embeddings to predict the probability of each edge being selected, similarly to an importance score. The approximated discrete masks are then sampled via the parameterization trick. Finally, the objective function maximizes the mutual information between the original predictions and new predictions. Similarly to GraphMask [4] and SubgraphX [3], the generated masks are combined with the input graph to obtain a new graph containing important input information, where the important input features captured by masks should lead to a similar prediction to the original one. GraphMask also trains a classifier to predict whether an edge can be dropped without affecting the original predictions, whereas SubgraphX explores different subgraphs via node pruning and selects the most important subgraph from the leaves of the search tree as the explanation of the prediction. CF-GNNExplainer [5] works by perturbing input data at the local, where the instances are nodes. The method iteratively removes edges from the original adjacency matrix based on matrix sparsification techniques, keeping track of the perturbations that lead to a change in prediction, and returning the perturbation with the smallest change w.r.t. the number of edges, after adding different edges to the subgraph. ZORRO [6] employs discrete masks to identify important input nodes and node features through a greedy algorithm, where nodes or node features are selected step by step. The goodness of the explanation is measured by the expected deviation from the prediction of the underlying model. For perturbation-based explanation methods for GNNs, the output consists of masks, indicating important input features, including node masks, edge masks or node feature masks depending on the explanation task. We can observe

three different types of masks that have been proposed, including soft masks [1][5], discrete masks [6] and approximated discrete masks [2]. These masks are then applied to the input graph(s) and fed into the trained GNNs to carry out predictions, which are targeted by the objective function to be similar to the original prediction. A further decomposition method is GNN-LRP, which studies the importance of different graph walks, with the GNN model being treated as a function and providing a view of high-order Taylor decomposition to develop the score decomposition rule [10].
A graph-specific surrogate explanation method is the PGM-Explainer [12], which entifies crucial graph components and generates an explanation in the form of a PGM approximating that prediction. Instead of drawing explanation from a set of linear functions of explained features, PGM-Explainer provides dependencies of explained features in the form of conditional probabilities. GraphLime [13] extends LIME [129] to deep graph models by studying the importance of different node features. A Hilbert-Schmidt Independence Criterion (HSIC) Lasso model is employed to fit the k-hop neighboring nodes and their predictions. The surrogate model selects important features to explain the HSIC Lasso predictions, which are then regarded as explanations of the original GNN prediction. However, graph structures such as nodes and edges aren't considered. RelEx [11] combines surrogate methods and perturbation-based methods, by employing a GCN model as a surrogate model to fit the local datasets and then applying perturbation-based methods to explain the predictions.
Reinforcement learning has been used in a global graph generation explainer approach [14], that proposes to explain GNNs via graph generation. global explanations make the prediction process of a particular model transparent, as opposed to a particular prediction instance. The generator predicts how to add an edge to the current graph, and the generated graphs are fed into the trained GNNs to obtain feedback to train the generator via policy gradient. The then generated graphs function as global explanations for a certain target class and should convey discriminating graph patterns.

In Table 2.4 it is shown how these methods fare in regard to the desired properties naturalness, sensitivity, fidelity, reference to graph topology and whether they provide local or global explanations. While there are a variety of methods that come with fidelity and reference to graph topology, human-centricity through sensitivity and naturalness is not given. [46] argues, that the main limitation of statistical methods is that it does not take domain knowledge or general background knowledge into account when generating an explanation. Based on experiences collected from real world applications, using a subgraph explainer

tends to be hard to understand, not only for domain experts but even for AI experts [128]. Furthermore, the majority of methods can only generate local explanations.

2.4.3 Symbolic explanation methods

As lined out above, symbolic methods come with inherent explainability. Using symbolic methods to create explanations for sub-symbolic machine learning models come with advantages, e.g., the establishing of reasoning rules that allows symbolic methods to be constrictive [95]. Since black-box models usually process data in a quantitative manner, it means that these probabilistic results have to be translated into qualitative notions containing causal links to provide a satisfying explanation [74]. As there is a trade-off between the simplicity of the information given by the system on its internal functioning and the exhaustiveness of this description or, in other words, between human-centricity and completeness [130], being selective when providing an explanation isn't straightforward. As symbolic methods process data directly in a qualitative way, they come with the ability for being selective and focusing solely on the main causes of a decision-making process [95].

2.4.3.1 Symbolic explanation methods: Non-ontological Explanations

There are a number of methods that create rule-based explanations without taking into account a knowledge base. Using decision trees to explain neural networks based on tabular data, has also gained some traction in the past, starting with TREPAN [17], which queries the neural network to induce a decision tree approximating the concepts represented by the networks by maximizing the gain ratio together with an estimation of the current model fidelity. In [131] a prototype for each target class is generated by using genetic programming to query the trained neural network, where the input features dataset is exploited for constraining the prototypes. The best prototypes are then selected for inducing the learning of the decision tree. Another method uses Genetic Programming to evolve decision trees [132] to mimic the behavior of the machine learning model. Similarly, Genetic Programming is used for rule extraction to explain a neural network, by using classical reverse engineering where random annotated permutations of the original dataset are used as input [133]. [134] also uses reverse engineering to generate classification rules for each class label, exploiting the data ranges previously identified by pruning input neurons. Decision rules and counterfactual rules are generated by a genetic algorithm on a synthetic neighborhood in [135]. Another rule extraction method proposes to learn rules in Conjunctive Normal or Disjunctive Normal Form [136], whereas in [137] model simplification is formulated as a model extraction process by approxi-

mating a transparent model to the complex one. In [138] a Monte Carlo algorithm is used for generation of decision rules as explanations by deriving a score based on formal requirements.

While some of these methods do provide fidelity and some naturalness, none of these methods include the grounding of explanations in domain knowledge and therefore enabling sensitivity or are able to consider graph-specific properties, as they have been designed for tabular data. More recently, approaches have been developed that integrate a KB when generating rules with [15] using free form attributes and captions. Another method is TCAV [16], where the authors use simple concept tags to produce human understandable explanations and therefore providing sensitivity.

2.4.3.2 Symbolic explanation methods: Ontological Explanations

Using a Knowledge Base such as an ontology or knowledge graph allows data to be processed directly in a qualitative way [66]. Furthermore, the integration of background knowledge via a KB into explanations carries the promise of being much closer to human conceptualizations and thus more useful for domain experts [46]. Using an ontological KB makes reasoning with symbolic representation possible [139].
Using background knowledge for enhanced explainability has gained attention recently, and there have been a number of XAI approaches developed that integrate Semantic Web Technologies with connectionist models [140]. The majority of these methods have been developed for image recognition, with one approach mapping network inputs or neurons to classes of an ontology or entities of a knowledge graph [15]. This enables the linking of a neuron's weight to semantically grounded domain knowledge. The authors in [19] map scene objects within images to classes of the Suggested Upper Merged Ontology. Based on the image classification outputted by the Neural Network, the authors run DL-Learner on the ontology to create class expressions that act as global explanations, resulting in an input-ouput matching. In [81], the authors extract image contents as RDF triples and then match them to DBpedia via the predicate same-concept, where user questions are translated into a SPARQL query which is run over the combined knowledge base. Explanations for image recognition on classes not included in the training data through zero-shot learning are generated in a similar fashion by [20]. In [21], KGs are used to represent symbolic expert knowledge to be leveraged by a convolutional neural network and Shapley explanations. Concept induction together with a class hierarchy is used in [22] to generate explanations for data differences in images.

Further neuro-symbolic approaches focus on how to integrate ontologies and how this influences the understandability of global explanations for artificial neural networks applied in finance and medicine domains (TREPAN Reloaded [141]), which is an extension of TREPAN [17]. [142] produces explanations for KG embeddings using domain ontologies. For generating local explanations of stock trend predictions, knowledge graphs were used [25]. A model-agnostic explainer [26] exploits temporal dimensions along with a medical ontology for explaining medical diagnosis predictions. The authors in [24] show how domain ontologies together with description logic based concept induction can be used to explain input output behavior of trained deep neural networks. In the domain of transfer learning, [23] use domain knowledge to enhance the explanations regarding which features are beneficial for the transfer.

In Table 1.1 it is shown how these methods fare in regard to the desired properties naturalness, sensitivity, fidelity, reference to graph topology, whether they provide local or global explanations and which KB they use. It can be seen that most of these methods provide naturalness, and all of them provide sensitivity as they give background knowledge by integrating a KB. Fidelity isn't considered at all, except partly in [18]. None of these methods can provide reference to graph topology and, except for [23], they provide either local or global explainability. Most KBs used are ontologies with [21] using knowledge graphs. For some of these methods, such as [19] [22] [23] [24], the causal links given by the KB do not directly reflect the operations that took place in the underlying machine learning model. There is no connection between the KB and the black-box, which makes it impossible to affirm that the explanation given is why the model predicted this output. Methods which come with a stronger integration of the inner workings of the ML model and the KB, such as [15] [21] also don't provide any fidelity metric, therefore making it hard to evaluate how close the student and teacher model are in reality. While the objective of reaching natural explanations, that provide sensitivity through integrating a KB is reached, even formulating a line of reasoning in some cases [23] [19], the explanation given might not be correct, defeating its whole purpose.

Further Model-Agnostic explanation methods There is a variety of model-agnostic explanation methods, that can be used on any type of machine learning algorithm with diversified data types and output features, decision trees or rules as explanations. While some of these methods do provide fidelity and some naturalness, none of these methods include the grounding of explanations in domain knowledge and therefore enabling sensitivity or are able to

Table 2.5: Comparison of different explanation methods containing symbolic elements with regard to requirements of naturalness, sensitivity, fidelity, reference to graph topology; local or global; and which knowledge base they contain.

	Naturalness	Sensitivity	Fidelity	Graph Topology	Local	Global	Knowledge Base
Non-Ontological Explanations:							
NIWT [15]	✓	✓	✗	✗	✓	✗	Free form
TCAV [16]	✓	✓	✗	✗	✓	✗	Free form
TREPAN [17]	✓	✗	✓	✗	✗	✓	NA
Ontological Explanations:							
TREPAN Rel. [18]	✓	✓	✓	✗	✗	✓	Ontology
Sarker et al. [19]	✓	✓	✗	✗	✗	✓	Ontology
Geng et al. [20]	✓	✓	✗	✗	✗	✓	Ontology
X-NeSyL [21]	✗	✓	✗	✗	✓	✗	Knowledge graph
Widmer et al. [22]	✓	✓	✗	✗	✗	✓	Ontology
Chen et al. [23]	✓	✓	✗	✗	✓	✓	Ontology
Labaf et al. [24]	✓	✓	✗	✗	✗	✓	Ontology
KDTCN [25]	✓	✓	✗	✗	✓	✗	Knowledge graph
Doctor XAI [26]	✓	✓	✓	✗	✓	✗	Ontology

consider graph-specific properties, as they have been designed for tabular data. That's why we will give only a short overview. Prominent examples here are LIME [93], which does not depend on the type of data or type of machine learning model. The explanations, which are features along with importance values, are derived locally from records generated randomly in their neighborhood and weighted according to their respective proximity. Extensions of LIME have been developed, analyzing particular aspects and use cases [129] [143]. Another well-known approach is SHAP (SHapley Additive exPlanations) [144], where an additive feature importance score is calculated for each particular prediction with a set of desirable properties.

2.4.4 Evaluation Measures and Metrics

With the emergence of a variety of different explainability techniques, the question of how to evaluate these methods is posed. There is no consensus what the most suitable metrics are to evaluate an explanation's quality, especially since the target audience for explainability is heavily heterogeneous [119] and scholars from different disciplines are focused on different XAI goals [71]. One way to evaluate the explanatory value of the generated explanations are questionnaires and interviews [71], such as evaluation explanation types based on satisfaction and transparency [145]. Even though the overarching goal is explainability, different measures and metrics are used to evaluate the respective goals [71]. Generally, the goodness of the explanation, user satisfaction and human-understandability as well as user trust are considered [66]. While user studies are commonly used to evaluate explanations and the underlying XAI, these studies may suffer from confirmation bias [119]. Here, the issue of user studies being constructed to confirm explanation effectiveness should not be underestimated [146]. Furthermore, there is a large variability of human understanding [147], leading to limited meaning of study results, which have been shown that they cannot capture the connection between explanations and user performance. To rely on user studies to evaluate explanations may lead to persuasive explanations rather than transparent systems due to user preference for simple explanations [148]. This is especially likely for explanations generated with a separate logic post-hoc via proxy methods, that differs fundamentally from the logic used by the underlying AI [73]. This means, a user could be satisfied with and trusting an explanation solely based on confirmation bias or preference for simple explanations, while the explanation itself is simply wrong [119]. Therefore, objective metrics should be used for evaluation. While both sub-symbolic and symbolic methods use accuracy metrics, their implementation can differ. Metrics used later on in the thesis are introduced below.

Further metrics commonly used for evaluating explainability include computational efficiency, stability of explanations or interpretability metrics.

Metrics for sub-symbolic explanation methods Many GNN explanation methods come with differing evaluation protocols, which makes their comparison difficult. As outlined in [149], evaluation of sub-symbolic explanation methods can come with a number of pitfalls, including the choice of evaluation metrics. Validating explanations is generally a challenging task because a ground-truth explanation is not always available. As there are several aspects that can be considered when evaluating an explanation, we have included a number of different metrics that have been recently proposed by [69], in order to evaluate our sub-symbolic explanation method RERE. We are including fidelity and, in case of ground-truth availability, accuracy in our quantitative analyses. Sparsity is used as a baseline to arrive at meaningful comparisons between the explanation methods.

Accuracy In synthetic datasets, even though it is unknown whether the GNNs make predictions in our expected way, the rules of building these datasets, such as graph motifs, can be used as reasonable approximations of the ground truth. That is why, for any input graph, we can compare its generated explanations with such ground truth motifs. For example, we can study the matching rate for edges in the explanations with those in the ground truth, e.g., all edges in a "house"-motif". For comparability with methods, where a threshold is necessary to arrive at the final explanation presented to the user, we choose F1-Score as accuracy metric.

Fidelity The model should identify input features that are important for the model, not intuitive for humans. To evaluate this, the Fidelity metric is recently proposed. The predictions shouldn't change significantly, when as not important identified input features, which aren't discriminative to the model, are being removed. Let G_i denote the i-th input graph and f denote the GNN model to be explained. Fidelity studies prediction change by keeping important input features and removing unimportant features. Fidelity can be computed as

$$\text{Fidelity} = \frac{1}{N}\sum_{i=1}^{N}(f(G_i)_{y_i} - f(G_i^{m_i})_{y_i}), \tag{2.7}$$

where G_i^m is the new graph when keeping important features of G_i based on explanation m_i. Lower values indicate less importance information are removed, resulting in better explanations.

Sparsity Good explanations should be sparse, which means they should capture the most important input features and ignore the irrelevant ones. Therefore, the sparsity metric measures the fraction of features selected as important by explanation methods [7]. Given the graph G_i and its features identified as important m_i, the Sparsity metric can be computed as

$$\text{Sparsity} = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{|m_i|}{|M_i|}), \tag{2.8}$$

where $|m_i|$, denotes the number of important edges identified in m_i and $|M_i|$ means the total number of edges in G_i. Higher values indicate the explanations are more sparse.

Metrics for symbolic explanation methods **Accuracy** We follow [17] and [141] and measure accuracy and fidelity on the examples in the test sets. Accuracy is defined as the percentage of test-set examples that are correctly classified, and is used to evaluate the explanation method regarding unseen data. If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over N is defined as

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}1(\hat{y}_i = y_i), \tag{2.9}$$

where 1 is the indicator function. It has to be kept in mind, that if the black-box model to be explained comes with low accuracy, the accuracy of the explanation method might be low as well [150]. Furthermore, recall and precision are reported, to show the full picture.

Fidelity Here, we use a definition of fidelity, where the percentage of test-set examples on which the classification made by a tree agrees with its black-box counterpart, is computed. When y_i^* is the predicted value of the black-box model, fidelity is defined as

$$\text{Fidelity} = \frac{1}{N}\sum_{i=1}^{N}1(\hat{y}_i = y_i^*), \tag{2.10}$$

This is a very simplified notion of fidelity and should only be considered as an indication. We use it to compare our method against the benchmark.

Comparative Evaluation In order to evaluate the quality of explanations in a quantitative way, several approaches have been developed. While there are different terms used

for this metric in the literature, the core concept is similar, and we will be referring to it as comparative evaluation. Several works [144] [151] [152] [153] use ground truth statements (annotations) given by humans, which are then compared with the generated explanation. Similarly to these concepts, we will use comparative evaluation. The comparison is qualitative and quantitative. For quantitative evaluation, the rate of ground truth statements covered by the explanations provided is computed.

3 Subgraph Explanations

3.1 Problem Definition

Existing perturbation-based methods as introduced in Section 2.4.2 focus on explaining GNN predictions by obtaining masks to indicate important input features, for example by maximizing the mutual information between the original predictions and predictions with the masked graph [69]. While the majority of these methods do come with explanation fidelity and reference to graph topology, they don't provide sensitivity. No domain or general background knowledge can be incorporated when generating explanations. Experiments show that this can for example lead to a Label-Flip problem [149].

3.2 Concept

In contrast, we are proposing a sequential method, which uses reinforcement learning to provide explanations in the form of a subgraph containing only the information that is most critical to the decision-making process of the GNN. The approach is model-agnostic and can explain predictions of any GNN. This includes node classification, link prediction, and graph classification models. The approach is designed as a reinforcement learning agent that operates within a subgraph reduction environment. Its state space is composed of the graph used by the GNN to make a prediction (i.e., the receptive field) with its action space containing the graph's edges. The action itself is the removal of an edge chosen by the agent, trained via policy gradient to optimize a reward composed of the prediction's resulting entropy.

Compared to other explainer methods, our approach can offer an even deeper insight into the decision-making of the GNN, as the edges are removed step by step. Thereby, the least informative nodes are removed first. The user can easily follow and understand the explanation generation process, providing a traceability mechanism, which is a key requirement for trustworthy artificial intelligence [72]. Due to its sequential operation, it is possible to

include constraints when generating the explanation. This enables the integration of domain knowledge, e.g., eliminating the Label-Flip issue that is exhibited by other explainer methods. The properties of this subgraph explanation method are shown in Figure 3.1. In our method, the actions to generate the explanation are conditioned on the prior state as by the Markov assumption, leading to a relaxation of the independence assumption found in existing methods [69]. Additionally, our approach does not suffer from the conceptual "introduced evidence" problem [125] as opposed to other perturbation-based methods [69], that output importance masks as explanations. Here, any non-zero or non-one value in these masks may introduce new semantic meaning or noise to the input graph, thus affecting the explanation results. In comparison to the state-of-the-art graph explanation method GNNExplainer [1], where masks are optimized for each input graph individually, we develop an inductive approach. This means that the agent's policy can be applied to graphs not appearing during policy optimization, which demonstrates a global understanding of the trained GNN.

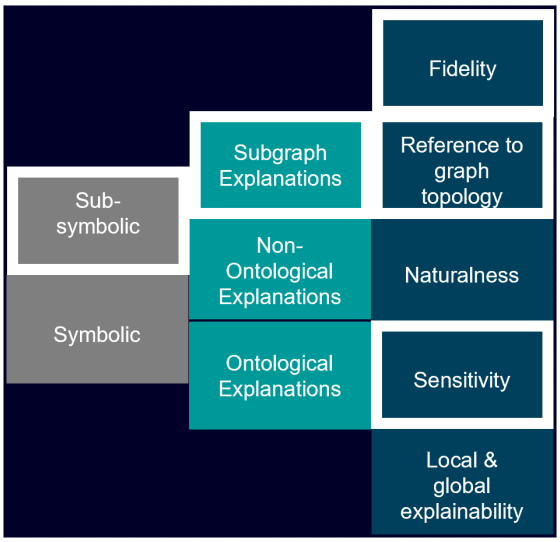


Figure 3.1: Technology, type and properties of our method

3.3 Proposed Method

We now introduce the details of our method, namely Receptive Field Reducer for Explaining Graph Neural Networks (RERE), beginning with the rationale for our choice to leverage reinforcement learning. A reinforcement learning approach for reducing the receptive field of the center node presents several advantages compared to alternative optimization methods. Firstly, through removing edges sequentially, the actions are conditioned on the prior state,

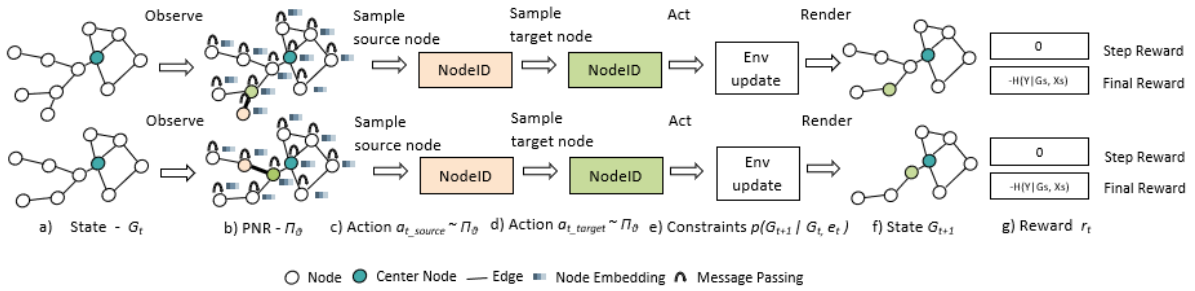


Figure 3.2: An overview of the proposed iterative explainer graph reduction method. Each row corresponds to one step in the reduction process. (a) The state is defined as the intermediate k-hop subgraph G_t (b) The policy network conducts message passing to encode the state as node embeddings to then produce a policy Π_θ. (c) An action $a_{t,\text{source}}$ and $a_{t,\text{target}}$ are sampled from the policy. (d) The environment performs a check on the intermediate state, and then returns (e) the next state G_{t+1} and (f) the associated reward r_t.

consequently relaxing the broader independence assumptions introduced by earlier methods where all actions are made independent of each other. Secondly, reinforcement learning is capable of directly integrating hard constraints and desired properties of the final subgraph into the optimization process (e.g., by integrating domain knowledge into the design of environment dynamics and reward function). Constraints like the desired connectedness of the subgraph or the avoidance of Label-Flips can be easily implemented. Thirdly, our approach comes with inductive properties. The policy can be optimized and then applied to explain instances that it hasn't seen yet without the need for any further computations.

Figure 3.2 shows an overview of the proposed iterative explainer graph reduction method, where each row corresponds to one step in the generation process. The state is defined as the intermediate k-hop subgraph G_t, on which a GNN operates to encode the state in the form of node embeddings. Based on the learned policy Π_θ, action $a_{t,\text{source}}$ and $a_{t,\text{target}}$ are drawn. Each action $a_{i,\text{source}}$ corresponds to a source node, for which subsequently a target node is chosen through action $a_{i,\text{target}}$. The source and target nodes must be connected by an edge, which is removed. The environment performs a validity check with respect to the specified constraints on the intermediate state, and then returns the next state G_{t+1} along with the associated reward r_t, as lined out in Section 3.3.

Explainer Subgraph Reduction as Markov Decision Process RERE is modeled as a Markov Decision Process, where we design an iterative subgraph reduction process defined as

$M = (S, D, R, p, \gamma)$, where $S = \{G_i\}$ is the set of states that consists of all possible subgraphs, $D = \{a_i\}$ is the set of actions, where each action corresponds to a specific edge removal resulting in adjacency matrix $\mathbf{A}$ without $e = (v_i, v_j)$. $R(G_i)$ is a reward function that computes a reward according to the generated state G_i, p defines the environment dynamics of the process and γ is the discount factor.
The process to reduce the subgraph sequentially can then be described by a trajectory $(G_0, a_0, r_0, ..., G_n, a_n, r_n)$, where G_n is the final subgraph. The reduction of the subgraph at each time step t can be viewed as a state transition distribution $p(G_{t+1}|G_t, ..., G_0) = \sum_{a_t} p(a_t|G_t, ...G_0)p(G_{t+1}|G_t, ...G_0, a_t)$, where $p(a_t|G_t, ...G_0)$ is represented as a policy network Π_θ. A policy is defined as the probability distribution of actions given a state, and the objective of a RL agent is to maximize the "expected" reward when following a policy Π_θ as detailed in Section 3.3.

RERE Environment RERE environment reduces the initial subgraph step by step through edge removal actions. The environment consists of the state space, action space and reward design.

State Space: Assuming there are k layers in the GNN, the prediction only relies on its k-hop computation graph, denoted as G_0. Therefore, the starting state is G_0, which corresponds to the GNN's receptive field. The subsequent state G_1 equals the G_0 with edge $e_0 = (v_{0,\text{source}}, v_{0,\text{target}})$ removed, determined by the action pair $a_{0,\text{source}}$ and $a_{0,\text{target}}$, unless the action was rejected due to a constraint violation defined by the environment dynamics p. In case the action was rejected, G_{i+1} equals G_i. The final state G_n is determined by the policy network.

Action Space: Based on the initial k-hop subgraph, we define a set of nodes $\{v_1, ..., v_n\}$ that can be removed. The action space contains all nodes in the entire current subgraph and a stop action. Each action $a_{i,\text{source}}$ corresponds to a source node, for which subsequently a target node is chosen through action $a_{i,\text{target}}$. The set of possible target nodes, are all nodes that are linked through a direct edge to the source node. The source and target nodes must be connected by an edge, which is then removed. Since the edge removal is modelled as the action space, RERE is as expressive as other perturbation-based approaches using edge masking methods. After the action has been carried out, the action space and the subgraph G_t is reduced by the chosen edge. With this approach, the "introduced evidence" problem

is avoided, as no new semantic meaning or noise is added to the input graph. The feature matrix $\mathbf{X_t}$ is analogously updated to $\mathbf{X_{t-1}}$. If the action has been rejected, the state space remains unchanged while the action space is reduced. If the stop action is selected at step t, the agent stops removing further edges, and the final state G_n equals G_{t-1}.

State Transition Dynamics: Domain-specific rules are incorporated in the state transition dynamics. The environment carries out actions that obey the given rules. Infeasible actions proposed by the policy network are rejected and the state G_t equals G_{t-1}. This includes a connectivity constraint for graph classifications, to arrive at a connected subgraph as explanation, since unlike node classification, it is no longer true that the explanation would have to be a connected subgraph. Therefore, the largest connected component is extracted as the explanation. Here, it is possible to include domain knowledge, by rejecting any action that would result in contradicting such. One such introduction of knowledge is the Label-Flip issue. With subgraph perturbation, it is possible that Label-Flips occur. The originally predicted label isn't predicted anymore with the final explainer subgraph as GNN input, therefore defeating its purpose of explaining the original prediction. To avoid this, any action that would result in a Label-Flip is rejected. In case of node classification explanations, any action that would lead to a removal of the center node is rejected. Furthermore, there is the possibility to include a size constraint, when a certain minimum size or sparsity for final explanation is desired.

