UNIVERSITY OF
CAMBRIDGE
Department of Computer
Science and Technology

# Interpretability of Graph Neural Networks

Han Xuanyuan

Churchill College

May 2022

A dissertation submitted in partial fulfillment of the requirements for the
Computer Science Tripos, Part III

Total page count: 57

Main chapters (excluding front-matter, references and appendix): 39 pages (pp 9–47)

Main chapters word count: 11815

Word count generated using:

```
\newcommand{\wordcountcommand}{%
  \immediate\write18{texcount -1 -sum -merge main.tex > main.sum }%
  \input{main.sum}%
}
```

Flags `%TC:ignore` and `%TC:endignore` are used to exclude the front-matter, references and appendix from the word count.

# Declaration

I Han Xuanyuan of Churchill College, being a candidate for the MEng in Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 11815

**Signed**: Han Xuanyuan

**Date**: 25/05/2022

This dissertation is copyright ©2022 Han Xuanyuan.
All trademarks used in this dissertation are hereby acknowledged.

# Interpretability of Graph Neural Networks

## Abstract

Graph neural networks (GNNs) are a class of machine learning models that take graph-structured data as input. They have shown promising results in tasks such as molecular property prediction, physics modeling, and language processing [1, 2, 3]. Like many neural methods, GNNs lack explainability and interpretability, and their complex decision-making is effectively a black-box process to a human observer [4]. Methods for explaining GNN have been proposed in recent years. Ying et al. proposed GNNExplainer, which produces local explanations by finding a set of edges in the input graph that best explains a certain decision [5]. Yuan et al. proposed XGNN, which produces global explanations by generating a graph that maximises the model's prediction [6].

Most existing solutions do not align with the recent trend of using *concepts* to explain deep models, which have been shown to make explanations more plausible due to their ability to express a model's decision making in terms of high-level interpretable units of information [7, 8]. Moreover, although there has been significant research into the behaviour of individual neurons of convolutional neural networks and recurrent neural networks [9, 10], the current literature on GNN explainability tends to treat the model as a black-box and only considers input attribution, thus ignoring the potential of neurons as a way of interpreting models [4].

In this work, I perform a **novel analysis** of the type of neuron-level concepts that are inherent to a trained GNN, and show that certain neurons are able to detect interpretable units of information. Using the extracted concepts, I propose novel concept-based approaches to address three areas of GNN explainability: global model-level explanations, local instance-level explanations, and inherent explainability. My results indicate that my model-level explanations **outperform the state-of-the-art approach** XGNN in terms of plausibility and usefulness for the machine learning practitioner. My instance-level explanations also score higher in terms of explanation fidelity than the highly prominent GNNExplainer on four different benchmark datasets for graph classification. Finally, I show that discovered neuron-level concepts can be used to train transparent models with only a minor reduction to test accuracy compared to deep GNNs, whilst being **significantly more interpretable**.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The impact of machine learning and artificial intelligence has been widespread, having already had a lasting influence in areas such as medicine [11], e-commerce [12] and finance [13]. Recent years have witnessed a rapid shift from classical learning methods to 'deep' methods which are generally agreed upon to be more powerful in modelling tasks where representations are hard to design by hand. Deep learning refers to the practice and application of machine learning using artificial neural networks, and has made breakthroughs in domains such as computer vision and language processing [14]. Whereas previous methods relied on feature engineering and injection of domain knowledge, deep neural networks are able to learn such representations automatically. The current state-of-the-art models in vision and language processing are based on very deep architectures such as transformers, with models such as GPT-3 having the capability to generate sophisicated-looking text that is hard to distinguish from text written by humans [15]. Although powerful, these models typically come with several downsides, with a notable drawback being that their outputs are inherently not *explainable* and are essentially black-boxes whose outputs are produced through the interaction of millions of individual weights and parameters [16]. This reduces the level of trust that practitioners and end-users of the model have in it, and makes most current deep methods unsuitable for deployment in safety-critical areas such as healthcare. For example, consider a model used to diagnose medical conditions of patients from X-ray images. In this case, it is crucial that the decision-making process is trustworthy and interpretable, since the stakes are high. The field of explainable artificial intelligence (XAI) has emerged to tackle this problem, and researchers have developed an array of techniques to unveil the decision-making processes within these black boxes, with the goal of producing *explanations* that describe a model's decision in human-understandable terms [16]. XAI is an important area, with potential to benefit the application of AI in healthcare, finance, and justice systems [17, 18, 19].

Graph neural networks (GNNs) are a class of deep models that take graph-structured data as input. In recent years GNNs have achieved highly promising results, especially in scientific disciplines where graph-structured data are common [21]. The relative recency

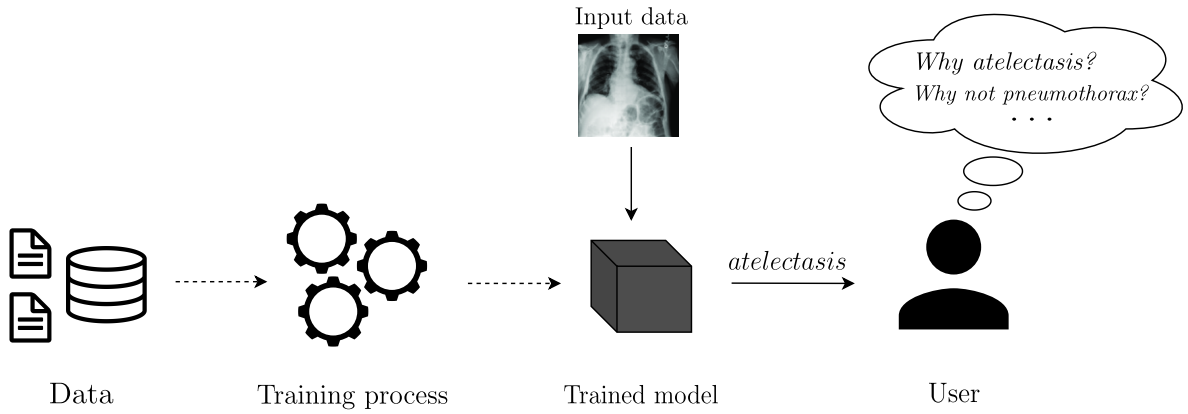


Figure 1.1: An example workflow in machine learning where explainability is desirable. The X-ray image is from the ChestX-ray dataset [20].

of GNNs means that fewer explainability methods have been developed for them. The earliest attempt at explaining GNNs was GNNExplainer [5], which aimed to identify a subset of edges of the input graph that is 'responsible' for the GNN decision. Later works have followed this direction and introduced improved methods for finding these important edges. This class of methods can be categorised as instance-level explanations, which only explain a single inference example, and has been criticised for the inability of explaining the model's general decision making process. Methods that aim to achieve 'model-level' explanations for GNNs are, in contrast, far less common. The most prominent[1] method XGNN [6] focuses on the task of generating a graph that maximises a certain GNN prediction, with the motivation that the generated graph may contain certain features and motifs that represent what the model believes to be important. However, the generated graphs may not capture all aspects of the model's decision-making.

Overall, most existing GNN explainability methods treat the model as a black box, and only consider the model inputs and outputs. This is in part due to the desire for such methods to be model-agnostic. However, there is a growing literature in the vision and NLP communities that focus on probing the individual neurons of a deep model to interpret the model's behaviour, and using such insights to produce more insightful explanations. Existing methods such as NetDissect [9] examine the high level 'concepts' that are learnt by probing the inner individual neurons of a deep model. This direction of research has insofar been lacking in the GNN community, with GCExplainer [22] being the only existing work that performs something similar[2]. In this dissertation, I attempt to fill this gap in the literature and provide new insights on how GNNs work.

[1] According to survey papers such as [4].
[2] They analyse the latent space, but do not probe individual neurons.

## 1.1 Contributions

This dissertation presents several contributions which I believe to be valuable to the research community:

- I propose a new framework that identifies the types of high-level semantic concepts that neuron activations within the latter layers of a GNN detect. I show that for certain tasks, neurons are able to detect a combination of multiple concepts, similar to what previous works such as NetDissect [9] have shown for convolutional neural networks. I show that concepts can be used to produce global explanations that outperform the existing state-of-the-art method XGNN in terms of plausibility.
- I propose a novel method for edge-mask explanations using our concept-extraction framework. Unlike existing methods, my approach is able to express the edge mask as logical formulae of human-interpretable concepts. Compared to highly influential GNNExplainer, I show that my method achieves higher on both visual quality and established metrics for evaluating explanations such as fidelity.
- I propose to create explainable models on graphs via a new method: transferring interpretable concepts from a black-box model to a transparent model via a concept distillation method. I show that this can produce transparent models with performance comparable to that of GNNs.

Overall, my work can be summarised as a new end-to-end framework for interpreting GNNs using concepts. Unlike the majority of existing work, my framework aims to tackle the three main paradigms of explainability: local post-hoc explanations, global post-hoc explanations, and inherent (self-explaining) interpretability.

# Chapter 2

# Background and related work

This chapter presents a comprehensive overview of the ideas which my contributions are rooted upon. Section 2.1 provides a brief dive into the concepts and terminologies of neural networks. Section 2.2 introduces the fundamental concepts of explainability and the current approaches which are related to our proposed methods. Then, Section 2.3 reviews the current state of research on explainability for GNNs, and finally Section 2.4 discusses the research directions that are most similar to my own.

## 2.1 Deep learning and neural networks

Deep learning is a family of machine learning methods that learn representations using artificial neural networks. Such methods have been shown to learn representations that generalise better than hand-crafted ones. Architectures including convolutional neural networks, autoencoders, recurrent neural networks and transformers have a strong ability to learn representations for noisy real-world data such as images and text. Likewise, for non-euclidean data with geometric properties, graph neural networks have made several breakthroughs. Models are typically trained using backpropagation and stochastic gradient descent. I will assume that the reader has a basic familiarity with how deep learning models are trained.

### 2.1.1 Perceptrons

The perceptron is the fundamental unit of neural networks [23]. It maps an input $\mathbf{x} \in \mathbb{R}^n$ to an output in $\mathbb{R}$. It has the general form

$$f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \sigma\left(b + \sum_{i}^{n} w_i x_i\right), \tag{2.1}$$

where $\mathbf{w} \in \mathbb{R}^n$ is a weight vector, $b \in \mathbb{R}$ is a bias term, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an activation function. In essence, it computes a weighted sum of the input features using a dot product.

Popular activation functions include the sigmoidal activation $\sigma(x) = (1 + e^{-x})^{-1}$ and the rectified linear unit $\sigma(x) = \max(0, x)$, also referred to as ReLU [24].

To model a function with $m$ outputs, we can use $m$ separate perceptrons. This is a *fully-connected* layer and is effectively a matrix-vector product of the form

$$\vec{f}(x) = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) = \begin{bmatrix} w_{11} & \dots & w_{n1} \\ \vdots & & \vdots \\ w_{1m} & \dots & w_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}. \tag{2.2}$$

A single fully connected layer can only learn to fit data that is linearly separable and is not expressive enough to model functions such as XOR. Stacking multiple fully-connected layers one after another gives a multi-layer perceptron (MLP). If the activation function after each layer is non-linear, then the MLP can fit non-linearly separable data. A *hidden layer* refers to any fully connected layer that is after the first layer and before the final layer. A neural network is considered to be *deep* if it consists of at least one hidden layer.

### 2.1.2 Graph neural networks

Graph neural networks (GNNs) are models that take graph-structured data as input and produce outputs such as graph-level classifications, node-level classifications, and edge predictions. GNNs are able to utilise both node feature information and structural information represented by the edges.

We formally define a graph $G = (E, V)$ as a set of edges $E$ and nodes $V$, and the neighbourhood $\mathcal{N}_i$ of a node $i$ as the set $\{j \mid (j, i) \in E\}$. Mathematically, we can express a GNN model as a function $\Phi(\mathbf{X}, \mathbf{A})$ such that:

- $\mathbf{X} \in \mathbb{R}^{|V| \times D}$ is a node feature matrix, where each row $\mathbf{x}_i$ is the $D$-dimensional feature vector of node $i$.
- $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$ is an adjacency matrix which represents the set of edges $E$.

In the forward pass, the GNN maintains a representation vector[1] $\mathbf{h}_i$ for each node $i$, and updates them at each layer, until the final layer representations are used for the downstream task [25]. Changing the order of the nodes does not fundamentally change the graph itself - therefore $\Phi$ must satisfy *permutation invariance* for tasks where a graph-level embedding is produced for the downstream task – that is, for any permutation matrix[2] $\mathbf{P}$ it must be true that $\Phi(\mathbf{PX}, \mathbf{PAP}^\top) = \Phi(\mathbf{X}, \mathbf{A})$.

A large number of GNN architectures have been proposed and most can be categorised as *message passing* GNNs [26]. Formally, a message-passing GNN $\Phi$ is a GNN such that

[1] The representations can be initialised as $\mathbf{h}_i := \mathbf{x}_i$, although other methods for initialising the embeddings exist.

[2] A permutation matrix permutes the rows of a second matrix when multiplying it.

at every layer $l$, $\Phi$ performs three computation steps:

1. Compute messages between each node and its neighbours: for each pair of nodes $i$ and $j$ such that an edge between $i$ and $j$, compute $\psi(\mathbf{h}_i, \mathbf{h}_j)$, where $\psi$ is some function that produces the message that $j$ 'sends' to $i$.

2. Aggregate messages received from neighbours: for every node $i$, compute $\text{AGG}_i = \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{h}_i, \mathbf{h}_j)$ where $\bigoplus$ is a permutation invariant aggregation function.

3. Update node representations. For each node $i$, we use the aggregated message to update its representation: $\mathbf{h}'_i = \phi(\mathbf{h}_i, \text{AGG}_i)$, where $\phi$ is some function, e.g. a multi-layer perceptron.

The entire message-passing mechanism for updating the representation of node $i$ is described by the following equation:

$$\mathbf{h}'_i = \phi\left(\mathbf{h}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{h}_i, \mathbf{h}_j)\right) \tag{2.3}$$

This layer is typically applied several times before the final representations are used for the downstream task. For instance, to perform graph classification, we can take the latent representations $\mathbf{h}_i$ and pool them to produce a graph level embedding $\mathbf{h}_{\text{graph}} = \sum_{i \in V} \mathbf{h}_i$ which can be inputted to an MLP.

The message-passing formulation of GNNs is a general framework for describing specific instantiations of GNNs. Popular architectures such as GCN (graph convolutional networks) [27], GIN (graph isomorphism networks) [28] and GAT (graph attention networks) [29] can all be described as specific instances of message passing, and in particular the other two 'flavours' of GNNs, convolutional and attentional, can also be grouped under message passing [30].

### Graph convolutional networks

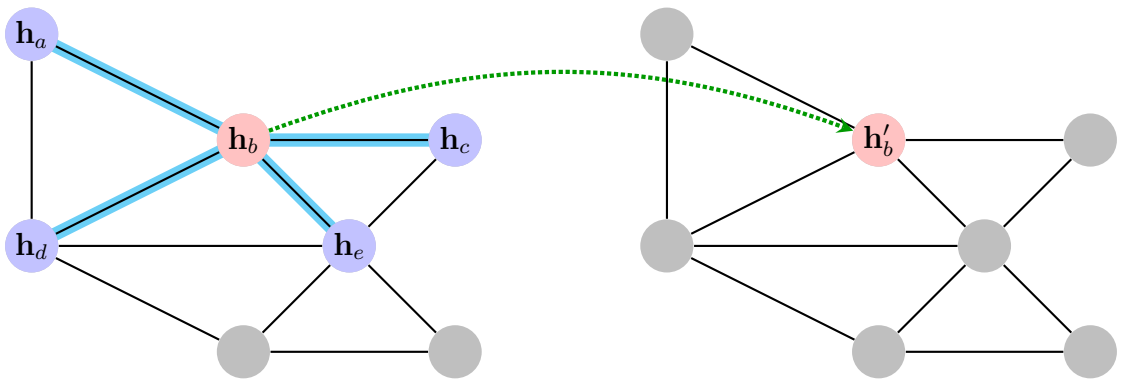


Figure 2.1: Convolution on graphs. Figure adapted from [31].

