Journal of Complex Networks, 2026, **14**(2), cnag006
https://doi.org/10.1093/comnet/cnag006
Advance access publication 23 March 2026
Article

OXFORD

Assessing the explainability of Graph Neural Networks in random graphs classification task

Darja Cvetković and Marija Mitrović Dankulov *

Institute of Physics Belgrade, University of Belgrade, Pregrevica 118, Belgrade 11080, Serbia

*Corresponding author. Institute of Physics Belgrade, University of Belgrade, Pregrevica 118, Belgrade 11080, Serbia.
E-mail: mitrovic@ipb.ac.rs

Abstract

Graph Neural Networks (GNNs) have become the dominant deep learning model for learning on graph-structured data, enabling breakthroughs in fields ranging from bioinformatics to social network analysis. Yet, as any deep learning model they suffer from 'black box' syndrome. Their decisions making process remains largely unknown. In this work, we want to advance our understanding of GNN explainability. We evaluate the performance and stability of GNNExplainer, a widely used post-hoc interpretability method, on the simple task of random graph classification. Using three very different GNN architectures, Graph Convolutional Networks, Graph Attention Networks, and Graph Isomorphism Networks, we examine the explainability of models trained to distinguish between Erdős-Rényi and Barabási-Albert random graphs, as well as between dk-randomized variants of four real-world networks. Our results show that despite the models achieving perfect classification accuracy, feature importance values identified by GNNExplainer exhibit substantial variability across architectures, hyperparameters, and random seed values. Moreover, the extracted explanations often fail to align with theoretical expectations based on established graph properties, such as degree distributions and degree correlations. These findings indicate that explanations produced by GNNExplainer are highly model-, configuration-, and seed value-dependent, challenging its reliability for deriving general insights into GNN decision mechanisms. Our work highlights fundamental limitations in the current generation of GNN explanations using GNNExplainer and suggests the need for more stable, theoretically grounded approaches to explainability in graph-based learning.

Keywords Graph Neural Networks, explainability, GNNExplainer, graph classification

1. Introduction

Many real-world systems can be naturally represented by complex networks or graphs [1]. Complex network theory provides a framework for analysing these systems. It focuses on the characterization of structural patterns and statistical regularities that emerge from interactions among components [1]. Concepts such as degree distributions, motifs, community structure, and centrality measures

Editor: Francisco Rodrigues
Received: 14 November 2025. **Accepted:** 18 February 2026

have been used for uncovering universal principles across diverse domains [1, 2]. However, these features are typically designed for descriptive analysis rather than predictive tasks.

Machine and deep learning methods have emerged as a powerful paradigm for predictive modelling in complex systems [3]. While graphs are a powerful representation of the relational data typical of complex systems, they are quite challenging for traditional machine learning methods to process effectively. Standard AI methods such as linear regression, decision trees, and neural networks struggle with the non-linear behaviours, complex interactions, and intricate dynamics inherent in graph structured data [4]. Graph Neural Networks (GNN) have emerged as a promising class of models to address these challenges. Standard neural networks are primarily designed for tabular data, where each observation is represented as a fixed-length feature vector and dependencies between entities are not explicitly encoded. GNN, by contrast, are designed to operate on graph-structured data. These models aim to learn meaningful representations of graph-structured data, enabling the application of deep learning techniques to a wide range of problems spanning from social network analysis to drug discovery [5].

Unlike conventional neural networks, which operate on fixed-size vectors or grid-structured data, Graph Neural Networks are explicitly designed to process graph-structured inputs where relationships between entities are encoded as edges. GNNs incorporate network topology into the learning process through neighbourhood aggregation, allowing node representations to evolve based on the structure of the underlying graph. This enables GNNs to model relational dependencies that are central to complex networked systems but are inaccessible to standard neural architectures.

Graph classification is an important problem that has applications in many fields of science. For instance, in bioinformatics and cheminformatics graph classification can be used to predict protein function or toxicity or physical properties of a chemical compound [6, 7]. In natural language processing graph labeling is used for document classification, while in social networks it can be used for fake news detection [8]. The goal of graph classification is to determine a class or a label of a graph. There are various models that can be used for graph classification [9–17], which are mostly supervised and/or they include graph structure as a feature. Graph Neural Networks [18, 19] have emerged as a the leading method for graph classification. In the GNN approach, the model combines node features with network structure to achieve high accuracy of network classification.

Despite its success, AI has certain disadvantages, one of the most pressing being its lack of explainability, which makes it difficult to understand how complex models arrive at their decisions and what they actually 'learned' from the data. AI methods, especially ones based on neural networks, are often criticized as 'black boxes', meaning that their internal decision-making process is not readily interpretable by humans. This lack of transparency is a significant limitation, as many real-world applications require not just accurate predictions but also an understanding of the underlying reasoning.

GNNs also suffer from this problem. To overcome this limitation, researchers have proposed various methods to improve the explainability of GNNs [20–23]. Yuan et al. [21] provide a taxonomy of explainability methods for GNNs—they categorize the methods into two main branches: instance-level methods that aim to identify the components of the input data that are responsible for model's predictions [24–33] and model-level methods that aim to provide a global understanding of a trained model [34–37]. Although model-level methods appear to be the most suitable choice for studying the inner workings of the GNN 'black box', they are currently unreliable, leading most explainability studies to focus on instance-level methods. Instance-level methods are further categorized into gradient-based methods that use the gradients of the neural network [26, 27], perturbation-based methods that perturb the input graphs to obtain important subgraphs or features for the prediction of the model [24, 25, 29, 31], decomposition-based methods that decompose the input graphs to obtain the explanations [20, 33], and surrogate-based methods that try to use simple interpretable surrogate models to explain the more complex model in question [28, 38].

Although these approaches provide valuable information about GNN decision-making, they also come with significant limitations. Gradient-based methods often suffer from high sensitivity to small changes in input data, making explanations unstable and difficult to generalize across different instances. Perturbation-based methods can be computationally expensive and may

introduce artificial changes that do not align with real-world graph structures, leading to misleading explanations. Decomposition-based methods struggle with interpretability when applied to large and complex graphs, as breaking down intricate relationships into meaningful components is inherently challenging. Surrogate-based methods, while useful, face the trade-off between fidelity and simplicity: interpretable surrogate models may fail to capture the true behaviour of the original GNN, leading to approximate and sometimes unreliable explanations [39]. Furthermore, many of these methods lack rigorous evaluation frameworks, making it difficult to assess the reliability and completeness of the generated explanations. Understanding these limitations is crucial for effectively applying existing methods and interpreting their results in practical scenarios, ensuring that the extracted explanations align with the underlying decision-making processes of GNNs.

In this work, we want to systematically evaluate the capabilities and limitations of one of the most popular explanation method, GNNExplainer [24], on the task of graph classification. We investigate the classification capabilities of different GNN models across different structural classes of graphs, as well as their decision making process using GNNExplainer. To this extent, we use GNNExplainer to find the importance of structural node features used in training the models, and then compare them with theoretical and empirical knowledge on what discerns different classes of graphs. GNNExplainer is perturbation based model that stands out from other explainability models for its ability to obtain node feature importance, which is important for this work, since we want to compare our findings with the ones from theory. Other explainability methods used for GNNs provide only the subgraph importance, and thus, they are not suitable for the task at hand.

We study two 'cases' of graph classification: (i) the classification between Erdos-Renyi (ER) and Barabasi–Albert (BA) graphs [2] and (ii) between 0k, 1k, and 2k random graph structures for four existing real-world networks (the Internet, US airport network, human protein interactions, and techno-social web of trust) [40]. We use different GNN architectures, Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Graph Isomorphism Network (GIN), and vary the value of hyperparameters to train classifiers that are able to distinguish between these classes of graphs. During the GNN training process, we utilize graph properties as features and employ GNNExplainer to identify importance of each feature in classification process for each GNN architecture and set of hyperparameters.

Our results show that the results of the GNNExplainer are dependent, to a certain extent, on factors like the type of mode, number of layers, random seed, and embedding size. This suggests that even in relatively simple network classification tasks, we cannot always fully trust which features are driving predictions. The number of layers in the model plays a crucial role, which aligns with theoretical justifications for considering a certain depth in the network. Furthermore, we find that embedding size and random seeds can significantly impact the feature importance given by GNNExplainer. The high variability of feature importance with the change of seed that we observe in this work is particularity alarming and urges to use feature importance found by GNNExplainer with caution.