Reward Design: In our environment, we consider only final rewards to guide the behavior of the reinforcement learning agent. The reward r_T is calculated based on the certainty of the classification decision given the final state of the reduced subgraph G_n. Since G_n functions as the explanation for the decision-making of the GNN, it should therefore contain the information that increased the certainty of the respective decision-making. To determine the certainty with which a classification decision has been made by the GNN, we consider the entropy of the predicted class distribution. Calculating the information of a random variable is the same as calculating the information for the probability distribution for the random variable. The entropy function $E(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$ gives us the information for a random variable. The lowest entropy is calculated for a random variable that has a single event with a probability of 1, a certainty. For example, in the case of node classification, removing an edge from the graph should not increase the entropy (i.e., uncertainty) regarding the center node's predicted class distribution but instead should preferably decrease or maintain the entropy. This results in a more interpretable subgraph without any information loss. Here, the change

in entropy is rewarded or penalized, depending on its increase or decrease. The reward is calculated as follows:

$$r_t = E(f(G_0, X_0)) - E(f(G_t, X_t)) \tag{3.1}$$

where f denotes the GNN model.

Policy Network Having outlined the receptive field reduction process, we now introduce the architecture of the policy network used by the agent to act in the environment. The policy network takes the intermediate graph G_t and outputs a distribution over the actions $a_{t,\text{source}}$ and $a_{t,\text{target}}$, which are used to select which edge to remove, as described in Section 3.3.
Action prediction We formalize the action selection according to a two-step link prediction task at time step t. First, either the stop action or a source node is selected. Secondly, in case of the latter, a target node is selected. The components are sampled according to a predicted distribution governed by the Equations 3.2 and 3.3.

$$a_{t,\text{source}} \sim \Pi_\theta(G_t) = \sigma\left(\mathbf{H}^{(L)}_{t,centre}\mathbf{W}_{\Pi,source}\left[\mathbf{H}^{(L)}_t || \mathbf{h}_{stop}\right]^\intercal\right) \tag{3.2}$$

$$a_{t,\text{target}} \sim \Pi_\theta(G_t, a_{t,\text{source}}) = \sigma\left(\mathbf{H}^{(L)}_{t,source}\mathbf{W}_{\Pi,target}\mathbf{H}^{(L)\intercal}_t\right) \tag{3.3}$$

where $\mathbf{H}^{(L)}_t = GNN_\theta(G_t, X_t)$, $\mathbf{W}_{\Pi,source}$ and $\mathbf{W}_{\Pi,target}$ are weight matrices and $\mathbf{h}_{stop}$ is the latent representation of the stop action. The information from the first action is incorporated in the selection of the second node. The stop action is part of the action space for the first action $a_{t,\text{source}}$. If the stop action is selected, the process terminates immediately, and no target action $a_{t,\text{target}}$ is sampled.

Policy Gradient Training To increase the stability of the training process by reducing the variance in the reward, we introduce an additional network $\hat{V}_\phi$ which approximates the value function as a state-dependent baseline [154]. The node embeddings $\mathbf{H}$ used by $\hat{V}_\phi$ are shared with the policy network.

$$\hat{V}_\phi(G_t) = \langle \mathbf{w}_\phi, [\mathbf{H}^{(L)}_{centre} || \mathbf{H}^{(L)}_{source}] \rangle \tag{3.4}$$

The intuition here is that the center node embedding is a good state representation to estimate the state's value. Once a source node has been picked, its node embedding is concatenated to the state for the target action state value estimation, otherwise $\mathbf{H}^{(L)}_{source} = \mathbf{0}$.
The policy network Π_θ is optimized according to the policy gradient objective [154]:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathop{\mathbb{E}}_{\tau\sim\Pi_\theta}\left[\sum_{t=0}^{T} J_t\right] = \mathop{\mathbb{E}}_{\tau\sim\Pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log\left(\Pi_\theta(a_t|J_t)\right)\left(J_t - \hat{V}_\phi(J_t)\right)\right], \tag{3.5}$$

where $J_t = \left(\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}\right)$ are the discounted rewards. The parameters of the value estimator $\hat{V}_\phi$ are optimized w.r.t. $\frac{1}{T}\sum_t^T \left(\hat{V}_\phi(J_t) - r_t\right)^2$.

Label-Flip rate

Reducing the size of the original subgraph is a post-processing step, carried out after training the GNN. It is possible that the originally predicted label can flip. This means, that the final explainer subgraph would lead to a different prediction than the original subgraph, defeating its purpose of explaining the original prediction. We pose the Label-Flip as domain knowledge, as we know a Label-Flip should not occur, meaning this metric is used as a proxy to measure sensitivity. The Label-Flip rate shows the average number of Label-Flips occurring for the generated explanation subgraphs:

$$\text{Label-Flip rate} = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i{}^{m_i} = \hat{y}_i), \tag{3.6}$$

with $\hat{y}_i$ being the predicted value of the i-th sample with the original graph G_i and $\hat{y}_i$ and $\hat{y}_i{}^{m_i}$ is the new prediction based on the new graph G_i^m.

3.4 Illustration for SportsNetwork

As described in Section 2.4.1, an explanation for GNN decision-making should come with certain properties, namely naturalness, sensitivity, reference to graph-topology and fidelity. We will look at to what extent our method RERE exhibits these properties illustrated on the toy example SportsNetwork. Figure 3.3 shows an explanation example for SportsNetwork. Here, the blue center node has been correctly classified as equestrian. Figure 3.3 (1) shows the original receptive field used by the GNN to make this prediction. In (2), (3), (4) and (5) the stepwise removal of edges can be seen. (6) shows the final explanation. The dashed red lines show which edge is predicted to be removed next.

Since the explanation is presented as an explainer subgraph, naturalness isn't given. However, since this method is a sub-symbolic explanation method that incorporates the GNN and outputs an explainer subgraph, reference to graph-topology is incorporated in the explanation

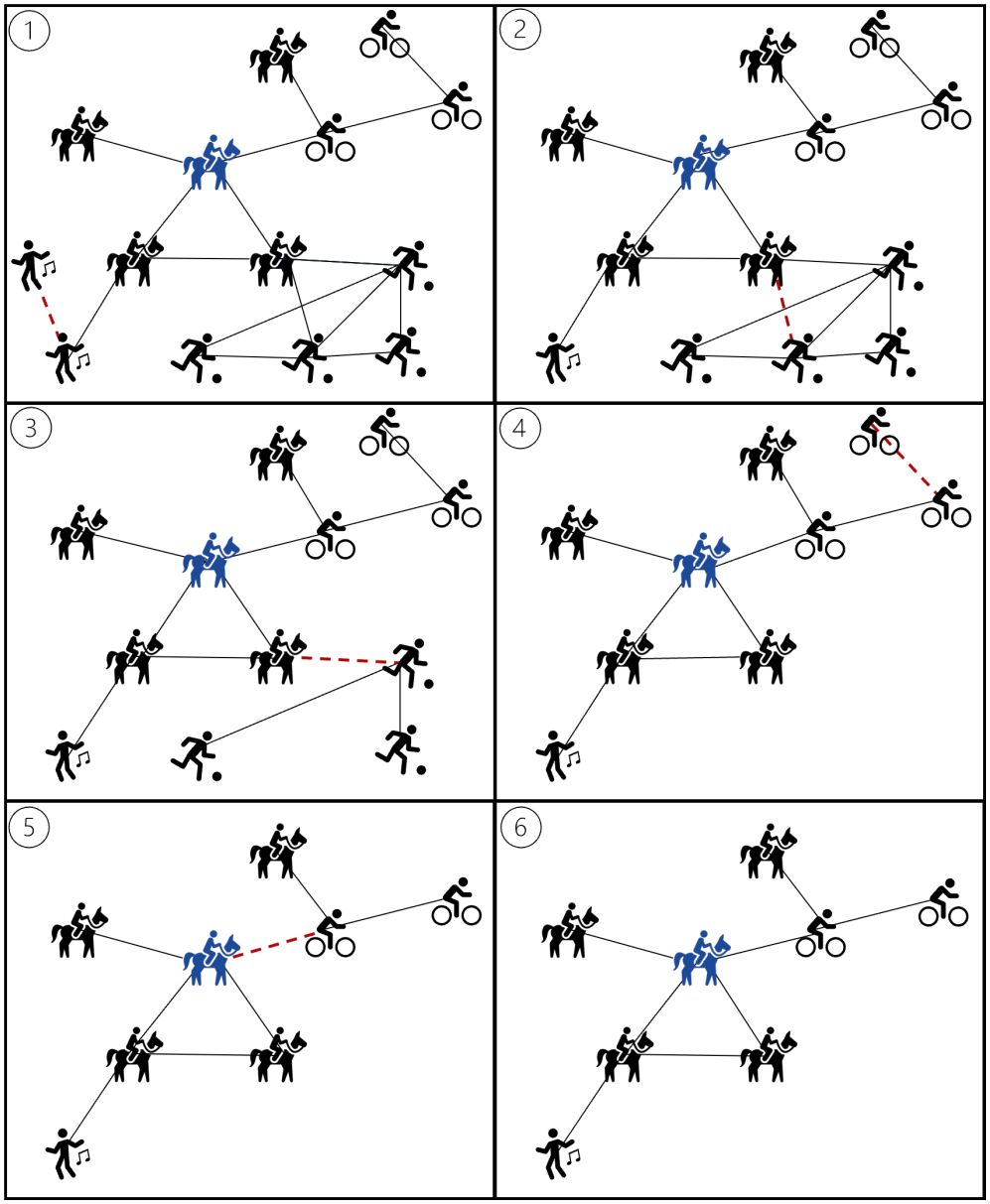


Figure 3.3: SportsNetwork example for an explanation for the classification of a node as an equestrian (blue). The edge identified to be removed in the next step by RERE is marked in dashed red.

method. Similarly, with fidelity, which will be shown in Section 6.

One advantage of our sub-symbolic explanation method over state-of-the-art methods is its capability to include domain knowledge through its stepwise removal process of edges, and therefore providing sensitivity to some extent. As introduced in Section 3.3, the Label-Flip issue can be posed as the incorporation of domain knowledge. As shown in Figure 3.3 (5), the red dashed edge has been predicted to be removed in the next step, however this action is rejected as it would be contradicting the knowledge that Label-Flips can lead to a wrong explanation. Therefore, the edge is not removed, and the domain knowledge is upheld.

4 Non-ontological Explanations

4.1 Problem Definition

While there are a number of different symbolic approaches extracting global explanations from neural networks for enhanced naturalness and sensitivity by integrating a KB, none of these methods provide reference to graph topology. An overview of these methods can be found in Section 2.4.3.2. Graph topology can have significant impact on the decision-making of a GNN. There, providing explanations that take it into account is essential for their accuracy in explaining the GNN's decision-making.

4.2 Concept

We address the lack of naturalness and sensitivity of sub-symbolic existing approaches and the lack of reference to graph topology of symbolic ones by proposing a post-processing rule-based companion to a sub-symbolic explanation method, with the conceptual schema shown in Figure 4.2. Thereby, we aim to complement the sub-symbolic local or instance-level explanations with global rules. By extracting and aggregating global rule-based explanations through a standard white-box machine learning method from the generated explainer subgraph, we reduce the amount of additional interpretation needed by the user and provide an explanation, that captures explanations about the global behavior of a model by investigating what input patterns can lead to a specific prediction.

The goal is to generate systems that can provide global rule-based explanations, which provide sensitivity and naturalness while including reference to graph topology, as shown in Figure 4.1. We can achieve this by combining the results of a sub-symbolic explanation method with a white-box rule generator, which satisfies the representation needs for human comprehensibility. The rule-based explanation generation isn't a stand-alone approach, but an add-on post-processing method in order to enhance the explanations and make them more user-centric. After training a GNN, the GNN's decision-making process is interpreted

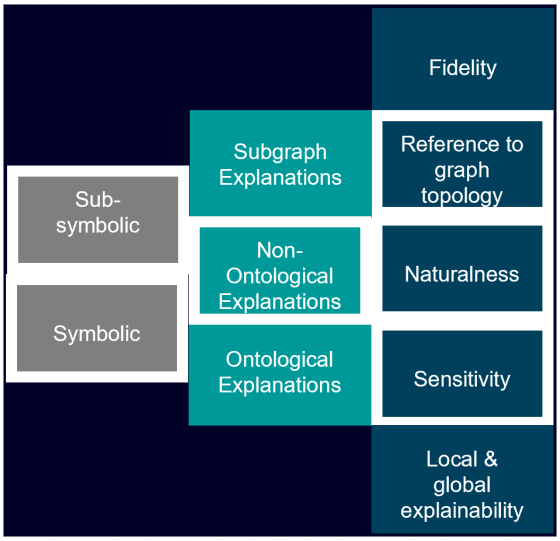


Figure 4.1: Technologies, type and properties of our method

by identifying a sparse receptive field containing influential elements. Our post-processing approach consists of taking these initial sub-symbolic explanations and lifting them to the level of rules.

An edge mask M_E and node feature mask M_X are generated by a sub-symbolic explainer model F_{ex}. The edge mask M_E contains all edges that have been identified as influential by the explainer model, the remaining edges are masked out, the node feature mask M_X is created analogously. Subsequently, rules for edge and node features are created by the white-box models D_E and D_X. The rules are created through a classification process, where the individual edges and features are assigned binary labels “influential" or “not-influential" based on their masking value. The white-box model is trained using the binary masks as the target variable. Note that depending on the learning and explanation tasks, different masks could be generated, including node masks, edge masks, and node feature masks. The approach works for any type of learning problem and mask. The generated rules then function as a global explanation for the respective classes of the original classification problem and furthermore, introduce a verbalization element that comes with higher naturalness than edge mask M_E and node feature mask M_X, resulting in an explainer subgraph G_S as is shown in Figure 4.3.

An important part of generating a user-centric explanation is to make the explanation customizable and include provenance, e.g. by including information about the domain knowledge utilized by the system to increase user-understandability and acceptability. Such domain knowledge can be included by extracting attributes from the subgraphs depending

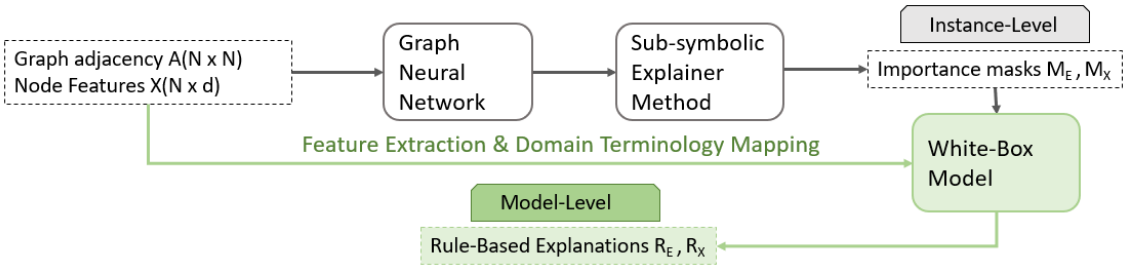


Figure 4.2: Conceptual schema of generating rule-based explanations

on the domain and classification task. An example of such inclusion of domain knowledge are graph-specific attributes, such as motifs mapped to the domain language.

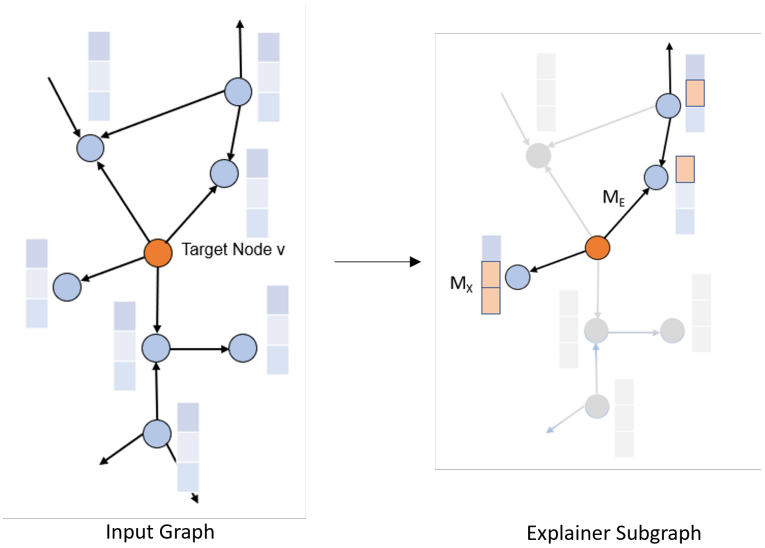


Figure 4.3: Explainer Subgraph as explanation of a node classification instance.

Personalization of explanations always comes with the drawback of reduced generalization. Which domain knowledge to chose for which approach is an open research question and very much dependent on the respective field and the needs of the user along with their current scenario and context. It is not the intention of our approach to provide a one-size-fit-all explanation method, but to show how such user-centric properties can be included in an explainer system. That means, that depending on the application and the user's need, guidelines should be established regarding the inclusion of any domain-knowledge.

Algorithm 1: Rule-Based Explanation Generation

Data: Explanation Subgraph Model: F_{ex};
Graph adjacency: $\mathbf{A}(N \times N)$;
Node features: $\mathbf{X}(N \times d)$;
Set of attributes: $L = \{a_1, \ldots a_L\}$;
Attribute mapping dictionary: $T = \{a_1 : t_1, \ldots a_L : t_T\}$;
White-box model: D_E, D_X
foreach $category = 1, 2, \ldots, K$ **do**
  foreach $node = 1, 2, \ldots, N$ **do**
    M_X, M_E = $F_{ex}(node, A, X)$ **foreach** *support_node in* M_E **do**
    end
    If $M_E(support\_node) \rightarrow y_{support}$
    $x_{support}$ = Extract-Attributes($support\_node$, M_X, M_E, L)
    $x_{support_m}$ = Map-Attributes($x_{support}$, T)
    D_E.fit($x_{support_m}$, $y_{support}$)
  end
  foreach *support_node in* M_X **do**
    If $M_E(support\_node) \rightarrow y_{support}$
    $x_{support}$ = Extract-Attributes($support\_node$, M_X, M_E, L)
    $x_{support_m}$ = Map-Attributes($x_{support}$, T)
    D_X.fit($x_{support_m}$, $y_{support}$)
  end
end

4.3 Proposed Method

To test this conceptual approach, we propose the method SUBGREX, which enables the generation of global explanations. We chose the GNNExplainer [1] as the sub-symbolic explanation method F_{ex}. Our method will work with any other sub-symbolic explanation method that generates importance scores for explainer subgraphs. The GNNExplainer takes a trained GNN and its prediction(s), and it returns an explanation in the form of a small subgraph of the input graph together with a small subset of node features that are most influential for the prediction, or in other words edge mask M_E and node feature mask M_X. For their selection, the mutual information between the GNN's prediction and the distribution of possible subgraph structures is maximized, as is described in Section 2. For example, consider node v_i, the mutual information quantifies the change in the probability of prediction when v_i's receptive field is limited to the explainer subgraph and its none masked-out node features as is shown in Figure 4.3. Then, if removing v_j from the subgraph strongly decreases the probability of prediction $\hat{y}_i$, the node v_j is a good counterfactual explanation for the prediction at v_i. For SUBGREX, the output of the GNNExplainer, edge mask M_E and node feature mask M_X, are used as input for the rule generation step. As method for extracting the rules, the standard machine learning mechanism decision tree is employed with $D_E = ID3_E, D_X = ID3_X$. The decision tree $ID3$ is a flowchart-like structure in which each internal node represents a "test" on an attribute such as a node feature and each branch represents the outcome of the test. Each leaf node represents a class label, which is the decision taken after computing all attributes. The decision trees can be linearized into decision rules R_E and R_X, where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if-clause.

The classification by the decision tree is binary and based on whether the node v_i is considered influential by the chosen subgraph generation method stemming from the edge mask M_E, which is a 0/1-valued vector and therefore our target attribute for the white-box model D_E. As lined out in Section 2, decision trees are a popular surrogate model to explain the global behavior of a black-box model, as is illustrated in Figure 4.4. As it is model-agnostic and doesn't require any information about the inner workings of the black-box, access to the data and prediction function is sufficient. TREPAN [17] and TREPAN Reloaded [18] are the most prominent examples of explaining sub-symbolic models, such as Neural Networks with a decision tree. While [17] matches the input data with the output, [18] takes into account domain knowledge by integrating ontologies into the explanations. The SUBGREX method goes further, by integrating a sub-symbolic explanation method and graph-specific domain

knowledge such as motifs, shown in Figure 4.5.

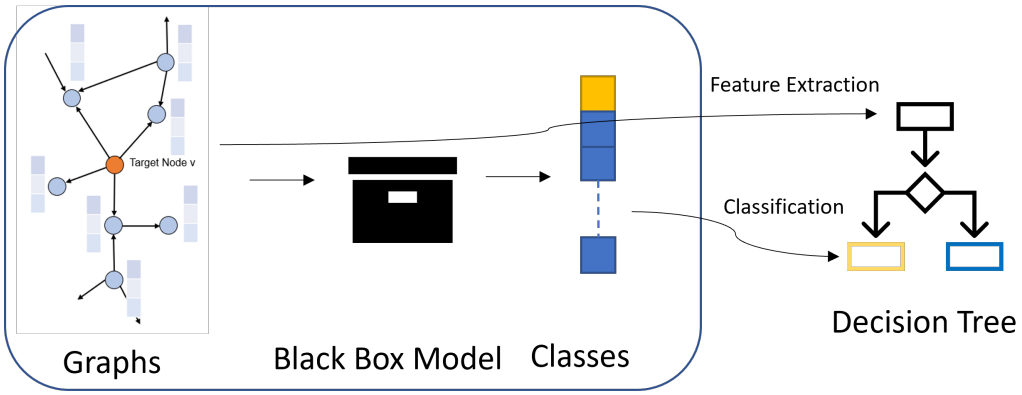


Figure 4.4: Decision Tree used to explain black-box model

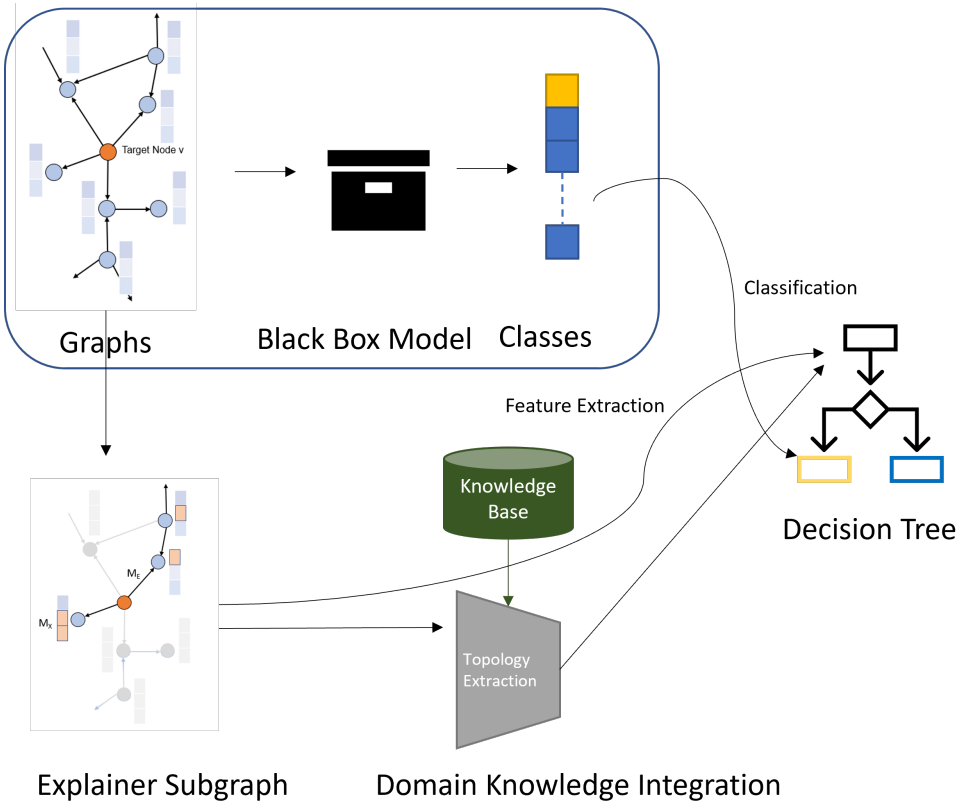


Figure 4.5: SUBGREX Workflow

Graph Attributes The attributes $a_1, \ldots a_L$ for the D_E input are extracted from node, edge, and graph meta information from the explanation subgraphs. Since graph architecture offers

more information than tabular data, it is vital to include reference to graph topology, such as network motifs. Further graph-specific data could include node degree (for node classification) or edge direction. In order to enhance the sensitivity of our approach, the attributes are personalized to the domain by mapping them to the respective domain terminology, as shown in Algorithm 1.
Especially, graph motifs have strong explanatory potential, as they usually contain semantic meanings that are indicative of the characteristics of the whole graph [39]. The used motifs should capture semantic meanings, e.g., similar to meaningful substructures in the respective domains. Therefore, a motif dictionary, containing all potential meaningful motifs, should come from domain knowledge. An example of a motif from the toy example SportsNetwork introduced in Section 2.3 is the clustering triadic motif.

The choice of graph-specific attributes is dependent on the learning task. For node classification problems, attributes such as whether the node is part of a motif, the distance to the target node (or number of hops) or the node degree are collected for each node. Node n has the following set of attributes:

$$L(n_i) = \{a_{in\_motif_1}(n_i), a_{in\_motif_2}(n_i), ..., a_{in\_motif_t}(n_i), a_{node\_distance}(n_i), a_{node\_degree}(n_i)\}$$

For graph classification or link prediction, the respective attribute whether the graph g_i contains a motif should be included:

$$L(g_i) = \{a_{contains\_motif_1}(g_i), a_{contains\_motif_2}(g_i), ..., a_{contains\_motif_t}(g_i), a_{connectivity}(g_i)\}$$

By choosing a subset of these features and therefore personalizing the method to the domain, it is inevitable that domain knowledge and potential bias is introduced into our approach. In order to enhance the user-friendliness of our approach, the attributes are personalized to the domain by mapping them to the respective domain terminology.

$$L(individual_i) = \{a_{in\_triadic\_cluster}(individual_i), a_{node\_degree}(individual_i)\}$$

SportsNetwork Explanations:
$R_{E1}(Equestrian)$ = If Individual is part of Equestrian triadic cluster (prob: 97%)
$R_{E2}(Equestrian)$ = If individual has node_degree smaller or equal than 5 (prob: 65%)
$R_{E1}(FootballPlayer)$ = If Individual is part of Football triadic cluster (prob: 78%)
$R_{E2}(FootballPlayer)$ = If individual has node_degree higher than 5 (prob: 66%)

Node Attributes The input used to generate feature selection rules are the instances x_i of the node features of all the nodes of the subgraph remaining based on the edge mask M_E. This means that the number of attributes L equals the combined amount of unique features x_i of all nodes in M_E. The corresponding node feature mask M_X is then used as target attributes for classification.

$$X(individual_i) = \{x_{gender}(individual_i), x_{age}(individual_i)\}$$

SportsNetwork Explanations:
$R_{X1}(Equestrian)$ = If Individual is female, then class 'Equestrian' (prob: 85%)
$R_{X1}(FootballPlayer)$ = If Individual is male, then class 'Football Player' (prob: 67%)

4.4 Illustration for SportsNetwork

As described in Section 2.4.1, an explanation for GNN decision-making should come with certain properties for making explanations user-centric through naturalness and sensitivity while upholding fidelity and reference to graph topology. As our white-box model is part of a framework that builds on a sub-symbolic explanation method, we will evaluate how and if the results of our white-box SUBGREX model enhance the explanations given by the sub-symbolic GNNExplainer as exemplified in terms of the listed properties.