Whilst convolution is often associated with CNNs, the same principles can be applied in

the graph domain – we can update a node's representation by aggregating the representations of its neighbours.

The GCN layer [27] is a popular graph convolution architecture following this principle, described as

$$\mathbf{h}'_i = \bigoplus_{j \in \mathcal{N}_i \cup \{i\}} c_{ij} \mathbf{h}_j, \tag{2.4}$$

where $c_{ij} = \frac{1}{\sqrt{\deg(i)\deg(j)}}$ is used to normalise messages to avoid high-degree nodes having much larger features than nodes with small degree. Note that this fits under the message passing framework with $\psi$ being set to $\psi(\mathbf{h}_i, \mathbf{h}_j) = c_{ij}\mathbf{h}_j$. This architecture is noted to have a strong inductive bias on *homophilous* graphs, those where neighbouring nodes tend to be more similar than distant nodes.

More recently, a different graph convolution architecture GIN was introduced and has been shown to be more expressive than GCNs, and as powerful as the Weisfeiler-Lehman test for distinguishing pairs of nonisomorphic graphs [28].

Although graph convolution architectures do not utilise the full expressivity offered by the message passing framework, they are the most commonly used flavour of GNNs in industrial applications, since they are easier to scale [32].

## 2.2 Explainability and interpretability

This section discusses the goals of explainability and recent efforts in bringing explainability to GNNs.

Explainable AI (XAI) as a field aims to construct methods which provide *explanations* to the outputs made by machine learning models. Consider a machine learning model $\mathcal{M}$ that maps inputs in $\mathbf{x}$ to outputs $\mathbf{y}$. There are multiple ways to define what an explanation means. For instance, we can define an explanation as a subset of features in $\mathbf{x}$ that are highly important to the model's decision, with importance being some metric that is specific to the architecture of $\mathcal{M}$ and its task. An explanation for an image classification model could be a mask of pixels that highlight the important regions of the input. For a graph classification model, subgraphs and edge masks have been used as explanations [4].

The subjective nature of explainability means that research directions often diverge and do not focus on the same goals. Existing methods can be very roughly grouped into two directions:

1. Methods which take a trained model $\mathcal{M}$ and provide explanations without further altering $\mathcal{M}$. These are known as *post-hoc* methods [33, 16].

2. Methods which design models that are inherently explainable and have compara-

ble performance to black box models. These are *self-explainable* models[3] and are designed to be transparent by design [33, 16, 34].

The area of developing post-hoc methods receives significant attention since it aims to preserve the behaviour of original model. Much attention has been directed towards explaining highly performant models with strong inductive biases such as deep CNNs and transformers [16]. As such, post-hoc methods are the main focus of this dissertation. However, self-explainable models are also of great interest, and I will also make use of them in later sections.

### 2.2.1 Prominent explainability methods for vision models

Explainability methods for CNNs are commonly studied in the literature, perhaps due to the ease of producing visual explanations. One of the most influential post-hoc methods is LIME, which trains a regression-based surrogate model that is locally faithful [35]. Other approaches such as SHAP [36] and layer-wise propagation (LRP) [37] are decomposition-based and are able to decompose a model's final prediction as an additive sum of feature contributions. class activation mapping (CAM) and GradCAM [38, 39] perform a mapping from the latent CNN activations to the original input space to produce a pixel mask.

### 2.2.2 Local and global post-hoc explanations

Post-hoc explanations can be *local* or *global*. A local explanation, or *instance-level* explanation tells the practitioner why the model made a specific decision (why was patient $x$ assigned treatment $y$?). On the other hand, a global or *model-level* explanation tells the practitioner how the model behaves as a whole (how does the model decide that a particular treatment $y$ is suitable?) [40].

### 2.2.3 Concept-based explanations

**Concept bottlenecks and latent concepts.** A new branch of explainability is that of *concept-based* explainability. A concept is a high-level semantic unit of information that is understandable by a human [41]. Recently, there has been an emergence of *concept bottleneck* models, which allows an interpretable model to be constructed using a set of pre-extracted concepts [8]. A motivating example is that the classification of an image as a *bird* can be made on the basis that it is composed of the concepts of *wing* and *beak*. Recent work by Barbiero et al. has combined concept bottlenecks with first-order logic, allowing explanations to be produced as logical formulae [42]. Another approach proposed by Ghorbani et al. [41] are cluster-based concepts, formed by clustering the latent activations of subpixels. These concepts refer to what the model learns inherently, and can be seen as a form of post-hoc interpretation.

[3]They are sometimes called *intrinsic* or *transparency-based* models in the literature

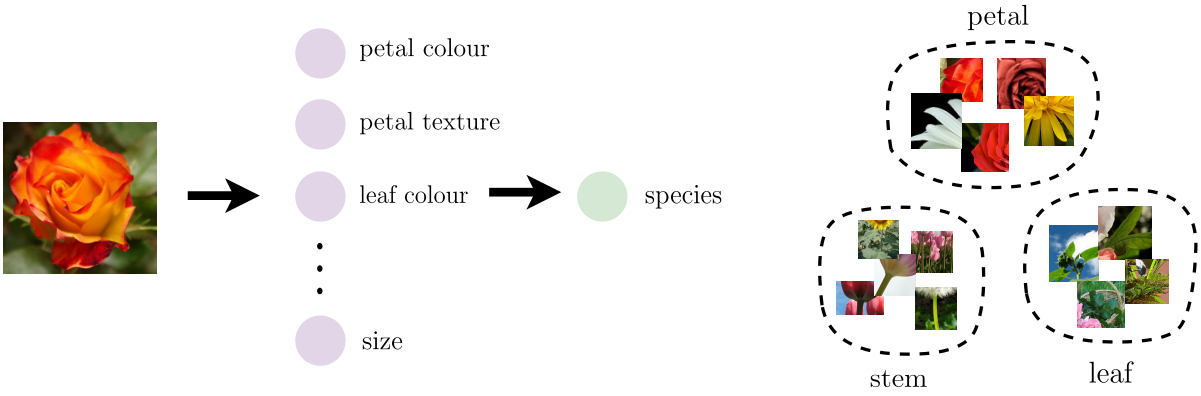


Figure 2.2: Different types of concepts. Left: bottleneck concepts. Right: latent space concept clusters.

**Neuron-level concepts.** A parallel line of work has been primarily motivated by the idea that convolutional neural networks learn units (also called filters or neurons) that correspond to feature detectors for certain concepts. Much work has focused on identifying these concepts and interpreting them. Bau et al. [9] proposed a framework NetDissect which aligns latent hidden units of a CNN with semantic concepts, and proposed metrics for evaluating the interpretability of hidden units. They showed that deep CNNs such as VGG [43] and ResNet [44] detect high-level concepts such as *house*, *dog* and *train*. Mu and Andreas [45] extended their work by using beam search to identify *poly-semantic* units that are activated by a logical combination of base concepts. The discovery of concepts inherent in the latent space not only provides insights into the models themselves, but can also serve as model-level explanations.

### 2.2.4 What makes a good explanation?

An important point to consider is the manner in which explanations are evaluated: what separates a good explanation from a poor one? Key ideas are faithfulness, plausibility and interpretability. An explanation is considered to be *faithful* to the model if it reflects the model's true decision making process, and is considered *plausible* if it appears convincing to a human [46]. For instance, consider the task of classifying images of cats and dogs. If our explanation for why an image is classified as a cat is *because the image contains whiskers* this would likely be plausible, since cats have whiskers but dogs do not, but not necessarily faithful, since the model could be doing something completely different – for example, it could be the case that all dog pictures were taken outdoors, and all cat pictures were taken indoors, causing the model to simply predict based on background colour. Another term for a faithful explanation is one that has *high fidelity*. Rudin [47] argues that all explanations are inherently unfaithful to the original model, since the only method to achieve perfect fidelity is to replicate the original model.

The third idea, that of *interpretability*, is rather conflated. Firstly, it can refer to the interpretability at the model level: Rudin [47] defines an interpretable model as one that is

constrained such that the decision-making is inherently clear to the human without needing any explanation techniques. The quantification of interpretability is difficult to define and is typically domain-specific. Note that self-explaining models are not necessarily interpretable in this sense of the term, since they still rely on an explanation mechanism. Interpretability may also refer to that of individual explanations. Ribeiro [35] defines an explanation to be interpretable if it is represented to the human in an understandable manner – for instance, as a binary mask, or a set of attention values.

A current dilemma within XAI research is whether explanations for black-box models are a viable choice in high-stake applications, or should black-box models be retired entirely in favour of interpretable models for these domains [47]. Nevertheless, black-box explanations are still invaluable to the practitioner for providing insights which will ultimately help develop an improved model.

## 2.3 Graph neural networks and explainability

### 2.3.1 The challenges

Traditional model-agnostic approaches such as backpropagation-based feature attribution techniques and surrogate modeling techniques such as LIME are known to produce misleading examples [5] and suffer from issues such as gradient saturation. A major challenge in interpreting GNNs is the difficulty in producing human-interpretable visualisations that capture both graph structure and node feature information. Unlike images or text, graphs have more complicated structure and their meaning is not immediately obvious to a human. This can be a problem, especially for tasks with very large graphs. Therefore, much attention has been focused on designing methods that are specifically suited towards GNNs, rather than relying on universal approaches.

### 2.3.2 Current approaches

The standard within current research is to use *masks* to highlight important edges and nodes. A major line of work attempts to produce *subgraph explanations* - that is, given a model $\mathcal{M}$ and graph $G$, find the subgraph $S$ that best explains the prediction of $\mathcal{M}$ given $G$. This is depicted in Figure 2.3. GNNExplainer [5] was the original work that attempted this problem and proposed a mutual-information maximisation approach. Their explanations are highly local and do not highlight the global mechanisms of the model [22]. Later works such as PGExplainer [48], PGMExplainer [49] and ReFine [50] attempt to address the various limitations of GNNExplainer. However, none of these methods are truly global – they at most only explain collections of instances and do not capture the global properties of the model [4, 22]. In other words, a practitioner may not be able learn the general rules that a model uses from such methods. A newer line of research aims to develop model-level explanations that are global. The prominent example

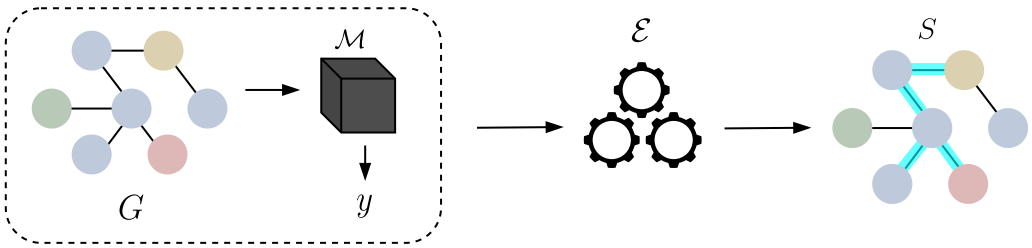


Figure 2.3: Subgraph explanations: the prediction $y = \mathcal{M}(G)$ can be explained with explainer $\mathcal{E}$, which produces subgraph $S$.

is XGNN [6], which aims to explain a GNN by generating an input that maximises a certain class prediction. However, a generated graph is still just an example, and does not demystify *how* the decision is reached. Another direction is the use of *concept-based* explanations. Magister et al. [22] proposed GCExplainer which uses $k$-means clustering [51] to identify clusters within the latent space of a GNN, and treats each cluster as a concept, thus providing post-hoc *global* explanations about the inner workings of a GNN. Their method emphasises that explainability methods can benefit from being *human-in-the-loop* and having a high degree of interactability. Georgiev et al. [52] were the first to use concept bottlenecks in an application of GNNs, in which they used a bottleneck layer to explain an algorithmic reasoning model. By using logic explained networks (LENs) they are able to provide both local and global explanations.

The appeal of concepts arises from their inherent interpretability, allowing the practitioner to abstract away the underlying mechanisms of GNN layers, and focus on high-level units of information [42]. The existing work on concept-based explanations for GNNs has no method of providing explanations that are instance-level unless a bottleneck is used. The drawback of bottleneck concept models is that labelling is required and that the practitioner must know the concepts beforehand. Though this may be feasible for image datasets where labelling sections of an image is intuitive for humans (e.g. different parts of a bird), the same is not true for graphs, where the concepts may not align well with human intuition, and for most tasks the concepts are not known beforehand. Therefore, it is desirable to have a way to obtain an interpretable model without resorting to bottleneck layers.

Furthermore, there is a lack of work investigating what happens *inside* a GNN, and most methods appear to treat the GNN as a black-box for the purpose of being model-agnostic. For vision and language models, much work has been done to probe the inner activations of the latent space and relating them to concepts [45, 9, 10], and I believe that insight can be gained from a similar analysis but for GNNs.

## 2.4 Closely related work

The main method proposed in this dissertation attempts to uncover a novel direction of research: analysing individual neurons of a GNN and understanding the concepts they detect, and thus there is no direct baseline to compare to. However, my work is inspired by existing works in the vision and language domains such as NetDissect [9] and the concept extraction method by Mu and Andreas [45], as well as existing concept-based methods for GNNs such as GCExplainer [22] and concept-based explainable reasoning [52]. In later chapters, I propose methods of producing instance-level explanations, and thus the existing works such as GNNExplainer [5], PGExplainer [48] and ReFine [50] are all relevant. The general framework of extracting concepts from a deep model is similar to *rule extraction*, which also considers the model's latent space [53].

## 2.5 Summary

Concept-based explainability is a promising direction in XAI, but has seen little application to GNNs. In particular, no existing work has performed a neuron-level analysis of GNNs. In the next chapter, I present my novel concept extraction framework for GNNs, allowing the practitioner to probe individual neurons and align them with interpretable concepts.

# Chapter 3

# Methodology

In this chapter, I introduce my proposed framework and discuss its implementation. In Figure 3.1, I outline the main components of the framework. My method aims to reveal the concepts that arise in GNNs in a way that allows both local and global explanations to be produced. I further show that the extracted concepts can be used to create interpretable models via *concept distillation* without resorting to using bottleneck or pretraining on labelled concepts.