The paper is organized in the following way. In Section 2, we describe graph types and graph features used in graph classification task. In the same section we explain in general how GNN operate, the GNN architectures that we are using in this work and GNNExplainer. In Section 3, we present our findings. The Section 4 gives discussion results and conclusion of this work.

2. Materials and methods

In this section, we describe the data and the models used. We use the *PyTorch Geometric* python library for building and training the models. (The code can be found at GitHub https://github.com/darjacvetkovic/gnn-xplain-dk.)

2.1. Graphs

Graphs, also known as networks, are mathematical structures comprising nodes (vertices) and edges (links) that establish connections between these nodes. They provide a way for representation of relationships and associations among objects or entities. This representation enables the analysis and modelling of diverse systems in fields such as social sciences, biology, chemistry, computer

science, and more [2]. Formally, graphs are denoted as $G(N, L)$ where N is the number of nodes and L is the number of links (edges), which also determine the size of the graph [1]. One of the ways the graph can be represented is an *adjacency matrix* $\hat{A}$ which elements a_{ij} denote if there is an edge between nodes i and j ($a_{ij} = 1$) or not ($a_{ij} = 0$). The edges between nodes can be directed on undirected, positive or negative, and can also have weights. In this paper, we will work with networks that have undirected, unweighted, and positive edges.

The structure of a graph can be described with various metrics, some of which are more relevant than others—depending of the nature of the graph and the problem that is being investigated [2]. Therefore, we will only mention the most fundamental metrics in this discussion, and the metrics we will use as node features for training the models.

- ***network diameter*** refers to the longest *shortest path* among all pairs of nodes in a graph. In a graph, a *path* is a sequence of edges that connects two nodes, and its *length* corresponds to the number of edges in that sequence. The *shortest path* between two nodes is the path with the minimum length. Therefore, the network diameter represents the maximum length among all the shortest paths in the graph, considering all possible pairs of nodes.
- ***node degree*** is the number of edges that a node has, representing the count of its 'first neighbours' [1]. Therefore, this metric indicates the level of connectivity of a node, and in that sense, how structurally important it is. Formally, it is most often denoted by k_i, where the index i represents a given node. A more detailed information about the structure of the network can be obtained from the ***degree distribution*** $P(k)$—a probability that randomly chosen node has a degree k. The degree distribution is a fundamental property used in network theory to describe the structural characteristics of a network, and it's explicit functional form is crucial for analytic calculation of many other metrics that influence a variety of phenomena occurring in complex networks [40].
- ***nearest neighbour degree*** $k_{nn,i}$ is simply the average degree of node i's neighbours: $k_{nn,i} = \frac{1}{|N(i)|} \sum_{j \in N(i)} k_j$.
- ***clustering coefficient*** C_i measures the local edge density which indicates the level of connectivity among the neighbours of node i. It can be calculated using a formula $C_i = 2L_i/k_i(k_i - 1)$, where L_i is the number of edges between the neighbours of i [41]. The value of C_i ranges between 0 and 1. A higher value signifies a more interconnected neighbourhood around node i, where $C_i = 1$ indicates that all neighbours are connected to each other, while $C_i = 0$ indicates no mutual connections among the neighbours. In this sense, the clustering coefficient C_i represents a probability that two neighbours of i are connected. To determine the clustering coefficient for the entire network (graph), we calculate the mean of all C_i values: $\langle C \rangle = \sum_{i=1}^{N} C_i/N$. This average represents the probability that two neighbours of a randomly selected node in the network are connected. Along with the nearest neighbour degree, it contributes to a more comprehensive view of the local graph topology.
- ***core number*** provides a more nuanced perspective on the local graph topology. A k-core is a maximal subgraph that contains nodes of degree k or more, and the core number of a node is the largest value k of a k-core containing that node. Core numbers can be important to discerning the significance of nodes in the network's structure, as well as it's connectivity.

We use Graph Geodesic Distance (GGD) [42] to measure the structural difference between two graphs. GGD is a spectral distance metric that measures dissimilarity between two graphs by comparing their spectral properties on a Riemannian manifold. It represents graphs through their modified positive-definite Laplacian versions and computes the geodesic distance between these matrices using generalized eigenvalues. GGD equals zero when two graphs are identical or isomorphic. Because this measure is derived from the affine-invariant Riemannian metric, it is not limited above, meaning that the distance can grow arbitrarily large as the spectral properties of the two graphs diverge.

2.2. Random graphs

2.2.1. Erdős–Rényi and Barabási–Albert models

We will begin our investigation with classification between Erdős–Rényi graphs [43] and Barabási–Albert graphs [44].

The Erdős–Rényi $G(N, M)$ model defines a random graph as a graph G containing a fixed number of N vertices, with M links randomly placed between them [43]. This results in a binomial degree distribution or Poisson distribution for large graphs. On the other hand, Barabási–Albert model uses the mechanisms of network growth and preferential attachment to generate graphs with degree distributions that follow a scale-free power law distribution, which is the case in many real world networks [44]. This means that, in such networks, a small number of nodes have a significantly higher number of connections compared to the majority of nodes, resulting in a heavy-tailed distribution.

As such, the resulting degree distributions for Erdős–Rényi (ER) graphs and Barabási–Albert (BA) graphs are fundamentally very different, meaning the graph structures of ER and BA graphs differ significantly from each other. Consequently, comparing the degree distributions allows for straightforward classification of graphs into their respective classes. However, the focus of this study is to explore how 'black-box' machine learning models perform when tasked with classifying these disparate graph classes. For this goal and for simplicity reasons, we generate ER and BA graphs with fixed number of nodes and average links per node: $N = 300$, $m = 3$.

2.2.2. *dk*-random graphs

dk−random graphs [40, 45], are graphs generated from already existing networks by rewiring them in such a manner that specific topological properties of the original network are preserved. Specifically, 0k graphs as graphs that have the same number of nodes and edges as the original graph. 0k graphs are ER graphs. 1k graphs have the same number of nodes and edges, and the same degree sequence as the original graphs. They differ from 0k graphs only by degree distribution. 2k graphs have the same number of nodes, edges, degree sequence, and joint degree matrix. Unlike 1k graphs, the degree correlations in 2k graphs are preserved and the same as in the original network. The main idea behind *dk* graphs is that for a high enough d the whole network can be reconstructed. In fact, in their publication [40], the authors demonstrated that many significant local and global structural properties of certain real-world networks can be accurately reproduced by *dk*-random graphs derived from these networks. Leveraging this insight as a baseline for our analysis and interpretation, we aim to investigate and interpret the outcomes obtained through our approach.

There are two main approaches to generate *dk*-random graphs based on a given real network G [45–48]. We use the RandNetGen (https://github.com/polcolomer/RandNetGen) GitHub repository. This approach involves swapping pairs of edges while preserving the desired *dk*-distribution at each swap.

In this study, using RandNetGen we generate 100 random graphs for each $0k$, $1k$, and $2k$ rewiring levels, for four distinct real networks: the Internet (INTERNET) [49], US airport network (AIRPORT) [50], human protein interactions (PPI) [51], and techno-social web of trust (PGP) [52]. This yields 300 *dk* random graphs for each real world network, and the models will learn to classify between these three *dk* classes, for each network separately.

The overview of these real world networks is given in Table 1.

2.3. Graph neural networks

The unique challenge of applying machine learning to network or graph data arises from the complex and non-Euclidean nature of graphs, where the relationships between nodes are crucial. Traditional neural networks and other machine learning models designed for datasets with independent data points (e.g. tabular data) are therefore ill-suited for learning and prediction tasks on graph-structured data. Euclidean mathematical operations, typically applicable to most data, are not viable due to the non-Euclidean structure of graphs [53]. Furthermore, graphs can have different adjacency matrix representations for the same underlying structure, and adding or removing nodes results in entirely new graphs. As a consequence, even convolutional neural networks, which are

Table 1 Table with the information about the number of nodes (N) and the number of (edges) of the networks we use in our study.