One of the desired properties is naturalness, entailing the use of natural language [114]. For generating coherent natural language explanations, standard patterns of discourse employed by humans should be followed. While the decision rules R_E and R_X linearized from decision trees don't reach the level of fluid natural language, their simple pattern of an if-clause, makes it very easy to follow as is shown in the SportsNetwork explanation examples

in Section 4.3. Especially compared to an explainer subgraph as exemplified in Figure 4.3 (right).

By extracting features from the subgraphs depending on domain and classification task and introducing domain terminology, domain knowledge is included, enabling sensitivity. Contrary to standard sub-symbolic graph explainer techniques, our SUBGREX method provides the user with a mapping of topology-based information to domain terminology, e.g. triangle motif to Equestrian triadic cluster/Football triadic cluster. This makes it easier for the user, to understand graph topology and set it into context. Therefore, there is less scope for a misinformed interpretation of the graph structure. By personalizing the rule-based explanations through choosing the attributes to be included, the output can also be customized according to the user's need by the SUBGREX method.

Through extracting graph topology features from the graph structure based on domain knowledge, these features feed into the explanations, whereas in a standard white box model, they would be ignored and crucial information relating structural connections could be missed. Furthermore, through domain knowledge mapping, there is less scope for a misinformed interpretation of the graph structure by the user.

Since this method uses a sub-symbolic explanation element in its framework instead of simple input-output mapping, it should come with a higher explanation fidelity.

Additionally, since our global approach is smoothing over many instance-level explanations, it may also be less susceptible to noise. A stable global explanation may also improve the classification decision understanding.

5 Ontological Explanations

5.1 Problem Definition

As explicated in Section 2.4.3.2, no approach exists that can provide explanations for GNNs that come with naturalness, sensitivity, fidelity, and reference to graph topology. Furthermore, most existing symbolic explanation methods either provide global or local explainability, but not both.

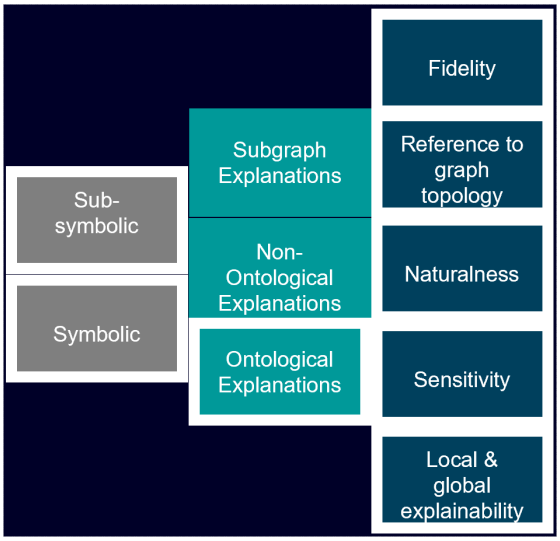


Figure 5.1: Technologies, type and properties of our method

5.2 Concept

With the method OntExplainer, we aim to develop a hybrid method by combining GNNs, sub-symbolic explanation methods and inductive logic learning. This enables human-centric and causal explanations through extracting symbolic explanations from identified decision drivers and enriching them with available background knowledge. These function as local as well as global explanations, as shown in Figure 5.1. With this method, high-accuracy sub-symbolic

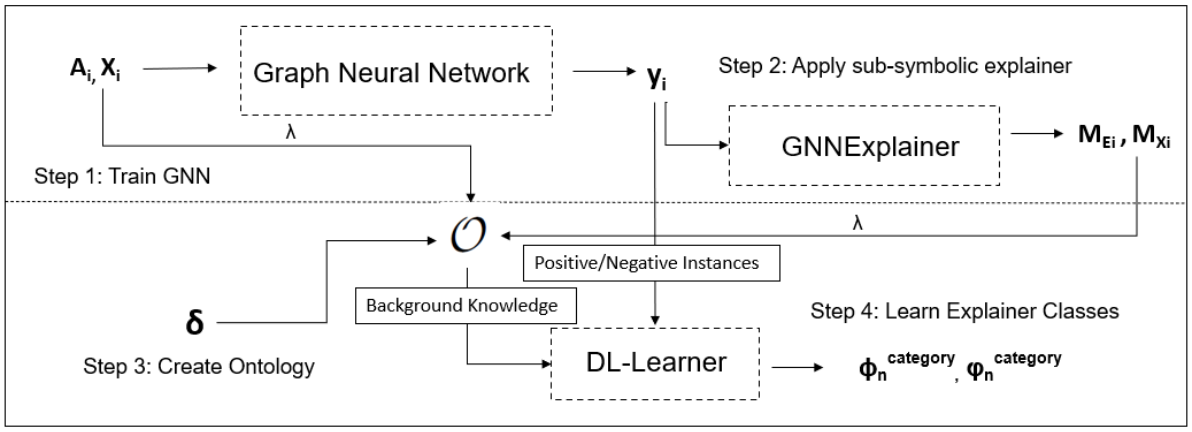


Figure 5.2: Learning Explainer Classes process flow.

predictions come with symbolic-level explanations, and provide an effective solution for the performance vs. explainability trade-off. The developed novel fidelity metric indicates how close an explanation is to the GNN's internal decision-making process. Additionally, the employment of justifications in our method provides a reasoning component that makes use of the background knowledge.

5.3 Proposed Method

We are proposing the hybrid method OntExplainer, within which the coupling of the sub-symbolic explanation method GNNExplainer with the symbolic DL-Learner is used to explain GNN local predictions. Our approach is task-agnostic and can be applied to graph classification, node classification or link prediction. The process flow of learning explainer classes can be seen in Figure 5.2.
For incorporating explicit domain knowledge into our explanation method on the side of symbolic representation, we use ontologies expressed in the W3C OWL 2 standard[1] [57] based on the description logic formalism. The concepts Semantic Web ontology, entailment, inductive logic learning as well as justifications have been introduced in Section 2.2. GNNs and sub-symbolic explanation methods in Section 2.1.2 and Section 2.4.2.

Explainer Class Learning Firstly, a GNN is trained on and applied to training and testing data and subsequently the sub-symbolic explanation method GNNExplainer is applied to all generated predictions, as can be seen in Figure 5.2 (Step 1 and Step 2). The sub-symbolic

[1] `https://www.w3.org/TR/owl2-overview/`

Table 5.1: Example excerpt of $\delta^{SportsNetwork}$.

(1)	`Football` $\sqsubseteq$ `TeamSports`	football is a team sport
(2)	`FootballTriadicMotif` $\sqsubseteq$ `ClusteringTriadicMotif` $\sqsubseteq$ `SocialStructure`	football triadic motifs are clustering triadic motifs, which are social structures
(3)	`FootballPlayer(Individual_2)`	Individual_2 is a football player
(4)	`hasGender(Individual_2, male)`	Individual_2 has gender male
(5)	`fromCountry(Individual_3, Germany)`	Individual_3 is from Germany
(6)	`hasFriend(Individual_2, Individual_1)`	Individual_2 has friend Individual_1

explanation method takes a trained GNN and its prediction(s), and it returns an explanation in the form of an explainer subgraph together with a small subset of node features that are most influential for the prediction, or in other words, importance scores. For their selection, the mutual information between the GNN prediction and the distribution of possible subgraph structures is maximized through optimizing the conditional entropy. The explanation method output is comprised of edge masks $M_{E_i} \in \{0,1\}^{n \times n} \subset \mathbf{A_i}$ and node feature masks $M_{X_i} \in \{0,1\}^{n \times d} \subset \mathbf{X_i}$, which are used as input to our framework. We chose the GNNExplainer [1] for our framework, but our approach will work with any other explainer subgraph generation method, including RERE.

Secondly, to create explainer classes for the GNN decision-making process, DL-Learner is applied for a specific predicted category, with positive and negative examples labelled accordingly through y_i (Step 4). The background knowledge used by the DL-Learner to learn explainer classes is comprised of the adjacency matrices $\mathbf{A_i}$ and node feature matrices $\mathbf{X_i}$, edge masks M_{E_i} and node feature masks M_{X_i} and domain knowledge δ. As the DL-Learner can only process ontologies, the matrices are mapped to an ontology (Step 3) through λ as detailed below in the next step.

Extraction and Mapping Step A set of graphs detailed in their associated matrices $\mathbf{A_i}$ and $\mathbf{X_i}$ [2] are modelled as set of individuals $\{\eta_i\}$. Their edges and node features are extracted from $\mathbf{A_i}'s$ and $\mathbf{X_i}'s$ edge and feature lists and modelled as set of individuals $\{v_j\}$ and $\{\chi_k\}$. If there are structures relating to graph topology common in the respective domain, such as certain

[2]Their size is dependent on the number of layers used by the GNN, to keep the consistency in coupling the sub-symbolic with the symbolic method.

motifs, e.g. a triadic or ring structure, the set of possible structures $\{motif_z\}$ along with their extraction functions $\{\gamma_z(A_i, X_i)\}$ is defined and mapped through mapping function $S : \{motif_z\} \mapsto \{\gamma_z(A_i, X_i)\}$.
If $motif_1$ is contained in (A_i, X_i), extraction function $\gamma_1(A_i, X_i)$ returns all individuals contained in the structure. The found structures are modelled as a set of individuals $\{\psi_g\}$. To assign all individuals their type declarations and roles, a set of roles $\{\rho_v\}$ and type declarations $\{\tau_w\}$ as well as further mapping functions based on domain knowledge δ are needed. Defining these sets and mapping functions has been done as a one-time manual step, with their complexity depending on the domain.

$P : \{\eta_i\} \times (\{v_j\} \cup \{\chi_k\} \cup \{\psi_g\}) \mapsto \{\rho_v\}$, maps a pair of individuals to their role.
$T : \{\eta_i\} \cup \{v_j\} \cup \{\chi_k\} \cup \{\psi_g\} \mapsto \{\tau_w\}$ maps individuals to their types.

All extracted individuals, roles and type declarations are added as axioms to ontology $\mathcal{O}$ through function $AddAxiom(\mathcal{O}, axiom)$ as is shown in Algorithm 2. Therefore, λ is defined as $\lambda(A_i, X_i, T, P, S) \mapsto \mathcal{O}$. Equivalently, λ is carried out for all corresponding sub-symbolic explainer subgraphs with their associated edge masks M_{E_i} and node feature masks M_{X_i}, with the set of explainer graphs modelled as individuals η_sub_i.
Additionally, mapping function μ is defined as bijective function, as is shown in Algorithm 2. This function is needed for the fidelity calculation. Function μ is defined in such a way, that if the input, e.g. σ_1, doesn't map to anything, σ_1 will be returned as output. An example ontology for our toy example SportsNetwork is shown in Table 5.1.

Example 5. (Mapping SportsNetwork Data with SportsNetwork Ontology) *The mapping functions* $S^{SportsNetwork} = \{FootballTriadicMotif : \gamma_{FootballTriadicMotif}$ *,* $EquestrianTriadicmotif : \gamma_{EquestrianTriadicMotif}, ...\}$*,* $R^{SportsNetwork} = \{(\eta_i, v_j) : hasFriend, (\eta_i, \chi_k) : hasAge, ...\}$ *and* $T^{SportsNetwork} = \{\eta_i : Individual, v_j : Link, \chi_k : Country, ...\}$ *are defined based on domain terminology* $\delta^{SportsNetwork}$*. For example, from sports network graph* G_1 *with associated matrices* $\mathbf{X_1}$ *and* $\mathbf{A_1}$*, the edge individuals edge_1_2, edge_1_3, etc., are modelled. For extracting structure* $FootballTriadicmotif$*, which is defined as three football players connected among each other, function* $\gamma_{FootballTriadicmotif}(A_1, X_1)$ *is employed. All accruing axioms are added to the ontology* $\mathcal{O}^{SportsNetwork}$*. Through* μ*, the set of edges forming the identified structure, e.g. { edge_1_2, edge_1_3, edge_1_4 } is mapped to the individual structure_1_1_1.*

Algorithm 2: Graph Structure Extraction λ

Data: Set of graphs with adjacency matrices $\mathbf{A_i}$, feature matrices $\mathbf{X_i}$, mapping functions for type declarations $T(\sigma)$, roles $P(\sigma_1, \sigma_2)$ and structures $S(x)$

Result: $\mathcal{O}$, μ

$\mathcal{O} : \{\}$

foreach *graph in range(i)* **do**

  AddAxiom($\mathcal{O}$, $T(graph)$ (η_{graph}))

  foreach *edge in Edgelist*($\mathbf{A_{graph}}$) **do**

    AddAxiom($\mathcal{O}$, $T(edge)(v_{edge\_graph})$)

    AddAxiom($\mathcal{O}$, $P(graph, edge)(\eta_{graph}, v_{edge\_graph})$)

  end

  foreach *feature in Featurelist*($\mathbf{X_{graph}}$) **do**

    AddAxiom($\mathcal{O}$, $T(feature)(\chi_{feature\_graph})$)

    AddAxiom($\mathcal{O}$, $P(graph, feature)(\eta_{graph}, \chi_{feature\_graph})$)

  end

  foreach *structure in* $\{motif_z\}$ **do**

    if $S(structure)(\mathbf{A_{graph}}, \mathbf{X_{graph}})$ *not* ***None*** **then**

      foreach *number in range(count*($S(structure)(A_{graph}, X_{graph})$)*)* **do**

        AddAxiom($\mathcal{O}$, $T(structure)(\psi_{graph\_structure\_number})$)

        AddAxiom($\mathcal{O}$, $P(graph, structure)(\eta_{graph}, \psi_{graph\_structure\_number})$)

        $\mu : S(structure)(\mathbf{A_{graph}}, \mathbf{X_{graph}}) \mapsto \psi_{graph\_structure\_number}$

      end

    end

  end

end

Definition 4 (Explainer Class Learning) *Given ontology $\mathcal{O}$ and a set of graph individuals $\{\eta_i\} \in \mathcal{O}$* [3] *with their respective classifications $\{y_1, y_2, ...y_i\}$ provided by a GNN for a certain category, we define explainer class learning as inductive logic learning such that $\eta_j | y_j = category \in E^+$ and $\eta_k | y_k \neq category \in E^-$.*

$\mathcal{O}$ provides the background knowledge for inductive logic learning, and the classification decision by the GNN provides the positive and negative examples in order to learn explainer

[3] Mapping sub-symbolic graph representations ($\mathbf{X_i}$, $\mathbf{A_i}$) $\mapsto \mathcal{O}$, resulting in individuals η_i is specified in Section 5.3

classes.

According to the GNN's classifications, positive and negative examples of graphs are distinguished, and explainer classes are learned. The background knowledge is the ontology $\mathcal{O} = \delta \cup \lambda(A_i, X_i, T, P, S) \cup \lambda(M_{E_i}, M_{X_i}, T, P, S)$. We differentiate between two types of explainer classes:

1. **Input-Output Explainer Classes**
   Given Section 2.2 Def. 4, background knowledge $\delta \cup \lambda(A_i, X_i, T, P, S)$, $\eta_i | y_i = category \in E^+$ and $\eta_i | y_i \neq category \in E^-$, a set of Input-Output Explainer Classes $\{\phi_i^{category}\}$ are learned. Input-output explainer classes are candidate explanations, that capture the global behavior of a GNN through investigating what input patterns can lead to a specific class prediction, comparable to the input-output mapping approach in [19].

2. **Importance Explainer Classes**
   Given Section 2.2 Def. 4, background knowledge $\delta \cup \lambda(M_{E_i}, M_{X_i}, T, P, S)$, $\eta\_sub_i | y_i = category \in E^+$ and $\eta\_sub_i | y_i \neq category \in E^-$, a set of Importance Explainer Classes $\{\varphi_m^{category}\}$ are learned. Importance Explainer classes show which edges, nodes, features, and motifs are important for the GNN to predict a certain class. These class expressions represent the inner workings of a GNN, by incorporating the output of the sub-symbolic explainer.

Explainer Class Application for Local Explanations The pool of possible explainer classes for all categories, consisting of $\{\phi_i\}$ and $\{\varphi_i\}$, are used in the application step to generate local explanations through explainer class entailment and justification steps.

Explainer Class Entailment

Given Section 2.2 Def. 1., a set of explainer classes $\{\phi_i^{category}\}$ and $\{\varphi^{category}\}$, ontology $\mathcal{O}$ and individual η_j classified as category, entailments for η_j are generated. By doing so, we check if the learned overall decision-making pattern of the GNN applies to a specific instance. For all available explainer classes, entailments for a specific individual η_j are generated. It is possible, that several entailments hold, just as it is possible that a classification decision of G_j is based on several factors. The set of entailments for η_j is given by $C_{Exp}(\eta_j) = \{\phi \mid \mathcal{O} \models \phi^{category}(\eta_j)\} \cup \{\varphi \mid \mathcal{O} \models \varphi^{category}(\eta_j)\}$.

Definition 5 (Entailment Frequency) *Given an ontology $\mathcal{O}$, explainer class $\phi_i^{category}$ and a set of individuals $\{\eta_i\}$, we define the entailment frequency as the number of entailments for $|\{\eta \in \{\eta_i\} : \mathcal{O} \models \phi_i^{category}(\eta)\}|$ over the number of instances $|\{\eta_i\}|$.*

Table 5.2: Example justification $\mathcal{J}(\mathcal{O}^{SportsNetwork}, \phi_3^m(\eta_1))$.

(1)	η_1 `is Equestrian`
(2)	`Equestrian refersTo HorseRiding`
(3)	`HorseRiding SubClassOf IndividualSports`
(4)	ϕ_3^m `EquivalentTo does some IndividualSports`

The entailment frequency gives insight over the generality or specificity of explainer classes and representing the average frequency with which a certain explainer class is entailed.

Explainer Class Entailment Justification

Given $\mathcal{O}$ and entailment $\mathcal{O} \models \phi_i^{category}(\eta_j)$, justification $\mathcal{J}(\mathcal{O}, \phi_i^{category}(\eta_j))$ is generated. The number of generated axioms gives some insight about the level of domain knowledge employed. As there can be several justifications for an entailment, we limit them to only one. Since a shorter justification tends to be more efficient, the justification with the minimum number of axioms is chosen.

Example 5. (Justification for SportsNetworks Explainer Class)
Table 6.15 shows an example justification for the entailment $\mathcal{O}^{SportsNetwork} \models \phi_3^m(\eta_1)$*, which contributes to a meaningful explanation, as it carries causal information present in expert knowledge about the conclusion.*

Fidelity Calculation

Fidelity is defined as the measure of the accuracy of the student model (DL-Learner) with respect to the teacher model (GNN). High fidelity is therefore fundamental, whenever a student model is to be claimed to offer a good explanation for a teacher model. Without high fidelity, an apparently perfectly good explanation produced by an explainable system is likely not to be an explanation of the underlying sub-symbolic system which it is expected to explain [120]. We calculate Fidelity as follows:

$$Fidelity(\phi_i, \eta_j) = \frac{|\mu^{-1}(ind(\mathcal{J}(\mathcal{O}, \phi_i(\eta_j)))) \cap \eta\_sub_j|}{|\mu^{-1}(ind(\mathcal{J}(\mathcal{O}, \phi_i(\eta_j))))|}, \tag{5.1}$$

where $ind()$ is a function that collects all individuals that are provable instances of a set of

axioms. The denominator equals the count of the set of edges or node features that have to be part of η_i, for the entailment of explainer class ϕ_i to hold. The fidelity metric is defined as the overlap of the sub-symbolic explainer output with the entailed explainer classes, as can be seen in Figure 5.3, which means that the effectiveness of the sub-symbolic explanation method in representing the GNN decision-making is therefore assumed.

Example 6. (Fidelity for Explainer Class *hasMotif some FootballTriadicMotif*) *As the explainer classes are represented through axioms, e.g.* ϕ_2^n `= hasMotif some Football-TriadicMotif`*, we apply the justification mechanism to arrive at the axioms containing the corresponding individual(s) for the specific example* η_1*, such as* η_1 `hasMotif motif_1_2_3` $\in \mathcal{J}(\phi_2^n, \eta_1)$*. Since there might be a multiplicity of individuals, function* $ind(\mathcal{J}(\mathcal{O}, \phi_2(\eta_1))$ *is applied, which collects all individuals that are provable instances of the justification. These individuals are then inversely mapped* (μ^{-1}) *to their corresponding set of individuals, in this example { edge_1_2, edge_1_3, edge_2_3 }. In case there is no corresponding set of individuals, the inverse mapping simply returns the given individual. For the numerator, we count the overlap of the identified set of individuals with the individuals in* η_sub_i*, the subgraph identified by the GNNExplainer.*

Definition 6 (Final Explanation)) *Given the set of entailments, that hold for* η_j*, we define the final explanation* $E(\eta_j)$ *as the set of the respective justifications* $E(\eta_j) = \{\mathcal{J}(\mathcal{O}, C(\eta_j))\} \mid C \in C_{Exp(\eta_j)}$.

Example 7. (SportsNetwork G_1) *In Figure 5.3, the final explanation for the classification for an individual as a football player can be seen, complete with justifications and fidelity score.*

Verbalization

For increased user-friendliness, the class expressions are verbalized further with the state-of-the-art *LD2NL* framework [62]. For example, through the verbalization step, the class expression ϕ_3^m `EquivalentTo inAgeGroup some YoungAdult` is translated to “Football player is in age group Young adult".

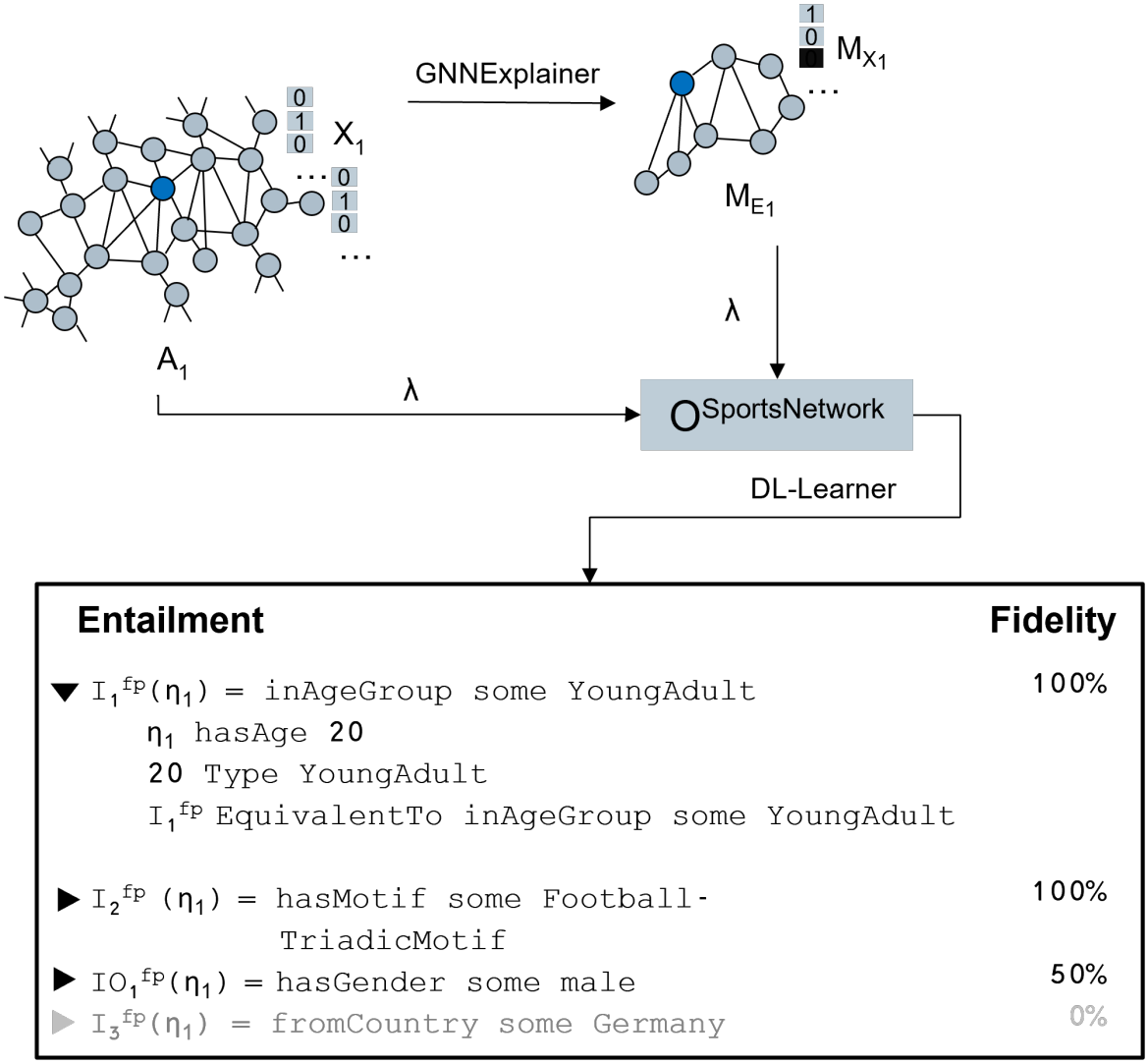


Figure 5.3: Final explanation for individual η_1 (blue node), which has been classified as football player (fp).

5.4 Illustration for SportsNetwork

As outlined in Section 2.4.1, an effective explanation of a GNN's decision-making process must meet four criteria: naturalness, sensitivity, reference to graph topology, and fidelity.

The OntExplainer generates class expressions with a sentence-like structure, providing explanations that are natural and easily understood. The verbalization step further strengthens this by translating the explanations into fluid natural language.

The inclusion of structured domain knowledge in the form of an ontology adds sensitivity to the explanations. Domain knowledge is incorporated, and the possibility of justifications for the explanations adds a reasoning component.

The extraction and mapping step, which is based on available domain knowledge in the ontology, enables important references to graph topology.

The OntExplainer uses a sub-symbolic explanation element in its framework, rather than a simple input-output mapping, leading to higher explanation fidelity. A fidelity measure for this framework ensures that the verbalized explanations are close to the actual decision-making process of the GNN. The measure also quantifies the level of accuracy of the verbalized explainer class.

6 Experiments and Results

The methods presented in this research are evaluated on two distinct datasets. The first dataset, Mutagenicity [44], is a chemical molecule use case commonly used in literature to evaluate the sub-symbolic explanation methods of GNNs. The second dataset describes a cybersecurity use case in an industrial setting and was generated using a demonstrator system that reflects the design of modern industrial systems integrating information technology (IT) and operational technology (OT) elements. To evaluate the accuracy of the RERE method, additional experiments were conducted using synthetic datasets that provide ground truth explanations.