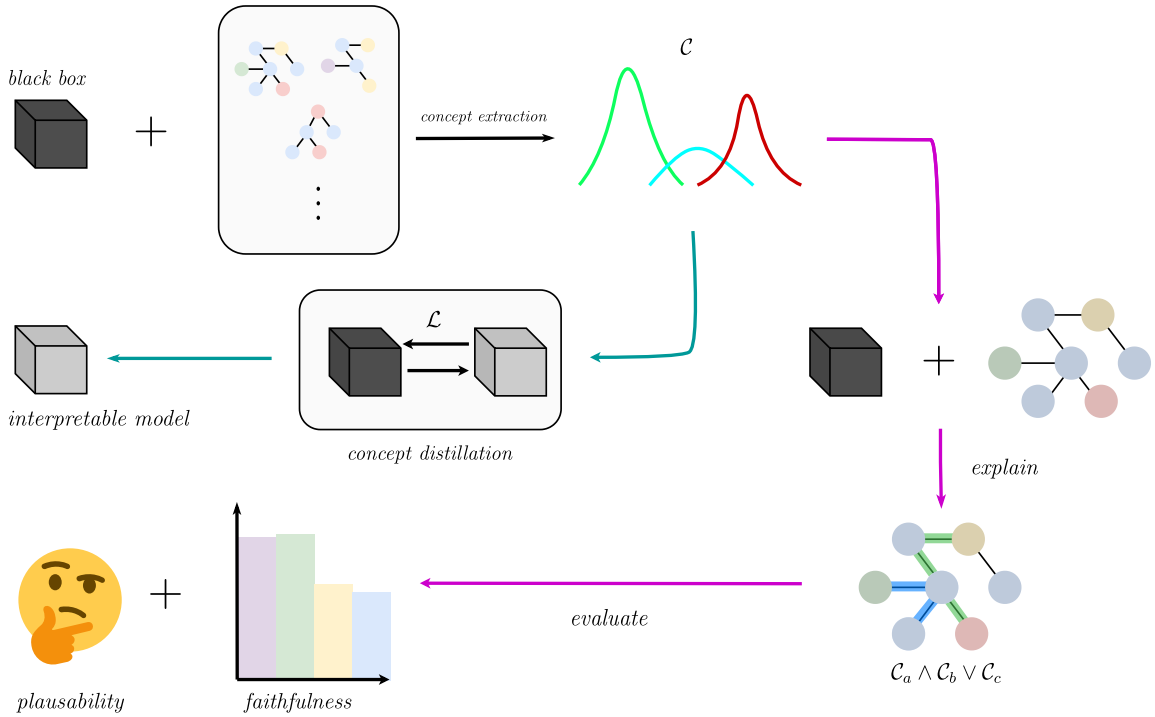


Figure 3.1: Overview of the proposed pipeline. Given a black-box model along with a training set of graphs, I extract concepts $\mathcal{C}$. I use extracted concepts to provide post-hoc explanations (purple line), and also to create interpretable models (blue line).

## 3.1 Concept extraction

With inspiration from the concept extraction method by Mu and Andreas [45] designed for vision and language models, I propose to perform a compositional concept search over the activation space of a GNN. Formally, given a GNN model $\mathcal{M}$, its activations at layer $l$ are the node representations after $l$ layers of message passing. Each node representation is a

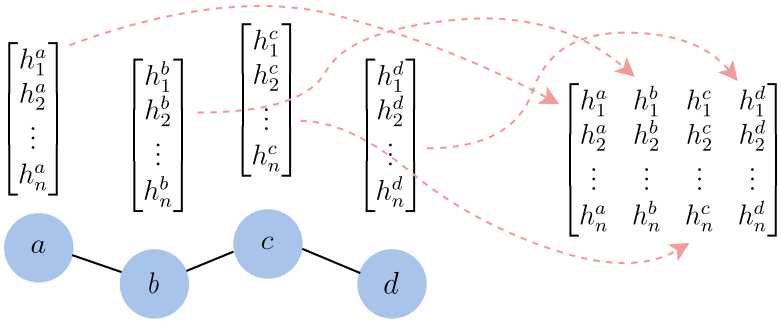


Figure 3.2: The activations of a GNN.

vector of the form $\mathbf{h}^{(l)} = [h_1, \ldots, h_n] \in \mathbb{R}^n$, where $n$ is the number of channels or *neurons*. For a graph with $m$ nodes, the activations can be represented as a matrix $\mathbf{X}^{(l)} \in \mathbb{R}^{n \times m}$. Letting $\mathbf{h}_i^{(l)}$ denotes the node representation of node $i$ at layer $l$ we can write $\mathbf{X}^{(l)}$ as

$$\mathbf{X}^{(l)} = \begin{bmatrix} | & & | \\ \mathbf{h}_1^{(l)} & \ldots & \mathbf{h}_m^{(l)} \\ | & & | \end{bmatrix}, \tag{3.1}$$

which is formed by stacking the node representations column-wise, as depicted in Figure 3.2. Recall that the embedding $\mathbf{X}[:, i] = \mathbf{h}_i^{(l)}$ represents some information about node $i$ within its $l$-hop neighbourhood[1], and is typically higher-level information if $l$ is deeper. Since this embedding is high-dimensional, it is difficult to interpret.

A major question is whether *concepts* are present in this activation space. GCExplainer identified concepts by $k$-means clustering on the activations of the final layer, and observed that different clusters correspond to different interpretable properties. However, since the permutation equivariance aggregation function is performed channel-wise in virtually all GNN architectures, it begs us to question if each individual *channel* act as *concept detectors*, as it has been shown in image CNNs. For instance, it could be the case that a certain channel is activated if the node has high centrality with respect to the graph, or if a node is a certain distance away from a particular substructure. The activations are fundamentally properties of the nodes, rather the edges - therefore, I propose to represent a concept as a mask over the nodes of the graph, analogous to the pixel masks of CNNs.

Formally, letting $\mathcal{G}$ be the space of input graphs, we treat concepts as functions $C : \mathcal{G} \to \{0, 1\}^m$ which outputs a boolean value indicating whether each node is part of the concept.

[1]We use slicing notation. $\mathbf{X}[i, :]$ indicates the $i$-th row of $\mathbf{X}$, and $\mathbf{X}[:, j]$ the $j$-th column.

We define a concept $C$ to be *interpretable* if there exists a natural language description of $C$ such that a practitioner is able to accurately predict the output of $C$ given any graph $G \in \mathcal{G}$ from solely knowing this description. Hence, concepts such as *"nodes that have exactly one neighbour"* would be interpretable. Furthermore, we refer to the *space* of concepts as a set $\mathcal{C}$ which contains all concepts that are interpretable. In practice, generating this set of concepts is intractable and ill defined due to the highly subjective nature of interpretability.

Our primary objective is to determine whether neurons are concept detectors. For a graph $G$, let $V_G$ be the nodes in $G$, $C(G)[i]$ indicate the mask value for the $i$-th node, and $\mathbf{X}_G^{(l)}$ be the activations of the model taking $G$ as input. Note that $\mathbf{X}_G^{(l)}[k,i]$ indicates the $k$-th neuron activation of the $i$-th node. We formally define a neuron $k$ to be a concept detector if

$$\exists C. \forall G \in \mathcal{G}. \left[ \forall i \in V_G. \left[ C(G)[i] \approx \tau_k \left( \mathbf{X}_G^{(l)}[k,i] \right) \right] \right]. \tag{3.2}$$

where $\tau_k : \mathbb{R}^m \to [0,1]^m$ is an element-wise thresholding function that is specific for neuron $k$. Later we will discuss how to learn $\tau_k$. However, this formulation assumes that there are concepts which near-perfect alignment with the activations, which in practice is unrealistic due to the noisy nature of deep neural networks. We cannot expect to find concepts which are perfectly faithful to the deep model's activations and interpretable at the same time. Even if we frame this as an gradient-based optimisation problem to search for an optimally satisfying concept $C$, we have no guarantees that the discovered $C$ will be interpretable.

Therefore, we relax the definition and consider whether there are concepts that approximately resemble the behaviour of a neuron $k$:

$$\exists C. \ \mathbb{E}_{G \sim \mathcal{G}} \left[ \text{Div} \left( C(G), \mathbf{X}_G^{(l)}[k,:], \tau_k \right) \right] \approx 0. \tag{3.3}$$

Here, Div is some measure of divergence between the concept mask $C(G)$ and the activations thresholded using $\tau_k$. Following previous works on image models [45, 9] that have used metrics intersection over union (IOU) scores, we choose a similar metric:

$$\text{Div}\left(\mathbf{a}, \mathbf{b}, \tau\right) = -\frac{\sum \mathbf{a} \cap \tau(\mathbf{b})}{\sum \mathbf{a} \cup \tau(\mathbf{b})} \cdot \frac{\mathbf{a} \cdot \tau(\mathbf{b})}{\sum b_i}. \tag{3.4}$$

This is the IOU score scaled by a factor representing how much of the activation signal is incorporated into the concept. Note that $\mathbf{a} \cdot \tau(\mathbf{b})$ represents the total activations within the concept mask, and $\sum b_i$ represents the sum of all activations in the graph. We find this necessary, since compared to the image models from prior work, the high-activation regions of GNN neurons have much greater variance in the magnitude of the activations.

Now, the threshold function $\tau$ is important, since the activations are continuous and thresholding will inevitably destroy some information. Prior work has used dynamic

thresholding [45], whereby a threshold is found for each individual neuron such that only a certain percentile of activations exceed the threshold. For the graph datasets I explore, I find that it is computationally feasible to apply dynamic thresholding and search in a particular range around each threshold, because the graphs require much less memory than images. Details are given in Appendix A.4.

Hence, the search objective becomes the process of finding a concept that maximises a score $f(C, k)$ for neuron $k$.

$$\arg\max_{C} f(C,k) \text{ s.t. } f(C,k) = \min_{\tau_k} \frac{1}{|\mathcal{D}|} \sum_{G \in \mathcal{D}} \text{Div}\left(C(G), \mathbf{X}_G^{(l)}[k,:], \tau_k\right) \tag{3.5}$$

in which we search through the concept space $\mathcal{C}$. This is tractable if $|\mathcal{C}|$ is small. However, following Mu and Andreas [45] we desire to search for *compositional* concepts, those which are a logical composition over base concepts. Formally, we define a compositional concept of length $k$ as an expression of the form

$$C_1 \oplus_1 C_2 \oplus_2 \ldots \oplus_3 C_k, \tag{3.6}$$

where $\{C_i\}_{i=1:k} \subseteq \mathcal{C} \cup \{\neg C \mid C \in \mathcal{C}\}$ and $\{\oplus_i\}_{i=1:k} \subseteq \{\wedge, \vee\}$. The space of compositional concepts is combinatorially large, which makes the objective intractable for large $k$. Therefore, we perform a beam search over the space of compositional concepts using the concept divergence as a priority metric to prune the less desirable concepts. At each iteration $i$ of the beam search, we maintain a set of concepts of length $i+1$ that are within the beam, and combine them with all unary concepts in $\mathcal{C} \cup \{\neg C \mid C \in \mathcal{C}\}$ to obtain a set of candidate concepts $\mathcal{C}_{\text{cand}}$ of length $i+2$. We then compute the expected divergence score for all concepts in $\mathcal{C}_{\text{cand}}$ and prune all but the top $T$ concepts. Since we only include the operators of disjunction and conjunction in our logical forms, unary concepts that are less desirable are unlikely to be part of the final formula. This motivates including the negated unary concepts in the set of base concepts.

### 3.1.1 Implementation notes

In practice, I use a GPU to make the search feasible. A vectorised version of the divergence function utilising scatter operations is used. Given a concept $C$, we can concatenate the concept masks for all graphs, and likewise the activations as follows:

$$\mathbf{a} = \Big\Vert_{G \in \mathcal{D}} C(G) \qquad \mathbf{b} = \Big\Vert_{G \in \mathcal{D}} \mathbf{X}_G^{(l)}[k,:]. \tag{3.7}$$

Both $\mathbf{a}$ and $\mathbf{b}$ have $N$ elements where $N$ is the total number of nodes in all graphs in $\mathcal{D}$. An index vector $I$ can be computed. Let $\sigma$ be a scatter addition operation such that $\sigma(\mathbf{x}) = \mathbf{y}$ where $y_i = \sum_{j|I_j=i} x_j$. The divergence score between $C$ and all graph activations

can be computed as

$$\overrightarrow{\text{Div}}(\mathbf{a}, \mathbf{b}, \tau, I) = -\sigma\left(\mathbf{a} \cap \tau(\mathbf{b})\right) \odot_{\div} \sigma(\mathbf{a} \cup \tau(\mathbf{b})) \odot_{\times} \sigma(\mathbf{a} \odot_{\times} \tau(\mathbf{b}))) \odot_{\div} \sigma(\mathbf{b}), \tag{3.8}$$

where $\odot_{\div}$ and $\odot_{\times}$ are element-wise division and multiplication. To simultaneously compute all pairwise divergences between all graphs in $\mathcal{D}$ and all concepts in $\mathcal{C}$, we can stack the $\mathbf{a}$'s for each concept into a matrix $\mathbf{A}$, and also the $\mathbf{b}$'s into a matrix $\mathbf{B}$ and perform the same computation, yielding a function $\overrightarrow{\text{Div}}(\mathbf{A}, \mathbf{B}, \tau, I)$ which computes a matrix $\mathbf{C} \in \mathbb{R}^{n \times |\mathcal{D}|}$ where the element at position $(i, j)$ indicates the divergence between concept $i$ and the activations in graph $j$.

This can be then used to compute the concept scores of all concepts. The exact procedure is shown in Algorithm 1.

**Algorithm 1** Vectorised concept scoring

**Input** $\mathbf{h} \in \mathbb{R}^N, I, \mathcal{T}$
**Output** $\mathbf{s} \in \mathbb{R}^{|\mathcal{T}|}$

**procedure** COMPARE($\mathbf{h}, I, \mathcal{T}$)
  $\mathbf{H} \leftarrow$ stack $|\mathcal{T}|$ rows of $\mathbf{h}$
  $\mathbf{s} \leftarrow [-\infty, \ldots, -\infty]$
  **for** $\tau \in$ thresholds **do**
    $\mathbf{C} \leftarrow \overrightarrow{\text{Div}}(\mathcal{T}, \mathbf{H}, \tau, I)$
    $\mathbf{s}' \leftarrow \text{mean}(\mathbf{C})$ ▷ compute the mean of each row
    $\mathbf{s} \leftarrow \mathbf{s} \odot_{\max} \mathbf{s}'$
  Return $\mathbf{s}$

The complete concept extraction process is shown in Algorithm 2, which requires a set of atomic concepts $\mathcal{A}$ as input.

### 3.1.2 Model-level explanations

Given model $\mathcal{M}$ and target class $y$, I propose to provide model-level explanations using the concept-extraction framework. Firstly, the concept map $\texttt{map} : \text{neurons}(\mathcal{M}) \times \mathcal{C} \to \mathbb{R}$ for the final-level of neurons can be extracted using Algorithm 2. To answer the question "how does $\mathcal{M}$ generally decide that a graph $G$ is of class $y$?" I will follow the concept-bottleneck paradigm of separating out the top-level of $\mathcal{M}$ (the section after the concept layer) from the rest of the model. Denoting this top level with $\mathcal{M}_T$, I propose to produce a model-level explanation for $\mathcal{M}_T$ where the input features are the neurons, to obtain a set of neurons $S_{\text{glob}}$ that globally explains the behaviour of $\mathcal{M}$ for class $y$. Then, a concept-based explanation can be produced by selecting $\mathcal{E}_{\text{glob}} = \{\arg\max_{C \in \mathcal{C}} \texttt{map}(i, C) \mid i \in S_{\text{glob}}\}$, which is the set of highest scoring concepts for neurons in $\mathcal{S}_{\text{glob}}$. There exist multiple ways of obtaining $S_{\text{glob}}$ - for models where $\mathcal{M}_T$ is a single perceptron, I propose to simply pick the neurons with positive weight contributions for class $y$.