Dataset	N	M
AIRPORT	500	2980
INTERNET	20 906	42 994
PGP	10 680	24 316
PPI	4099	13 355

typically applied to images by exploiting fixed local pixel neighbourhoods, are not naturally suited for graph-structured data. With all of that in mind, a machine learning method on graphs must be such that it (i) includes the connections between the nodes, that is the structure of the graph, (ii) is independent of the graph size, (iii) is permutation invariant. *Graph neural networks* (GNNs) offer a solution to these challenges, as they are deep learning models specifically designed for learning on graph structures [18, 54]. The key idea behind GNNs is getting the *node embeddings* depending on their local neighbourhoods, and then using them for downstream predictions. GNNs achieve this by employing a message passing framework known as 'neural message passing'. Given an input graph represented as $G = (N, V)$ and a set of node features denoted as $X \in \mathbb{R}^{d \times |V|}$, GNNs have the ability to generate node embeddings, denoted as z_u for all nodes u in the graph V. Hence, these node embeddings serve as vector representations, capturing crucial information derived from both node features and the local graph structure. The fundamental mechanism of graph neural networks can be described as follows: at each layer of the GNN, the embedding for each node is calculated by aggregating the messages from neighbouring nodes, including the node itself, and subsequently applying an update operation. This iterative process generates new embeddings, which serve as messages for the next layer of the GNN. Each layer represents a 'neighbourhood hop', meaning that in an n-layer GNN, nodes receive information from their nth order neighbourhood (e.g. in a two layer GNN the embedding for a node will contain information from it's neighbours and it's neighbour's neighbours). Notably, this mechanism enables each node to define its own *computation graph*, effectively creating a personalized neural network architecture based on its local neighbourhood. However, all these individual neural networks within a layer share the same set of parameters. The following mathematical formulation, elucidates the process [55]:

$$\vec{h}_u^{(k+1)} = UPDATE^{(k)}\left(\vec{h}_u^{(k)}, AGGREGATE^{(k)}\left(\{\vec{h}_v^{(k)}, v \in \mathcal{N}(u)\}\right)\right)$$

That is

$$\vec{h}_u^{(k+1)} = UPDATE^{(k)}\left(\vec{h}_u^{(k)}, \vec{m}_{\mathcal{N}(u)}^{(k)}\right)$$

where $\vec{h}_u^{(k)}$ is the embedding of a node u in layer k, and $\vec{m}_{\mathcal{N}(u)}^{(k)}$ is the message from the neighbourhood of node u $\mathcal{N}(u)$. Functions $UPDATE^{(k)}$ i $AGGREGATE^{(k)}$ are essentially arbitrary differentiable functions, but such that $AGGREGATE^{(k)}$ is permutation invariant (because the order of message aggregation from nodes shouldn't matter), and $UPDATE^{(k)}$ is a non-linear activation function. The parameters for both of the functions are learned during the training process. Different GNN models essentially differ only by these two functions [19].

In this paper, we use three different GNN models, representing different architectural approaches within the graph neural network framework: the Graph Convolutional Network (GCN [56]), Graph Attention Network (GAT [57, 58]), and Graph Isomorphism Network (GIN, [59]). Each of these model variants introduces unique design choices, such as convolutional layers, attention mechanisms, or isomorphism-based learning, offering a range of options for tackling graph-related prediction tasks. We provide a brief overview and descriptions of each model below:

- **GCN** model operates by computing a simple weighted sum of the features of neighbouring nodes (including the current node), and then applying a *ReLU* activation function

$$\vec{h}_u^{(k)} = ReLU\left(W^{(k)} \frac{\sum_{v \in \mathcal{N}(u) \cup \{u\}} \vec{h}_v^{(k-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$$

  Here, $W^{(k)}$ is the weight matrix for layer k and it's learned and optimized during the training process.
- **GAT** model incorporates attention mechanisms to capture important dependencies between nodes in a graph [57]. Unlike the previous GCN model, GAT assigns different *attention weights* to different neighbours of a node, allowing for more expressive modelling of the graph structure. In GAT, the node embeddings are computed by combining the features of neighbouring nodes with attention weights. Each node calculates its attention weights by attending over its neighbours' features, and the importance of each neighbour is determined based on its relevance to the current node. These attention weights are then used to compute a weighted sum of the neighbour features, resulting in the node embedding. The formula for the embedding can be written as:

$$\vec{h}_u^{(k)} = ReLU\left(\alpha_{uu}^{(k)} W^{(k)} \vec{h}_u^{(k-1)} + \sum_{v \in \mathcal{N}(u)} \alpha_{uv}^{(k)} W^{(k)} \vec{h}_v^{(k-1)}\right)$$

  where $\vec{h}_u^{(k)}$ represents the embedding of node u in layer k, $\mathcal{N}(u)$ denotes the set of neighbours of node u, $\alpha_{uv}^{(k)}$ represents the attention weight between nodes u and v in layer k ($\alpha_{uu}^{(k)}$ being 'self-attention' for node u), $\mathbf{W}^{(k)}$ is a weight matrix specific to layer k, and $\vec{h}_v^{(k-1)}$ denotes the embedding of node v in the previous layer $k-1$. The ReLU activation function is applied element-wise to introduce non-linearity. Both the weight matrix and the attention weights are learned during training. The attention weights are computed using an attention mechanism that measures the relevance or similarity between the features of node u and node v. In the original paper [57] the attention weights are defined as:

$$\alpha_{uv} = \frac{exp(a^\intercal [W\vec{h_u} \oplus W\vec{h_v}])}{\sum_{v' \in \mathcal{N}(u)} exp(a^\intercal [W\vec{h_u} \oplus W\vec{h_{v'}}])}$$

  where a is a trainable attention vector, and the operator $\oplus$ denotes concatenation. However, the authors in [58] point to the problem of 'static' attention, and propose a solution in the form of 'dynamic' attention which is implemented in the appropriate *PyTorch Geometric* modules, and we use this implementation in our present work. In short and simple terms, the proposed solution enables the model to capture more complex and nuanced patterns in the graph, potentially leading to improved performance.
- **GIN** model incorporates the framework of the Weisfeiler-Lehman (WL) isomorphism test [59, 60] to generate the highly expressive node embeddings and a model with strong discriminative power. The expressivity of a GNN is largely influenced by its aggregation function. Specifically, some aggregation functions may have limitations in distinguishing between different neighbourhoods—so, if we think of aggregations as functions on *multisets* of node neighbourhoods, to achieve maximum expressivity the aggregation function needs to be *injective*. In the GIN model, the authors leverage the universal approximation theorem [61] to construct an injective framework for the aggregation scheme. This theorem states that every function can be approximated by a deep enough neural network. Consequently, multi-layer perceptrons (MLPs) are used to model and learn the update and aggregation functions. The node embedding formula for the GIN model is as follows:

$$\vec{h}_u^{(k)} = MLP^{(k)}\left((1 + \epsilon^{(k)})\vec{h}_u^{(k-1)} + \sum_{v \in \mathcal{N}(u)} \vec{h}_v^{(k-1)}\right)$$

Here, $\vec{h}_u^{(k)}$ represents the node embedding of node u at layer k, $\epsilon^{(k)}$ is a learnable parameter that controls the self-loop contribution. The aggregation and the update steps are performed using a multi-layer perceptron $MLP^{(k)}$. The MLP applies non-linear transformations to the combined node features and self-loop contribution, allowing the GIN model to capture complex relationships and patterns in the graph, and is also learned during training.

In the task of graph classification, it is necessary to obtain the embedding of an entire graph by pooling the embeddings of all its nodes. There are several approaches to perform this pooling operation, but for the purposes of this paper, we adopt the method known as *global mean pooling*—it simply averages the embeddings of all the nodes to get the embedding of the entire graph. Additionally, the graph embeddings from each layer are concatenated, rather than using the embeddings only from the last layer, since features from the first layers can sometimes generalize better [59]. In this way, more of the structural information from the graph is encoded for the final prediction of the model.