As our identified properties are naturalness, sensitivity, reference to graph topology and fidelity, we want to evaluate these properties, along with the XAI method's ability to provide global or local explanations. A number of accuracy and fidelity metrics exist and can be employed. Naturalness entails the use of natural language explanations. As this is a qualitative notion, it has to be evaluated as such, whether a coherent textual explanation is produced. For evaluating sensitivity, we use proxy metrics (data points which can be used to represent the value of something else) and comparative qualitative evaluations given ground truth concepts. Since the property reference to graph topology is a qualitative notion, measuring it is difficult. Either topological information is considered or not. Therefore, we use accuracy to determine in how far integrating graph topology into the explanation leads to improvements compared to benchmark methods.

As we are using a variety of different methods, including a reinforcement learning algorithm, decision trees and inductive logic learning as XAI, different metrics are necessary to evaluate the accuracy and fidelity of the explanations. This is especially important in order to compare our methods to the state-of-the-art methods in the respective XAI sub-field. The type of metric used to evaluate the respective method and corresponding property is listed in Table 6.1. The used metrics have been introduced in Section 2.4.4 and Section 5.

For subgraph explanations (RERE), the accuracy is calculated with F1-Score, however this is only possible for the additional experiments, as ground-truth explanations are required. Fidelity is calculated according to Equation 2.7 and the Label-Flip rate (Eq. 3.6) is included as a proxy to measure sensitivity.
For non-ontological explanations (SUBGREX), predictive accuracy and fidelity are calculated according to Equations 2.9 and 2.10 respectively. The naturalness of the explanations is evaluated qualitatively, and the sensitivity is evaluated based on comparative evaluation.
For ontological explanations (OntExplainer), the predictive accuracy of the explanations is calculated according to 2.9 for the Mutagenicity use case. Furthermore, we are looking at the entailment frequency, defined in Section 5 Def. 5, to measure the generality or specificity of the explanations. As the cybersecurity use case is qualitatively different from the Mutagenicity use case, the evaluation metrics for the utility of explanations differs somewhat. In the cybersecurity use case, one of the tasks is to reduce the amount of false positives in order to decrease the workload of the cybersecurity analysts. That's why the reduction rate of false positives, i.e., by how much percent is the number of false positives reduced, is taken into account for evaluating our method instead of the predictive accuracy. For fidelity measurement, we are employing a novel fidelity metric, that is based on the sub-symbolic explanation method in the framework. The fidelity metric is defined in Section 5 Equation 5.1. To measure the sensitivity, we are employing comparative evaluation. Furthermore, we can look at the average number of justification axioms to see the structural depth of the domain knowledge provided.

Table 6.1: Properties and Metrics for Evaluation

Method	**Property**	**Metric**
Subgraph Explanations	Fidelity	Fidelity (Equation 2.7)
	Sensitivity	Label-Flip rate (Equation 3.6)
	Reference to graph topology	F1-Score
Non-ontological Explanations	Fidelity	Fidelity (Equation 2.10)
	Naturalness	Qualitative Evaluation
	Sensitivity	Comparative Evaluation Score
	Reference to graph topology	Predictive Accuracy (Equation 2.9)
Ontological Explanations	Fidelity	Fidelity (Equation 5.1)
	Naturalness	Qualitative Evaluation
	Sensitivity	Comparative Evaluation Score
		Number of Justification Axioms
	Reference to graph topology	Predictive Accuracy (Equation 2.9)
		Entailment Frequency (Equation 5)
		False Positive Reduction rate

6.1 Chemical Molecule Use Case

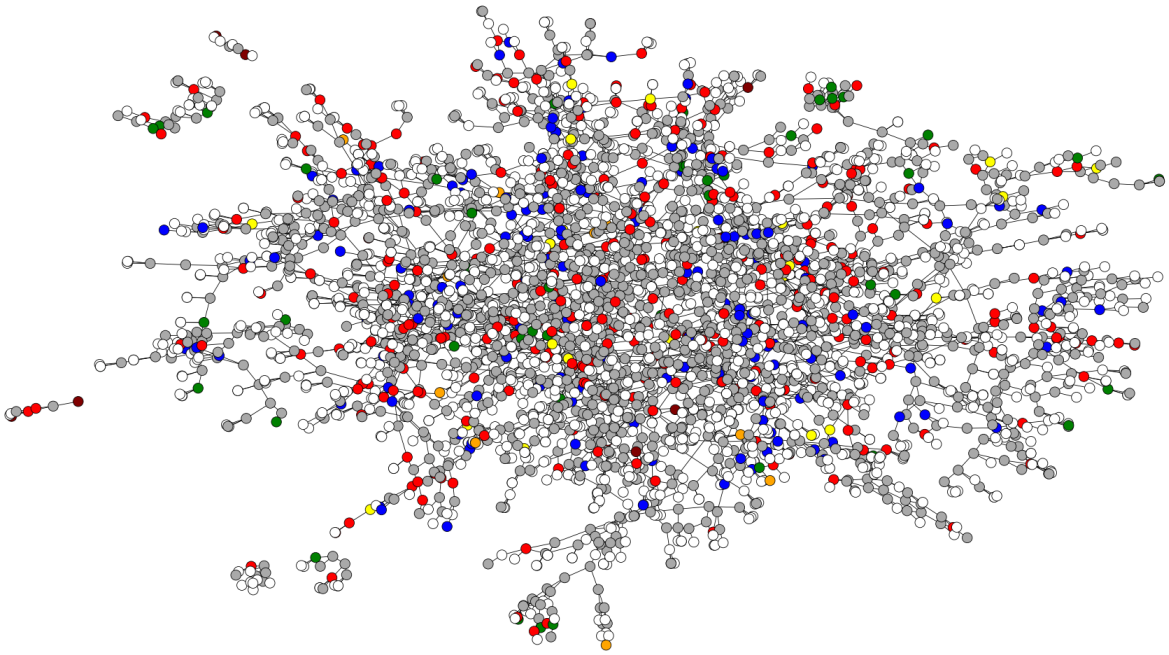


Figure 6.1: Excerpt of the Mutagenicity dataset. Number of edges of total dataset: 133447. Max. graph size: 103, avg. graph size: 29.76, std of graph size: 15.89.

For the first use case, we use data from a chemical domain, as it comes with complex domain knowledge that is universally accepted and can therefore be considered as ground truth when evaluating explanations. Furthermore, it comes with a benchmark dataset that is widely used, for evaluating our method against others. Here, a collection of nitro aromatic compounds or molecules are used with the goal to predict their mutagenicity, which refers to the induction of permanent transmissible changes in the amount or structure of the genetic material of cells or organisms. We use a real-life dataset, Mutagenicity, for graph classification. The dataset contains 4377 molecule graphs labeled according to their mutagenic effect on the gram-negative bacterium Salmonella typhimurium. An excerpt of the graphs can be seen in Figure 6.1. Graph nodes have 14 labels, and each graph is labelled as belonging to 1 of 2 classes [44]. Input graphs are used to represent chemical compounds, where nodes stand for atoms and are labeled by the atom type, which is represented by one-hot encoding. The atoms included are listed in Table 6.2 along with their chemical symbol. They are color-coded according to the widely used Corey–Pauling–Koltun coloring system [155]. Edges between vertices represent bonds between the corresponding atoms.

Table 6.2: Mutagenicity graph node labels with chemical symbol in its corresponding color.

	Chemical Symbol	Element
0	C	Carbon
1	O	Oxygen
2	Cl	Chlorine
3	H	Hydrogen
4	N	Nitrogen
5	F	Fluorine
6	Br	Bromine
7	S	Sulfur
8	P	Phosporus
9	I	Iodine
10	Na	Natrium
11	K	Kalium
12	Li	Lithium
13	Ca	Calcium

It is well-established that molecules containing carbon rings (as depicted in Figure 6.2) and/or Azanide NH_2 groups (as shown in Figure 6.3) or Nitrogen NO_2 have mutagenic properties, according to the study by Debnath et al. [44]. Additionally, it has been found that chemicals with three or more fused rings exhibit higher mutagenic potency compared to those with one or two fused rings. These five observations and their inverse serve as the benchmark for comparative evaluation.

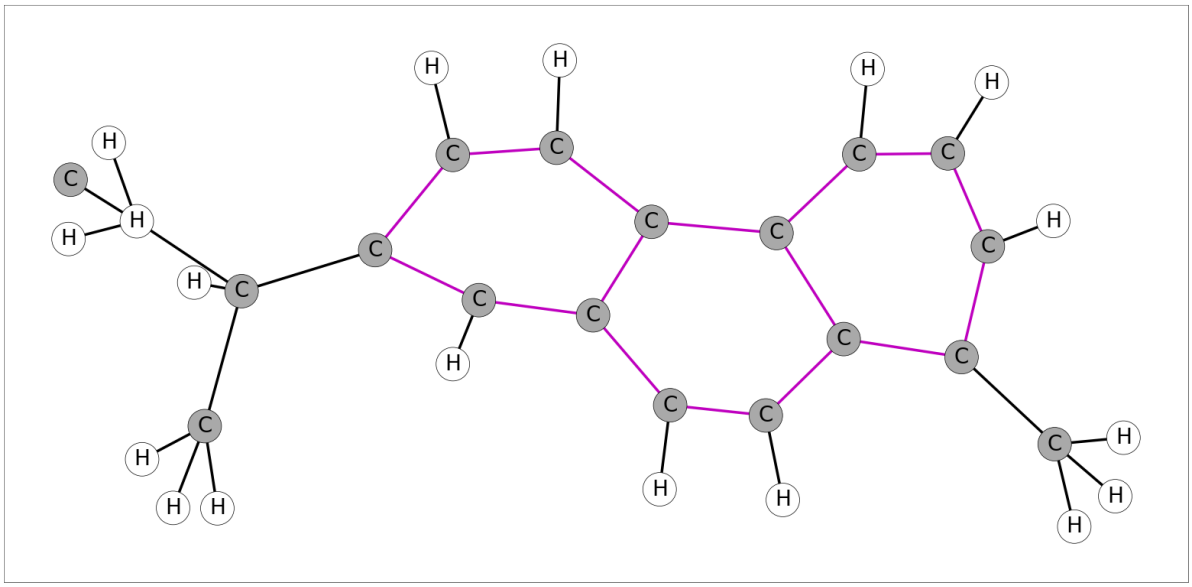


Figure 6.2: Example for molecule graph containing three fused carbon rings C_6. Corresponding edges are colored magenta.

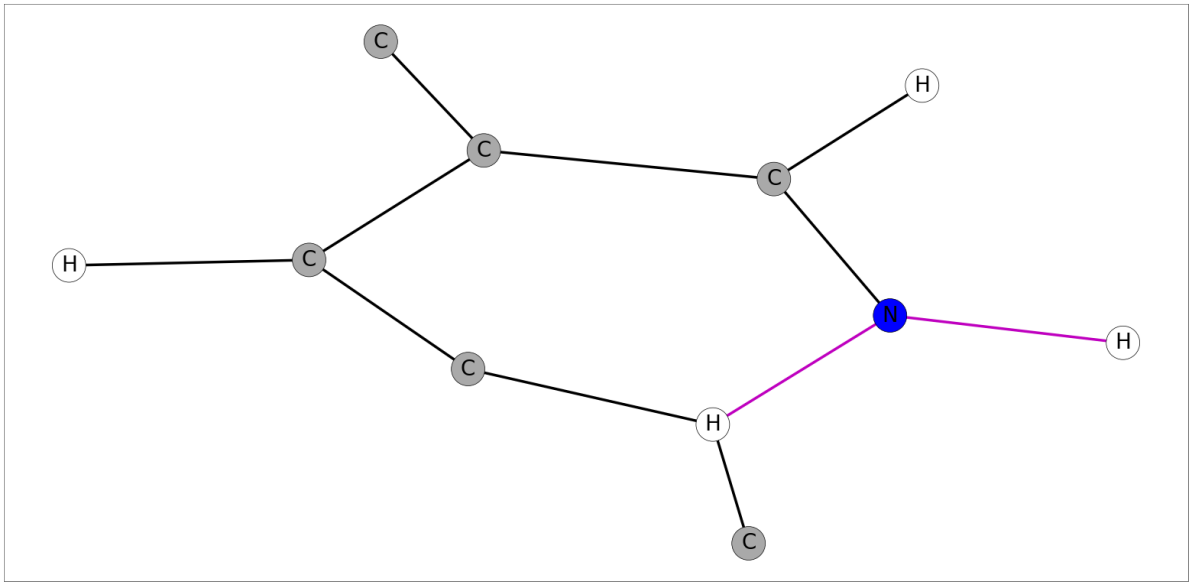


Figure 6.3: Example for molecule graph containing the functional group NH_2 or Azanide. Corresponding edges are colored magenta.

6.1.1 Subgraph Explanations

We follow the experimental settings in GNNExplainer [1]. We train a three-layer GNN and then apply all introduced explainer methods to explain predictions made by the GNN. We use the Adam optimizer to train the GNN. All GNN models are trained for 1000 epochs with learning rate 0.001, reaching accuracy of at least 85% for the graph classification datasets. The train/validation/test split is 80/10/10% for all datasets.
For RERE we train for at least, 30000 episodes. This is efficient, as RERE policy only needs to be trained once and can then be applied to unseen nodes. We use the Adam optimizer and learning rate 0.001. For RERE, we train ten policies based on randomly chosen graphs for each class and average the results. For better comparison of the quantitative results, we include a size constraint to arrive at a similar sparsity as the alternative baseline approaches.

Benchmark Methods We compare our approach with the benchmark methods GNNExplainer [1], PGExplainer [2] and SubgraphX [3]. These methods have been introduced in Section 2.4.2. All three methods are perturbation-based and produce a subgraph as explanation for a GNN model prediction based on importance scores. The results for these methods in terms of fidelity, Label-Flip rate and sparsity can be seen in Table 6.3. Is it shown that GNNExplainer performs best in terms of fidelity.

Table 6.3: Performance evaluation of RERE and alternative baseline explainability approaches for Mutagenicity. Bold indicates best result.

	Fidelity (Eq. 2.7)	Label-Flip rate	Sparsity
GNNExplainer	**0.32**	0.43	0.69
PGExplainer	0.34	0.4	0.65
SubgraphX	0.39	0.49	0.63
RERE	0.33	**0**	0.62

Quantitative Results The quantitative results, including fidelity and Label-Flip rate, are shown in Table 6.3. The metric Label-Flip rate functions as a proxy for measuring the capability for sensitivity. The results show, that for the Mutagenicity dataset, a Label-Flip rate of 0% can be achieved with similarly high levels of fidelity compared to the baseline methods.

Qualitative Results The successive removal of edges by the agent is shown in Figure 6.4, with the respective last step being the final explanation. As demonstrated in these Figures,

mutagenic properties such as certain chemical groups are being correctly identified by RERE. While the baseline methods GNNExplainer and PGExplainer have also shown to identify such motifs, RERE comes with the option to follow the edge removal process successively, where the least informative edges are being removed first. This property can increase the trustworthiness of the explanations, as the user can follow the subgraph reduction procedure more closely, which provides traceability [72].

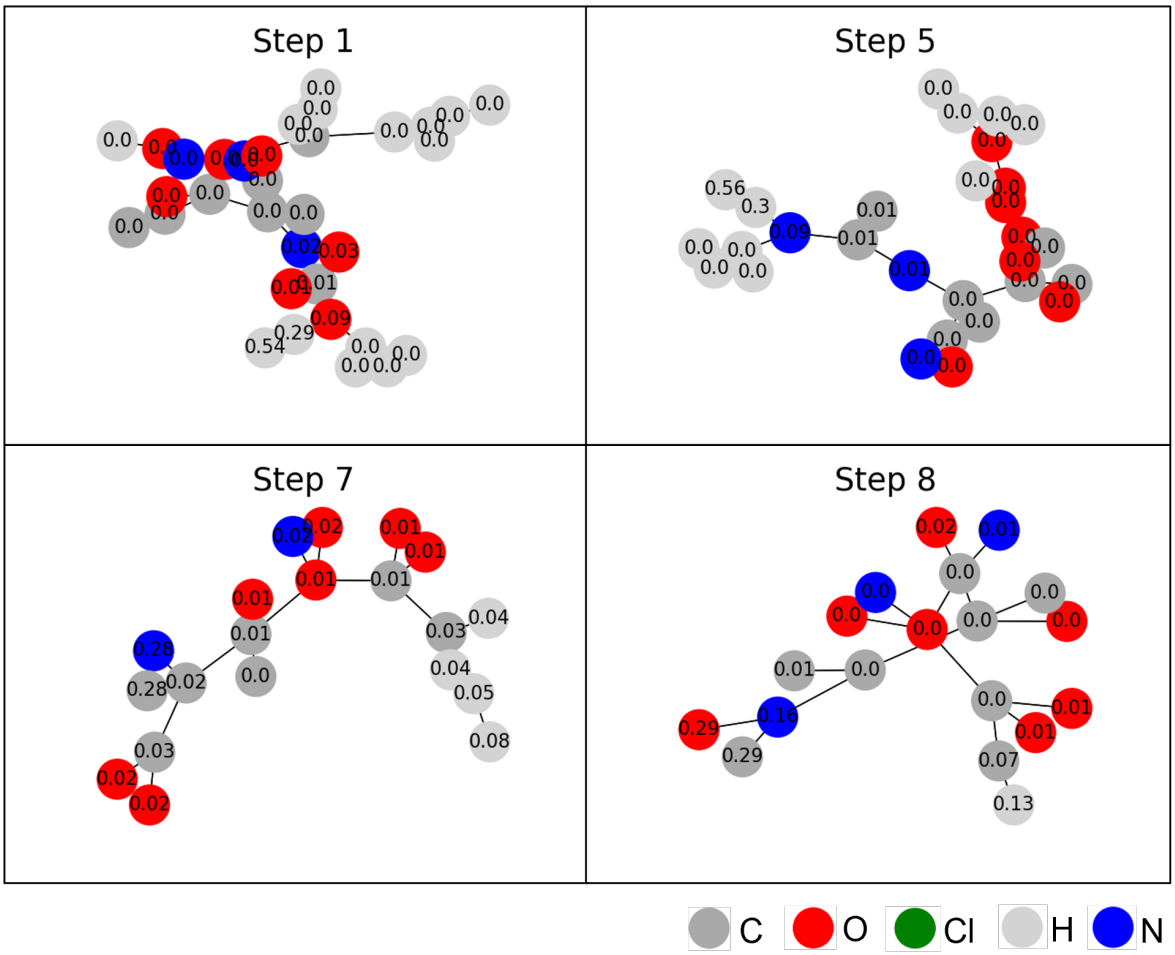


Figure 6.4: Step-wise removal of edges in a molecule based on RERE. The node labels give the action probability of an edge connected to the respective node being removed.

Figure 6.4 shows, that RERE correctly identifies the chemical group NO_2 as mutagenic property, with this chemical group having continuously a low probability of being removed and being part of the final subgraph. Figure 6.5 shows how edges with attached Chlorine,

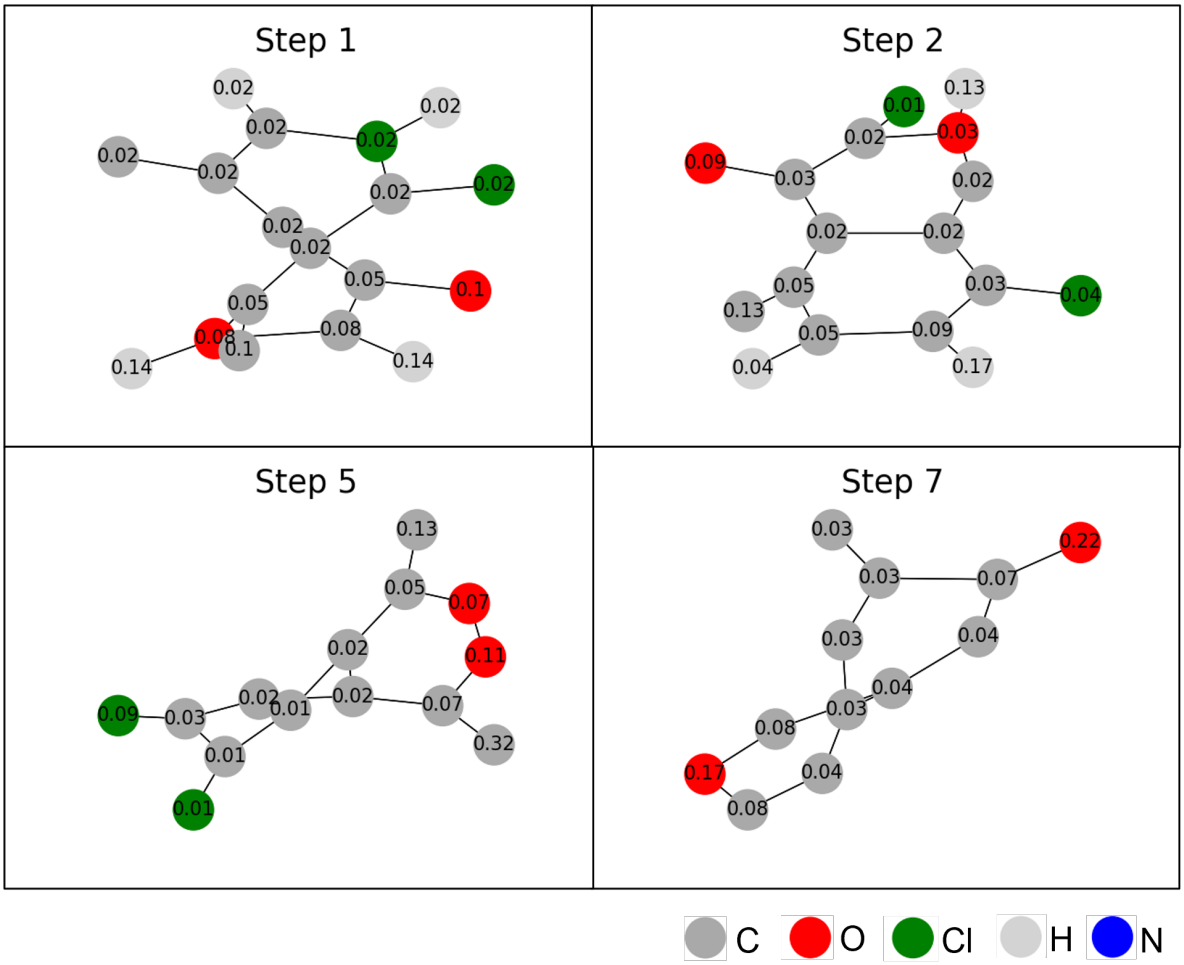


Figure 6.5: Step-wise removal of edges in a molecule based on RERE. The node labels give the action probability of an edge connected to the respective node being removed.

Carbon and Hydrogen nodes are being removed, resulting in a final explanation containing a remaining carbon ring, another chemical group that is known to be mutagenic.

6.1.2 Non-ontological Explanations

We carry out graph classification with a Graph Convolutional Network. We use the Adam optimizer to train both the GNN for 1000 epochs with learning rate 0.001, reaching accuracy of more than 85%. The train/validation/test split is 80/10/10%, that means we have 3469 training graphs, 434 validation graphs and 434 testing graphs. In the next step, we generate the edge mask M_{Ei} and node feature mask M_{Xi} for the molecule graph g_i with the GNNExplainer. The same optimizer and learning rate is used as with the GNN, and we train for 100 epochs. Table 6.4 shows the free form knowledge used as domain knowledge. All motifs are extracted from the edge and node features masks. Decision trees are generated for graph attributes $L(molecule_i)$ with the motifs taken from the domain knowledge as well as for the node attributes $X(molecule_i)$:

$$L(molecule_i) = \{a_{contains\_Carbon\_5-Ring}(molecule_i), a_{contains\_Carbon\_6-Ring}(molecule_i), a_{contains\_Azanide}(molecule_i), a_{contains\_Methyl}(molecule_i), a_{contains\_Nitro}(molecule_i)\}$$

$$X(molecule_i) = \{x_{Carbon}(molecule_i), x_{Oxygen}(molecule_i), ..., x_{Calcium}(molecule_i)\}$$

Decision tree training and testing data follows a random 80% - 20% split.

Table 6.4: Graph Motifs for Mutagenicity.

Motif	Functional Group
Carbon 5-Ring	C_5
Carbon 6-Ring	C_6
Azanide	NH_2
Methyl	CH_3
Nitro	NO_2

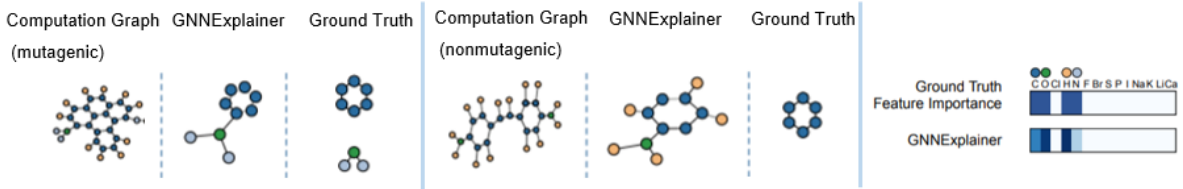


Figure 6.6: GNNExplainer results for Mutagenicity dataset classifications. Figure adapted from Figure 4 and Figure 5 in [1].

GNNExplainer Output for Mutagenicity Dataset Classifications
The GNNExplainer is applied to identify the most influential parts of the respective graph for the classification decision. Figure 6.6 shows the original graph, its edge mask M_E as identified by the GNNExplainer and the ground truth for a mutagenic (left) and nonmutagenic (middle) molecule as well as the identified node feature mask M_X (right). It can be seen that the identified important graph motifs and node features align with some ground truth mutagenic properties, as given by [44]. These include ring structures and the node features C, O, N and H. However, the fact that these results represent a carbon ring as well as the chemical group NO_2 (Nitrogen dioxide) is left to the user for interpretation.

Benchmark Methods In order to test the effectiveness of our method for being a global explainer method, we compare it to a decision tree explainer method without taking into account any edge mask M_E and node feature masks M_X. This means, the features are extracted from the original input graphs as opposed to the explainer subgraphs, similarly to [18], which we call Standard Decision Tree Explainer (SDTE). The free form domain knowledge included in this method is identical for better comparability. The quantitative comparison to a method analogously to [17], as illustrated in Section 4 Figure 4.4, which doesn't include any domain knowledge (SDTE w/o KB) results in decision rules that only include node features, comparably to SDTE R_X. All insight gained from SDTE D_X or SUBGREX D_X respectively is missing.

Quantitative Results We report the accuracy as defined Def. 2.9 to show to which extent our explanations can function as a global explanation. Accuracy indicates the ability of the decision tree to correctly classify a node or feature as influential. The accuracy for SUBGREX D_E is at 82% and therefore 27 percentage point higher than the benchmark method SDTE D_E, as shown in Table 6.19 Similarly, the accuracy for SUBGREX D_X lies at 73% compared to 54% for SDTE D_X, which is little higher than random. In terms of fidelity, defined as the percentage of test-set examples on which the classification made by a decision tree agrees with its GNN counterpart (Eq. 2.10), the results are analogous. In order to quantitatively evaluate sensitivity, we look at the percentage of coverage regarding the ground truth explanations provided by humans. The results in Table 6.6 show that without including domain knowledge (SDTE w/o KB), none of the ground truth statements are covered, while for SDTE 30% are. With SUBGREX 80% of ground truth statements are covered by the generated explanation.