A set of concepts in logical form might be insufficient as there is no visualisation. There-

**Algorithm 2** GNN concept extraction framework

**Input** Model $\mathcal{M}$, training set $\mathcal{D}_{\text{train}}$, atomic concepts $\mathcal{A}$, depth $l$, beam width $w$
**Output** Neuron concept scores $\texttt{map} : \text{neurons}(\mathcal{M}) \times \mathcal{C} \to \mathbb{R}$

**procedure** UPDATE
  $G_{\text{batch}}, I_{\text{batch}} \leftarrow \text{batch}(\mathcal{D}_{\text{train}})$
  $\mathbf{A} \leftarrow \mathcal{M}_{\text{acts}}(G)$
  **for** $u \in \{1, \ldots, |\text{neurons}(\mathcal{M})|\}$ **do**
    $\mathbf{s} \leftarrow \text{COMPARE}(\mathbf{A}_{\cdot,u}, I_{\text{batch}}, \mathcal{T}_u)$ ▷ get scores for each concept
    $\mathcal{T}_u \leftarrow \text{prune}_w(\mathcal{T}_u, \mathbf{s})$ ▷ only keep top $w$ concepts
    update map with $\mathcal{T}_u$ and $\mathbf{s}$
**procedure** EXPAND
  **for** $u \in \{1, \ldots, |\text{neurons}(\mathcal{M})|\}$ **do**
    **for** $c \in \mathcal{T}_u$ **do** $\mathcal{T}_u \leftarrow \{C_1 \oplus C_2 \mid C_1, C_2 \in \mathcal{T} \wedge \oplus \in \{\texttt{and}, \texttt{or}\}\}$
**procedure** EXTRACT
  Initialise $\mathcal{C} \leftarrow \mathcal{A}$, $\texttt{map} \leftarrow \texttt{empty map}$
  Initialise $\mathcal{T}_u \leftarrow \{\}$ $\forall u \in \{1, \ldots, |\text{neurons}(\mathcal{M})|\}$
  **for** $i \in \{1, \ldots, l\}$ **do**
    UPDATE()
    **if** $i < l$ **then** EXPAND() **end if**
Call EXTRACT

fore, for each concept in $\mathcal{E}_{\text{glob}}$ I also produce an example visualisation to accompany the concept. Namely, for each neuron $i \in \mathcal{S}_{\text{glob}}$ I choose $\arg\max_{G \in \mathcal{D}} \sum_j \mathbf{X}_G[j, i]$ as the graph $G$ that has the highest presence of neuron $i$, and choose this graph to visualise the corresponding concept of $i$. Note that the idea of using an extreme input (one that maximises a certain condition) as an explanation has been done before in the case of vision models [54], and also by XGNN[6][2]. Therefore, the form of our model-level explanation is a set of concepts, along with an instance-level visualisation of each.

## 3.2 Concept-based instance level explanations

I propose a novel method for producing instance-level explanations, using the concept extraction approach as a preliminary step. An instance-level explainer $\mathcal{E}$ for a GNN model $\mathcal{M}$ and an input graph $G = (E, V)$ is such that $\mathcal{E}(\mathcal{M}, G) \in \mathbb{R}^{|E|}$ is a mask over the edges of $G$ where higher values indicate that the edge is more important to the class prediction $\mathcal{M}(G)_j$ for class $j$. Given model $\mathcal{M}$, let $\mathcal{C}_{\text{neurons}} = \{C_1, \ldots, C_k\}$ denote the concepts extracted for the $k$ neurons in the final activation layer of $\mathcal{M}$ before the fully connected layers. Existing edge importance methods such as GNNEXPLAINER [5] and REFINE [50] do not utilise concepts, and existing concept-based methods such as GCExplainer [22] do not provide edge masks.

On a high level, my solution can be described in two independent stages or sub-problems:

[2] in their case an example graph is generated to explain a class

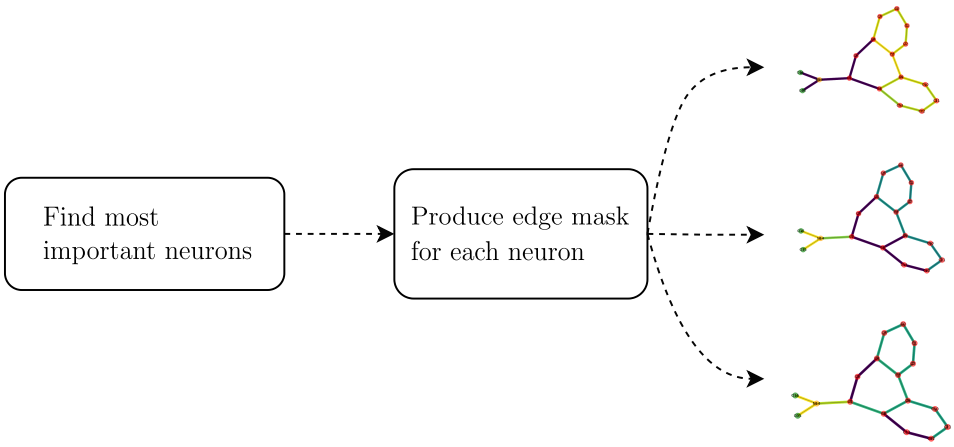


Figure 3.3: Producing instance-level explanations.

1. **Edge masking**: Given neuron $n_i$ with associated concept $C_i$, find the edge mask $M_i \in [0,1]^{|V|\times|V|}$ that best represents $C_i$ in the graph instance $G = \langle E, V \rangle$.
2. **Neuron masking**: Given all neurons $N = \{n_1, \ldots, n_k\}$, select a certain subset of neurons $S$ and aggregate per-neuron edge masks $M_i$ to obtain the final edge mask $M$ with associated concept set $\{C_i \mid n_i \in S\}$.

### 3.2.1 Activation-based edge masking

The first problem can be considered a general case of finding the edge mask that best explains a prediction, since we are now trying to explain a specific neuron. The neuron values at each node contain information about the structural information of the subgraphs containing the nodes. A simple solution is taking the value of neuron $n_i$ at each node, and computing the edge mask using function $f(u,v) = \frac{n_i(u)+n_i(v)}{2}$ for each edge $(u,v)$. This is motivated by the fact that a neuron $n_i(v)$ for a node $v$ is directly computed using the representations of $v$ and its neighbours $\mathcal{N}_v$ from the previous layer. Hence, if $n_i(v)$ is high, it could potentially be the result of the edges to $\mathcal{N}_v$. Note that the edge mask will likely be over-approximated, since the information is not necessarily flowing from $\mathcal{N}_v$. However, I later show experimentally that despite this limitation, high explanation fidelity is achieved.

### 3.2.2 Entropy-based neuron masking

For the second problem, an intuitive approach is to find *neuron importance scores* $I_1, \ldots, I_k$ and compute the final edge mask as $M = \sum_{i=1}^{k} I_i M_i$. Firstly, one can use a gradient-based metric to judge the importance of individual neurons. Given neuron $n_i$, we can measure $I_i = \frac{\partial \mathcal{M}(G)_j}{\partial n_i}$ i.e. the derivative of class $j$ with respect to neuron $n_i$. Hence, if the downstream classifier is a single-layer perceptron with weights $w_1, \ldots, w_k$, the final edge mask is $M = \sum_{i=1}^{k} w_i M_i$. Note that the instance level explanation $M$ is the GNN analogue of Grad-CAM [39]. However, I argue that this approach is unsuitable since

gradient-based sensitivity methods are known to not produce counterfactual explanations, and that the original Grad-CAM was more intended to be a visualisation tool than a method for producing high quality explanations. I therefore propose an *entropy-based approach* that is rooted upon well-defined principles and inspired by the masking method from GNNExplainer and Neural Relational Inference [5, 55].

Given neurons $n_i, \ldots, n_k$ our objective is to find a subset $S$ which counterfactually explains the prediction of the model $\hat{y} = \mathcal{M}(G)$. That is, removing the neurons in $S$ from the model should change the prediction $\hat{y}$. By removing a neuron $i$ we refer to the action of setting all $n_i(v) := 0$ for all nodes $v$. Let $Y$ be the class probability distribution produced by the model, and let $\mathcal{X}$ be the set of neurons being used for the downstream prediction. We use the notion of mutual information and adopt the following objective:

$$\arg\max_S \mathrm{MI}(Y, \mathcal{X}) = \arg\max_S \Big( H(Y) - H(Y \mid \mathcal{X} = S) \Big). \tag{3.9}$$

Note that $H(Y)$ is the entropy of the probability distribution outputted for the original input (i.e. when $S$ contains all neurons). Since $Y$ is constant, this is equivalent to $\arg\min_S H(Y \mid \mathcal{X} = S)$. Hence the objective becomes

$$\arg\min_S \; -\frac{1}{r} \sum_{c=1}^{r} \log P_{\mathcal{M}}(Y = y_c \mid \mathcal{X} = S), \tag{3.10}$$

which is intractible since the number of possible neuron subsets $S$ is exponential in the number of neurons $k$. Following Ying et al. [5] we interpret $S$ probabilistically and consider a multivariate Bernoulli distribution $\mathcal{S}$ from which $S$ is sampled, such that $\Pr_{\mathcal{S}}(S) = \prod_{i=1}^{k} \eta_i$, where we use a continuous relaxation and consider $S$ to be a *neuron mask* $\eta \in [0, 1]^k$. The objective becomes

$$\arg\min_{\mathcal{S}} \; -\mathbb{E}_{S \in \mathcal{S}} \left[ \sum_{c=1}^{r} \log P_{\mathcal{M}}(Y = y_c \mid \mathcal{X} = S) \right]. \tag{3.11}$$

Using Jensen's inequality we obtain

$$\arg\min_{\mathcal{S}} \; -\sum_{c=1}^{r} \log P_{\mathcal{M}}(Y = y_c \mid \mathcal{X} = \mathbb{E}_{S \in \mathcal{S}}[S]), \tag{3.12}$$

which gives us the following optimisable objective for finding $\eta$:

$$\arg\min_{\mathcal{S}} \; -\sum_{c=1}^{r} \log P_{\mathcal{M}}(Y = y_c \mid \mathcal{X} = \sigma(\eta)), \tag{3.13}$$

in which $\sigma$ is the sigmoid function. Note that the convexity assumption of Jensen's inequality is not guaranteed - however, similar to GNNExplainer we show that experimentally good local optima can still be reached.

After the mask $\eta$ is learned, we take $S = \text{round}(\sigma(\eta))$ as the final set of neurons to explain the prediction, where $\text{round}(\cdot)$ performs element-wise rounding to the closest integer. Now, to indicate the relative importance of each neuron, we take $I_i = w_i n_i \mathbf{1}_{i \in S}$ where $\mathbf{1}$ is an indicator function. In other words, we weight each chosen neuron by its absolute contribution to the class logit. This is purely an adhoc process for the purpose of giving the user a rough indication of relative importance.

Note that an alternative approach for choosing $S$ is by picking all neurons that have positive absolute contribution to the class logit, and likewise setting $I_i$ to the absolute contribution. This has the benefit of being a simple interpretable process where no learning is required, and I later use this as a baseline to compare the entropy-based approach with.

Note that my method has a different goal than feature attribution methods such as Shapley values [36]. Whereas feature attribution aims to compute importance scores for every feature, we aim to extract a small *subset* of features that are relevant and discard all other features. With only feature importance scores, the decision of discarding a set of features becomes completely arbitrary.

## 3.3 Concept distillation

The standard literature in knowledge distillation studies the problem of transferring the knowledge inherent in a teacher network to a secondary student model, that has certain advantages such as being more memory efficient or having lower inference times [56]. Whilst existing applications of distillation have mainly been performance-centric, I aim to apply distillation in a different direction: model interpretability. This is done for two reasons: (i) it allows us to evaluate the extracted concepts by seeing *whether they are useful* as features; (ii) it is an attempt to achieve a grand goal of XAI for GNNs - to create white-box models that are just as accurate as black-box ones.

Recall my GNN concept extraction framework which extracts concept set $\mathcal{C}$ from model $\mathcal{M}$. That is, given blackbox model $\mathcal{M}_\text{b}$ and concepts $\mathcal{C}$, I aim to transfer $\mathcal{C}$ onto a white-box model $\mathcal{M}_\text{w}$. Recall that we treat concepts $C \in \mathcal{C}$ as functions $C : \mathcal{G} \to \{0,1\}^m$. My method is shown in Algorithm 3.

We are effectively extracting a concept vector for each node, and performing channel-wise summation to obtain a concept vector for each graph. Thus, the $i$-th element of each concept vectors indicates the *number of times concept $C_i$ occurs in the graph.* Concept vectors are stacked to form training data $\mathbf{D}$.

The white box model $\mathcal{M}_\text{w}$ should be chosen by the practitioner depending on the problem at hand and the desired form of explanations. Later in Section 4.4 I perform experiments using different white box models including perceptrons and decision trees.

**Algorithm 3** Concept distillation

**Input**: $\mathcal{M}_{\text{b}}, \mathcal{D}_{\text{train}} = \{G_1, \ldots, G_{|\mathcal{D}_{\text{train}}|}\}$
**Output**: $\mathcal{M}_{\text{w}}$

**procedure** DISTILL
  $\mathcal{C} \leftarrow \text{EXTRACT}(\mathcal{M})$
  $\mathbf{M} \leftarrow \mathbf{0}^{|\mathcal{C}| \times |\mathcal{D}_{\text{train}}|}$
  Intialise $\mathcal{D}_{\text{w}}$
  **for** $j \in \{0, 1, \ldots, |\mathcal{D}_{\text{train}}|\}$ **do**
    **for** $i \in \{0, 1, \ldots, |\mathcal{C}|\}$ **do**
      $\langle E, V \rangle \leftarrow G_j$
      $\mathbf{D}_{i,j} \leftarrow \sum_{v \in V} C_i(v)$
  Train $\mathcal{M}_{\text{w}}$ on $\mathbf{D}$.

In addition, the practitioner has the choice of determining the number of concepts to transfer to the white box model. One can simply use *all* concepts, but at the cost of faithfulness, since not all concepts have good alignment with the original model.

# Chapter 4

# Experiments and evaluation

I present my findings using the methods discussed in Section 3. My results aim to showcase the role of concepts in many different ways, both qualitatively and quantitatively. In Section 4.1 I explain my experimental setup. In Section 4.2 I perform a dive into the types of concepts that arise in GNNs, and show that they align with human intuition. In Section 4.3, I evaluate the strength of my concept-based instance level explainer, and show that it outperforms existing baselines. In Section 4.4, I show that the extracted concepts can used to train interpretable models with no loss in performance, outperforming even black-box GNNs.

## 4.1 Experimental setup

**Datasets** Following standard practice in the field, I perform experiments on four popular real-world benchmark datasets for binary graph classification: MUTAG [57], Reddit-Binary [58], Proteins [59] and IMDB-Binary [58]. A summary of each dataset is shown in Table 4.1.

| Dataset | #Graphs | Description |
|---|---|---|
| MUTAG | 188 | Given nitroaromatic compounds, classify if each is mutagenic on *Salmonella enterica*. |
| Reddit-Binary | 2000 | Graphs are threads from `reddit.com`. Nodes represent users, and an edge exists beween $A$ and $B$ is $A$ replies to a comment by $B$. Classify graphs as either a discussion-type thread or a Q&A type. |
| Proteins | 1113 | Graphs are proteins and nodes are amino acids. An edge exists between two nodes if they are within 6 Angstroms apart. Classify proteins as either enzymes or non-enzymes. |
| IMDB-Binary | 1000 | Given a network of actors, classify if graphs as belonging to either the *action* or *comedy* genre. An edge exists between actors if both appear in the same movie. |

Table 4.1: Description of evaluation datasets.