2.4. GNNExplainer

GNNExplainer [24] is a model-agnostic technique designed to provide explainability for any GNN based model on any graph-based machine learning task. The main idea behind GNNExplainer is to learn 'importance scores' for nodes and edges, indicating their significance for the GNN model's predictions. Specifically, GNNExplainer learns a *graph mask* and a *feature mask*, which jointly specify the subgraph that maximizes the mutual information with the model's prediction. This mutual information quantifies the change in the probability of the prediction when a part of the computation graph is masked. Simply put: if there is a significant difference in the predictions for a graph with the perturbed feature, and the unperturbed graph, then that feature is important. In mathematical terms, the goal of GNNExplainer is to optimize the following function (a modified cross entropy loss) using gradient descent:

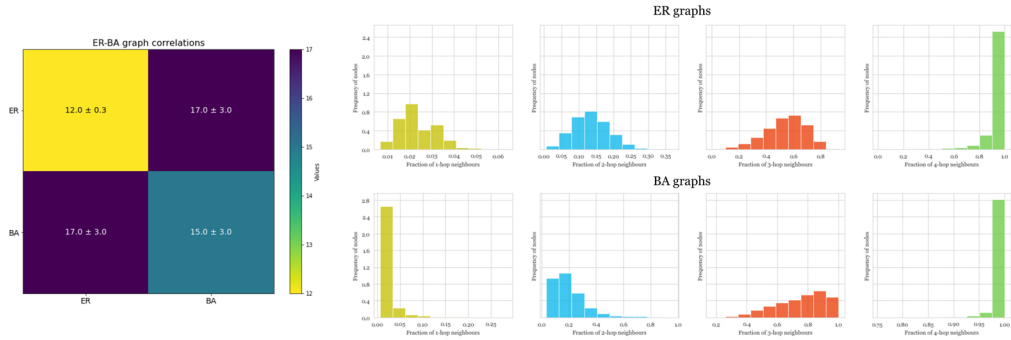
$$\min_{M} -\sum_{c=1}^{C} \mathbb{1}[y=c] log P_{\Phi}(Y=y|G=A_c \odot \sigma(M), X=X_c)$$

Here, M represents a mask $M \in \mathbb{R}^{n \times n}$, where n is the number of nodes. y denotes the predicted label, c is the true label, $P_{\Phi}(Y=y|G=A_c \odot \sigma(M), X=X_c)$ denotes the conditional probability of the model predicting y given the masked computation graph ($A_c \odot \sigma(M)$, with $\odot$ indicating element-wise multiplication) and a subset of features X_c [24].

In this paper we use GNNExplainer to identify the most important features for classifying between (i) generated ER and BA graphs, and (ii) 0k, 1k, and 2k graphs from four real world networks. For each graph in a specific dataset, we average the feature importances, ranging from 0 (not important) to 1 (important), calculated by GNNExplainer across all nodes—getting the mean importance of each node feature in predicting the class of the graph. We then plot the distributions of these mean feature importances for all graphs and each class in the dataset.

3. Results

In order to train the models, we initialize each of the graphs in the dataset with 4-dimensional node feature vectors. Specifically, each node in the graph has a starting feature vector that describes it's structural information in the graph, consisting of it's degree, clustering coefficient, average neighbour degree, and core number. As discussed above, these node features provide valuable insights into the structural characteristics of individual nodes, and should contribute to the models' capacity to discern and interpret intricate relationships within the graph. Additionally, these metrics have been proven [40] to distinguish different graph structures considered in this paper, and these insights are used as a comparison baseline with the importances given by GNNExplainer. The idea is that during message passing, nodes share mutual information about each other, and inour case,



(a) (b)

Figure 1 (a) Geodesic distance [62] as a measure of similarity between ER and BA graphs in the dataset. (b) The distributions of neighbours for different number of layers $n = 1, 2, 3, 4$ for ER and BA graphs, respectively.

this is the information of the structural role of the nodes. Then, utilizing the GNNExplainer, we extract the importances that these features have in training the GNN models, and compare them from the knowledge we have from previous research and theory on the classes of graphs that we use.

3.1. Erdős-Rényi and Barabási–Albert graphs

We first explore feature importance in classification between ER and BA graphs. All models reached 100% accuracy and a steady minimum of the loss function, see Supplementary Fig. S1. As explained in Section 2.2, the main difference between these two types of graphs is degree distribution [2]. While ER graphs have the Poisson degree distribution, the distribution of BA graphs is broad and has power-law behaviour [2]. Otherwise, both types of networks are uncorrelated and unclustered, see Supplementary Fig. S2. GGD shown in Fig. 1a shows that we observe the highest difference between networks of different type. The distribution of reachable neighbours, shown in Fig. 1b, shows that reachability for lower number of hops is higher in ER graphs, than in BA, which is consistent with the heterogeneous distribution of node degrees in BA graphs. While reachability is changing gradually for ER graphs for 1, 2, and 3 hops, the reachability of BA graphs has very clear shift between 2 and 3 hops. The diameter of these networks is between 6 and 9 for ER graphs and between 5 and 6 for BA graphs, thus it is not surprising that most of the nodes are reachable from every node for $n = 4$ hops.

The main difference between ER and BA graphs is degree distribution, thus we would expect degree to be the most important feature for differentiation between these two type of graphs. Figure 2 shows distribution of feature importance in classification of ER and BA graphs for GCN, GAT, and GIN graph neural networks for different number of layers. There are few striking observations. Degree is not the feature with the highest importance for GCN model. The average degree of nearest neighbours and core number are the most important features for ER graphs for all except the model with $n = 2$ layers. For GAT and GCN models, degree is either equally important, or lower importance than k_{nn} or core number for ER graphs. Similar situations is for BA graphs. Degree has equal of lower importance than average degree of nearest neighbours and core number. The only exception is GCN model with $n = 4$ layers. We observe the only agreement with the theory in the case of clustering coefficient. The importance of c is low for all models. This is expected, since both graph types have low values of clustering coefficient. It follows from Fig. 2 that feature importance depends on number of layers and model type.

The complexity of the graph class also influences width of the distribution of feature importance. The variability of feature importance from graph to graph is higher for BA graphs. Variability of feature importance between the same number of layers in the model n is more explicit for BA graphs for all three models, suggesting that complexity of graph type has a certain influence. The GIN model 'produces' the highest variability of feature importance for both graph types and number