Table 6.5: Accuracy, Recall, and Fidelity for D_E and D_X for Mutagenicity

Model	Accuracy (Def. 2.9)	Recall	Precision	Fidelity (Def. 2.10)
SUBGREX D_E	0.82	0.64	1	0.82
SDTE D_E	0.55	0.07	0.42	0.48
SUBGREX D_X	0.73	0.71	0.74	0.74
SDTE D_X	0.54	0.2	0.51	0.53
SDTE w/o KB D_X	0.54	0.2	0.51	0.53

Table 6.6: Comparative Evaluation for Mutagenicity: Percentage of ground-truth statements covered.

SUBGREX	SDTE	SDTE w/o KB
0.8	0.3	0

Qualitative Results In the following, linearized rule-based explanations R_E and $R_X$from the respective decision trees for Mutagen and Non-Mutagen classifications are listed:

Table 6.7: $R_{Ei}(Mutagen)$

Rule	Probability
if (Methyl > 0.5)	100.0%
if (Methyl <= 0.5) and (Nitro > 0.5)	100.0%
if (Methyl > 0.5) and (Nitro > 0.5) and (Azanide > 0.5)	100.0%
if (Methyl > 0.5) and (Nitro <= 0.5) and (CarbonSixRing > 0.5)	66.67%

The generated decision rules $R_{Ei}(Mutagen)$ (Table 6.7) give the user a comprehensible idea of the network motifs that are influential in the GNN's decision-making. While an if-then rule doesn't provide fully fluent natural language, the explanation is in textual form and provides enhanced naturalness compared to an explainer subgraph. Rules including Methyl, Nitro, Azanide and Carbon 6-Ring as decision drivers are generated. The latter three motifs are known to have mutagenic properties. Analogously, $R_{Ei}(Non - Mutagen)$ (Table 6.9) shows that the non-existence of these motifs and motif Carbon 5-Ring additionally in the explainer subgraph, are the decision rules for Non-Mutagenicity.
The node feature rules $R_{Xi}(Mutagen)$ (Table 6.8) are somewhat more convoluted and also come with lower probabilities. While there are some clear rules indicating the existence of

Table 6.8: $R_{Xi}(Mutagen)$

Rule	Probability
if (Hydrogen > 0.5) and (Bromine <= 0.5) and (Oxygen <= 0.5) and (Carbon > 0.5) and (Chlorine <= 0.5) and (Phosphorus <= 0.5) and (Fluorine <= 0.5) and (Nitrogen <= 0.5) and (Potassium <= 0.5) and (Sulfur <= 0.5)	87.07%
if (Hydrogen > 0.5) and (Bromine <= 0.5) and (Nitrogen <= 0.5) and (Phosphorus <= 0.5)	74.6%
if (Carbon > 0.5)	74.4%
if (Carbon > 0.5) and (Bromine <= 0.5) and (Nitrogen <= 0.5)	69.49%
if (Carbon > 0.5) and (Bromine <= 0.5) and (Chlorine <= 0.5)	68.34%
if (Hydrogen > 0.5) and (Bromine <= 0.5)	66.15%

Table 6.9: $R_{Ei}(Non - Mutagen)$

Rule	Probability
if (Methyl <= 0.5) and (Nitro <= 0.5) and (Azanide <= 0.5)	71.43%
if (Methyl <= 0.5) and (CarbonSixRing <= 0.5) and (Azanide <= 0.5) and (CarbonFiveRing <= 0.5) and (Nitro > 0.5)	62.5%
if (Methyl <= 0.5) and (Nitro <= 0.5)	57.49%

Carbon as a decision driver for mutagenicity, Nitrogen's existence is not identified as such. However, Hydrogen and Nitrogen's non-existence, which make up the known mutagen Nitro, are included in $R_{Xi}(Non - Mutagen)$ (Table 6.10). The difference between R_{Ei} and R_{Xi} shows, that the inclusion of domain knowledge and reference to graph topology in the shape of motifs leads to much more user-friendly and more accurate explanations.

Table 6.10: $R_{Xi}(Non - Mutagen)$

Rule	Probability
if (Hydrogen <= 0.5) and (Nitrogen > 0.5) and (Carbon <= 0.5)	100.0%
if (Hydrogen <= 0.5) and (Nitrogen <= 0.5) and (Oxygen > 0.5) and (Carbon <= 0.5)	100.0%
if (Hydrogen <= 0.5) and (Nitrogen > 0.5)	91.51%
if (Carbon <= 0.5) and (Nitrogen <= 0.5) and (Oxygen <= 0.5) and Hydrogen <= 0.5) and (Sulfur <= 0.5) and (Chlorine <= 0.5) and (Bromine <= 0.5) and (Phosphorus <= 0.5)	74.61%
if (Hydrogen <= 0.5)	63.51%

6.1.3 Ontological Explanations

We carry out graph classification with a Graph Convolutional Network. We use the Adam optimizer to train both the GNN for 1000 epochs with learning rate 0.001, reaching accuracy of more than 85%. The train/validation/test split is 80/10/10%, that means we have 3469 training graphs, 434 validation graphs and 434 testing graphs. We used a subset of 530 molecule graphs as training data to learn explainer classes, and 800 molecule graphs as testing data. All molecule graphs come with adjacency matrices A_i^{Mutag} and feature matrices X_i^{Mutag} and their corresponding GNNExplainer importance masks ($M_{E_i}^{Mutag}$ and $X_{E_i}^{Mutag}$), equally split between mutagenic and nonmutagenic classifications. The DL-Learner can create arbitrarily many class expressions, functioning as explainer classes, which are ordered by predictive accuracy (number of correctly classified examples divided by the number of all examples). We are taking a cut-off point of $> 50\%$ predictive accuracy, as an explainer class with less than 50% predictive accuracy, would not represent a pattern for mutagenic classification decisions, but rather the opposite, and v.v. for nonmutagenic classification decisions.

As we are combining GNNs and ontologies, graph data has to be available as triples as well as background knowledge. The domain knowledge used in our approach is given by the Mutagenesis ontology $\mathcal{O}^{Mutag}$ [1], which is exemplified in Table 6.11.

Table 6.11: Example excerpt of $\mathcal{O}^{Mutag}$.

(1)	`Carbon` $\sqsubseteq$ `Atom`	carbons are atoms
(2)	`Hetero_aromatic_5_ring` $\sqsubseteq$ `Ring_Size_5` $\sqsubseteq$ `RingStructure`	hetero-aromatic rings of size 5 are rings of size 5, which are ring structures
(3)	`Nitrogen(feature_100_5)`	feature_100_5 is a nitrogen
(4)	`Compound(graph_100)`	graph_100 is a compound
(5)	`hasAtom(graph_100, feature_100_5)`	graph_100 has atom feature_100_5

Benchmark Methods Classifications and their accompanying explanations can be generated using only a symbolic classifier like DL-Learner, which performs inductive logic learning as outlined in Section 2.2. Hence, we evaluate our method against a purely symbolic approach

[1] https://github.com/SmartDataAnalytics/DL-Learner/tree/develop/examples/mutagenesis

to determine if and what benefits our hybrid approach offers.

Additionally, we aim to examine the advantages of incorporating a sub-symbolic explainer into our framework, rather than relying on the input-output matching method for explaining GNN predictions, as seen in previous works such as [19] and [22].

Quantitative Results The entailment frequency gives us insight over the generality or specificity of explainer classes. As can be seen in Table 6.12 (Avg. Entailment Rate), there is a wide range of entailment rates. Some explainer classes, e.g. ϕ_4^m = `hasAtom some Carbon` always apply, while others are quite rare, such as ϕ_4^n = `hasAtom some Phosphorus`, that comes with only a 4% entailment rate. As expected, we have an overall lower entailment rate for nonmutagenic explainer classes, as there are also less distinct factors indicating nonmutagenicity [44]. Most nonmutagenic classifications come with about 3 entailments, while mutagenic classifications come with more than 5 entailments on average. This is due to a lower generality of the explainer classes, which implies that such an explainer class only applies to specific instances. This is also confirmed by the lower average predictive accuracy of the DL-Learner results for nonmutagenic (57%) as opposed to mutagenic (63%) explainer classes, as can be seen in Table 6.12 (Avg. Pred. Acc). The predictive accuracy of the DL-Learner is defined as the number of correctly classified examples divided by the number of all examples [156].

Table 6.12: Input-output(ϕ_i^m for mutagenic classifications, ϕ_i^n for nonmutagenic) and importance explainer classes (φ_i) with avg. pred. accuracy (DL-Learner), entailment rate and fidelity with their respective standard deviations (SD).

Explainer Class Type	Number (n)	Avg. Pred. Acc. (SD)	Avg. Entailment Rate (SD)	Avg. Fidelity (SD)
ϕ_i^m	1,...,10	0.56 (0.04)	0.64 (0.3)	0.88 (0.12)
ϕ_i^n	1,...,5	0.59 (0.03)	0.09 (0.04)	0.82 (0.12)
φ_i^m	1,...,4	0.77 (0.06)	0.86 (0.15)	0.99 (0.01)
φ_i^n	1,...,7	0.56 (0.01)	0.41 (0.25)	0.81 (0.05)

Fidelity gives the user a measure of reliability of the explanation, with the average fidelity ranging from 64% for ϕ_5^n = `hasStructure some Carbon_5_ring` to 100% for e.g. φ_2^m = `hasAtom some Hydrogen`. While an explainer class with an average fidelity of 64% might still give the user some insight, its explanatory value cannot be considered as reliable as for an explainer

Table 6.13: Average fidelity for true positives and false positives.

	TP^m	FP^m	TP^n	FP^n
Number of instances	371	29	374	26
Average fidelity	0.96	0.66	0.82	0.44

class with a higher fidelity. An explainer class, that has a low generality, meaning it is rarely applied to explain a classification, can nonetheless come with a high fidelity such as ϕ_4^n (100%). This suggests that also low generality explainer classes can be valuable for specific instances.

We can observe a positive correlation of 88% between the average fidelity and predictive accuracy for $\{\varphi_i\}$ and of 50% between the average fidelity and $\{\varphi_i\}$ and $\{\phi_i\}$ respectively, signalizing the effectiveness of representing the sub-symbolic decision-making process with the DL-Learner. As the predictive accuracy of the output given by the DL-Learner is the metric on which we base our choice of explainer classes included in the pool, the correlation with the fidelity indicates that this approach leads to reliable explanations.

Explainability of sub-symbolic methods is desirable not only to justify actions taken based on the predictions made by the system, but also to identify false predictions. Therefore, it is also important to evaluate our method based on its ability to not generate explanation for wrong predictions. Table 6.13 shows the difference in entailments for the correctly classified (true positives TP) and incorrectly classified graphs (false positives FP). We can see, that the average fidelity for entailments is 30 percentage points lower for mutagenic FP than mutagenic TP, and 38 percentage points for nonmutagenic FP. While this might not be sufficient to clearly identify a wrong classification, it indicates the validity of the fidelity metric, as it is significantly lower for explainer classes applied to incorrect classification.

In terms of comparative evaluation, 100% of all ground truth statements are covered by the explanations by our method, as is shown in Table 6.14. However, if there was no domain knowledge (including reference to graph topology) used, only 47% of ground truth statements would be covered.

Justification Axioms: Through justifications, we provide causality for explanations, based on domain knowledge. The ontology $\mathcal{O}^{Mutag}$ utilized has little structural depth, as can be seen

Table 6.14: Comparative Evaluation for Mutagenicity: Percentage of ground-truth statements covered.

OntExplainer	OntExplainer without $\mathcal{O}^{Mutag}$
1	0.47

in the example excerpt in Table 6.10. Nonetheless, there is a minimum of 3 axioms for all entailments in any justification. For 20% of explainer classes, 4 axiom justifications and for 8% of explainer classes, 5 axiom justifications are generated. This means, that for all explanations generated, the explanations carry some causal information about the conclusion, supported by expert knowledge.

Comparison of our OntExplainer with DL-Learner Explanations: When comparing this purely symbolic approach with our hybrid method, we find that using only the DL-Learner comes with significantly lower prediction accuracy and also explanatory value. The predictive accuracy of the GNN using the same subset of training data is 85%, so, considerably above the DL-Learner result, as shown below. When applying the DL-Learner to carry out classifications, we are restricted to only one classifier. The DL-Learner might generate several possible classifiers, but only one can be chosen to carry out the classification. This would usually be the one with the highest predictive accuracy. This means, even if we allow more complex class expressions, we only have one explanation for the target predicate mutagenic:

`hasStructure some Nitrogen_dioxide or hasThreeOrMoreFusedRings value true` (pred. acc.: 65.76%)

Comparison of our OntExplainer with Input-Output Explanations: We can see, that for some explainer classes such as $\phi_3^m = \varphi_3^m$ `= hasAtom some Nitrogen`, we have overlap of the importance explainer classes with the input-output explainer classes. However, the importance explainer classes come with a significantly higher predictive accuracy of 77% as can be seen in Table 6.12, indicating their significance for the classification decision. For the nonmutagenic classifications, explainer class φ_2^n `= hasStructure some Carbon_6_ring`, which is equivalent with the ground truth as shown in Figure 6.6, would not have been included in ϕ_i^n. Here, we can clearly see the added benefit of generating explainer classes from the GNNExplainer as opposed to only observing the input-output behaviour of a GNN. The main benefit of including such a sub-symbolic explainer, however, is the provision of the fidelity

metric. Without such a metric, there is no means to quantify the reliability of the explanation.

Qualitative Results The generated pool of explainer classes provides a total of 14 explainer classes for mutagenic and 12 explainer classes for nonmutagenic classifications. All the comprehensible explanation for mutagenic classification decisions that can be identified and interpreted from the GNNExplainer output, as outlined above, have been learnt by the DL-Learner. These include

ϕ_2^m = `hasStructure some Carbon_6_ring`,
ϕ_7^m = `hasStructure some Nitrogen_dioxide`,
φ_1^m = `hasAtom some Carbon`,
φ_2^m = `hasAtom some Hydrogen`,
φ_3^m = `hasAtom some Nitrogen`,
φ_4^m = `hasAtom some Oxygen`,

along with several others, which have not been identified by the GNNExplainer.
The explainer class ϕ_6^m = `hasStructure some Phenanthrene` is a compelling example for the effectiveness of our hybrid approach, as Phenanthrene is a strong indicator for mutagenic potency [44], but isn't identifiable in the GNNExplainer output. This shows that our hybrid method can identify and verbalize decision-making processes of the GNN, which a comprehensible sub-symbolic explainer system, whose output might not be easily understood and interpreted by a user, is missing. An example for a justification of an entailment is given in Table 6.15, along with its verbalization.

Table 6.15: Example justification $\mathcal{J}(\mathcal{O}^{Mutag}, \phi_8^m(\eta_1))$.

(1)	η_1 `hasStructure structure_1_1_1`	Graph 1 has Structure 1_1_1
(2)	`structure_1_1_1 Type Hetero_aromatic_5_ring`	1_1_1 is of type Hetero Aromatic Five Ring
(3)	`Hetero_aromatic_5_ring SubClassOf Ring_size_5`	Hetero Aromatic Five Ring is a subclass of Ring of size 5
(4)	ϕ_8^m `EquivalentTo hasStructure some Ring_size_5`	Explainer Class 8 is equivalent to has Structure Ring of size 5

6.1.4 Discussion

All three methods have been applied to the Mutagenicity dataset, integrating different levels of domain knowledge and providing different types and levels of explainability.
The results for RERE applied to the Mutagenicity dataset show that our methods achieve similar or higher fidelity compared to sub-symbolic benchmark methods. Sensitivity is evaluated with the Label-Flip rate. With a resulting 0% Label-Flip rate, it can be shown that such a constraint can be integrated without harming the quality of the explanations. As the Mutagenicity dataset doesn't come with ground truth explanations, an accuracy score cannot be calculated here. However, common motifs known to indicate mutagenicity, such as carbon rings, can be found in the final explanation subgraphs.
This approach is sensible for users who desire local interpretability and have a basic knowledge of graphs to comprehend the explanation effectively. The option of incorporating domain knowledge constraints is not overly complex and can be utilized whenever deemed appropriate.

The results for SUBGREX show, that using decision trees to extract if-then rules from explainer subgraphs enhances the naturalness of explanations. Integrating this sub-symbolic element furthermore leads to rules with significantly higher accuracy and comparative evaluation scores. This is due to the filtering effect, where only things that have been considered as influential by a sub-symbolic explainer are taken into consideration for the global rules. Furthermore, it can be shown that by including domain knowledge, even unstructured free-form domain knowledge, sensitivity, and reference to graph topology can be provided. The utility of these properties is explicated in the substantially increased accuracy score as well as the comparative evaluation score. Without any domain knowledge and therefore reference to graph topology, the explanations would be very limited. While the fidelity metric employed here is only an indication, the integration of sub-symbolic importance scores shows promising results.
This approach is ideal for users that want global rules to understand the overall decision-making pattern of a GNN, without prior knowledge of graph theory. While it comes with some added complexity and matching efforts, integrating both a sub-symbolic explainer element and domain knowledge with reference to graph topology is indispensable for this use case to arrive at meaningful and accurate explanations.

The OntExplainer results show, that naturalness of explanations is given through verbal-

ized complete axioms containing the class expressions. The level of sensitivity provided is dependent on the domain knowledge available in the used ontology. With integrating the Mutagenesis ontology it has been shown that for all explanations generated, the explanations carry some causal information about the conclusion, supported by expert knowledge. Without including domain knowledge and also reference to graph topology, only 47% of ground truth statements would be covered by the explanations as opposed to 100%, showing their utility. Furthermore, the difference in fidelity between false positives and true positives, shows the validity of the fidelity metric used. As a positive correlation between average fidelity and predictive accuracy could be found and integrating sub-symbolic explainer subgraphs leads to higher predictive accuracy, the validity of this hybrid approach is shown.

This approach is suitable for users who desire both global and local explanations of the decision-making process of a GNN, without prior knowledge of graph theory. The key requirement is access to a domain knowledge ontology, which provides a reference point for the explanation. However, if such structured domain knowledge is not available, the explanation may not be presented in the user's domain language, and will lack reference to graph topology and reasoning. Integrating a sub-symbolic element can increase complexity, as the results from the sub-symbolic explainer must be integrated into the ontology. Nevertheless, if one wants to validate the explanation, the additional effort may be worthwhile, depending on the level of risk associated with the use case.

6.2 Cybersecurity Use Case

The second use case is in a real-life industrial setting which has the focus of the evaluation on the efficacy of the method in an industrial context. The continuous increase in cyberattacks has given rise to a growing demand for modern intrusion detection approaches that leverage ML to detect both simple security risks as well as sophisticated cyberattacks [157]. These approaches identify patterns in data and highlight anomalies corresponding to attacks. Such detection tasks are particularly poised to benefit from the ability to automatically analyze and learn from vast quantities of data. There are many relevant examples of the application of deep learning and similar techniques for intrusion detection systems (IDS) [158] based on anomaly detection methods able to find deviations from a previously learned baseline [159]. However, their drawbacks include alarm flooding problems [160] and a lack of explainability, e.g., for why certain network traffic is flagged as anomalous by the IDS [161].

This is not just relevant for the defense of conventional IT systems, but also in the context of modern operational technology (OT) systems, such as those used in factories and other industrial automation settings. While these industrial control systems used to be exclusively deterministic in their operation, modern Industry 4.0 automation settings are characterized by a convergence of IT and OT infrastructure [162]. This convergence comes with increasingly complex activity patterns and network topologies that make extensive use of autonomous systems and components such as AI-enabled software applications [159]. While this has the potential to substantially improve the flexibility, reliability and efficiency of industrial systems and consumer-oriented manufacturing, it also poses new cybersecurity challenges [163] and demands a high degree of domain-specific knowledge from analysts assessing potential integrity issues or indications of security compromises.

Therefore, there is a clear need for XAI that enables analysts to understand how the system is reaching its conclusions and allow them to interact with it in a collaborative manner [157]. One of the biggest drivers for successful adoption of ML models is how well human users can understand and trust their functionality. Trust in automation can break down quickly and be hard to reestablish, in cases of conspicuous systems faults or when there are alarm flooding problems [119]. The benefits afforded by explanations only fully come to bear when these are human-centered and the users are able to understand and interact with them. This is especially crucial in the cybersecurity domain, where experts require far more information from the model than a simple binary output for their analysis [159].

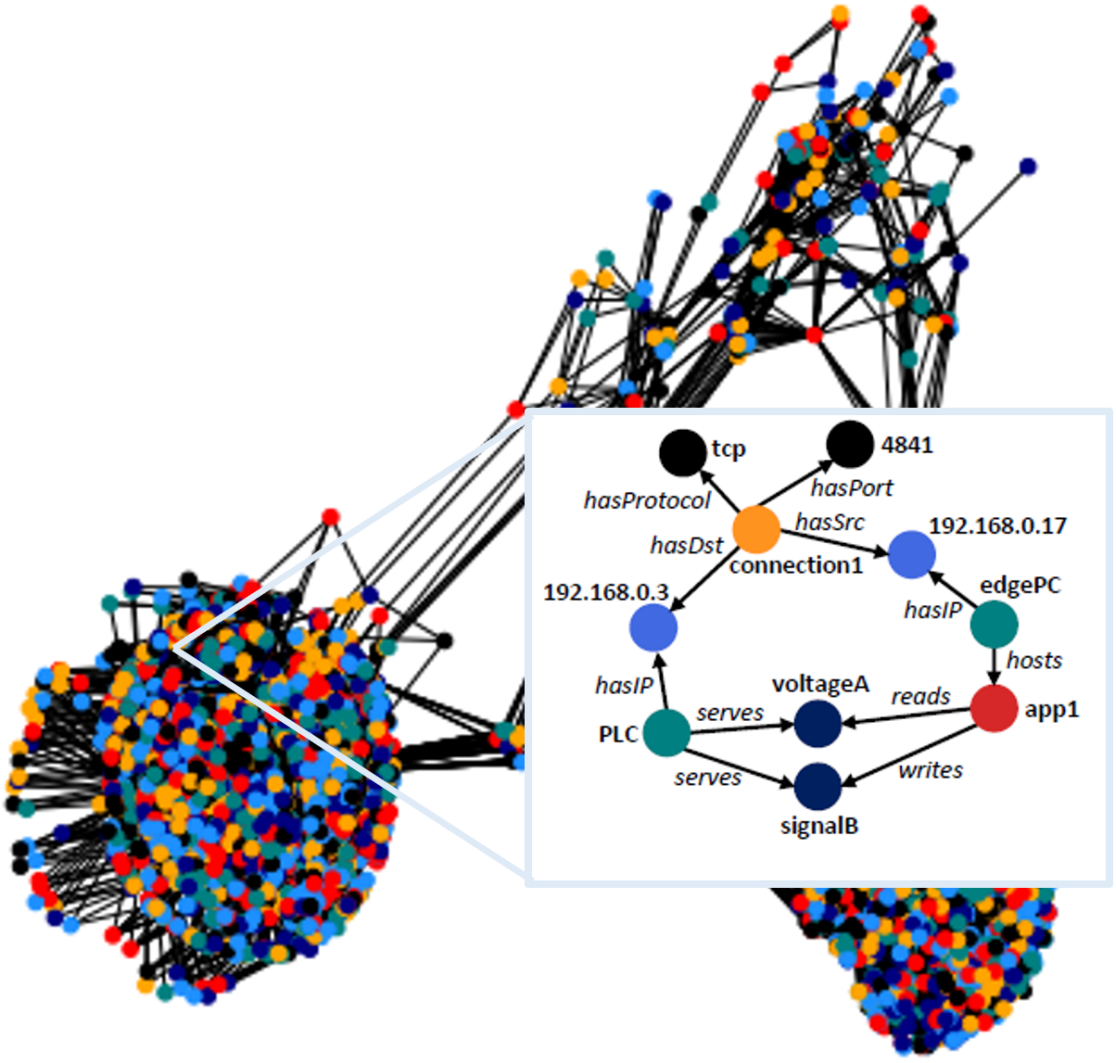


Figure 6.7: The hardware demonstrator of an OT system is represented here as a multi-relational graph.

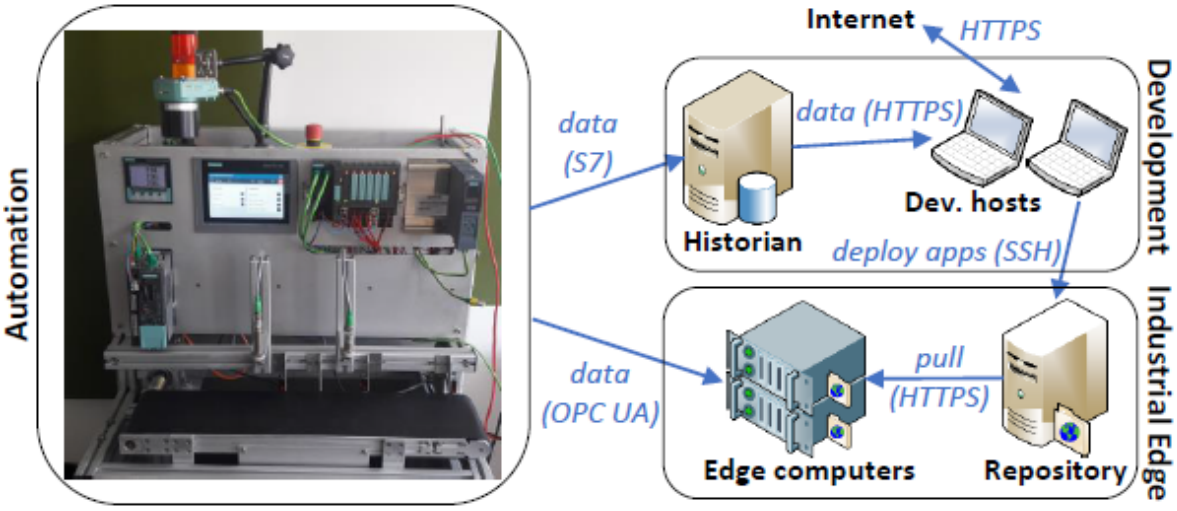


Figure 6.8: Industrial automation demonstrator consisting of an automation part, a development environment and an industrial edge. Image adapted from [167] (©2021 IEEE).

XAI in the Cybersecurity Domain XAI in cybersecurity has been gaining more attention in the last two years. The authors of [157] and [164] focus on what constitutes a good explanation for the user, including what information requirements a human needs for decision-making. In [159], the authors propose a taxonomy for XAI methods and introduce a black box attack for analyzing the consistency, correctness, and confidence properties of gradient-based XAI methods. In [165] a rule extraction process is proposed that allows to explain the causes of cyber threats, while in [21] a system is proposed that combines experts' written rules and dynamic knowledge generated by a decision tree algorithm. Similarly, in [166] a concrete proposal for an Explainable Intrusion Detection System including a neural network combined with decision trees is presented, together with an empirical evaluation of its prototype implementation.