**Models** Following existing work in the area, I choose to train models which fit under the convolutional flavour of GNNs, since these architectures are easier to train and more

commonly used. I choose to use Graph Isomorphism Network (GIN) layers [28], since they can be shown to have greater expressive power than Graph Convolutional Network (GCN) layers. For each graph classification task I train a GIN model with three layers, followed by an addition pooling operation across nodes to produce the graph-level representation. The exception is the Reddit-Binary dataset, where I use GCN as I observed training to be more stable given the large graph sizes. All models are given a hidden embedding size of 64. The exact descriptions of the models and their training parameters are shown in Appendix A.3.

## 4.2 What concepts do GNNs learn?

I apply my GNN concept extraction framework on the models trained on the benchmark datasets. Recall that we require a set of atomic concepts $\mathcal{A}$ to be inputted in order to search for compositional concepts. To align with the human-in-the-loop nature of explainability, I use a tailored set of atomic concepts for each task. I intentionally choose *simple* concepts that non-domain experts are able to understand, such as *next-to(X)* (being next to a node with label $X$), and *is(X)* (having label $X$). The full set of atomic concepts for each task is shown in Appendix A.3.

A notable finding is that it appears that interpretable concepts *are* in fact detected by the neurons of a GNN, and they vary across the different tasks. I provide a visual summary of the most intepretable concepts along with their logical forms. I also show the concept contribution scores for each example graph using *absolute contribution* and *entropy score*. The absolute contribution, defined in Section 3.2, is the absolute contribution of a neuron to the class logit. The entropy score is the mask value $\eta_i$ when the entropy-based masking approach is applied to the instance.

**MUTAG**. In the MUTAG dataset, we find $NO_2$ detector concepts and carbon ring concepts, which corroborate with the findings of Ying et al. [5]. The $NO_2$ concept is in fact the *aromatic nitro toxicophore* which has been shown to be a strong indicator of mutagenicity [60], and the presence of carbon rings is also an indicator. In the examples, **neuron 0** is a carbon ring detector and **neuron 11** is an $NO_2$ detector. We observe that carbon ring concepts typically have much higher absolute contribution to the final prediction. Unlike prior work, we additionally observe concepts which contribute to the prediction of non-mutagenicity. In particular we observe concepts that are highly concentrated on parts of compounds that hang off carbon rings but are not $NO_2$ substructures - for instance, **neuron 22** appears to detect any part of the input which is not part of a carbon ring or an $NO_2$ substructure. Other concepts appear to trigger for only certain types of atoms hanging off carbon rings, e.g. **neuron 29** detects Chlorine atoms which have exactly one carbon neighbour.

**Reddit-Binary**. In Reddit-Binary, graphs typically have one or two nodes with high

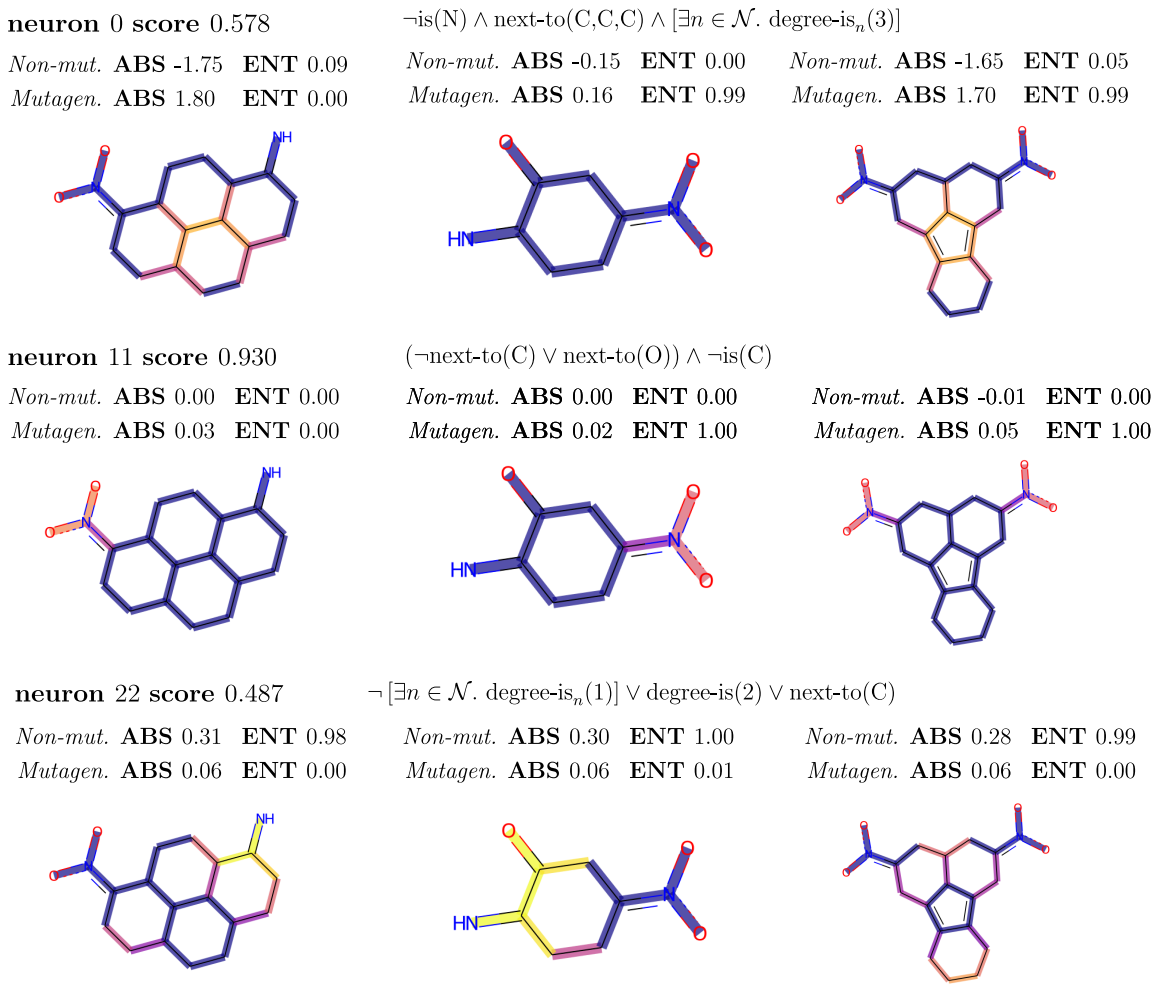


Figure 4.1: MUTAG Concepts. Lighter colours indicate higher importance. Each row is thresholded using the learnt threshold $\tau_k$. Neuron contributions are shown for both classes.

degree, corresponding to the highly upvoted comments that receive the most attention. we observe the emergence of the 'few-to-many' subgraphs, resembling a small number of experts replying to several questions, as was mentioned by Ying et al. [5], and is a strong indicatin that the graph is of the Q&A class. An example of such a concept detector is **neuron 51**. We additionally find neurons that appear to trigger when their degree is above a certain threshold, such as **neuron 20**. Intuitively, a user with lots replies is a strong indication of Q&A, since the host of the Q&A will likely receive a large number of follow-up questions. We also see neurons which appear to detect the nodes that are more than a single hop from a high-degree node, representing users at the end of a long comment chain, e.g. **neuron 46**. We observe that these concepts penalise the discussion class in terms of absolute contribution. One possible explanation is that discussion threads typically involve the same users commenting several times in the same chain, which would cause the chain to collapse in the graph (since nodes represent users, not comments).

**Proteins**. We discover concepts that are sensitive to the type of amino acid in the immediate neighbourhood of a node: for instance, **neuron 42** is activated in regions with high

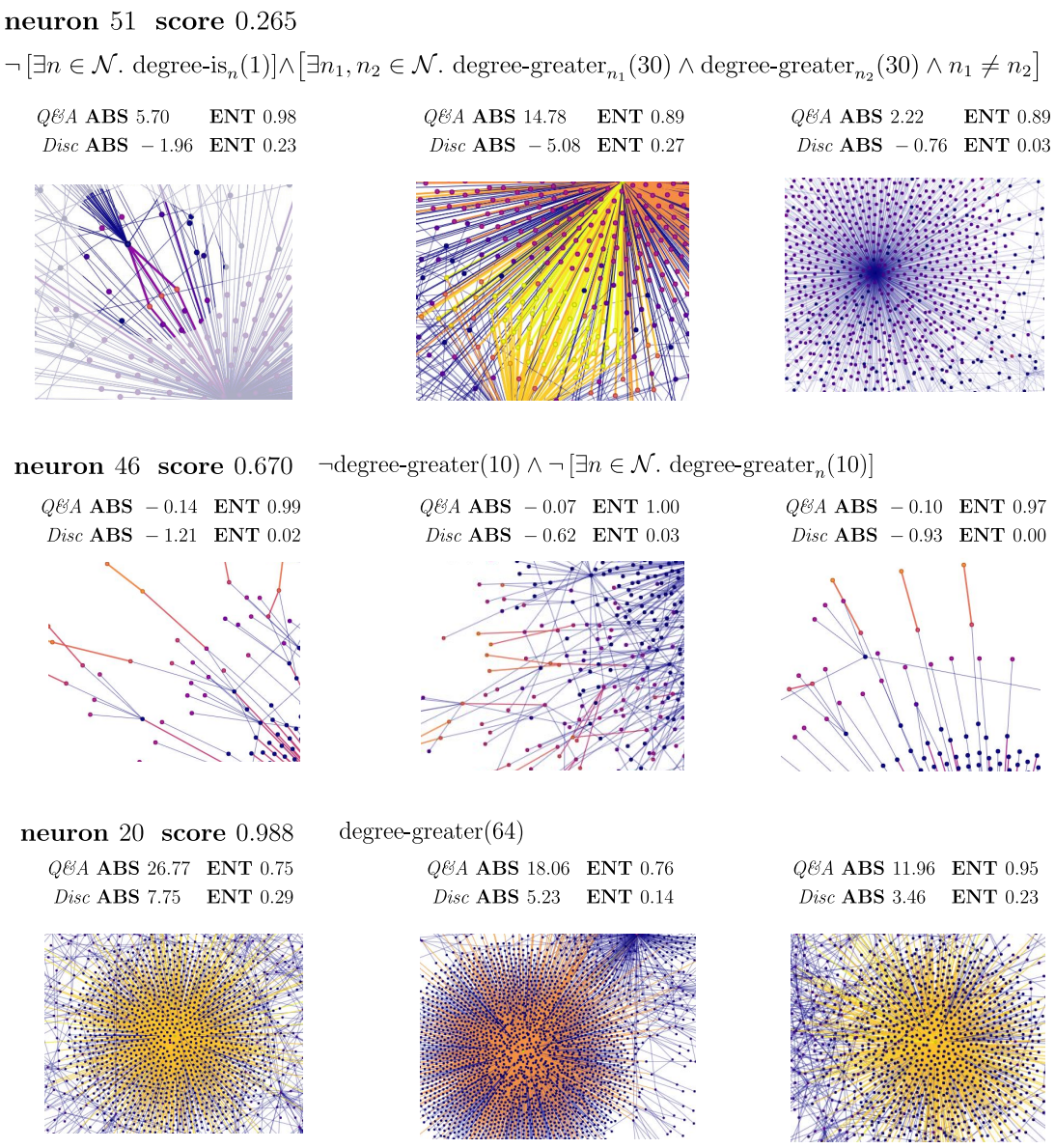


Figure 4.2: Reddit-Binary Concepts. For graphs where the concept is hard to see by eye, we have highlighted the concept area by changing the opacity of the image.

concentration of $A$ amino acids, but is surpressed in regions with high concentration of $C$ amino acids. Likewise, **neuron 12** is only activated in regions with high $C$ concentration.

**IMDB-Binary**. The typical graph has one or two 'popular' nodes that have high degree, and have multiple smaller communities attached to them. We observe that almost all active neurons are sensitive to the *degree* of a node, with some neurons only activating if the degree exceeds (or is less than) a certain amount. We find that **neuron 21** is activated for the small 'communities' that are attached to the popular node, and is accurately summarised as detecting if the degree is no greater than 7. It acts as a strong indicator of the genre being *action*. Interestingly, **neuron 12** behaves similarly - activating for nodes with degree no more than 5 - but is a strong indicator for the opposite genre *comedy*. This suggests the GNN is learning to discriminate genres based on subtle differences in the size of the small communities. There exists high-degree concepts, e.g. **neuron 7** which are

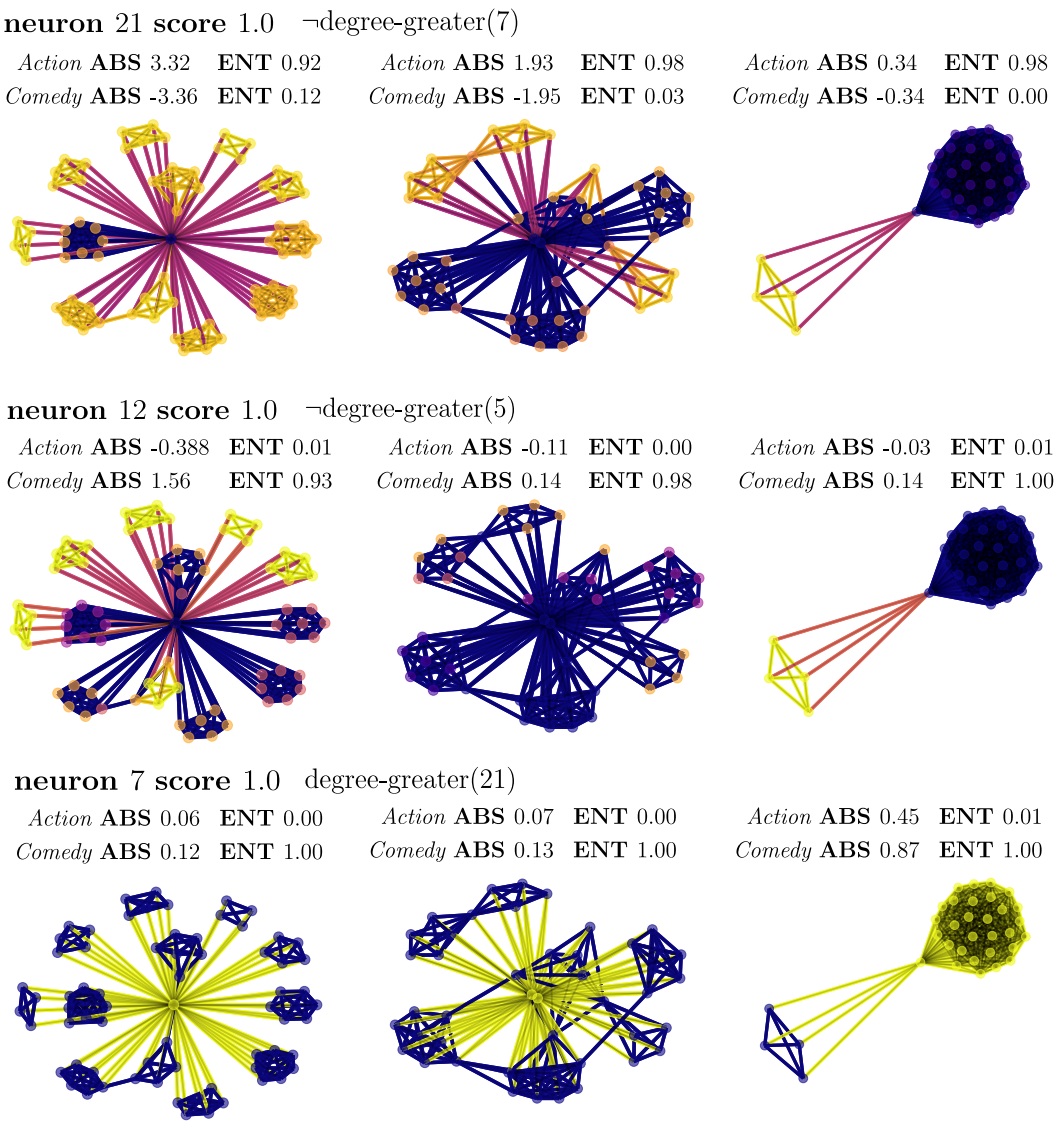


Figure 4.3: IMDB-Binary Concepts

indicative of the comedy class. Moreover, we are surprised by the high interpretability score of these neurons, with many having a perfect score of 1, i.e. the concepts learnt by the GNN can be summarised accurately using only degree information.