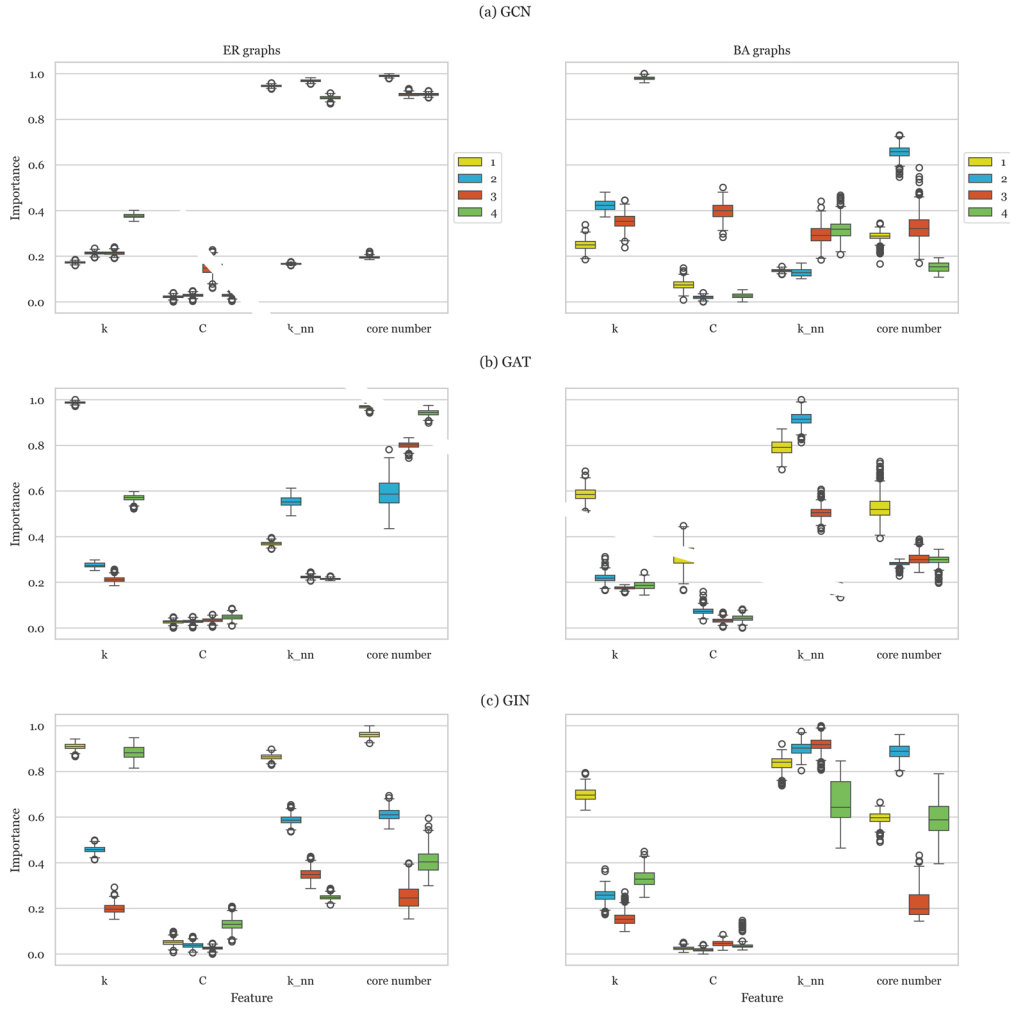


Figure 2 Classification between ER and BA graphs—model feature importances for different number of layers $n = 1, 2, 3, 4$. Panels show results for (a) GCN, (b) GAT, and (c) GIN models. The model output corresponds to the predicted class of an input graph (ER or BA). Each subplot presents boxplots of feature importance values for either ER graphs (left) or BA graphs (right). The x-axis corresponds to input features; for each feature, four boxplots represent models with different number of layers ($n = 1, 2, 3, 4$), indicated by color. Boxplots summarize the distribution of feature importance scores across all graphs of a given class, showing the median, interquartile range, whiskers, and outliers. Wider boxes indicate greater variability in feature importance values across graphs.

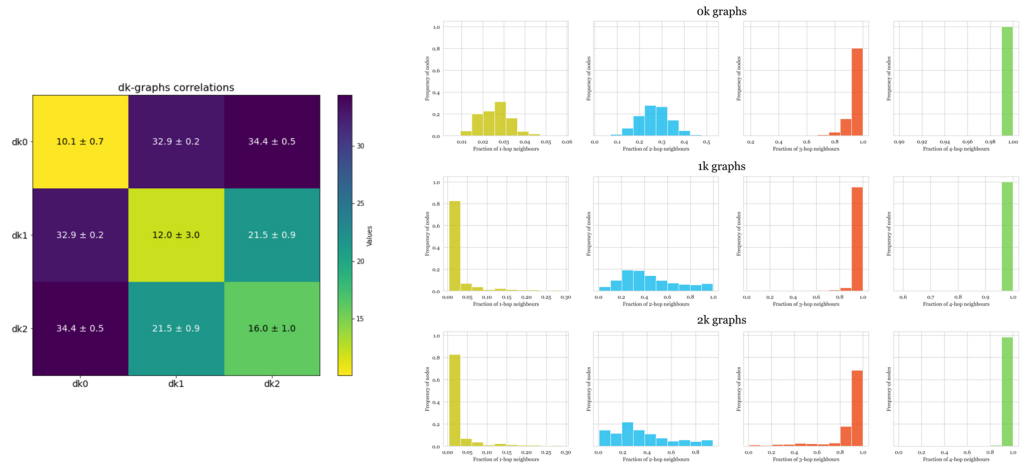
of layers. This could be explained by the higher expressiveness of the GIN model, which should be more sensitive to structural and feature distributions in the graph itself. In general, variability of feature importance with number of layers seems larger for ER graphs than for BA graphs, although the width of feature importance distribution is higher for BA networks.

We observe similar patterns as described in previous paragraphs for different seed value and size of the embedding vector, see Fig. 3. However, we observe significant changes in feature importance when we vary model hyperparameters. For instance, if we change the dimension of the embedding vectors and the value of random seed, the feature importance obtained through GNNExplainer change, see Fig. 3.

Figure 3 Classification between ER and BA graphs—comparison of mean feature importances and their standard deviations between models with 16 dimensional embedding size and with models initialized with a different random seeds. The results are shown for each model variation (CLEAN—the original model, 16EMB—the model with embedding_size = 16 and SEED—the model initialized with different random seed) and number of layers $n = 1, 2, 3, 4$ for (a) GCN, (b) GAT, (c) GIN models.

The feature importance changes varies with GNN model type and the value of hyperparameters. The clustering coefficient is the only feature whose importance varies little with the change of model and value of hyperparameters.

Feature importance in distinguishing ER and BA graphs does not align with theoretical expectations. The degree is not the dominant discriminating feature. The importance of a feature depends strongly on the model type, number of layers, and hyperparameters. This suggest that GNN feature explanations are not stable nor directly aligned with theoretical graph differences.



(a) (b)

Figure 4 (a) Geodesic distance [62] as a measure of similarity between 0k, 1k, and 2k graphs from the AIRPORT network. (b) The distributions of neighbours for different number of layers $n = 1, 2, 3, 4$ for 0k, 1k, and 2k graphs from the AIRPORT network.

3.2. dk-randomized graphs

ER and BA graph models were explored in detail, thus they represent a good golden standard for testing different classification approaches on them. However, their relative simplicity may lead to oversimplified conclusions. For these reasons, we apply the same approach described in previous section to dk-graphs. Similar to GNN for classification between ER and BA graphs, GNNs for classification between dk graphs all reach accuracy 1 (see Supplementary Figs S3, S5, S9, and S13). The GGD shown in Fig. 4a indicates that there are clear differences between these three classes of graphs. We see that reachability of nodes has almost the same distribution for 1k and 2k graphs, while most of the nodes are reached for $n = 3$ layers in all three classes of graphs. Reachability in dk graphs for other three networks is slightly different, see Supplementary Figs S6, S10, and S14. While reachability is 100% for the AIRPORT network for $0k$,$1k$ and $2k$ graphs and $n = 4$, we observe high reachability only for INTERNET and PPI $1k$ and $2k$ graphs for $n = 4$.

Reachability indicates clear differences in structural properties between dk graphs obtained from AIRPORT, INTERNET, PGP and PPI networks. $0k$ graphs for all four networks are uncorrelated and unclustered, see Supplementary Figs S4, S7, S11, and S15. Clustering coefficient is relatively low for $1k$ and $2k$ graphs for all four networks. $1k$ and $2k$ graphs for are disassortative for INTERNET, weekly disassortative for AIRPORT and PPI, and weekly assortative for PGP network. All four networks have broad degree distributions, Supplementary Figs S4, S7, S11, and S15.

Comparison of Figs 2 and 5 shows that there are many more variations of feature importance for different n. Clustering coefficient has higher importance, but its variability is lower compared to the importance of other features. $0k$ graphs are ER graphs generated in different way [40]. However, we observe higher variability of node features with number of layers for GCN model. The variability of node features with n is similar for GAT and GIN models. In the case of the GCN, we observe clear reliance on the k_{nn} and core number for $0k$ and $2k$ graphs and higher number of layers ($n = 3, 4$), while importance of node degree is lower for $2k$ graphs. For $1k$ graphs we observe increase of node degree importance, while k_{nn} and core number and k_{nn} importance decreases with increase of number of layers. The feature importance has the smallest variability with number of layers for $2k$ graphs. The wider distribution of feature importance and existence of outliers is observed for networks with more complex topology. Unlike ER and BA graphs, AIRPORT $2k$ graphs have nontrivial value of clustering coefficient which is reflected in higher importance c for these networks. This is particularly evident for the GCN the GAT models.