Experimental Setup The demonstrator system for e.g., measuring the height of objects for quality control amongst other capabilities, is described in Figure 6.8, following the design of modern industrial systems integrating IT and OT elements. The automation side is equipped with a programmable logic controller (PLC) connected to peripherals via an industrial network. These include a drive subsystem controlling the motion of a conveyor belt, an industrial camera, a human-machine interface, and a distributed I/O subsystem with modules interfacing with various sensors for object positioning and other measurements

(Fig. 6.8, left). The PLC exposes values reported by these sensors as well as information about the state of the system by means of an OPC-UA server[2]. The variables exposed by the server are consumed on the IT part of the demonstrator by applications hosted on edge computing servers (Fig. 6.8, bottom right), i.e., computing infrastructure directly located at the factory floor which is typically devoted to data driven tasks that require close integration with the automation system and short response times, such as real-time system monitoring, fault prediction or production optimization. Industrial edge applications have dynamic life cycles, and this is captured in the prototype by recreating a development environment (Fig. 6.8, top right). This cycle starts with development hosts consuming potentially high volumes of data from a historian, a database that constantly stores process data from the automation system. Finally, edge computing hosts fetch application updates periodically. To make the behavior more realistic, development hosts occasionally access the internet with low traffic volumes. The environment is fully virtualized and performs these activities in an autonomous manner, with an option to manually induce different types of anomalous behaviors in order to test the response of our IDS system. A knowledge graph is built out of the running prototype by integrating three main sources of knowledge: information about the automation system, observations at the network level (e.g., connections between hosts), and observations at the application level (e.g., data access events). A sizeable portion of the information is related to the automation system, which is extracted from engineering tools in the Automation ML format and ingested into the graph using a readily available ontology [169]. Information about application activity is obtained from the OPC-UA server logs, including session information, i.e., which variables are accessed and in which way. Finally, all network traffic is passed through the security monitoring tool Zeek [170], which produces a stream of observed connections that are ingested using a simple custom data model.

Anomaly Detection Initially, a baseline is captured with the system under normal operating conditions, and the collected data is used to train the link prediction algorithm in an unsupervised manner. Thereafter, in order to qualitatively evaluate its predictions, we trigger a set of actions which result in events not observed during normal operation, but which would be assigned a wide range of severity levels by a human expert upon detailed analysis of the available contextual information. Suspicious behavior is novel behavior given the baseline definition. These scenarios are defined following the ATT&CK framework for Industrial

[2]OPC-UA server is a machine to machine communication protocol for industrial automation [168]

Control Systems[3], i.e., a standardized collection of cyberattack patterns, to guarantee a high degree of realism. The employed scenarios are listed in Table 6.16. One example is sniffing, where an app accesses data variables completely unrelated to those usually accessed (Scenario 1.2), e.g., not served by the PLC, or with a different data type, such as strings instead of numeric data types like int, real, etc. This could be an event where system information is extracted, like serial numbers of devices or firmware versions, which is useful information for discovering back doors and vulnerabilities of the system.

Graph Neural Networks for Anomaly Detection We apply a GNN to detect unexpected activity in industrial automation systems, which are control systems, such as computers or robots, and information technologies for handling different processes and machines in a factory. Use of machine learning methods is possible on knowledge graphs, typically by means of so-called graph embeddings: vector representations of graph entities which are more suitable for processing via neural networks and similar methods than their original symbolic representations. In [159], relational learning on knowledge graphs is applied to security monitoring and intrusion detection by mapping the events in an industrial automation system to links in a knowledge graph. This way, the anomaly detection task can be rephrased as a link prediction task in the knowledge graph representation of the modeled system. In particular, [159] report that the collective learning properties of graph embedding methods allow the resulting models to generalize beyond individual observations, benefiting from the context provided by a rich set of entity and relationship types. Here, we follow the same paradigm of phrasing the security monitoring task as a knowledge graph link prediction task, with a 2-step process. (1) Learning a baseline of normal behavior by training a GNN on a graph built from a training dataset. (2) Applying the GNN in a link prediction setting to rank the likelihood of triple statements resulting from events observed at test time and determine whether they represent an anomaly. A GNN usually consists of graph convolution layers which extract local substructure features for individual nodes and a graph aggregation layer which aggregates node-level features into a graph-level feature vector [90].

[3]https://collaborate.mitre.org/attackics/index. php/Main Page

Application activity
1.1 App changes the way it accesses some variables (e.g. writes instead of reads).
1.2 App accesses variables completely unrelated to those accessed usually.
Network activity (HTTPS)
2.1 A local address not corresponding to a dev. host (e.g. an edge server) accesses the historian.
2.2 A local address not corresponding to a dev. host (e.g. an edge server) accesses a public IP address.
2.3 A high-volume HTTP access is made to a public IP address (high volumes only from historian in baseline).
Network activity (SSH)
3.1 The historian host (not a dev. host, but on the same network) accesses the app repository via SSH.
3.2 A dev. host accesses an edge server via SSH, but during training, no edge servers received SSH connections.
3.3 SSH connection between two edge servers. Usually, no edge servers started or received SSH connections.
Credential use
4.1 Access to OPC-UA server from an IP address that corresponds to a development host.
Network Scan
5.1 Connection which does not match any source-destination pair usually observed.
5.2 Attempt to connect to an IP which is not assigned to any host.

Table 6.16: Attack Scenarios

6.2.1 Subgraph Explanations

We train a two-layer GraphSAGE model and then apply all introduced explainer methods to explain predictions made by the GNN. Training is done for 1000 epochs with learning rate 0.01, reaching a ROC AUC score of at least 81%. The train/validation/test split is 80/10/10%. For RERE we train for at least 10000 episodes. We use the Adam optimizer and learning rate 0.001. We train ten policies based on randomly chosen graphs for each class and average the results. For better comparison of the quantitative results, we include a size constraint to arrive at a similar sparsity as the alternative baseline approaches.

Benchmark Methods Again, we compare our RERE approach with the benchmark methods GNNExplainer [1], PGExplainer [2] and SubgraphX [3]. The results for these methods in terms of fidelity, Label-Flip rate and sparsity can be seen in Table 6.17.

Table 6.17: Performance evaluation of RERE and alternative baseline explainability approaches for cybersecurity. Bold indicates best result.

	Fidelity (Eq. 2.7)	Label-Flip rate	Sparsity
GNNExplainer	0.11	0.38	0.96
PGExplainer	0.21	0.39	0.94
SubgraphX	0.14	0.41	0.95
RERE	**0.08**	**0**	0.97

Quantitative Results The quantitative results, including Fidelity and Label-Flip are shown in Table 6.17. The results show, that for the cybersecurity dataset, that in terms of Label-Flip rate and fidelity, RERE outperforms the baseline methods with similar levels of sparsity.

Qualitative Results The successive removal of edges by the agent is shown in Figure 6.9, with the respective last step being the final explainer subgraph. The figure shows a link predicted as anomalous by the GNN between the nodes opc#app3 and zeek#192.168.0.80, a credential use attack scenario. It can be seen that RERE correctly identifies the connection to a development host as influential in the decision-making of the GNN, as the node new#devNetwork remains in the final explainer subgraph. The corresponding attack scenario

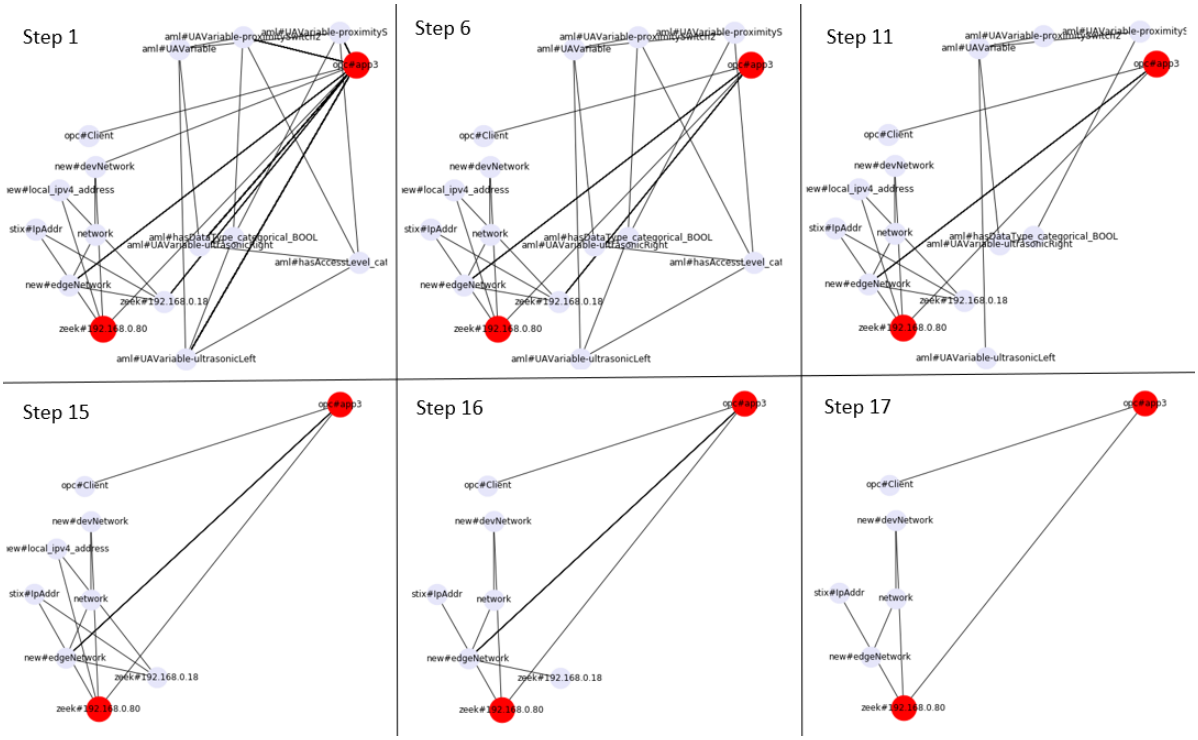


Figure 6.9: Step-wise removal of edges in a for link classified as suspicious between the node opc#app3 and the node zeek#192.168.0.80. Step 17 shows the final explainer subgraph.

can be found in Table 6.16: "Access to OPC-UA server from an IP address that corresponds to a development host".

6.2.2 Non-ontological Explanations

We carry out link prediction with a 2-layer GraphSAGE model. We use the Adam optimizer to train the GNN. Training is done for 1000 epochs with learning rate 0.01, reaching a ROC AUC score of at least 81%. The train/validation/test split is 80/10/10%. We generate the edge masks M_{Ei} and node feature masks M_{Xi} for the cybersecurity graph with the GNNExplainer. Decision trees are learned for links that have been classified as an anomaly and are therefore considered suspicious, which are a total of 105 links over 5 categories. As a 2-layer GNN is used for the prediction, the 2-hop neighborhoods of the two nodes between the link predicted as anomaly is used as $graph_i$. The same optimizer and learning rate is used as with the GNN, and we train for 100 epochs.
Table 6.18 shows an excerpt of the free form knowledge used as domain knowledge. As there are a total of 4317 different node features, including all of them would lead to convoluted and incomprehensible decision rules. Therefore, the extracted node features are mapped to one of 37 entity types. Decision trees are generated for edge attributes $L(molecule_i)$ with the available edge features as well as for the node attributes $X(molecule_i)$ taken from the domain knowledge:

$$L(graph_i) = \{a_{initiated\_from}(graph_i), a_{read}(graph_i), ..., a_{write}(graph_i)\}$$

$$X(graph_i) = \{x_{Destination\_port}(graph_i), x_{Network}(network_i), ..., x_{PLCtag}(graph_i)\}$$

Decision tree training and testing data follows a random 80% - 20% split.

Benchmark Methods Like in Section 6.1.2 we compare SUBGREX to SDTE (features being extracted from the original input graphs as opposed to the explainer subgraph) and SDTE w/o KB.

Quantitative Evaluation The accuracy for SUBGREX D_E is at 90% and therefore 66 percentage point higher than the benchmark method SDTE D_E and 82 percentage points higher than SDTE w/o KB, as shown in Table 6.19. Similarly, the accuracy for SUBGREX D_X lies at 92% compared to 26% for SDTE D_X. In terms of fidelity, the results are analogous. In order to quantitatively evaluate sensitivity, we look at the percentage of coverage regarding the ground truth explanations provided by humans. The ground truth statements are taken from the introduced attack scenarios, shown in Table 6.16, and encompass a total of 11 statements.

Table 6.18: Entity Type Dictionary Excerpt

Entity type	Data type	Example values
Destination_port (id.resp_p)	addr	*id.resp_p_categorical*_5355.0, *id.resp_p_categorical*_49155.0, *id.resp_p_categorical*_443.0
Network	string	*edgeNetwork*, *automationNetwork*, *devNetwork*
PLCtag	string	$PLCtag - currentL1$, $PLCtag - activePowerL2$, $PLCtag - driveEnabled$
UAVariable	string	$UAVariable - ExecuteReaderParameters.objectId$, $UAVariable - ReaderData.hwConnect.Static.buf199$
Client	string	$app4$, $app5$, $app1$
Responder-payload-bytes (resp_bytes)	count	$resp\_bytes - logCategorical\_10to2$, $resp\_bytes - logCategorical\_10to6$, $resp\_bytes - logCategorical\_10to3$
IpAddress	addr	192.168.0.3, 192.168.0.18
Application-protocol (service)	string	*service_categorical_ssh*, *service_categorical_s7comm*, *service_categorical_ssl*
Transport_layer-protocol	transport-proto	*proto_categorical_tcp*, *proto_categorical_udp*, *proto_literal*
Duration_literal	string	*duration_kmean_cluster*_305.6, *duration_kmean_cluster*_0.39, *duration_kmean_cluster*_140.35

Table 6.19: Accuracy, Recall, and Fidelity for D_E and D_X for Mutagenicity

Model	Accuracy (Def. 2.9)	Recall	Precision	Fidelity (Def. 2.10)
SUBGREX D_E	0.9	1	1	0.88
SDTE D_E	0.24	0.28	0.1	0.21
SDTE w/o KB D_E	0.08	0.05	0.04	0.08
SUBGREX D_X	0.92	0.98	1	0.93
SDTE D_X	0.26	0.2	0.33	0.23
SDTE w/o KB D_X	0.24	0.28	0.1	0.21

The results in Table 6.20 show that without including domain knowledge (SDTE w/o KB) 9% of the ground truth statements are covered, while for SDTE 27% are. With SUBGREX 73% of ground truth statements are covered by the explanations.

Table 6.20: Comparative Evaluation for Cybersecurity

SUBGREX	SDTE	SDTE w/o KB
0.73	0.27	0.09

Qualitative Evaluation Linearized rule-based explanations R_E and R_X from the respective decision trees for as anomalous predicted links are listed in Table 6.21 and 6.22.
The generated decision rules $R_{X_i}(Anomaly)$ and $R_{Ei}(Anomaly)$ are sorted according to their category as given in Table 6.16. For category application activity, the decision rules clearly find the attack scenario pattern introduced, where the way variables are accessed (e.g., read instead of write) by a client is the anomaly. For network activity HTTPS, the decision rules fall short in finding the introduced patterns of local addresses not corresponding to a dev. host accessing the historian or a public IP address. Instead, both $R_{X_i}(Anomaly)$ and $R_{Ei}(Anomaly)$ find the duration to be the decision driver, which likely relates to the high-volume HTTP access made (Scenario 2.3). For network activity SSH, the decision rules $R_{X_i}(Anomaly)$ show an anomaly regarding the destination IP, which relates to Scenario 3.2. The decision rules

Table 6.21: $R_{X_i}(Anomaly)$

Rule	Probability
Application activity	
if (read > 0.5) and (initiated_from <= 0.5)	100.0%
if (read > 0.5) and (initiated_from > 0.5) and (write > 0.5)	100.0%
HTTPS	
if (duration > 0.5)	100.0%
SSH	
if (Destination_IP <= 0.5)	100.0%
Credential Use	
if (Application_protocol (> 0.5)	100.0%
if (Application_protocol (<= 0.5) and (initiated_from > 0.5) then class	100.0%
if (Originator_payload_bytes > 0.5)	100.0%
if (Originator_payload_bytes <= 0.5) and (initiated_from > 0.5)	100.0%
if (Client > 0.5)	100.0%
Network Scan	
if (Application_protocol > 0.5) and (Destination_port > 0.5) and (Originator_payload_bytes > 0.5) and (initiated_from <= 0.5)	100.0%
if (Application_protocol > 0.5) and (Destination_port > 0.5) and (Originator_payload_bytes > 0.5) and (initiated_from > 0.5) and (Connection_state <= 0.5) and (Transport_layer_protocol <= 0.5)	66.67%

Table 6.22: $R_{Ei}(Anomaly)$

Rule	Probability
Application activity	
if (IPAddress <= 0.5)	100.0%
if (Client > 0.5)	100.0%
HTTPS	
if (Duration_literal > 0.5)	100.0%
SSH	
if (Client > 0.5) and (Destination_Port> 0.5)	100.0%
Credential use	
if (Destination_Port > 0.5)	100.0%
if (IP_Address> 0.5)	100.0%
Network Scan	
if (Client <= 0.5)	100.0%
if (IP <= 0.5)	100.0%

$R_{E_i}(Anomaly)$ covers Scenario 3.1, finding that accessing apps from an unusual destination port being the decision driver for finding an anomaly. Also, for the credential use category, the destination port being unusual is identified. For category network scan, an unusual destination port is identified among others, covering Scenario 5.1.

6.2.3 Ontological Explanations

We carry out link prediction with a 2-layer GraphSAGE model. We use the Adam optimizer to train the GNN. Training is done for 1000 epochs with learning rate 0.01, reaching a ROC AUC score of at least 81%. The train/validation/test split is 80/10/10%. The training data encompasses more than 37k data events, with 4347 nodes and 37 edge types. The testing data contains 1367 data events, evenly distributed across all 5 attack scenarios. The links predicted come with adjacency matrices A_i and feature matrices X_i and their corresponding GNNExplainer importance masks (M_{E_i} and X_{E_i}) based on their 2-hop receptive field. The DL-Learner can create arbitrarily many class expressions, functioning as explainer classes, which are ordered by predictive accuracy (number of correctly classified examples divided by the number of all examples). We are taking a cut-off point of > 50% predictive accuracy, as an explainer class with less than 50% predictive accuracy, wouldn't represent a pattern for anomalous links.

Ontology Creation: Analysis of security incidents typically requires consideration of multiple data sources, some of which are often exchangeable between organizations. In order to facilitate this, common schemas and data representation formats, such as STIX [171], have been introduced that enable organizations to exchange threat intelligence in a consistent way. More recently, these have evolved into fully-fledged ontologies enabling inference and reasoning [172]. These ontologies model a wide range of cybersecurity-relevant knowledge such as product information, known vulnerabilities and attack patterns, and can additionally be linked to domain-specific knowledge, e.g. coming from industrial automation systems [173]. Once constructed, these knowledge graphs find a wide range of applications, e.g., intrusion and threat detection [174], [175]. Construction of high-quality knowledge graphs is a challenging task, especially when it requires extraction of information from unstructured textual or heterogeneous data. To use the DL-Learner, we first have to transform the knowledge graph into an ontology. Here, a class hierarchy that follows the separation of the industrial automation system into three domains following STIX [171] is adapted:

- **Automation part**: Summarizes the engineering design of the manufacturing prototype. Further separated into InternalStructure, containing InternalElements, ExternalInterfaces, Attributes and InternalLinks as subclasses as well as UAVariables, containing PLCtags and Attributes as subclasses.
- **Edge part**: Contains app initialization events, data events and the applications.
- **Network part**: Contains network connections and their properties as subclasses, as well

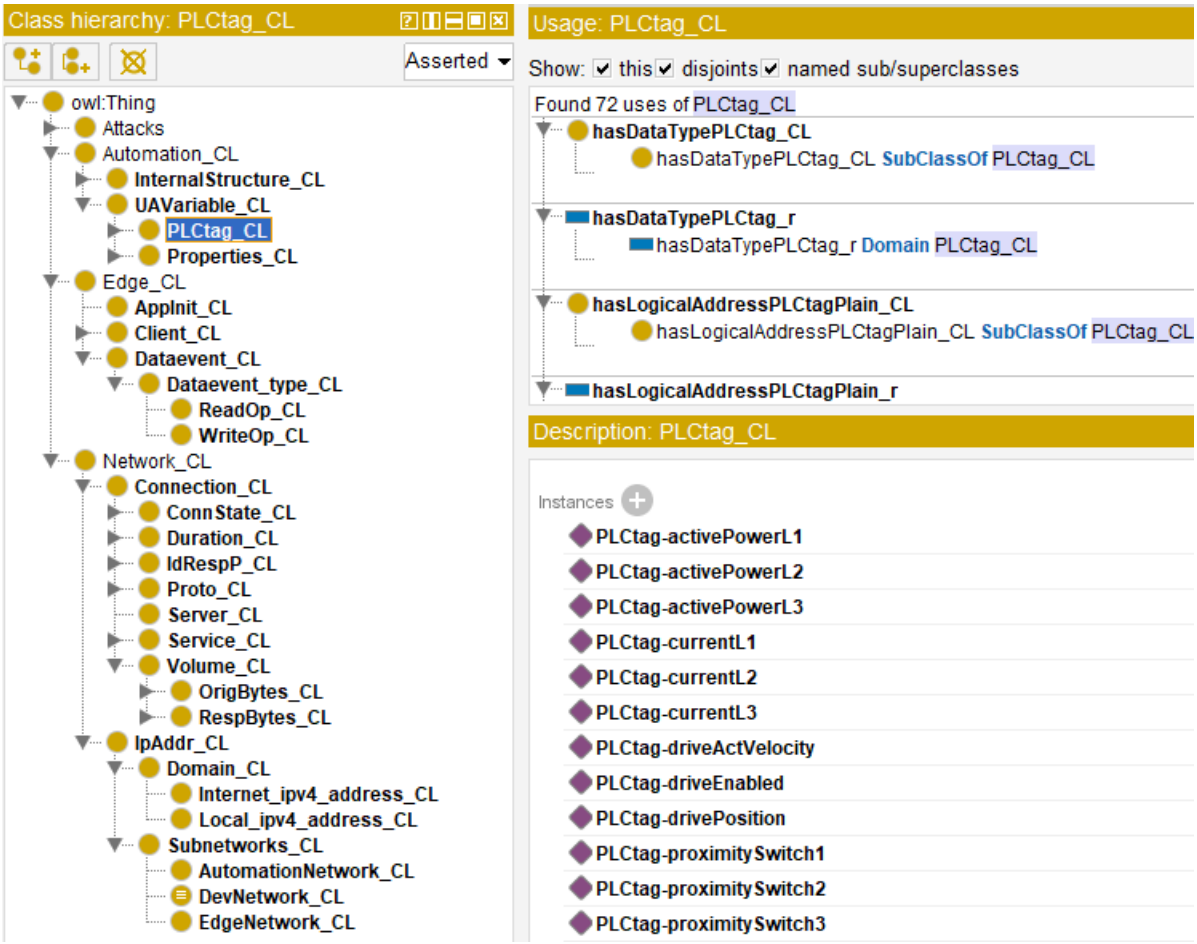


Figure 6.10: Screenshot of Cybersecurity Ontology in Protégé.

as IPs and their subdomains with local and global and automation, development and edge networks as subclasses.

For relations, domain and range are provided. To enable DL-Learner to use properties (like network connection properties), these have to be promoted to classes, e.g., every possible instance for network volumes is its own class. We do the same for applications, InternalElements and ExternalInterfaces, network types (DevNetwork etc.) and attributes. Figure 6.10 shows a screenshot of the described ontology in the Protégé application [176].

Explanation Generation: Tools specially designed to address tasks related to the detection of malicious behavior typically tend to focus more on events or observations that are considered to be unexpected or unusual [157]. Similarly to that, detected unexpected events serve as the trigger for explanations in our approach, where the type of abnormality is identified. For testing, events of a certain device (e.g., activity of a certain app, or network connections between two certain IPs) are compared based on severity classes against each other, such as the suspicious class against the baseline. We do this by using model predictions, based on a ranked list with suspiciousness scores. Thus, one can compare, e.g., the top entry with compatible baseline events. This is how positive and negative examples for the DL-Learner are generated.
Sub-symbolic explanations in the shape of subgraphs are generated by the GNNExplainer, as can be seen in Figure 6.11. Here, the flagged data event $App4\ read\ UAVariable-HardwareRevision$ can be seen. The greyed out nodes and edges are considered not influential in the flagging by the GNN, as opposed to the remaining subgraph. This information is then used to calculate the fidelity score of the entailed explainer class for each flagged event. As can be seen in the figure, the entailed explainer class is part of the identified subgraph, with the data event client being $App4$ and the $UAVariable-HardwareRevision$ (orange nodes) having datatype string (blue node). Therefore, the explanation generated by the DL-Learner shows fidelity with respect to the GNN.

Benchmark Methods Classifications and their accompanying explanations can be generated using only a symbolic classifier like DL-Learner, which performs inductive logic learning as outlined in Section 2.2. While link prediction isn't possible with inductive logic learning, we can frame the problem as a classification task by using anomalous data events as positive examples and non-anomalous data events as negative examples. Hence, we evaluate our method against a purely symbolic approach to determine if and what benefits our hybrid

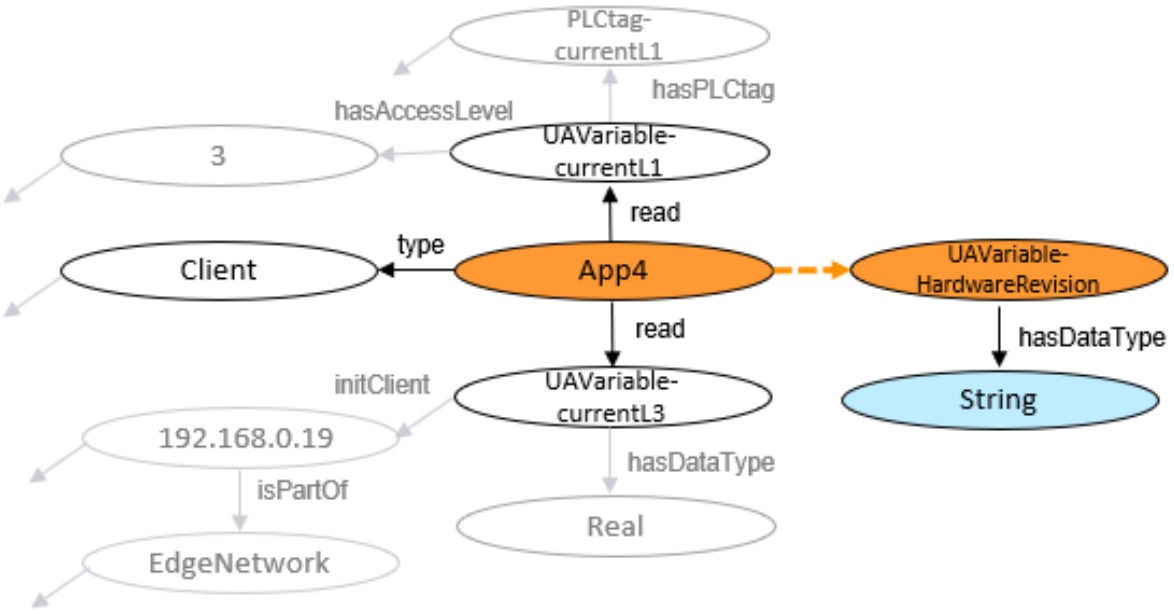


Figure 6.11: Explanation subgraph for flagging data event “App4 Read UAVariable-HardwareRevision" as suspicious generated by GNNExplainer. They greyed out nodes and edges are considered non-important and are not part of the explanation.