### 4.2.1 General takeaways

Overall we observe that a range of concepts are being detected by the GNN neurons. We make three important observations:

- **GNN neurons detect concepts that are lower-level than the concepts observed for deep image models and RNNs**. This is attributed to the fact that GNNs tend to be much shallower than CNNs and RNNs, and the ones used in my experiments have at most three layers. From the perspective of my method, this is a good sign, since high-level concepts are unlikely to be captured by composing together simple rules, and the fact that high-level concepts will require manual

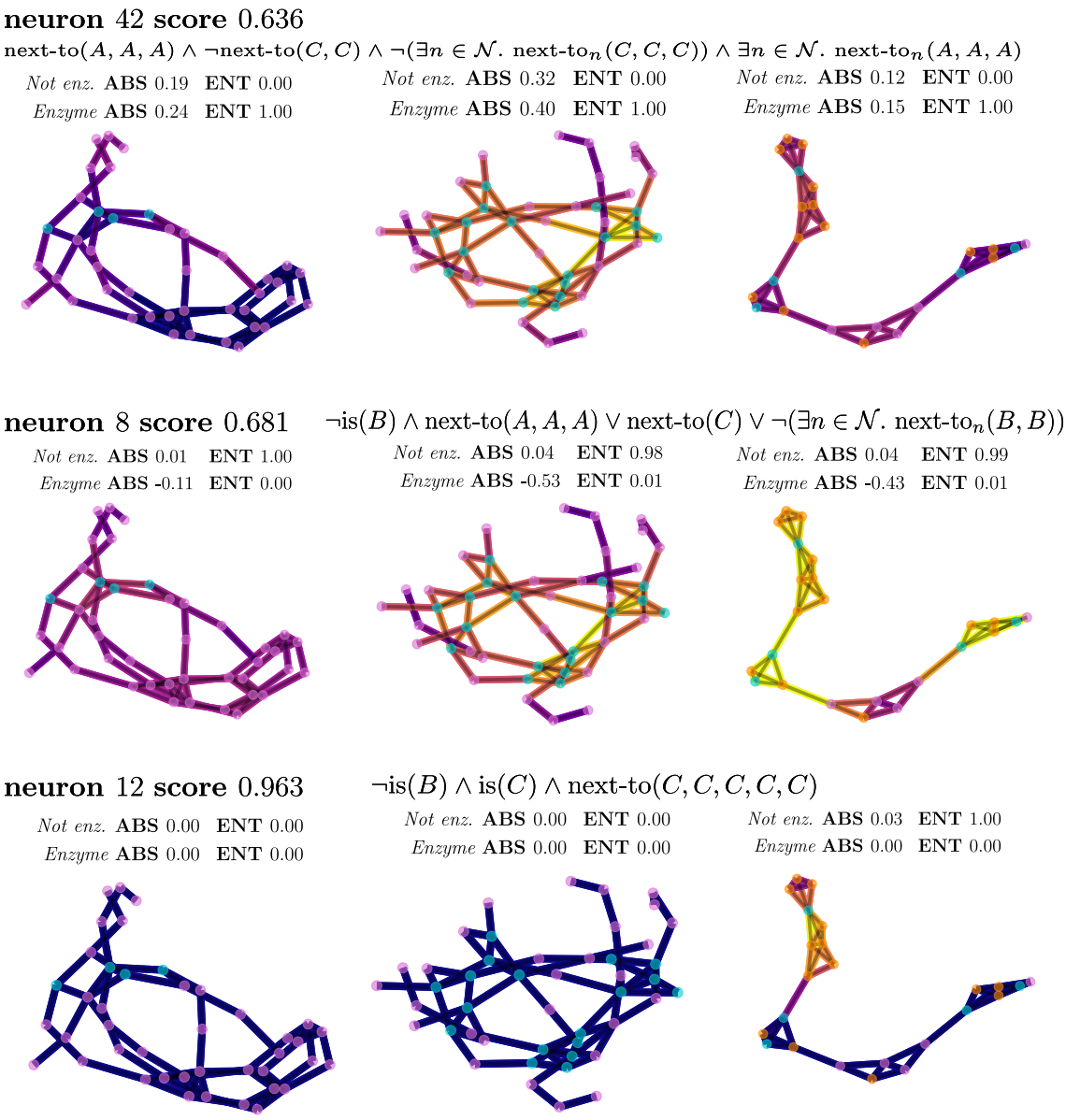


Figure 4.4: Protein Concepts

labelling.

- **Not all concepts are useful for the model**. There exist neurons which detect concepts, but have low contribution to predictions. In addition, there exists neurons that the model believes to be useful for *both* classes and therefore do not act as useful discriminators for the model. For example, in IMDB-Binary, there are neurons that detect the presence of a node with large degree which have positive weights for both *romance* and *comedy* genres.

- **GNNs learn neurons with high redundancy**. There are often neurons that detect similar concepts. For instance, the MUTAG model contains 12 neurons that act more or less as $NO_2$ detectors, but their associated concepts are marginally different. This can be attributed to the nature of SGD-based algorithms used to train GNNs - there is no explicit constraint that enforces each neuron to behave

differently. We note that despite there being many neurons detecting similar concepts, their activation thresholds can still vary greatly. This suggests my method can be potentially used to identify neurons that are redundant in the information they provide.

### 4.2.2 Are more interpretable concepts more important?

A natural question to ask is whether the neurons identified by my system are actually beneficial to decision-making process of the model. Previously it has been observed for image models that there is a positive correlation between neuron interpretability and contribution to model accuracy, whereas for language models there is a negative correlation [45]. We define a neuron's *interpretability* as the score of its best matching concept. We note that several general methods of determining neuron importance do not necessarily work for GNNs as they do for CNNs. For example, we found that *erasure* – whereby neurons are zeroed and the resulting change in performance is measured – is a poor metric for determining neuron importance due to the high presence of *redundant neurons* learnt by GNNs. Likewise, solely looking at the weights of the connections attaching the neurons to the prediction can be misleading, since some neurons have drastically different activation thresholds than others. Therefore I choose two alternative metrics of my own design.

**Neuron importance**. For a model with $r$ class outputs, we measure the importance of a neuron $n_i$ in making a single prediction as the variance of individual logit contributions: $\mathrm{Var}(w_i^1 n_i, \ldots, w_i^r n_i)$ where $w_i^j$ is the weight connecting $n_i$ to the $j$-th logit. We aim to measure the neuron's ability to discriminate between classes, rather than its absolute contribution.

**Neuron correctness**. Following Mu and Andreas [45] we also measure how much each neuron contributes to the model being *accurate*. They chose to measure the frequency of correct predictions when each neuron is activated. While this yielded good results for CNNs, it does not transfer well to GNNs due to the redundant nature of neurons. We choose to compute a correctness score for each neuron, defined as

$$\text{correctness}(n_i) = \mathrm{Corr}(|h_i^{\text{pool}}|, -\mathcal{L}(y_{\text{pred}}, \hat{y})) \tag{4.1}$$

in which we measure the Pearson product-moment correlation between the neuron's magnitude with the negated value of the loss function used to train the model.

I show the results of my quantitative analysis in Figure 4.5. First, for the MUTAG and Proteins tasks, the distribution of neuron interpretability scores appears to resemble a bell curve, where more neurons have scores closer to 0.5 than either extremes. For Reddit-Binary, we observe the most neurons are either not interpretable or are very interpretable. In terms of the relationship with neuron importance, we observe that the most important neurons are situated around the centre of the interpretability spectrum. This suggests

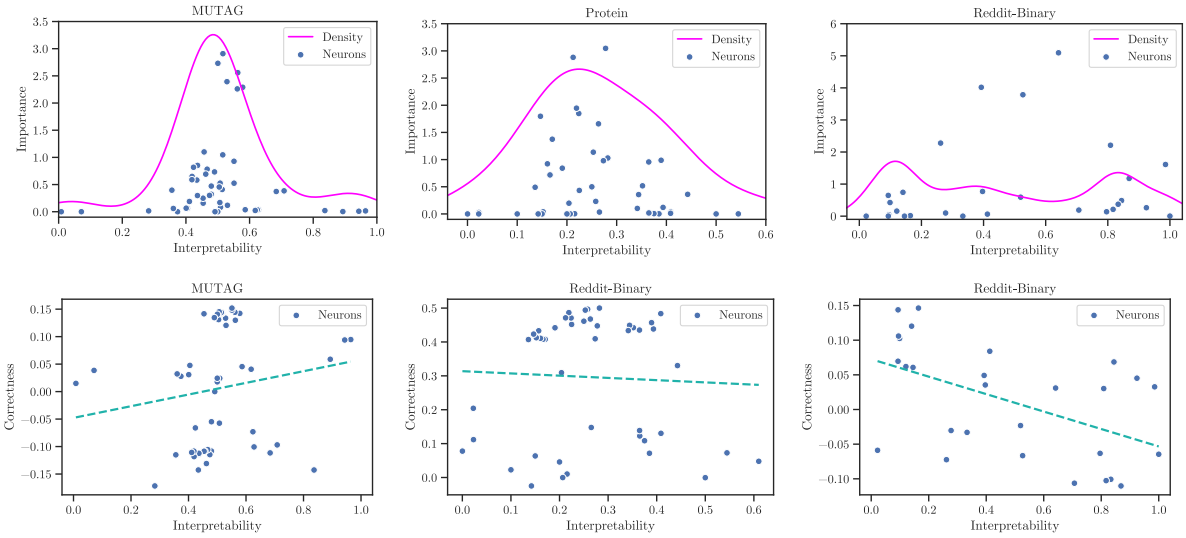


Figure 4.5: Investigating the interpretability of neurons and their importance. The pink curves are kernel density estimations [61] to indicate the relative spread of neurons across different interpretability scores. The blue dashed lines are lines-of-best-fit.

that the neurons which are most important are also more difficult to summarise using a set of interpretable rules, which can be unsurprising. Nevertheless, we do not observe a negative correlation between neuron importance and interpretability, and the most important neurons typically have around 0.5 interpretability, which I emperically observe to be sufficient in terms of explanation quality. In terms of contribution to model accuracy, we observe that neuron interpretability is anticorrelated with having accurate predictions on the Proteins and Reddit-Binary datasets, suggesting that more interpretable neurons tend to capture more spurious patterns in the data and do not generalise as well as less interpretable neurons.

### 4.2.3 Beyond final layer concepts

Works on interpreting deep models with concepts typically focus on the last few hidden layers, since high-level concepts are more likely to emerge there. Models are also typically well-trained to be representative of the real-world use case. This motivates us to ask two follow-up questions:

1. Do different types of concepts emerge at different depths of the model?

2. How does training affect the concepts that emerge? Do models with only a few epochs of training exhibit different concepts than models that are trained for extended numbers of epochs?

I perform two smaller experiments on the MUTAG dataset in an attempt to answer these questions and to provide some preliminary steps for future research.

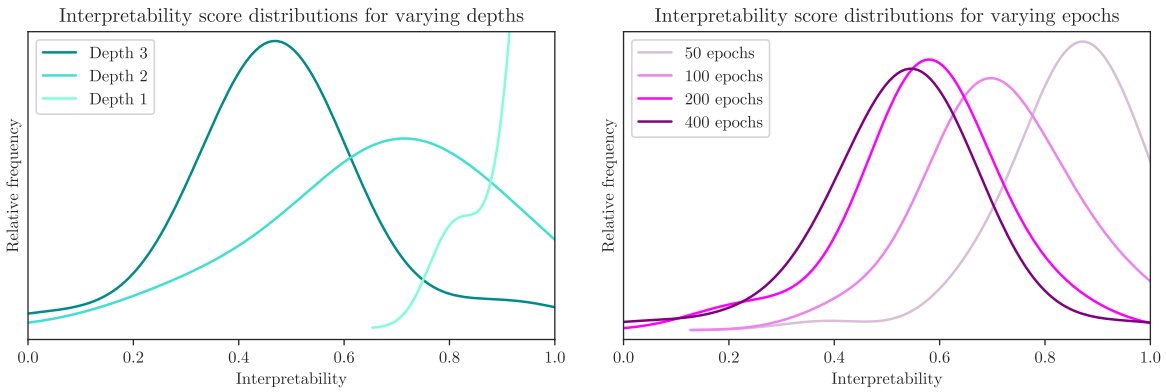


Figure 4.6: Understanding how depth and duration of training affect concepts. The left figure shows interpretability distributions at varying *depths*, while the right shows interpretability distributions at varying *epochs*.

**Concepts at different layers.** Using the MUTAG model, I probe the first two layers and observe the interpretability spectrum, which is shown in Figure 4.6. In earlier layers, the concepts tend to be more interpretable, with concepts in the first layer being almost perfectly interpretable. This is not surprising and corroborates with what was previously known about deep models. An important point is that the useful concepts (carbon rings and $NO_2$ detectors) do not emerge until the second layer.

**Concepts at different training epochs.** I train three-layer GIN models for 50, 100, 200 and 400 epochs. For each model I extract concepts from the final layer. I notice a similar relationship between the duration of training (in terms of the number of epochs) compared with the activation depths. Training a model for longer causes the concepts to be less interpretable in general. Note that in some cases, there is a small difference in test set performance, despite a large difference in concept interpretability. For example, both the models trained for 100 epochs and 400 epochs have the same test accuracy of 87%, but one has significantly more interpretable concepts, likely because the model trained for 400 epochs shows signs of overfitting. This suggests that there may be some general tradeoff between interpretability and model accuracy even when the same model architecture is used. I leave this idea for future investigation.