Figure 5 Classification between 0k,1k and 2k graphs of the AIRPORT network—model feature importances for different number of layers $n = 1, 2, 3, 4$. the model output corresponds to the predicted class of an input graph (0k,1k or 2k). Each subplot presents boxplots of feature importance values for either 0k graphs (left), 1k graphs (middle) or 2k graphs (right). the x-axis corresponds to input features; for each feature, four boxplots represent models with different number of layers ($n = 1, 2, 3, 4$), indicated by color. Boxplots summarize the distribution of feature importance scores across all graphs of a given class, showing the median, interquartile range, whiskers, and outliers. Wider boxes indicate greater variability in feature importance values across graphs.

Unlike the GCN and GAT models, GIN exhibits a visibly different pattern for the importance of k and k_{nn}. We observe for all three classes of graphs the increase of importance of k_{nn} is followed by decrease of importance of degree, except in the case of 0k and $n = 3$. For all classes networks we observe the degree importance is always important for $n = 1$, but for $n > 1$ this importance significantly decreases for 1k and 2k graphs, except in the case of 0k and $n = 3$. The core number has higher importance for 0k graphs than for 1k and 2k graphs. What is surprising is that clustering coefficient has almost zero importance for all three classes of graphs which is stable over layers.

In summary, for 0k, as ER graphs, GCN model has the most stable feature importance while, GAT and GIN exhibit similar variability of node features. For 1k graphs we observe anti-correlations between importance of degree and k_{nn} for all three models. This is also true for 2k graphs and GCN and GIN models.

Figure 6 and Supplementary Figs S8, S12, and S16 show that change of SEED value or the size of embedding vector causes the change of feature importance, as in the case of classification of ER and BA graphs. These findings further confirm that feature importance found with GNNExplainer do not generalize over different models.

What we find most surprising is that when comparing the results for the feature importance between all networks (INTERNET, PPI, PGP and AIRPORT), there seems to be nontrivial amount similarities, Fig. 7 and Supplementary Figs S17, S18, and S19, qualitatively speaking. This is unexpected because these networks are very different from one another, and we would expect that different features are more important for their classification in most cases. Specifically, looking at Fig. 7 we can obtain three 'levels' of information:

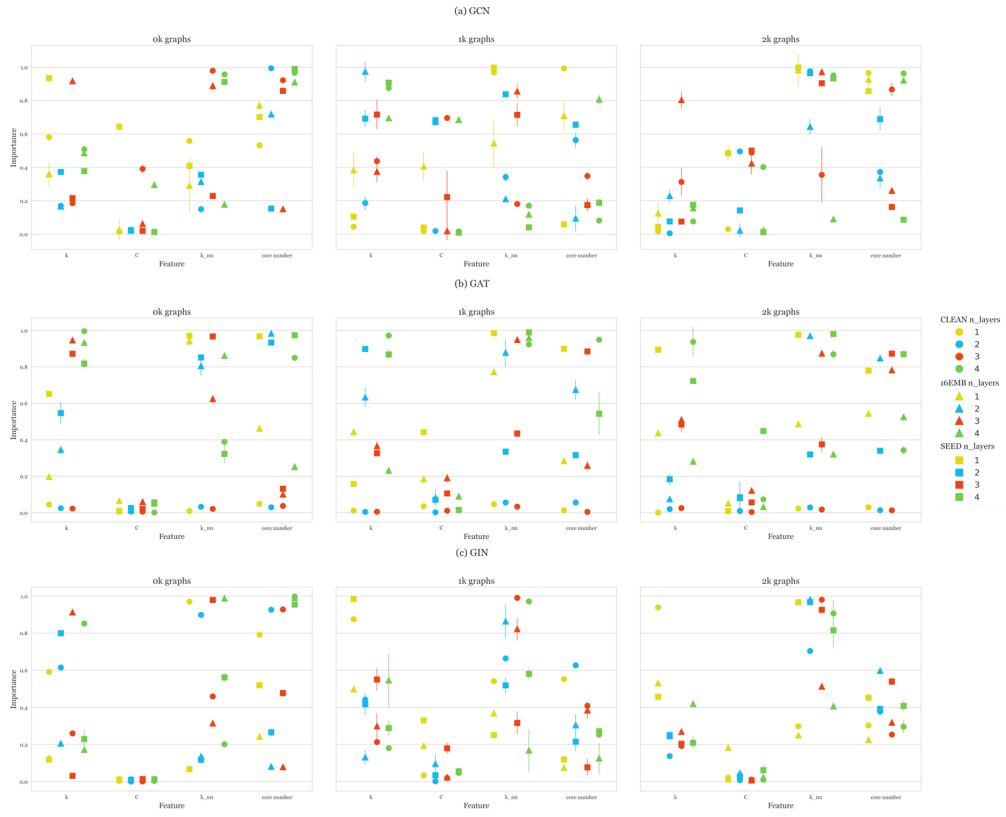


Figure 6 Classification between $0k, 1k$ and $2k$ graphs for the AIRPORT network—comparison of mean feature importances between models with 16 dimensional embedding size and with models initialized with a different random seeds. The results are shown for each model variation (CLEAN—the original model, 16EMB—the model with embedding_size = 16 and SEED—the model initialized with different random seed) and number of layers $n = 1, 2, 3, 4$ for (a) GCN, (b) GAT, (c) GIN models.

- mean differences of feature importances for a specific model for all networks—e.g. a dot represents the mean feature importance for GCN classification across all networks, while the error bars represent their standard deviations. When the error bar is small, that means that there is a small difference between that features importance between classifying different networks. In other words, predicting classes with the GCN model in this case yields a similar importance of that specific feature for all networks: AIRPORT, INTERNET, PPI and PGP. This is the case, e.g. in the first row (when using $n = 1$) in Fig. 7 for $0k$ graphs and the GAT model and all features, while for $1k$ that is the case for the GCN model. For $2k$ graphs in this case, we see that behaviour for all features and both GCN and GAT. Of course, this is the case sometimes only for specific features (e.g. for the clustering coefficient and the core number for the GIN model, represented by the green crosses in the case of $n = 1$).
- differences of the of feature importances between different models (represented by different colours and markers)—e.g. for $n = 4$, for the core number all models have similar importance. As in the previous point, ofcourse, there are also similarities between different pairs of models.
- how bothof these points differ across different number of layers n, which is represented by different rows. Again, we observe, overall, that the clustering coefficient is not important across all networks and number of layers n.

Figure 7 Classification between $0k, 1k$ and $2k$ graphs for the all networks (AIRPORT, INTERNET, PPI and PGP)—comparison of mean feature importances. Each point represents the mean importance for the feature across all networks for a specific model, while the error bars represent their standard deviation. This way we can see how the importances differ between both the different networks and different models. Smaller error bars show that there is a small difference between the importances of features for a specific model's prediction for different networks, and vice versa.

4. Conclusions and discussion

Most of the AI models often operate as black boxes, making their decisions difficult to interpret, which can hinder trust, especially in high-stakes domains like healthcare, finance, or autonomous systems. Explainability addresses this by providing human-understandable reasons for a model's predictions, thereby enhancing transparency and user trust. Explanations of how model's makes decisions allow stakeholders to verify that these decisions are fair and justified. Furthermore, it helps developers detect and correct errors or biases in the model's reasoning. Explainability is crucial for development of trustworthy AI, bridging the gap between complex model computations and human comprehension.

Explaining the predictions of GNNs poses unique challenges due to the non-Euclidean nature of graph data. Attributed graphs carry information in both node features and the graph's connectivity. GNN architectures, such as GCN, GAT and GIN, vary in how they propagate and aggregate information, meaning explainability methods must be adaptable to different model mechanisms.

GNNs perform tasks at different scales. Explanation of local prediction, for instance single node label, may require focusing on a node's neighbourhood, whereas explaining a global graph classification involves identifying patterns in the entire graph. These factors make GNN explainability non-trivial. A variety of methods have been proposed to tackle these challenges. Understanding their scope and limits is thus of great importance to achieving GNN explainability.

For graph classification tasks ranging from molecular graphs to social networks, explainability is a key component for deploying GNNs in practice, where interpretability, trust, and accountability are as important as accuracy. However, evaluation of GNNs is a challenging task. Evaluating GNN explanations is difficult because most graph datasets do not have ground-truth explanations for why a certain label is true. There is also the challenge of making explanations useful to end-users. The explanations should ideally incorporate domain knowledge to be truly insightful.