Explainer Class	Verbalization
(*dataevent_client* **some** *App*5) **and** (*dataevent_operation* **some** WriteOp) ↦ Attack Scenario 1.1	Sniffing is something whose data event client is an App5 and whose data event operation is a Write Operation
(*dataevent_client* **some** *App*4) **and** (*dataevent_variable* **some** (*hasDataType* **some** (not (*DataType_REAL*)))) ↦ Attack Scenario 1.2	Sniffing is something whose data event client is App4 and whose data event variable is something that has data type something that is not data type real
Network_CL **and** (*service* **some** *service_ssh*) ↦ Attack Scenario 3.2	Security Breach is something whose service is SSH
(*id.resp_p* **some** *id.resp_p*_22.0) **and** (*orig_bytes* **some** *orig_bytes_log*_10*to*2) ↦ Attack Scenario 3.3	Security Breach is something whose port is an 22 and whose origin volume is 10 *to* 2
(*init_client* **some** *App*3) **and** (*init_server* **some** (*isPartOf* **some** *DevNetwork*)) ↦ Attack Scenario 4.1	Credential Use is something whose initial client is an App3 and whose initial server is something that is part of a development network
id.resp_p **some** *id.resp_p*_22.0 ↦ Attack Scenario 5.1	Network Scan is something whose port is 22

Table 6.23: Explainer Classes with Verbalization

approach offers.

As already explicated, we aim to examine the advantages of incorporating a sub-symbolic explainer into our framework, rather than relying on the input-output matching method for explaining GNN predictions, as seen in previous works such as [19] and [22]. In these approaches, no connection between the KB and the black-box are considered, which makes it impossible to affirm that the explanation given is why the model predicted this output.

Quantitative Results A total of 16 explainer classes have been created through framing the problem as classification task and therefore explaining anomalies. As can be seen in Table 6.24, a total of 105 data events have been predicted as unexpected, or in other words, 105 alerts have been triggered, with nearly 40% being false positives. The learned explainer

classes apply to only 14 of the total 41 false positives, while covering 100% of all true positive alerts. This gives the domain expert the possibility to filter out all triggered alerts that have been created based on the availability of an explanation, reducing the need to investigate false positives by 66%. In a further step, the fidelity of the explanation per alert can be taken into account. Such an additional filtering step would lead to a reduction of the false positives by 93%. However, 39% of true positives would be missed.
Therefore, depending on the preferences of the domain expert, it might be preferable to use the fidelity score as a means to prioritize the alerts. The use of these filter and prioritization techniques significantly reduces the time and resources need by the domain expert. Additionally, more time will be saved in the analysis of the remaining alerts, as explanations for these are available as seen in Section 6.2.3. While this is a relatively small example, we can see the potential benefit of applying such a method in a large-scale OT system.
In terms of comparative evaluation, 90% of all ground truth statements are covered by the explanations generated by our method.

To evaluate our method based on its ability to not generate explanation for wrong predictions and therefore validating them, we look at the fidelity of false and true positives. Table 6.25 shows the difference in entailments for the correctly classified (true positives TP) and incorrectly classified data events (false positives FP). We can see, that the average fidelity for entailments is 54 percentage points lower for false positive than true positives, showing the efficacy of our method.

Justification Axioms: Through justifications, we provide causality for explanations, based on domain knowledge. As the domain knowledge for cybersecurity use case comes with some structural depth as elaborated in 6.2, the size of the justifications ranges from 3 axioms to 9 axioms.
Comparison of our OntExplainer with DL-Learner Explanations: When comparing this purely symbolic approach with our hybrid method, we find that using only the DL-Learner comes with significantly lower prediction accuracy and also explanatory value. The predictive accuracy of the GNN using the same subset of training data is 81%. When applying the DL-Learner to carry out classifications, we are restricted to only one classifier. This means, even if we allow more complex class expressions, we only have one explanation for the target predicate, i.e., `id.orig_h some (isPartOf some devNetwork` (pred. acc.: 72.73%)

	Manual alert security check by analyst	**Alert with Explanation**	Number of alerts reduced by %	**Average fidelity for alert > 0**	Number of alerts reduced by %
Total (TP + FP)	**105**	**78**	**0.26**	**42**	**0.60**
App. activity	12	12	0	6	0.50
Network (HTTPS)	29	23	0.21	18	0.38
Network (SSH)	23	13	0.43	11	0.52
Credential Use	12	6	0.50	3	0.75
Network Scan	29	23	0.21	4	0.86
TP (total)	**64**	**64**	**0**	**39**	**0.39**
App. activity	11	11	0	6	0.45
Network (HTTPS)	23	23	0	18	0.22
Network (SSH)	8	8	0	8	0
Credential Use	3	3	0	3	0
Network Scan	18	18	0	4	0.78
FP (total)	**41**	**14**	**0.66**	**3**	**0.93**
App. activity	1	1	0	0	1
Network (HTTPS)	6	0	1	0	1
Network (SSH)	14	5	0.64	3	0.79
Credential Use	9	3	0.67	0	1
Network Scan	11	5	0.55	0	1

Table 6.24: Quantitative Results for all as suspicious flagged data events (True Positives TP and False Positives FP)

Table 6.25: Average fidelity for true positives and false positives.

	TP	FP
Number of instances	64	41
Average fidelity	0.61	0.07

Qualitative Results Through the verbalization step the class expressions' human-centricity is enhanced, where entailments reach a real level of human readability and are not relying on technical expressions for experts anymore. E.g. the class expression ($dataevent\_client$ **some** $App4$) **and**
($dataevent\_variable$ **some** ($hasDataType$ **some** (not ($hasDataType\_REAL$)))) of the class $Sniffing$ is translated to "Sniffing is something whose data event client is App4 and whose data event variable is something that has data type something that is not data type real". In Table 6.23, a selection of explanations can be seen, along with their verbalization and correspondence with a sub-scenario. For example, the verbalized explanation "Credential Use is something whose initial client is an App3 and whose initial server is something that is part of a development network" corresponds to Scenario 4.1. "Access to OPC-UA server from an IP address that corresponds to a development host." The explanation captures the anomaly and is even more specific to the concrete data example, as it also gives information about the initial client, providing the analyst with more details to work with. This explanation of a Credential Use anomaly is, for example, entailed for the triple $App3$ $initiatedFrom$ 192.168.0.80.
The explanation "Network Scan is something whose port is 22", which is, amongst others, entailed for connection 192.168.0.18 *to* 192.168.0.60, may need some additional information for a layman, but should give ample information for a domain expert. Here, a network scan is carried out with a certain IP Address and, of course, also IPs are scanned that it normally connects to, but with the wrong port - SSH (22) instead of HTTPS (443). Overall, we can see that the explanations capture the general scenarios, while often being more specific in describing the concrete anomaly in the data. An example for a justification of an entailment for an as suspicious flagged data event is given in Table 6.26, showing that since in the specific data event App3 is accessed and and the IP Address is part of a development network, the explainer class is entailed.

Table 6.26: Example justification $\mathcal{J}(\mathcal{O}^{Cybersecurity}$, (data_event_231)).

(1)	`data_event_231 init_server 192.168.0.80`
(2)	`data_event_231 init_client app3_cl`
(3)	`192.168.0.80 isPartOf devNetwork`
(4)	`app3_cl Type App3`
(5)	`CredentialUse EquivalentTo (init_client some App3) and (init_server some (isPartOf some devNetwork))`

6.2.4 Discussion

All three methods have been applied to the cybersecurity dataset, integrating different levels of domain knowledge and providing different types and levels of explainability.
The results for RERE applied to the cybersecurity dataset show that our method achieves higher fidelity compared to sub-symbolic benchmark methods. Sensitivity is evaluated with the Label-Flip rate. With a resulting 0% Label-Flip rate, it can be shown that such a constraint can be integrated without harming the quality of the explanations. As the cybersecurity dataset doesn't come with ground truth explanations, an accuracy score cannot be calculated here.

The results for SUBGREX show, that using decision trees to extract if-then rules from explainer subgraphs enhances the naturalness of explanations, albeit the explanations might be hard to understand for a non-expert. It can be shown that by including domain knowledge, in this case an entity type dictionary, sensitivity can be provided. The utility of these properties is explicated in the substantially increased accuracy score as well as the comparative evaluation score compared to the baseline methods. For this use case, reference to graph topology plays only a secondary role, as there are no recurrent graph motifs or structures present in the domain ontology. However, without any domain knowledge, the explanations would be very convoluted or non-existent. While the fidelity metric employed here is only an indication, the integration of sub-symbolic importance scores shows promising results, similarly to the Mutagenicity use case.

The OntExplainer results show, that naturalness of explanations is given by the class expressions and further enhanced by the verbalization step. An extensive amount of cybersecurity

domain knowledge has been modelled in the ontology used, as described above, providing a high level of sensitivity and reference to graph topology as nodes and edges, along with their features, are modelled likewise. This leads to a high level of ground truth statement coverage in the results and explanations that come with domain knowledge vocabulary with justifications that contain up to 9 axioms. In this use case, one of the tasks is to reduce the amount of false positives in order to decrease the workload of the cybersecurity analysts. That's why the reduction rate of false positives is taken into account for evaluating our method. Here, the fidelity metric can provide the additional function to prioritize explanations based on their fidelity score in case triage has to be done by cybersecurity analysts. Furthermore, the significant difference in fidelity between false positives and true positives, shows the validity of the fidelity metric used.

6.3 Additional Experiments

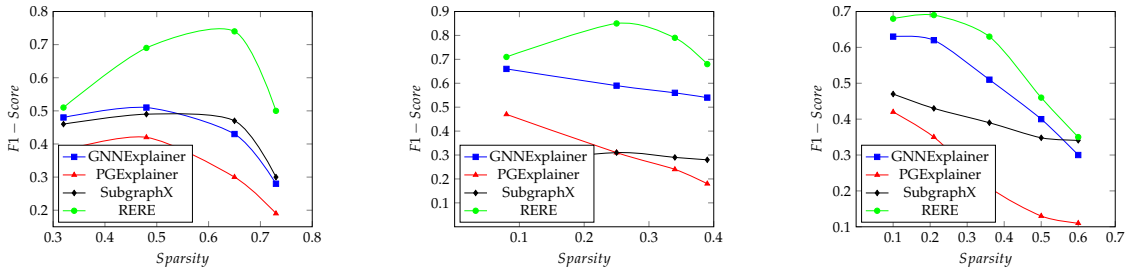
We include this additional experiments for the RERE method, in order to test the efficacy of the method for synthetic datasets that come with ground truth. This enables computing the accuracy score.

6.3.1 Subgraph Explanations

Synthetic Data: We follow the setting in the baseline approaches and construct different kinds of node classification datasets including BA-SHAPES, TREE-CYCLES, and TREE-GRID. Using synthetic data sets come with the benefit of intuitive motifs and labelling, which is understandable by humans [149] and makes it therefore possible to quantify the accuracy of the explanations compared to real-world datasets. BA-SHAPES is a node classification dataset with a base graph of 300 nodes and a set of 80 five-node "house"-structured network motifs, which are attached to randomly selected nodes of the base graph. The resulting graph is further perturbed by adding $0.1N$ random edges and nodes are assigned to 4 classes based on their structural roles (the top, middle, and bottom node of the house and nodes not belonging to a house). TREE-CYCLES is a node classification dataset with two different labels, that consists of a base 8-level balanced binary tree and 80 six-node cycle motifs, which are attached to random nodes of the base graph. TREE-GRID is the same as TREE-CYCLES, except that 3-by-3 grid motifs are attached to the base tree graph in place of cycle motifs.

REDDIT-BINARY Data: REDDIT-BINARY is a dataset of 2000 graphs, each representing an online discussion thread on Reddit [177], classified based on the type of discussion.

We follow the experimental settings in GNNExplainer [1]. We train a three-layer GNN and then apply all introduced explainer methods to explain predictions made by the GNN. We use the Adam optimizer to train the GNN. All GNN models are trained for 1000 epochs with learning rate 0.001, reaching accuracy of at least 85% for graph classification datasets, and 95% for node classification datasets. Like for the other experiments, the train/validation/test split is 80/10/10for all datasets. For RERE we train for at least 30000 episodes. We use the Adam optimizer and learning rate 0.001. For RERE, we train ten policies based on randomly chosen graphs for each class and average the results.



BA-SHAPES **TREE-CYCLES** **TREE-GRID**

Figure 6.4: F-1 Scores plotted for different levels of sparsity for BA-SHAPES, TREE-CYCLES and TREE-GRID data sets.

Benchmark Methods As with the other RERE experiments, we compare our approach with the benchmark methods GNNExplainer [1], PGExplainer [2] and SubgraphX [3].

Quantitative Results As you can see in Table 6.27 for fidelity, RERE delivers the best results. Furthermore, when looking at the Label-Flip rate, RERE significantly outperforms the other methods, as it can include constraints that prevent any Label-Flips. This is a substantial improvement over the baseline approaches, which can include Label-Flips for up to 85% of explanations for the PGExplainer, 81% for SubgraphX and 70% for the GNNExplainer. The low Label-Flip rate for RERE aligns with its comparatively low overall fidelity score. The change in prediction, by keeping important input features and removing unimportant features, should be as low as possible. In terms of F1-Score, RERE continuously outperforms the other three methods, by up to 15 percentage points. For a more comprehensive comparison, we compare the different methods using F1-Score under a variety of different levels of sparsity. The results are reported in Figure 6.4 where the curves of F1-Scores are plotted with respect to the sparsity scores. We can see that for all experiments, RERE outperforms the comparing methods significantly and consistently for different sparsity levels.

Qualitative Results The successive removal of edges by the agent is shown in Figure 6.5, with the respective last step being the final explanation. As demonstrated in these Figures, the ground-truth motifs are being correctly identified by RERE. While the baseline methods GNNExplainer and PGExplainer have also shown to identify such motifs, RERE comes with the option to follow the edge removal process successively, where the least informative edges are being removed first. This property can increase the trustworthiness of the explanations, as the user can follow the subgraph reduction procedure more closely, which provides

Table 6.27: Performance evaluation of RERE and alternative baseline explainability approaches. Bold indicates best result.

	Fidelity (Eq. 2.7)	Label-Flip	F1-Score	Sparsity
BA-SHAPES				
GNNExplainer	0.26	0.18	0.59	0.68
PGExplainer	0.26	0.29	0.35	0.69
SubgraphX	0.21	0.31	0.41	0.69
RERE	**0.16**	**0**	**0.74**	0.68
TREE-CYCLES				
GNNExplainer	0.12	0.14	0.68	0.13
PGExplainer	0.12	0.13	0.44	0.1
SubgraphX	0.04	0.19	0.30	0.11
RERE	**0.02**	**0**	**0.71**	0.09
TREE-GRID				
GNNExplainer	0.54	0.7	0.56	0.36
PGExplainer	0.7	0.85	0.24	0.43
SubgraphX	0.39	0.81	0.35	0.38
RERE	**0.11**	**0**	**0.60**	0.38
REDDIT-BINARY				
GNNExplainer	0.12	0.18	NA	0.47
PGExplainer	0.17	0.12	NA	0.48
SubgraphX	0.16	0.21	NA	0.46
RERE	**0.06**	**0**	NA	0.47

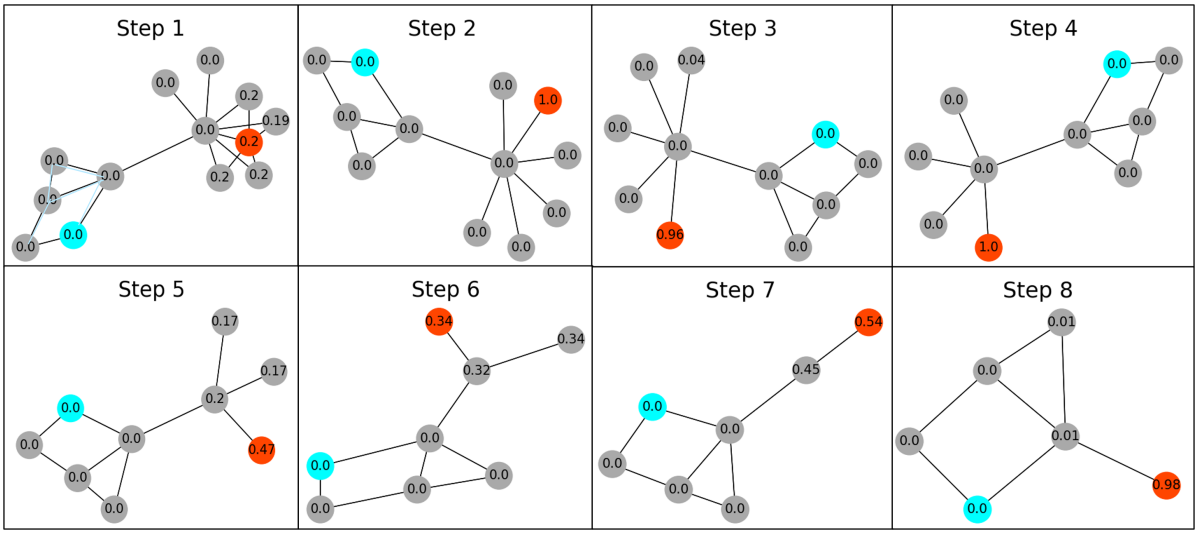


Figure 6.5: Step-wise removal of edges based on RERE for BA-SHAPES. The node labels give the action probability of an edge connected to this node being removed. The red node has the highest probability of a connected edge being removed in the next step, and the turquoise node is the centre node for the node classification.

traceability [72]. In Figure 6.5, the individual steps are shown for a BA-SHAPES graph to explain a node classification. In this example, the bottom node of a "house"-motif is the centre node. The node labels give the action probability of an edge connected to this node being removed, indicating its respective importance for the prediction of the centre node label. Through this step-wise process we get some additional insights into the GNN, as we don't only get a final explanation, but can also analyze how RERE arrives there. Here, we can see that up until Step 8, all nodes in the house motif have zero probability of having an edge removed next, whereas the edges outside the house motif are being removed step by step. Furthermore, we can detect, that the node which is closest to the centre node with only 2 hops distance, is removed only in the last step. This indicates that proximity to the centre node plays a role in the classification decision by the GNN.

6.3.2 Discussion

We included the synthetic dataset into the evaluation in order to evaluate RERE's capability to produce accurate explanations. The results for RERE applied to the synthetic datasets show that our method furthermore achieves higher levels of fidelity compared to sub-symbolic benchmark methods. Additionally, even with integrating a domain knowledge constraint and therefore providing sensitivity, the accuracy of our method outperforms the baseline.

7 Conclusions and Outlook

In this chapter, we draw conclusions from the preceding chapters. We present the main conclusions from each of the previous chapters. Lastly, in Section 7.2, we provide an outlook.

7.1 Conclusions

In this thesis, we created three distinct methods that incorporate the identified properties naturalness, sensitivity, reference to graph topology and fidelity to varying degrees, and evaluated them using two different datasets. We can draw several general conclusions from our experiments:

- *To effectively create explanations for domain experts, it is necessary to possess a certain level of expertise in the relevant field.*

  In other words, an explanation for a domain expert comes with the requirement to have the respective domain knowledge available. Hence, the profile of the intended audience, or "explainee," should be considered as part of the modeling process. Structured domain knowledge, such as ontologies, is not always readily accessible. For technical applications that, for example, involve chemical or physical knowledge, this information is usually available. However, in other domains, the knowledge may not be as well-structured. If the necessary domain expertise isn't too complex, as seen in our toy example, common sense explainability might be sufficient. In our evaluation, we focus on use cases where structured domain knowledge is present, and our method is parameterizable using the available background knowledge, allowing it to be adapted to the specific requirements of the explainee. In cases where no information about the explainee is available, the challenge of explaining becomes more complex.

- *Integrating domain knowledge improves properties sensitivity (RERE +38pp, SUBGREX +18pp, OntExplainer + 53pp) and reference to graph topology (RERE + 6%).*

Through the integration of domain knowledge (even unstructured free-form domain knowledge), the quality of explanations in terms of sensitivity is improved as well as reference to graph topology, as is shown by the experiments. Using the RERE method for the cybersecurity use case, the sensitivity as quantified by the Label-Flip rate is increased by 38pp in comparison with the best state-of-the-art method GNNExplainer. Using the SUBGREX method for the cybersecurity use case, the sensitivity as quantified by the comparative evaluation score is increased by 18pp in comparison with same method not including domain knowledge. Using the OntExplainer method for the chemical molecule use case, the sensitivity as quantified by the comparative evaluation score is increased by 53pp. Furthermore, using the RERE method, for the synthetic data sets, the reference to graph topology, as quantified by the accuracy method, is increased by +6% in comparison with the best state-of-the-art method GNNExplainer.

- *The integration of sub-symbolic and symbolic explainer methods increases complexity, but it results in a level of explainability for GNNs, that cannot be achieved by either approach alone:*
  - *Fidelity +7% (OntExplainer)*
  - *Sensitivity +46pp (SUBGREX)*
  - *Reference to graph topology +77% (OntExplainer), +35% (SUBGREX)*

  This combination of methods satisfies properties that are not attainable through either approach individually. The added complexity is justifiable if it leads to the desired level of explainability, which encompasses both naturalness and sensitivity, as well as reference to the graph topology and fidelity. The integration of symbolic and sub-symbolic models can be challenging due to the intricate mapping processes involved. However, if a more limited set of explanation properties is sufficient, a simpler solution may be adequate.
  The experiments demonstrate that the integration of a sub-symbolic explanation component into the method leads to an improvement in the fidelity of the explanations, as anticipated. Furthermore, by selectively determining which features have a significant impact on the decision-making process of a GNN, the results show that the sensitivity and reference to graph topology of the results are also improved. Using the OntExplainer method for the chemical molecule use case, the fidelity as quantified by the fidelity metric is increased by 7% in comparison with omitting the sub-symbolic element. Reference to graph topology is increased by 77% as quantified by predictive accuracy. Using the SUBGREX method for the cybersecurity use case, the sensitivity as

quantified by the comparative evaluation score is increased by 46pp in comparison with omitting the sub-symbolic element. Reference to graph topology is increased by 35% as quantified by the predictive accuracy.

- *Additional criteria for the choice of the best explanation method are: use case risk, availability of structured domain knowledge and whether local and global explanations are needed.*

  A number of methods exist, each with its own set of benefits and drawbacks, and the choice of approach depends on the user's specific requirements and considerations. Using ontological explanations or in other words, utilizing structured domain knowledge, if available, provides more insightful results compared to non-ontological explanations as it enables reasoning. Through the entailment capabilities of ontologies, both local and global explanations can be achieved. Furthermore, as it is also necessary to map free-form knowledge for integration, there isn't necessarily a complexity reduction for non-ontological explanations. Nevertheless, if no structured domain knowledge is available, incorporating free-form knowledge is still advantageous as it can reference graph topology. The risk associated with low-fidelity explanations should be evaluated based on the use case, and measures to enhance fidelity should be incorporated as needed.

- The proposed method RERE which provides subgraph explanations, has properties fidelity, reference to graph topology and sensitivity and is therefore very well suited for users that want a local explanation with the option to include domain knowledge constraints and have a basic knowledge of graphs, and can due to its sensitivity be considered superior to other sub-symbolic explanation methods.

- The proposed method SUBGREX which provides non-ontological explanations, has properties naturalness, reference to graph topology and sensitivity and is therefore well suited for users that want a global explanation, and can due to its reference to graph topology be considered superior to other symbolic explanation methods.

- The proposed method OntExplainer which provides ontological explanations, has properties naturalness, sensitivity, reference to graph topology and fidelity and is therefore well suited for users that want a local or global explanation, and can due to its reference to graph topology and fidelity be considered superior to other explanation methods.

7.2 Outlook

As this thesis has argued, we believe that explainability for GNNs should come with a certain set of properties to ensure comprehensible user-friendly and faithful explanations. In our work we only discussed domain experts as explainees, but along the AI life cycle there are further types of users that have to be taken into consideration, such as users affected by model decisions, regulatory entities or product owners. These user groups might require a different type of explainability or background knowledge. Furthermore, there might be no information about the potential explainee available. If there is none, one possible solution could be to gather more information about the explainee through various means, such as surveys or user interviews, to better understand their needs and tailor the explanation accordingly. Additionally, analyzing the effect on explanations when the coupling of available domain knowledge with GNNs training is deepened before training is another interesting research direction. In summary, we believe that explaining Graph Neural Networks by using structured domain knowledge such as ontologies is well-suited to generate explanations for many user groups. We look forward to future developments in this field and to the deployment of such models on real-world tasks.

List of Figures

List of Tables

Bibliography

[1] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. "Gnnexplainer: Generating explanations for graph neural networks". In: *Advances in neural information processing systems*. 2019, pp. 9244–9255.

[2] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. "Parameterized explainer for graph neural network". In: *Advances in neural information processing systems* 33 (2020), pp. 19620–19631.

[3] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji. "On explainability of graph neural networks via subgraph explorations". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12241–12252.

[4] M. S. Schlichtkrull, N. De Cao, and I. Titov. "Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking". In: *International Conference on Learning Representations*. 2020.

[5] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri. "Cf-gnnexplainer: Counterfactual explanations for graph neural networks". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 4499–4511.

[6] T. Funke, M. Khosla, M. Rathee, and A. Anand. "Z ORRO: Valid, Sparse, and Stable Explanations in Graph Neural Networks". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[7] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann. "Explainability methods for graph convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10772–10781.

[8] F. Baldassarre and H. Azizpour. "Explainability Techniques for Graph Convolutional Networks". In: *International Conference on Machine Learning (ICML) Workshops, 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*. 2019.

[9] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schutt, K.-R. Mueller, and G. Montavon. "Higher-Order Explanations of Graph Neural Networks via Relevant Walks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[10] P. Xiong, T. Schnake, G. Montavon, K.-R. Müller, and S. Nakajima. "Efficient Computation of Higher-Order Subgraph Attribution via Message Passing". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 24478–24495.

[11] Y. Zhang, D. Defazio, and A. Ramesh. "Relex: A model-agnostic relational model explainer". In: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 2021, pp. 1042–1049.

[12] M. Vu and M. T. Thai. "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks". In: *Advances in neural information processing systems* 33 (2020), pp. 12225–12235.

[13] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang. "Graphlime: Local interpretable model explanations for graph neural networks". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[14] H. Yuan, J. Tang, X. Hu, and 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, pp. 430–438.