### 4.2.4 To what extend do our concepts serve as model-level explanations?

Recall that a *model-level* (global) explanation is one that aims to explain how a model reaches its decision in *general*, rather than for specific instances. The concepts we investigate, in short, can be described as human-interpretable units that closely resemble the activation patterns of a black-box GNN. Since our concept-extraction process fundamentally acts over an entire training set of graphs $\mathcal{D}$ that accurately represents the real distribution of graphs the model will see in practice, it is clear that our concepts

are global. In addition, the concepts correspond to *neurons*, which are the fundamental decision-making mechanisms of the model and behave in the same deterministic manner regardless of the input graph. This strongly suggests that the concepts are representative of the model in a global sense, rather than local. The current state-of-the-art baseline for

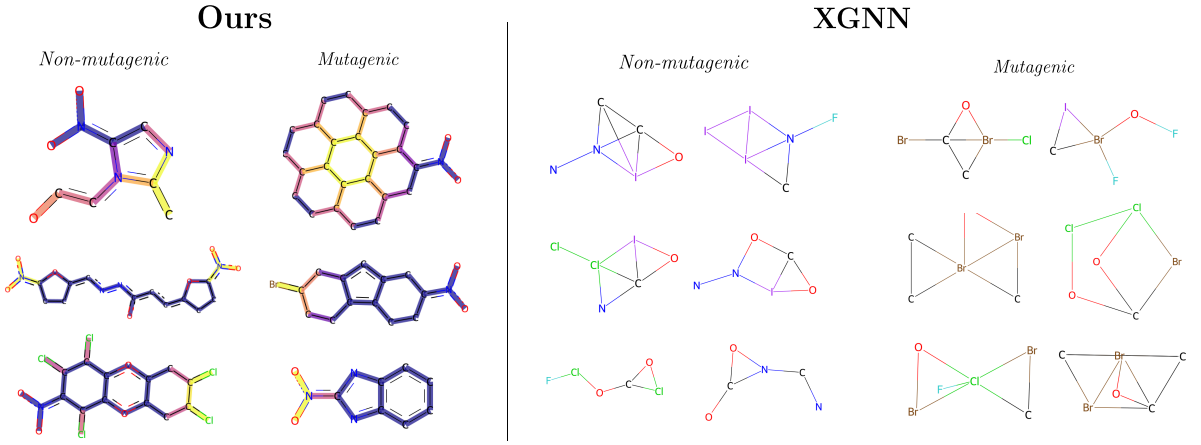


Figure 4.7: Comparison of class-specific model-level explanations against XGNN. Each of our graphs corresponds to the graph in the dataset which maximises a neuron that is important for the class. Each XGNN graph is generated using reinforcement learning.

class-specific model-level explanations for GNNs is XGNN[1][6, 4], which produces class-specific global explanations via *graph generation*. We compare against XGNN in terms of the model-level explanations produced by both methods. We make two important observations from the qualitative comparison:

- While XGNN is able to generate a 'prototypical' graph maximising a certain class, the generated graph typically does not contain all the concepts which have been extracted by our method. For example, we do not see the appearance of carbon-ring concepts for the MUTAG class *non-mutagenic*. Similarly, we do not see the *few-to-many* concept for Reddit-Binary. The graphs generated by XGNN are also very different from the typical graphs for the task and in many cases do not resemble anything the model would receive as input in practise.

- XGNN has to generate multiple graphs to get an idea of how a class is maximised. There is no guarantee that each generated graph corresponds to a separate unit of decision making (unlike concepts). Furthermore, the explanations come only in the form of the genenerated graphs, with no additional description such as logical formulas as in our method. I argue that this makes XGNN more subjective to human bias compared to our approach.

[1]There have been other model-level methods proposed in recent months, but non have received the same amount of attention and experimental validation as XGNN. GCExplainer is model-level, but does not explain specific classes.

## 4.3 Instance-level explanations

### 4.3.1 Qualitative overview

The concept extraction pipeline can be naturally extended to provide visual explanations at the instance level in the form of edge masks. Recall from Section 3.2 that for a given model $\mathcal{M}$ and input graph $G = (E, V)$ we produce an edge mask $\mathcal{E}(\mathcal{M}, G) \in \mathbb{R}^{|E|}$ highlighting the important edges with higher values. Whereas previous approaches output a single edge mask for a prediction, we are able to output an edge mask for each concept. In 3.2 I proposed methods to find subsets of concepts that explain a prediction using few interpretable concepts. I provide an example explanation in Figure 4.8.

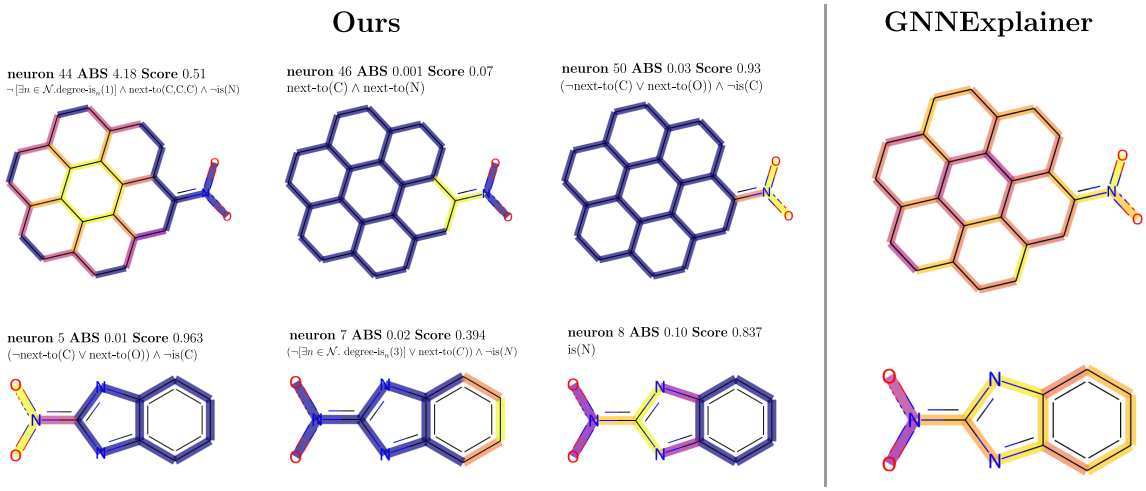


Figure 4.8: Comparison of instance-level explanations with GNNExplainer.

My method is able to provide several edge masks instead of a single one. This disentangles the explanation into constituent parts. For example, we can decompose the prediction of a compound as mutagenic into the composition of the carbon ring and $NO_2$ concepts. Similarly, the prediction of a Reddit thread as a Q&A type can be explained as the composition of both having long message threads *and* having a few-to-many substructure. I also provide concept-important scores along with each concept to indicate to the user the relative importance of each concept. I use the absolute contribution method to select the top concepts.

To my knowledge, no existing method has attempted to decompose an edge mask explanation into concepts. My proposed method of producing explanations aligns with the concept-based paradigm of utilising human-understandable categories instead of raw features and, in a sense, can improve plausibility of explanations from a human perspective.

Compared to the edge masks produced by GNNExplainer, we observe a significant difference in quality:

- First, my explanations are more focused in local regions, whereas GNNExplainer in many cases appears to highlight irrelevant parts of the graph. For example, in a

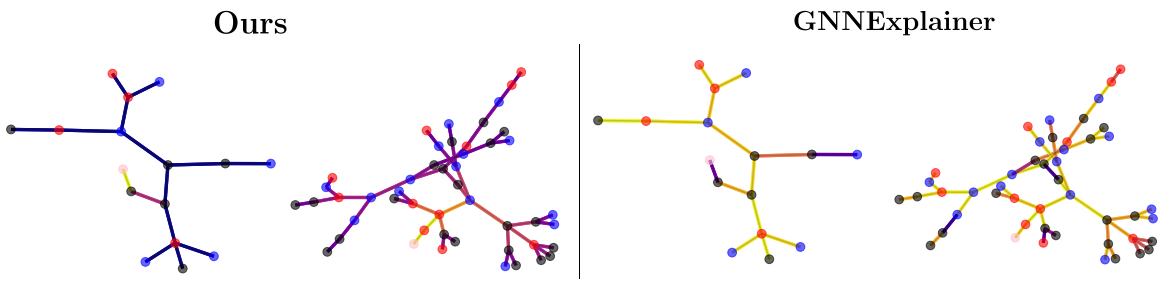


Figure 4.9: Comparing edge masks on the synthetic ThreeHops dataset, where the goal is to determine if the special node (in pink) is within three hops of a red node.

MUTAG graph with seven carbon rings, we attribute the central ring as the most important, likely because removing that single ring would simultaneously break all rings in the graph, whereas removing any other ring would not. GNNExplainer, on the other hand, highlights all rings. I attribute this to the fact that the optimisation objective of GNNExplainer may be difficult to converge to, especially since it relies on a convexity assumption that does not hold for GNNs.

- In addition, my decomposition approach is superior for input graphs where most of the edges are important, since we are able to separate out the concepts. GNNExplainer and virtually all other existing approaches would most likely highlight most of the graph, which is not useful to the user.

The statement I make in the first point is quite bold, especially since GNNExplainer is a commonly used explanation method. Therefore, I further investigate this by constructing a synthetic dataset. I call this dataset **ThreeHops**, which consists of 2000 randomly generated trees of depth 5 with an average size of 30. Each tree has a node called the *special node*, and all other nodes are black, red, or blue. The goal is to classify if there is a red node within three hops from the special node. Using a train-test split of 50:50, I find that three GIN layers are capable of achieving above 99% test accuracy. Upon evaluating the instance-level explanations, I find that my approach is able to highlight the region around the special node, whereas GNNExplainer appears to highlight edges without any interpretable basis. More examples are shown in Appendix B.1.

### 4.3.2 Are the explanations faithful?

Although my concept-based explanations are sensible from a qualitative inspection, a numerical way of evaluating them is desired. I choose to use the fidelity metric – an explanation has high fidelity if removing the explanation from the input graph significantly alters the prediction. This has been used before in the literature [62]. To compute the fidelity of an explanation, we greedily remove edges from the input $G$ according to the mask $\mathcal{E}$ and record how much the model prediction changes using the metric $(y - y')/y$ where $y$ is the initial prediction and $y'$ is the new prediction. The predictions are taken as the softmax probabilities – therefore, we assess not only if the explanation is important

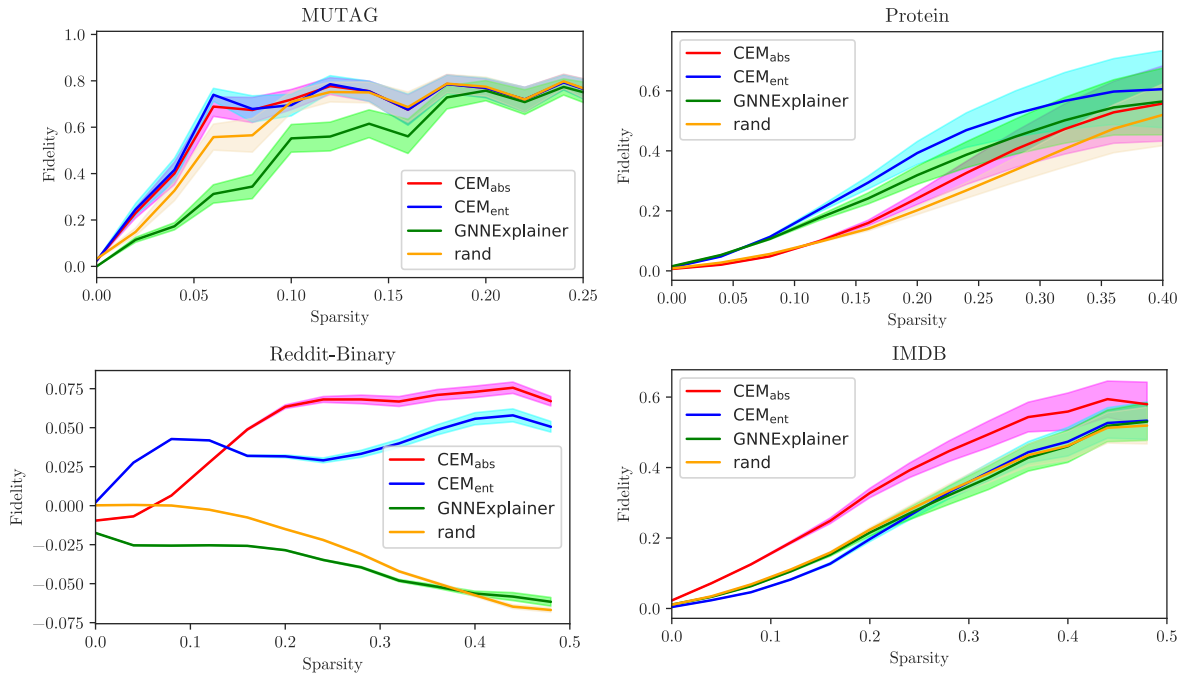


Figure 4.10: Fidelity evaluation on benchmark datasets. The area around each line indicates the variance. Fidelity is defined as the percentage change in the logit of the predicted class after a certain percentage (the sparsity level) of the graph's edges is removed.

for the predicted class, but also its role in suppressing the predictions of the other classes. I plot the fidelity score as a function of *sparsity* - the percentage of the input that has been pruned. The fidelity score over a subset of graphs in the dataset and the mean fidelity at each sparsity level is computed.

I use the fidelity metric to evaluate the instance-level explanation method proposed in Section 3.2. I refer to my masking method as CEM (concept edge masks) and evaluate my two proposed schemes: (1) $\text{CEM}_{\text{abs}}$, the approach of ranking concepts according to the absolute weight contribution, and (2) $\text{CEM}_{\text{ent}}$, the entropy-based learning-based approach. GNNExplainer [5] is used as a baseline. In addition, unlike prior work, I include a random baseline in which the edge mask is completely randomised to give a more absolute indiciation of how good each method is. From the results in Figure 4.10, I observe that in MUTAG and Reddit-Binary, the $\text{CEM}_{\text{ent}}$ and $\text{CEM}_{\text{abs}}$ both outperform GNNExplainer, which scores lower on fidelity than the random explainer. On Reddit-Binary, although $\text{CEM}_{\text{abs}}$ performs better for the high sparsity regime, $\text{CEM}_{\text{ent}}$ dominates the low sparsity regime, which is perhaps more important because edge-mask explanations should ideally be small. On the Proteins task, GNNExplainer performs slightly better and shows superior fidelity to the random explainer. The $\text{CEM}_{\text{ent}}$ method still shows the highest fidelity and has an advantage over $\text{CEM}_{\text{abs}}$. On IMDB, $\text{CEM}_{\text{abs}}$ dominates the other three approaches. This is likely due to the IMDB task having concepts that vary greatly in size (the number of nodes in which the concept is activated): the high-degree concepts have a large presence over the edges in the graph, whereas the low-degree

concepts cover only a small proportion of the edges. For the IMDB task, the low-degree concepts typically have greater absolute contribution, which leads to $\text{CEM}_{\text{abs}}$ giving them higher priority.

I attribute the higher performance of my approaches to the fact that they are based on neuron activations, rather than being based on the model-agnostic principle of treating the entire model as a black box. The $\text{CEM}_{\text{ent}}$ method marginally outperforms $\text{CEM}_{\text{abs}}$, likely because looking at absolute contributions can be misleading. Recall that I observe concepts that have positive absolute contributions for both classes. In this case, even if a concept has a positive contribution to the target class, it may have greater contribution to the other class. In addition, I note that my application of the entropy-based learning objective is more realistic, since the loss surface is likely simpler due to the mask being learnt over the concept layer rather than the original graph.

In general, I can state that my proposed methods are capable of matching or exceeding the fidelity of GNNExplainer, and also offer the benefit of being concept-based.