In this work we advance our understanding of scopes and limits of GNNExplainer in the task of classification of simple graphs. We use domain knowledge from complex networks theory to explore explanations given by GNNExplainer compared to domain knowledge. Furthermore, we examine the stability of provided explanations against the hyperparameter change.

It follows from domain knowledge that main difference between ER and BA graphs is degree distribution. ER graphs have Poasonian distribution, while other properties such as clustering coefficient, average degree of nearest neighbour or core number should not be relevant for decision making process. However, we see from Fig. 2 that average neighbour degree and core number have similar or higher importance in decision making process. Moreover, the feature importance varies with hyperparameter change, see Figs 2 and 3. The only constant is the low importance of clustering coefficient. The variability is even more striking for dk graphs. Again, based on domain knowledge the main difference between these graphs is based on degree distribution and joint degree matrix, which is directly related to k_{nn} [40]. One would expect that these two features have higher importance than clustering and core number. However, we see that clustering coefficient while predominantly has low importance, sometimes this importance can be higher than one of k and k_{nn} for some number of layers and values of hyperparameters, Figs 5 and 6.

GNNExplainer was designed to identify a compact subgraph structure and a subset of node and/or edge features that are most influential for that prediction. The synthetic graphs used in this work do not have specific subgraphs that contain sufficient information for the model's decision, thus we focused on node structural features. Importance of node features should help us to interpret the predictability of GNN, to debug and improve the model and to build trust. For all of this, we need stable importance of node features that do not vary too much with values of hyperparameters. Clearly, feature importance obtained from GNNExplainer does not correlate with domain knowledge even in the case of classification of very simple graphs. The variability of feature importance with change in number of layers, size of embedding vector and value of seed is high for all three models, suggesting that we can not make any universal conclusions about the node feature importance.

While certain variability is expected for different values of number of layers and size of embedding vector, the variability with the value of seed, which in some cases can be quite extreme, should not occur. High variability of feature importance with seed shown in Figs 3 and 6 further proofs GNNExplainer instability. Different seed value usually leads to different local minima found by the model and GNNExplainer. Model is equally accurate regardless of the seed value, but GNNExplainer finds that different features are important for the model successful prediction. The difference can be caused by two reasons, or their combination. Either model finds another minima where features have different importance, further showing that we cannot use feature importance to derive general conclusions. Or, GNNExplainer finds different minima and thus different feature importance, which further proves its instability.

The selection of features is also arbitrary to a certain extent. Furthermore, the values of features can be noisy. However, some experiments show that even in these cases one can obtain the model that classifies the classes of networks correctly with sufficient accuracy. GNNExplainer will produce the importance of features for these models which will be relevant for that specific model, but cannot be used for making any general conclusion about the given phenomena.

Additionally, specific to graph classification task is the concept of pooling, or getting the feature representation of a whole graph, which can also be done in a variety of ways as long as it is permutation invariant. Specifically, pooling can be done by summing, averaging, getting the maximum of each feature, or even using more sophisticated methods. On top of this, one can, e.g. stack the pooled features from each layer as we did in this work and use this stacked feature vector for predicting the graph class, or make similar architectural choices if they are considered suitable for the problem at hand. We have not tested the different choices for the pooling methods, but with the intuition gained from these results, we assume that this could also add additional variability to the results.

Thus, although these behaviours make sense in the context of how neural networks and the learning process works, the observed variability shows that we cannot use node features obtained from GNNExplainer to make general conclusions about the features that enable us to distinguish between different classes of networks, or about the classification process in general. We can, to a certain extent, understand what are the important features for specific model, defined by architecture and the number and value of hyperparameters. However, GNNExplainer does not provide universal explanation, not even in the case of specific model architecture, when it comes to classification of networks without characteristic subgraphs.

Acknowledgements

D.C. and M.M.D. acknowledge funding provided by the Institute of Physics Belgrade, Serbia, through the grant by the Ministry of Education, Science, and Technological Development of the Republic of Serbia. Model training and analysis, and datasets generation were performed on the PARADOX-IV supercomputing facility at the Scientific Computing Laboratory, National Center of Excellence for the Study of Complex Systems, Institute of Physics Belgrade.

Supplementary material

Supplementary material is available at *COMNET Journal* online.

Conflicts of interest

None declared.

Funding

This work was financially supported by the Science Fund of the Republic of Serbia, Prizma programm 7416, Topology-derived methods for the analysis of collective trust dynamics—CTRUST.

Data availability

The data that support the findings of this study are openly available at GitHub https://github.com/darjacvetkovic/gnn-xplain-dk.

References

1. Barabási AL, Pósfai M. *Network Science*. Cambridge: Cambridge University Press; 2016. http://barabasi.com/networksciencebook/
2. Boccaletti S, Latora V, Moreno Y *et al.* Complex networks: structure and dynamics. *Phys Rep* 2006;**424**:175–308. https://www.sciencedirect.com/science/article/pii/S037015730500462X
3. Qi D, Majda AJ. Using machine learning to predict extreme events in complex systems. *Proc Natl Acad Sci USA* 2020;**117**:52–9.
4. Samoaa P, Samoaa H. *From Trees to Graphs: Advancing Regression Analysis through Model-Centric AI, Data-Centric AI, and Active Learning*. Sweden: Chalmers Tekniska Hogskola; 2024.
5. Veličković P. Everything is connected: graph neural networks. *Curr Opin Struct Biol* 2023;**79**:102538.

6. Cvetković D, Dankulov MM, Bogojević A *et al.* Enhancing Hansen solubility predictions with molecular and graph-based approaches. *Chemometrics Intelligent Lab Syst* 2024;**251**:105168.
7. Wieder O, Kohlbacher S, Kuenemann M *et al.* A compact review of molecular property prediction with graph neural networks. *Drug Discov Today Technol* 2020;**37**:1–12.
8. Phan HT, Nguyen NT, Hwang D. Fake news detection: a survey of graph neural network methods. *Appl Soft Comput* 2023;**139**:110235. https://doi.org/10.1016/j.asoc.2023.110235
9. Ketkar NS, Holder LB, Cook DJ. Empirical comparison of graph classification algorithms. In: *IEEE Symposium on Computational Intelligence and Data Mining*. IEEE; 2009, 259–266. https://doi.org/10.1109/CIDM.2009.4938658
10. Riesen K, Bunke H. Graph classification by means of Lipschitz embedding. *IEEE Trans Syst Man Cybern B Cybern* 2009;**39**:1472–83. https://doi.org/10.1109/TSMCB.2009.2019264
11. Tsuda K, Saigo H. *Graph Classification*. Boston, MA: Springer US; 2010, 337–363. https://doi.org/10.1007/978-1-4419-6045-0_11
12. Kong X, Fan W, Yu PS. Dual active feature and sample selection for graph classification. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM; 2011, 654–62. https://doi.org/10.1145/2020408.2020511
13. Li G, Semerci M, Yener B *et al.* Effective graph classification based on topological and label attributes. *Stat Anal Anal* 2012;**5**:265–83. https://doi.org/10.1002/sam.11153
14. Schmidt M, Palm G, Schwenker F. Spectral graph features for the classification of graphs and graph sequences. *Comput Stat* 2012;**29**:65–80. https://doi.org/10.1007/s00180-012-0381-6
15. Zhang M, Cui Z, Neumann M *et al.* An end-to-end deep learning architecture for graph classification. *Proc AAAI Conf Artif Intell* 2018;**32**:4438–45. https://doi.org/10.1609/aaai.v32i1.11782
16. Motallebi S, Aliakbary S, Habibi J. Generative model selection using a scalable and size-independent complex network classifier. *Chaos Interdiscipl J Nonlinear Sci* 2013;**23**:043127.
17. Nagy M, Molontay R. Network classification-based structural analysis of real networks and their model-generated counterparts. *Net Sci* 2022;**10**:146–69.
18. Scarselli F, Gori M, Tsoi AC *et al.* The Graph Neural Network Model. *IEEE Trans Neural Netw* 2009;**20**:61–80.
19. Wu Z, Pan S, Chen F *et al.* A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning system* 2020;**32**:4–24.
20. Pope PE, Kolouri S, Rostami M *et al.* Explainability methods for graph convolutional neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; Long Beach, CA, USA: IEEE; 2019, 10772–80.
21. Yuan H, Yu H, Gui S *et al.* Explainability in graph neural networks: a taxonomic survey. *IEEE Trans Pattern Anal Mach Intell* 2023;**45**:5782–99.
22. Agarwal C, Queen O, Lakkaraju H *et al.* Evaluating explainability for graph neural networks. *Sci Data* 2023;**10**:144.
23. Amara K, Ying R, Zhang Z *et al.* GraphFramEx: towards systematic evaluation of explainability methods for graph neural networks. In: *Proceedings of the First Learning on Graphs Conference (LoG)*. Proceedings of Machine Learning Research, vol. 198. London, UK: PMLR; 2022.
24. Ying R, Bourgeois D, You J *et al.* GNNExplainer: generating explanations for graph neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 32. Red Hook, NY, USA: Curran Associates, Inc.; 2019, 9240–51.
25. Luo D, Cheng W, Xu D *et al.* Parameterized explainer for graph neural network. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20*. Red Hook, NY, USA: Curran Associates Inc.; 2020.
26. Selvaraju RR, Cogswell M, Das A *et al.* Grad-CAM: visual explanations from deep networks via gradient-based localization. *Int J Comput Vis* 2020;**128**:336–59.
27. Gu J, Tresp V. Saliency methods for explaining adversarial attacks. arXiv, arXiv:1908.08413, https://api.semanticscholar.org/CorpusID:201309174, 2019, preprint: not peer reviewed.
28. Huang Q, Yamada M, Tian Y *et al.* GraphLIME: local interpretable model explanations for graph neural networks. *IEEE Trans Knowl Data Eng* 2023;**35**:6968–72.