[15] R. R. Selvaraju, P. Chattopadhyay, M. Elhoseiny, T. Sharma, D. Batra, D. Parikh, and S. Lee. "Choose your neuron: Incorporating domain knowledge through neuron-importance". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 526–541.

[16] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)". In: *International conference on machine learning*. PMLR. 2018, pp. 2668–2677.

[17] M. Craven and J. Shavlik. "Extracting tree-structured representations of trained networks". In: *Advances in neural information processing systems* 8 (1995).

[18] R. Confalonieri, T. Weyde, T. R. Besold, and F. Moscoso del Prado Martın. "Trepan reloaded: A knowledge-driven approach to explaining artificial neural networks". In: (2020).

[19] M. K. Sarker, N. Xie, D. Doran, M. Raymer, and P. Hitzler. "Explaining trained neural networks with semantic web technologies: First steps". In: *arXiv preprint arXiv:1710.04324* (2017).

[20] Y. Geng, J. Chen, and E. O. Jimenez-Ruiz. "0000-0002-9083-4599 and Chen, H. Human-centric Transfer Learning Explanation via Knowledge Graph". In: *AAAI-19 Workshop on Network Interpretability for Deep Learning*. Vol. 27. 2019, p. 28.

[21] N. Dıaz-Rodrıguez, A. Lamas, J. Sanchez, G. Franchi, I. Donadello, S. Tabik, D. Filliat, P. Cruz, R. Montes, and F. Herrera. "EXplainable Neural-Symbolic Learning (X-NeSyL) methodology to fuse deep learning representations with expert knowledge graphs: The MonuMAI cultural heritage use case". In: *Information Fusion* 79 (2022), pp. 58–83.

[22] C. Widmer, M. Kamruzzaman Sarker, S. Nadella, J. Fiechter, I. Juvina, B. Minnery, P. Hitzler, J. Schwartz, and M. Raymer. "Towards Human-Compatible XAI: Explaining Data Differentials with Concept Induction over Background Knowledge". In: *arXiv e-prints* (2022), arXiv–2209.

[23] J. Chen, F. Lécué, J. Z. Pan, I. Horrocks, and H. Chen. "Knowledge-based transfer learning explanation". In: *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*. 2018.

[24] M. Labaf, P. Hitzler, and A. B. Evans. "Propositional Rule Extraction from Neural Networks under Background Knowledge." In: *NeSy*. 2017.

[25] S. Deng, N. Zhang, W. Zhang, J. Chen, J. Z. Pan, and H. Chen. "Knowledge-driven stock trend prediction and explanation via temporal convolutional network". In: *Companion Proceedings of The 2019 World Wide Web Conference*. 2019, pp. 678–685.

[26] C. Panigutti, A. Perotti, and D. Pedreschi. "Doctor XAI: an ontology-based approach to black-box sequential data classification explanations". In: *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 2020, pp. 629–639.

[27] A. Himmelhuber, S. Zillner, S. Grimm, M. Ringsquandl, M. Joblin, and T. A. Runkler. "A New Concept for Explaining Graph Neural Networks." In: *NeSy*. 2021, pp. 1–5.

[28] A. Himmelhuber, S. Grimm, S. Zillner, M. Joblin, M. Ringsquandl, and T. Runkler. "Combining Sub-symbolic and Symbolic Methods for Explainability". In: *International Joint Conference on Rules and Reasoning*. Springer. 2021, pp. 172–187.

[29] J. B. Hamrick, K. R. Allen, V. Bapst, T. Zhu, K. R. McKee, J. B. Tenenbaum, and P. W. Battaglia. "Relational inductive bias for physical construction in humans and machines". In: ().

[30] M. Welling and T. N. Kipf. "Semi-supervised classification with graph convolutional networks". In: *J. International Conference on Learning Representations (ICLR 2017)*. 2016.

[31] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. "Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017, pp. 1802–1808.

[32] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. "Protein interface prediction using graph convolutional networks". In: *Advances in neural information processing systems* 30 (2017).

[33] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. "Interaction networks for learning about objects, relations and physics". In: *Advances in neural information processing systems* 29 (2016).

[34] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. "Learning combinatorial optimization algorithms over graphs". In: *Advances in neural information processing systems* 30 (2017).

[35] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications". In: *arXiv preprint arXiv:1812.08434* (2018).

[36] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[37] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81.

[38] L. Romijn, B. Ó. Nualláin, and L. Torenvliet. "Discovering motifs in real-world social networks". In: *International Conference on Current Trends in Theory and Practice of Informatics*. Springer. 2015, pp. 463–474.

[39] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C.-K. Lee. "Motif-based graph self-supervised learning for molecular property prediction". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15870–15882.

[40] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[41] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". In: *ICLR* (2017).

[42] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph attention networks". In: *ICLR* (2017).

[43] W. Hamilton, Z. Ying, and J. Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[44] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity". In: *Journal of medicinal chemistry* 34.2 (1991), pp. 786–797.

[45] E. Ilkou and M. Koutraki. "Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies?" In: *CIKM (Workshops)*. 2020.

[46] P. Hitzler, F. Bianchi, M. Ebrahimi, and M. K. Sarker. "Neural-symbolic integration and the semantic web". In: *Semantic Web* 11.1 (2020), pp. 3–11.

[47] M. K. Sarker, L. Zhou, A. Eberhart, and P. Hitzler. "Neuro-symbolic artificial intelligence: Current trends". In: *arXiv preprint arXiv:2105.05330* (2021).

[48] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al. "Knowledge graphs". In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37.

[49] E. N. Benderskaya and S. V. Zhukova. "Multidisciplinary trends in modern artificial intelligence: Turing's way". In: *Artificial Intelligence, Evolutionary Computing and Metaheuristics*. Springer, 2013, pp. 319–343.

[50] I. Hatzilygeroudis and J. Prentzas. "Neuro-symbolic approaches for knowledge representation in expert systems". In: *International Journal of Hybrid Intelligent Systems* 1.3-4 (2004), pp. 111–126.

[51] A. d. Garcez, S. Bader, H. Bowman, L. C. Lamb, L. de Penning, B. Illuminoo, H. Poon, and C. Gerson Zaverucha. "Neural-symbolic learning and reasoning: A survey and interpretation". In: *Neuro-Symbolic Artificial Intelligence: The State of the Art* 342 (2022), p. 1.

[52] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. "The world-wide web". In: *Communications of the ACM* 37.8 (1994), pp. 76–82.

[53] S. Chari, D. M. Gruen, O. Seneviratne, and D. L. McGuinness. "Foundations of Explainable Knowledge-Enabled Systems". In: *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*. IOS Press, 2020, pp. 23–48.

[54] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao. "Prov-o: The prov ontology". In: *World Wide Web Consortium* (2013).

[55] A. Hogan, A. Harth, and A. Polleres. "Scalable authoritative OWL reasoning for the web". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.2 (2009), pp. 49–90.

[56] M. Loskyll, I. Heck, J. Schlick, and M. Schwarz. "Context-based orchestration for control of resource-efficient manufacturing processes". In: *Future Internet* 4.3 (2012), pp. 737–761.

[57] D. L. McGuinness, F. Van Harmelen, et al. "OWL web ontology language overview". In: *W3C recommendation* 10.10 (2004), p. 2004.

[58] M. Horridge and P. F. Patel-Schneider. "Manchester Syntax for OWL 1.1." In: *OWLED (Spring)*. Citeseer. 2008.

[59] M. Horridge, J. Bauer, B. Parsia, and U. Sattler. "Understanding Entailments in OWL." In: *OWLED*. 2008.

[60] J. Lehmann. "DL-Learner: learning concepts in description logics". In: *The Journal of Machine Learning Research* 10 (2009), pp. 2639–2642.

[61] M. C. Daconta, L. J. Obrst, and K. T. Smith. *The Semantic Web: a guide to the future of XML, Web services, and knowledge management*. John Wiley & Sons, 2003.

[62] A.-C. N. Ngomo, D. Moussallem, and L. Bühmann. "A Holistic Natural Language Generation Framework for the Semantic Web". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. 2019, pp. 819–828.

[63] S. Wasserman and K. Faust. “Social network analysis in the social and behavioral sciences”. In: *Social network analysis: Methods and applications* 1994 (1994), pp. 1–27.

[64] L. Stone, D. Simberloff, and Y. Artzy-Randrup. “Network motifs and their origins”. In: *PLoS computational biology* 15.4 (2019).

[65] P. W. Holland and S. Leinhardt. “Local structure in social networks”. In: *Sociological methodology* 7 (1976), pp. 1–45.

[66] A. B. Arrieta, N. Dıaz-Rodrıguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcıa, S. Gil-López, D. Molina, R. Benjamins, et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115.

[67] D. Castelvecchi. “Can we open the black box of AI?” In: *Nature News* 538.7623 (2016), p. 20.

[68] A. Preece, D. Harborne, D. Braines, R. Tomsett, and S. Chakraborty. “Stakeholders in explainable AI”. In: *arXiv preprint arXiv:1810.00184* (2018).

[69] H. Yuan, H. Yu, S. Gui, and S. Ji. “Explainability in graph neural networks: A taxonomic survey”. In: *arXiv preprint arXiv:2012.15445* (2020).

[70] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. “A survey of methods for explaining black box models”. In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[71] S. Mohseni, N. Zarei, and E. D. Ragan. “A multidisciplinary survey and framework for design and evaluation of explainable AI systems”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 11.3-4 (2021), pp. 1–45.

[72] *Ethics guidelines for trustworthy AI*. `https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai`, Accessed: 2022-09-08. 2019.

[73] C. Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[74] T. Miller, P. Howe, and L. Sonenberg. “Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences”. In: *IJCAI Workshop on Explainable Artificial Intelligence* (2017).

[75] U. Ehsan, P. Wintersberger, Q. V. Liao, E. A. Watkins, C. Manger, H. Daumé III, A. Riener, and M. O. Riedl. "Human-Centered Explainable AI (HCXAI): beyond opening the black-box of AI". In: *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 2022, pp. 1–7.

[76] D. M. Best, A. Endert, and D. Kidwell. "7 key challenges for visualization in cyber network defense". In: *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*. 2014, pp. 33–40.

[77] J. R. Goodall, E. D. Ragan, C. A. Steed, J. W. Reed, G. D. Richardson, K. M. Huffer, R. A. Bridges, and J. A. Laska. "Situ: Identifying and explaining suspicious behavior in networks". In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 204–214.

[78] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1721–1730.

[79] J. Krause, A. Perer, and K. Ng. "Interacting with predictions: Visual inspection of black-box machine learning models". In: *Proceedings of the 2016 CHI conference on human factors in computing systems*. 2016, pp. 5686–5697.

[80] M. Liu, S. Liu, X. Zhu, Q. Liao, F. Wei, and S. Pan. "An uncertainty-aware approach for exploratory microblog retrieval". In: *IEEE transactions on visualization and computer graphics* 22.1 (2015), pp. 250–259.

[81] X. Wang, S. Liu, J. Liu, J. Chen, J. Zhu, and B. Guo. "TopicPanorama: A full picture of relevant topics". In: *IEEE transactions on visualization and computer graphics* 22.12 (2016), pp. 2508–2521.

[82] C. Robinson, F. Hohman, and B. Dilkina. "A deep learning approach for population estimation from satellite imagery". In: *Proceedings of the 1st ACM SIGSPATIAL Workshop on Geospatial Humanities*. 2017, pp. 47–54.

[83] Q. V. Liao and K. R. Varshney. "Human-centered explainable ai (xai): From algorithms to user experiences". In: *arXiv preprint arXiv:2110.10790* (2021).

[84] D. Sacha, H. Senaratne, B. C. Kwon, G. Ellis, and D. A. Keim. "The role of uncertainty, awareness, and trust in visual analytics". In: *IEEE transactions on visualization and computer graphics* 22.1 (2015), pp. 240–249.

[85] Y. Ming, H. Qu, and E. Bertini. “Rulematrix: Visualizing and understanding classifiers with rules”. In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 342–352.

[86] F. Hohman, H. Park, C. Robinson, and D. H. P. Chau. “Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations”. In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 1096–1106.

[87] Y. Ahn and Y.-R. Lin. “Fairsight: Visual analytics for fairness in decision making”. In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 1086–1095.

[88] A. Weller. “Transparency: Motivations and Challenges”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* 11700 (2019), p. 23.

[89] B. Lim. “Improving understanding, trust, and control with intelligibility in context-aware applications”. In: *Human-Computer Interaction* (2011).

[90] Q. Zhang, W. Wang, and S.-C. Zhu. “Examining cnn representations with respect to dataset bias”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[91] R. C. Fong and A. Vedaldi. “Interpretable explanations of black boxes by meaningful perturbation”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3429–3437.

[92] M. D. Zeiler and R. Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[93] M. T. Ribeiro, S. Singh, and C. Guestrin. “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[94] L. A. Hendricks, K. Burns, K. Saenko, T. Darrell, and A. Rohrbach. “Women also snowboard: Overcoming bias in captioning models”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 771–787.

[95] A. Bennetot, J.-L. Laurent, R. Chatila, and N. Dıaz-Rodrıguez. “Towards Explainable Neural-Symbolic Visual Reasoning”. In: *IJCAI* (2019).

[96] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. “Towards better analysis of deep convolutional neural networks”. In: *IEEE transactions on visualization and computer graphics* 23.1 (2016), pp. 91–100.

[97] H. Lakkaraju, S. H. Bach, and J. Leskovec. "Interpretable decision sets: A joint framework for description and prediction". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1675–1684.

[98] M. Wu, M. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. "Beyond sparsity: Tree regularization of deep models for interpretability". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[99] M. Chromik, M. Eiband, F. Buchner, A. Krüger, and A. Butz. "I think i get your point, AI! the illusion of explanatory depth in explainable AI". In: *26th International Conference on Intelligent User Interfaces*. 2021, pp. 307–317.

[100] H. Yuan, H. Yu, S. Gui, and S. Ji. "Explainability in graph neural networks: A taxonomic survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[101] R. Calegari, G. Ciatto, and A. Omicini. "On the integration of symbolic and sub-symbolic techniques for XAI: A survey". In: *Intelligenza Artificiale* 14.1 (2020), pp. 7–32.

[102] A. Krajna, M. Kovac, M. Brcic, and A. Šarčević. "Explainable Artificial Intelligence: An Updated Perspective". In: *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE. 2022, pp. 859–864.

[103] X. Chen, R. Cai, Y. Fang, M. Wu, Z. Li, and Z. Hao. "Motif Graph Neural Network". In: *arXiv preprint arXiv:2112.14900* (2021).

[104] Z. Yu and H. Gao. "MotifExplainer: a Motif-based Graph Neural Network Explainer". In: *arXiv preprint arXiv:2202.00519* (2022).

[105] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei. "Trustworthy Graph Neural Networks: Aspects, Methods and Trends". In: *arXiv preprint arXiv:2205.07424* (2022).

[106] Y. Alufaisan, L. R. Marusich, J. Z. Bakdash, Y. Zhou, and M. Kantarcioglu. "Does explainable artificial intelligence improve human decision-making?" In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 6618–6626.

[107] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. Wortman Vaughan. "Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning". In: *Proceedings of the 2020 CHI conference on human factors in computing systems*. 2020, pp. 1–14.

[108] D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[109] R. E. Petty and J. T. Cacioppo. “The elaboration likelihood model of persuasion”. In: *Communication and persuasion*. Springer, 1986, pp. 1–24.

[110] Z. Buçinca, P. Lin, K. Z. Gajos, and E. L. Glassman. “Proxy tasks and subjective measures can be misleading in evaluating explainable AI systems”. In: *Proceedings of the 25th international conference on intelligent user interfaces*. 2020, pp. 454–464.

[111] M. Nourani, C. Roy, J. E. Block, D. R. Honeycutt, T. Rahman, E. Ragan, and V. Gogate. “Anchoring Bias Affects Mental Model Formation and User Reliance in Explainable AI Systems”. In: *26th International Conference on Intelligent User Interfaces*. 2021, pp. 340–350.

[112] U. Ehsan, S. Passi, Q. V. Liao, L. Chan, I. Lee, M. Muller, M. O. Riedl, et al. “The who in explainable ai: How ai background shapes perceptions of ai explanations”. In: *arXiv preprint arXiv:2107.13509* (2021).

[113] M. Szymanski, M. Millecamp, and K. Verbert. “Visual, textual or hybrid: the effect of user expertise on different explanations”. In: *26th International Conference on Intelligent User Interfaces*. 2021, pp. 109–119.

[114] J. D. Moore and C. L. Paris. “Requirements for an expert system explanation facility”. In: *Computational Intelligence* 7.4 (1991), pp. 367–370.

[115] A. Springer and S. Whittaker. “Progressive disclosure: empirically motivated approaches to designing effective transparency”. In: *Proceedings of the 24th international conference on intelligent user interfaces*. 2019, pp. 107–120.

[116] M. J. Druzdzel. “Explanation in probabilistic systems: Is it feasible? Will it work”. In: *Proceedings of the Fifth International Workshop on Intelligent Information Systems (WIS-96)*. Citeseer. 1996, pp. 12–24.

[117] D. Doran, S. Schulz, and T. Besold. “What Does Explainable AI Really Mean? A New Conceptualization of Perspectives”. In: *CEUR Workshop Proceedings*. Vol. 2071. 2018.

[118] K. Sokol and P. Flach. “Explainability fact sheets: a framework for systematic assessment of explainable approaches”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 56–67.

[119] A. Rosenfeld. “Better metrics for evaluating explainable artificial intelligence”. In: *Proceedings of the 20th international conference on autonomous agents and multiagent systems*. 2021, pp. 45–50.

[120] A. d. Garcez and L. C. Lamb. "Neurosymbolic AI: The 3rd Wave". In: *arXiv preprint arXiv:2012.05876* (2020).

[121] M. T. Ribeiro, S. Singh, and C. Guestrin. "Nothing Else Matters: Model-Agnostic Explanations By Identifying Prediction Invariance". In: *NIPS Workshop on Interpretable Machine Learning in Complex Systems* 1050 (2016), p. 17.

[122] A. Papenmeier, G. Englebienne, and C. Seifert. "How model accuracy and explanation fidelity influence user trust in AI". In: *IJCAI Workshop on Explainable Artificial Intelligence (XAI) 2019*. 2019.

[123] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[124] C. Olah, A. Mordvintsev, and L. Schubert. "Feature visualization". In: *Distill* 2.11 (2017).

[125] P. Dabkowski and Y. Gal. "Real time image saliency for black box classifiers". In: *Advances in neural information processing systems* 30 (2017).

[126] J. Chen, L. Song, M. Wainwright, and M. Jordan. "Learning to explain: An information-theoretic perspective on model interpretation". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 883–892.

[127] U. Alon. "Network motifs: theory and experimental approaches". In: *Nature Reviews Genetics* 8.6 (2007), pp. 450–461.

[128] V. B. Kumar, B. Ganesan, M. Ameen, D. Sharma, and A. Agarwal. "Automated Evaluation of GNN Explanations with Neuro Symbolic Reasoning". In: *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR. 2022, pp. 314–318.

[129] M. T. Ribeiro, S. Singh, and C. Guestrin. "Model-agnostic interpretability of machine learning". In: *ICML Workshop on Human Interpretability in Machine Learning* (2016).

[130] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. "Explaining explanations: An overview of interpretability of machine learning". In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE. 2018, pp. 80–89.

[131] R. Krishnan, G. Sivakumar, and P. Bhattacharya. "Extracting decision trees from trained neural networks". In: *Pattern recognition* 32.12 (1999).

[132] U. Johansson and L. Niklasson. "Evolving decision trees using oracle guides". In: *2009 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE. 2009, pp. 238–244.

[133] U. Johansson, L. Niklasson, and R. König. "Accuracy vs. comprehensibility in data mining models". In: *Proceedings of the seventh international conference on information fusion*. Vol. 1. Citeseer. 2004, pp. 295–300.

[134] M. G. Augasta and T. Kathirvalavakumar. "Reverse engineering the neural networks for rule extraction in classification problems". In: *Neural processing letters* 35.2 (2012), pp. 131–150.

[135] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. "Local rule-based explanations of black box decision systems". In: *arXiv preprint arXiv:1805.10820* (2018).

[136] G. Su, D. Wei, K. R. Varshney, and D. M. Malioutov. "Interpretable two-level boolean rule learning for classification". In: *arXiv preprint arXiv:1511.07361* (2015).

[137] O. Bastani, C. Kim, and H. Bastani. "Interpretability via Model Extraction". In: ().

[138] R. Turner. "A model explanation system". In: *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*. IEEE. 2016, pp. 1–6.

[139] I. Donadello, M. Dragoni, and C. Eccher. "Persuasive explanation of reasoning inferences on dietary data". In: *PROFILES/SEMEX@ ISWC*. 2019.

[140] A. Seeliger, M. Pfaff, and H. Krcmar. "Semantic web technologies for explainable machine learning models: A literature review." In: *PROFILES/SEMEX@ ISWC* 2465 (2019), pp. 1–16.

[141] R. Confalonieri, T. Weyde, T. R. Besold, and F. Moscoso del Prado Martın. "TREPAN Reloaded: A Knowledge-Driven Approach to Explaining Black-Box Models". In: *ECAI 2020*. IOS Press, 2020, pp. 2457–2464.

[142] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen. "Interaction embeddings for prediction and explanation in knowledge graphs". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019, pp. 96–104.

[143] S. Mishra, B. L. Sturm, and S. Dixon. "Local interpretable model-agnostic explanations for music content analysis." In: *ISMIR*. Vol. 53. 2017, pp. 537–543.

[144] S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[145] F. Gedikli, D. Jannach, and M. Ge. “How should I explain? A comparison of different explanation types for recommender systems”. In: *International Journal of Human-Computer Studies* 72.4 (2014), pp. 367–382.

[146] D. Wang, Q. Yang, A. Abdul, and B. Y. Lim. “Designing theory-driven user-centric explainable AI”. In: *Proceedings of the 2019 CHI conference on human factors in computing systems*. 2019, pp. 1–15.

[147] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez. “How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation”. In: *arXiv preprint arXiv:1802.00682* (2018).

[148] B. Herman. “The Promise and Peril of Human Evaluation for Model Interpretability”. In: *arXiv e-prints* (2017), arXiv–1711.

[149] A. Himmelhuber, M. Joblin, M. Ringsquandl, and T. Runkler. “Demystifying Graph Neural Network Explanations”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 67–75.

[150] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso. “Machine learning interpretability: A survey on methods and metrics”. In: *Electronics* 8.8 (2019), p. 832.

[151] S. Mohseni, J. E. Block, and E. D. Ragan. “A human-grounded evaluation benchmark for local explanations of machine learning”. In: *arXiv preprint arXiv:1801.05075* (2018).

[152] P. Lertvittayakumjorn and F. Toni. “Human-grounded evaluations of explanation methods for text classification”. In: *EMNLP-IJCNLP 2019* (2019).

[153] N. Poerner, H. Schütze, and B. Roth. “Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 340–350.

[154] E. Greensmith, P. L. Bartlett, and J. Baxter. “Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning.” In: *Journal of Machine Learning Research* 5.9 (2004).

[155] W. L. Koltun. *Space filling atomic units and connectors for molecular models*. US Patent 3,170,246. Feb. 1965.

[156] J. Lehmann, S. Auer, L. Bühmann, and S. Tramp. “Class expression learning for ontology engineering”. In: *Journal of Web Semantics* 9.1 (2011), pp. 71–81.

[157] J. N. Paredes, J. C. L. Teze, G. I. Simari, and M. V. Martinez. "On the Importance of Domain-specific Explanations in AI-based Cybersecurity Systems (Technical Report)". In: *arXiv preprint arXiv:2108.02006* (2021).

[158] S. Mane and D. Rao. "Explaining Network Intrusion Detection System Using Explainable AI Framework". In: *arXiv e-prints* (2021), arXiv–2103.

[159] J. S. Garrido, D. Dold, and J. Frank. "Machine learning on knowledge graphs for context-aware security monitoring". In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE. 2021, pp. 55–60.

[160] D. Rao and S. Mane. "Zero-shot learning approach to adaptive Cybersecurity using Explainable AI". In: *arXiv preprint arXiv:2106.14647* (2021).

[161] A. Aldweesh, A. Derhab, and A. Z. Emam. "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues". In: *Knowledge-Based Systems* 189 (2020), p. 105124.

[162] A. Chemudupati, S. Kaulen, M. Mertens, and S. Zimmermann. "The convergence of IT and Operational Technology". In: *Atos Scientific Community* (2012), pp. 3–16.

[163] R. Paes, D. C. Mazur, B. K. Venne, and J. Ostrzenski. "A guide to securing industrial control networks: Integrating IT and OT systems". In: *IEEE Industry Applications Magazine* 26.2 (2019), pp. 47–53.

[164] L. Vigano and D. Magazzeni. "Explainable security". In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2020, pp. 293–300.

[165] S. Mahdavifar and A. A. Ghorbani. "DeNNeS: deep embedded neural network expert system for detecting cyber attacks". In: *Neural Computing and Applications* 32.18 (2020), pp. 14753–14780.

[166] M. Szczepański, M. Choraś, M. Pawlicki, and R. Kozik. "Achieving explainability of intrusion detection system by hybrid oracle-explainer approach". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[167] D. Doldy and J. S. Garridoy. "An energy-based model for neuro-symbolic reasoning on knowledge graphs". In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2021, pp. 916–921.

[168] W. Mahnke, S.-H. Leitner, and M. Damm. *OPC unified architecture*. Springer Science & Business Media, 2009.

[169] O. Kovalenko, M. Wimmer, M. Sabou, A. Lüder, F. J. Ekaputra, and S. Biffl. "Modeling automationml: Semantic web technologies vs. model-driven engineering". In: *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE. 2015, pp. 1–4.

[170] *Zeek documentation.* `https://docs.zeek.org/en/stable/intro/`, Accessed: 2021-02-04.

[171] S. Barnum. "Standardizing cyber threat intelligence information with the structured threat information expression (stix)". In: *Mitre Corporation* 11 (2012), pp. 1–22.

[172] Z. Syed, A. Padia, T. Finin, L. Mathews, and A. Joshi. "UCO: A unified cybersecurity ontology". In: *Workshops at the thirtieth AAAI conference on artificial intelligence*. 2016.

[173] M. Eckhart, A. Ekelhart, and E. R. Weippl. "Automated security risk identification using AutomationML-based engineering data". In: *IEEE Transactions on Dependable and Secure Computing* (2020).

[174] E. Kiesling, A. Ekelhart, K. Kurniawan, and F. Ekaputra. "The SEPSES knowledge graph: an integrated resource for cybersecurity". In: *International Semantic Web Conference*. Springer. 2019, pp. 198–214.

[175] S. N. Narayanan, A. Ganesan, K. Joshi, T. Oates, A. Joshi, and T. Finin. "Early detection of cybersecurity threats using collaborative cognition". In: *2018 IEEE 4th international conference on collaboration and internet computing (CIC)*. IEEE. 2018, pp. 354–363.

[176] M. A. Musen. "The protégé project: a look back and a look forward". In: *AI matters* 1.4 (2015), pp. 4–12.

[177] P. Yanardag and S. Vishwanathan. "Deep graph kernels". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1365–1374.