29. Schlichtkrull MS, De Cao N, Titov I. Interpreting graph neural networks for NLP with differentiable edge masking. arXiv, arXiv:2010.00577, 2020, preprint: not peer reviewed.
30. Lucic A, ter Hoeve M, Tolomei G *et al.* CF-GNNExplainer: counterfactual explanations for graph neural networks. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 4499–511. London, UK: PMLR (Proceedings of Machine Learning Research), 2022.
31. Yuan H, Yu H, Wang J *et al.* On explainability of graph neural networks via subgraph explorations. In: *Proceedings of the 38th International Conference on Machine Learning (ICML), Virtual conference (planned Vienna, Austria),* pp. 12241–52. London, UK: PMLR (Proceedings of Machine Learning Research); 2021. https://proceedings.mlr.press/v139/yuan21c.html
32. Li W, Li Y, Li Z *et al.* DAG Matters! GFlowNets Enhanced Explainer For Graph Neural Networks. In: *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*. Amherst, MA, USA: OpenReview.net; 2023.
33. Schnake T, Eberle O, Lederer J *et al.* Higher-order explanations of graph neural networks via relevant walks. *IEEE Trans Pattern Anal Mach Intell* 2022;**44**:7581–96.
34. Yuan H, Yu H, Gui S, Ji S. Explainability in graph neural networks: a taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. New York, NY, USA: IEEE; 2023;**45**:5782–99.
35. Azzolin S, Longa A, Barbiero P *et al.* Global explainability of GNNs via logic combination of learned concepts. In: *The Eleventh International Conference on Learning Representations*. Amherst, MA, USA: OpenReview.net; 2023. https://openreview.net/forum?id=OTbRTIY4YS
36. Wang X, Shen HW. GNNInterpreter: a probabilistic generative model-level explanation for graph neural networks. In: *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*. Amherst, MA, USA: OpenReview.net; 2023.
37. Lv G, Chen L. On data-aware global explainability of graph neural networks. *Proc VLDB Endow* 2023;**16**:3447–60. https://doi.org/10.14778/3611479.3611538
38. Vu MN, Thai MT. PGM-Explainer: probabilistic graphical model explanations for graph neural networks. *Advances in Neural Information Processing Systems* 2020;**33**:12225–35. https://doi.org/10.1145/3696444
39. Longa A, Azzolin S, Santin G*et al*. Explaining the explainers in graph neural networks: a comparative study. *ACM Comput Surv* 2025;**57**:1–37. https://doi.org/10.1145/3696444
40. Orsini C, Dankulov MM, Colomer-de Simón P *et al.* Quantifying randomness in real networks. *Nat Commun* 2015;**6**:8627. https://doi.org/10.1038/ncomms9627
41. Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. *Nature* 1998;**393**:440–2. https://doi.org/10.1038/30918
42. Shuvo SS, Aghdaei A, Feng Z. Geodesic distance between graphs: a spectral metric for assessing the stability of graph neural networks. arXiv, arXiv:240610500, 2024, preprint: not peer reviewed.
43. Erdős P, Rényi A. On random graphs I. *Publ Math Debrecen* 1959;**6**:290.
44. Barabási AL, Albert R. Emergence of scaling in random networks. *Science* 1999;**286**:509–12.
45. Mahadevan P, Krioukov D, Fall K *et al.* Systematic topology analysis and generation using degree correlations. *SIGCOMM Comput Commun Rev* 2006;**36**:135–46. https://doi.org/10.1145/1151659.1159930
46. Maslov S, Sneppen K, Alon U. *Correlation Profiles AND Motifs in Complex Networks*. Weinheim, Germany: John Wiley & Sons, Ltd., 2002, 168–98. https://onlinelibrary.wiley.com/doi/abs/10.1002/3527602755.ch8
47. Maslov S, Sneppen K, Zaliznyak A. Detection of topological patterns in complex networks: correlation profile of the Internet. *Phys A Stat Mech Appl* 2004;**333**:529–40. https://doi.org/10.1016/j.physa.2003.06.002
48. Gjoka M, Kurant M, Markopoulou A. 2.5K-graphs: from sampling to generation. In: *Proceedings of IEEE INFOCOM*, pp. 1968–76. New York, NY, USA: IEEE, 2013.
49. Mahadevan P, Krioukov D, Fomenkov M *et al.* The Internet AS-level topology: three data sources and one definitive metric. *Sigcomm Comput Commun Rev* 2006;**36**:17–26.

50. Colizza V, Pastor-Satorras R, Vespignani A. Reaction–diffusion processes and metapopulation models in heterogeneous networks. *Nat Phys* 2007;**3**:276–82.
51. Rolland T, Taşan M, Charloteaux B *et al.* A proteome-scale map of the human interactome network. *Cell* 2014;**159**:1212–26.
52. Boguná M, Pastor-Satorras R, Díaz-Guilera A *et al.* Models of social networks based on social distance attachment. *Phys Rev E Stat Nonlinear Soft Matter Phy* 2004;**70**:056122.
53. Bronstein MM, Bruna J, LeCun Y *et al.* Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine* 2017;**34**:18–42.
54. Zhou J, Cui G, Hu S *et al.* Graph neural networks: a review of methods and applications. *AI Open* 2020;**1**:57–81. https://doi.org/10.1016/j.aiopen.2021.01.001
55. Hamilton WL. *Graph Representation Learning*. Vol. **14**. San Rafael, CA, USA: Morgan and Claypool.
56. Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Amherst, MA, USA: OpenReview.net; 2017.
57. Veličković P, Cucurull G, Casanova A *et al.* Graph Attention Networks. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Amherst, MA, USA: OpenReview.net; 2018.
58. Brody S, Alon U, Yahav E. How attentive are graph attention networks? In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Amherst, MA, USA: OpenReview.net; 2020.
59. Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Amherst, MA, USA: OpenReview.net; 2019.
60. Morris C, Ritzert M, Fey M *et al.* Weisfeiler and Leman go neural: higher-order graph neural networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 4602–9. Palo Alto, CA, USA: AAAI Press, 2019. https://ojs.aaai.org/index.php/AAAI/article/view/4384
61. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Netw* 1989;**2**:359–66. https://www.sciencedirect.com/science/article/pii/0893608089900208
62. Shuvo SS, Aghdaei A, Feng Z. Geodesic distance between graphs: a spectral metric for assessing the stability of Graph Neural Networks. arXiv, 2024, preprint: not peer reviewed.