## 4.4 Can concepts be used to make more transparent models?

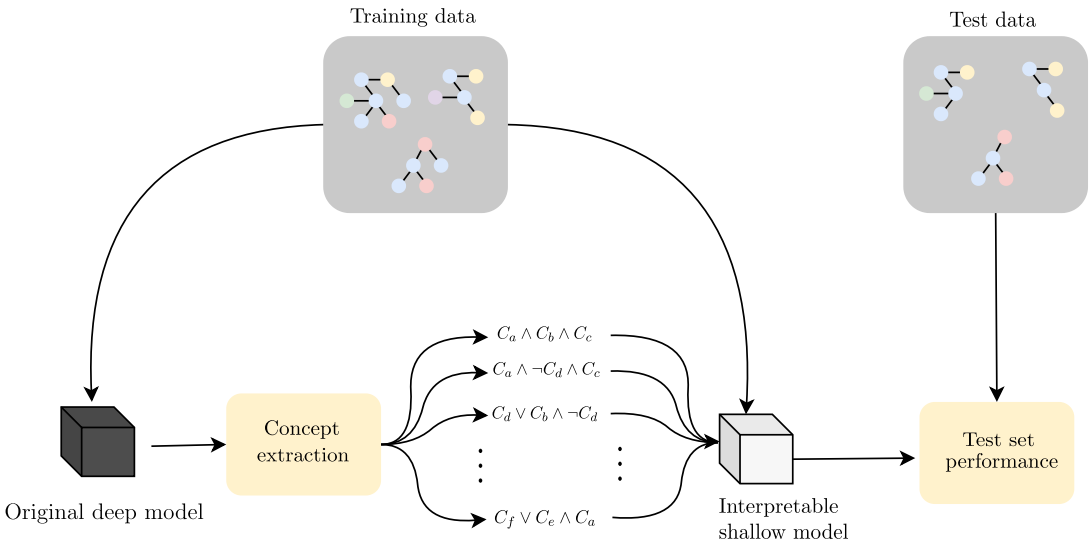


Figure 4.11: Evaluation framework for concept distillation.

In this section, I present my findings in transferring extracted concepts from black-box GNN models to produce more interpretable models. I perform experiments using the framework proposed in 3.3. For each black-box model I trained to evaluate the concept extraction method, I take the logical forms of the concepts as interpretable feature extractors and train a white-box model using three types of downstream classifiers: a single layer perceptron, random forests, and logic-explained networks (LENs). The performance of these white-box models is evaluated using the test set. I report my figures in Table 4.2.

| | MUTAG | PROTEINS | IMDB-B | REDDIT-B |
|---|---|---|---|---|
| *Original* | $\mathbf{88.7 \pm 3.4}$ | $\mathbf{76.7 \pm 2.0}$ | $\mathbf{74.3 \pm 2.5}$ | $86.1 \pm 0.5$ |
| concepts+perceptron | $84.3 \pm 3.1$ | $61.0 \pm 0.1$ | $72.4 \pm 3.7$ | $82.7 \pm 1.5$ |
| concepts+forest | $82.8 \pm 3.1$ | $75.3 \pm 0.2$ | $\mathbf{73.0 \pm 1.4}$ | $\mathbf{88.1 \pm 1.1}$ |
| concepts+LEN | $\mathbf{86.8 \pm 1.5}$ | $74.8 \pm 1.3$ | $69.3 \pm 1.3$ | $87.5 \pm 0.6$ |
| GRAPH2VEC | $55.4 \pm 1.5$ | $39.3 \pm 2.4$ | $51.4 \pm 2.4$ | $68.6 \pm 4.2$ |
| FEATHER | $63.2 \pm 6.4$ | $68.3 \pm 1.9$ | $72.9 \pm 1.3$ | $70.6 \pm 1.7$ |
| LDP | $80.9 \pm 1.5$ | $75.0 \pm 2.4$ | $70.9 \pm 0.6$ | $77.6 \pm 0.5$ |
| GCNCONV | $77.9 \pm 1.5$ | $\mathbf{76.9 \pm 0.7}$ | $52.0 \pm 4.1$ | $\mathbf{88.3 \pm 1.6}$ |
| SAGECONV | $80.9 \pm 4.4$ | $76.6 \pm 0.8$ | $51.3 \pm 5.4$ | $75.4 \pm 0.4$ |
| GATCONV | $77.0 \pm 3.1$ | $76.7 \pm 0.6$ | $52.1 \pm 4.1$ | $75.5 \pm 0.4$ |

Table 4.2: Comparison of performance of concept-based white-box models against black-box baselines. Standard deviations are reported.

I compare against both graph-embedding based methods and GNN-based methods as baselines. All methods are evaluated using the same train-test split, and I perform the experiments three times using different random seeds. The graph embedding methods are GRAPH2VEC [63], FEATHER [64] and LDP [65], which extract a embedding vector given an input graph. The implementations are obtained from the library `karateclub` [66], and are models are trained by using the graph embeddings as input features to a support vector machine. The GNN-based models are trained in the same conditions as the original baseline, but with the GNN layers swapped and any hyperparameters adjusted to increase model performance.

For each dataset, the top two results are highlighted. The concept-based white-box models have only marginally worse performance than the original model, and significantly outperform graph-embedding methods. Compared to GNN models, white-box models achieve similar performance, but with *considerably higher interpretability*. The datasets used are all popular choices for benchmarking and comparing different GNNs, but it appears that even combinations of simple logical rules can achieve similar or better performance if these rules are discovered in the right way. Furthermore, the white-box models are not only interpretable, but they have far fewer parameters.

## 4.5 Summary

I have evaluated three components of my work: concept discovery, concept-based instance-level explanations, and concept distillation. My evaluation shows that the proposed methods have the ability to outperform previous baselines and yield new insights.

# Chapter 5

# Conclusions

## 5.1 Contributions

This project has attempted to open a novel area of GNN explainability research and has made several contributions.

- I perform an analysis of the behaviour of individual GNN neurons and have given significant empirical evidence that individual neurons can learn to detect distinct concepts. I show that there are major differences between the behaviour of GNN neurons compared what has been previously observed in vision and language models. The concept extraction can be used to produce model-level explanations and unlike previous works such as XGNN, my approach reveals the decision-making of a GNN to a greater degree.
- I propose a concept-based approach of producing instance-level explanations and show both visually and quantitatively that it is superior to GNNExplainer across multiple datasets in terms of usefulness to the user, and also in terms of explanation fidelity.
- I propose to create explainable models on graphs via a new paradigm: transferring interpretable concepts from a black box model to a transparent model via a concept distillation method. I show that this can produce transparent models with only minor performance reduction compared to GNNs.

## 5.2 Future work

There is still much work to be done in the area of GNNs and concepts. The methods which I have proposed rely on one important procedure: the discovery and extraction of concepts. I have shown that this can be done by combining simple logical predicates acting over the neighbourhood of nodes, but I believe that even greater insights may be

revealed if more base concepts are explored: for example, subgraphs related to semantic information for NLP tasks, or subgraphs representing some visual concepts for graph-based computer vision tasks. These have not been explored in my work as they likely require significant amounts of manual labelling, but I believe that if such an experiment were to be carried out, it would lead to novel insights.

Next, I would like to point out some interesting directions of research that have not yet been attempted or have received little attention. This work has mainly focused on explaining models post-hoc and producing transparent models using the insights found, but I believe there may be significant benefit from the middleground of training GNNs that are more interpretable but are still fundamentally GNNs. For instance, one could invent a new loss function that encourages the concepts to be more interpretable, or encourage the concepts to be more distinct. In fact, there is already work on designing GNNs that learn *disentangled* representations, where the neuron activations are less correlated with each other [67]. An investigation into the application of this for explainable GNNs would be worthwhile.

Although concept-based explainability inherently aims to improve the ethics of AI in society, it does pose potential risks. For example, an attacker can use the presence of spurious concepts to create adversarial examples that can be used to trick a machine learning system [45]. The discovery of individual concepts can be used to identify properties about the training data used, and potentially lead to data privacy issues. Therefore, I believe that there is great value in researching the relationship between concept-based explainability and other areas such as differential privacy and adversarial robustness.

On a final note, explainability is no doubt an important area. It strives to not only increase human trust in AI in high-stake areas such as medicine, finance, and judicial systems, but will also open up new insights and help demystify machine learning methods. I truly hope this work will bring us one step closer to a future where AI is used responsibly, transparently, and for the benefit of humanity.

# Bibliography


[1] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37: 1–12, 2020. ISSN 1740-6749. doi: https://doi.org/10.1016/j.ddtec.2020.11.009. URL `https://www.sciencedirect.com/science/article/pii/S1740674920300305`.

[2] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. *ArXiv*, abs/2002.09405, 2020.

[3] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. Graph neural networks for natural language processing: A survey. *ArXiv*, abs/2106.06090, 2021.

[4] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Jun Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *ArXiv*, abs/2204.08570, 2022.

[5] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating Explanations for Graph Neural Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf`.

[6] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: Towards Model-Level Explanations of Graph Neural Networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[7] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*, 2018.

[8] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. *ArXiv*, abs/2007.04612, 2020.

[9] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network Dissection: Quantifying Interpretability of Deep Visual Representations. In *Computer Vision and Pattern Recognition*, 2017.

[10] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James R. Glass. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *AAAI*, 2019.

[11] Alvin Rajkomar, Jeffrey Dean, and Isaac Kohane. Machine learning in medicine. *The New England Journal of Medicine*, 2019.

[12] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 368–377, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219846. URL https://doi.org/10.1145/3219819.3219846.

[13] Machine learning (in finance), Jan 2022. URL https://corporatefinanceinstitute.com/resources/knowledge/other/machine-learning-in-finance/.

[14] A short history of deep learning – everyone should read. https://www.forbes.com/sites/bernardmarr/2016/03/22/a-short-history-of-deep-learning-everyone-should-read/?sh=7b470de35561. Accessed: 2022-05-20.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[16] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215, 2018. doi: 10.23919/MIPRO.2018.8400040.

[17] Julia Amann, Alessandro Blasimme, Effy Vayena, Dietmar Frey, and Vince I Madai. Explainability for artificial intelligence in healthcare: a multidisciplinary perspective. *BMC Medical Informatics and Decision Making*, 2020.

[18] Ambreen Hanif. Towards explainable artificial intelligence in banking and financial services. *ArXiv*, abs/2112.08441, 2021.

[19] Ashley Deeks. The judicial demand for explainable artificial intelligence. *Virginia Public Law and Legal Theory Research Paper No. 2019-51*, 2019.

[20] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.

[21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

[22] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889*, 2021.

[23] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[24] Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375, 2018.

[25] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[26] Petar Velivckovic. Message passing all the way up. 2022.

[27] Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv*, abs/1609.02907, 2017.

[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? *ArXiv*, abs/1810.00826, 2019.

[29] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. Graph Attention Networks. *ArXiv*, abs/1710.10903, 2018.

[30] Michael Bronstein. Graph Neural Networks beyond Weisfeiler-Lehman and vanilla Message Passing. `https://towardsdatascience.com/graph-neural-networks-beyond-weisfeiler-lehman-and-vanilla-message-passing-bc8605` 2022. [Accessed: 10/4/2022].

[31] Graph attention networks. `https://petar-v.com/GAT/`. Accessed: 2022-05-25.

[32] Petar Velickovic. Theoretical Foundations of Graph Neural Networks. `https://www.youtube.com/watch?v=uF53xsT7mjc`. [Accessed: 10/4/2022].

[33] Interpretable machine learning. `https://christophm.github.io/interpretable-ml-book/taxonomy-of-interpretability-methods.html`. Accessed: 2022-05-20.

[34] David Alvarez-Melis and T. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*, 2018.

[35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[36] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf`.

[37] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In *ICANN*, 2016.

[38] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016.

[39] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep

networks via gradient-based localization. *International Journal of Computer Vision*, 128:336–359, 2019.

[40] Valérie Beaudouin, Isabelle Bloch, David Bounie, Stéphan Clémençon, Florence d'Alché Buc, James R. Eagan, Winston Maxwell, Pavlo Mozharovskyi, and Jayneel Parekh. Flexible and context-specific ai explainability: A multidisciplinary approach. *ArXiv*, abs/2003.07703, 2020.

[41] Amirata Ghorbani, James Wexler, James Y. Zou, and Been Kim. Towards automatic concept-based explanations. In *NeurIPS*, 2019.

[42] Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Li'o, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. *ArXiv*, abs/2106.06804, 2021.

[43] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015. doi: 10.1109/ACPR.2015.7486599.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[45] Jesse Mu and Jacob Andreas. Compositional Explanations of Neurons. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17153–17163. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/c74956ffb38ba48ed6ce977af6727275-Paper.pdf`.

[46] Bernease Herman. The Promise and Peril of Human Evaluation for Model Interpretability. *ArXiv*, abs/1711.07414, 2017.

[47] Cynthia Rudin. Please Stop Explaining Black Box Models for High Stakes Decisions. *ArXiv*, abs/1811.10154, 2018.

[48] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 33, 2020.

[49] Minh N. Vu and My T. Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *ArXiv*, abs/2010.05788, 2020.

[50] Xiang Wang, Yingxin Wu, An Zhang, Xiangnan He, and Tat-Seng Chua. Towards multi-grained explainability for graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18446–18458. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/99bcfcd754a98ce89cb86f73acc04645-Paper.pdf`.

[51] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489.

[52] Dobrik Georgiev, Pietro Barbiero, Dmitry Kazhdan, Petar Velivckovi'c, and Pietro Lio'. Algorithmic concept-based explainable reasoning. *ArXiv*, abs/2107.07493, 2021.

[53] Mateo Espinosa Zarlenga, Zohreh Shams, and Mateja Jamnik. Efficient decompositional rule extraction for deep neural networks. *ArXiv*, abs/2111.12628, 2021.

[54] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

[55] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems, 2018. URL `https://arxiv.org/abs/1802.04687`.

[56] Jianping Gou, B. Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *ArXiv*, abs/2006.05525, 2021.

[57] 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. *Journal of Medicinal Chemistry*, 1991.

[58] Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783417. URL `https://doi.org/10.1145/2783258.2783417`.

[59] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, S V N Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 2005.

[60] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 2005.

[61] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1:161 – 187, 2017.

[62] Mohit Bajaj, Lingyang Chu, Zihui Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. Robust counterfactual explanations on graph neural networks. In *NeurIPS*, 2021.

[63] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *ArXiv*, abs/1707.05005, 2017.

[64] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020.

[65] Chen Cai and Yusu Wang. A simple yet effective baseline for non-attribute graph classification. *ArXiv*, abs/1811.03508, 2018.

[66] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 3125–3132. ACM, 2020.

[67] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4212–4221. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ma19a.html.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

# Appendix A

# Details of experimental setup

## A.1 Software and hardware

| Component | Configuration |
|---|---|
| CPU | i7 8700k @ 4.90GHz |
| Memory | 32 GB |
| GPU | NVIDIA RTX 3080 Founders Edition |
| Operating system | Windows 10 |
| Development environment | Pycharm 2021.3 |
| Python version | 3.8.2 |
| PyG version | 1.8 |
| PyTorch [68] version | 2.4.0 |
| CUDA version | 11.2 |
| DIG version | `https://github.com/divelab/DIG/commit/83eefb10fa0d5e690dfabd352ef197bdf249d229` |
| karateclub version | 1.2.3 |
| torch-explain version | 0.7.0 |

Table A.1: Experimental setup details.

## A.2 Black-box model architectures and training

All GNN models I try to explain consist of a stack of $N$ GNN layers (with latent representation size $D$), followed by a global pooling operation and a single fully-connected layer to produce the output logits. Models are trained using the Adam optimiser with a L2 weight decay. The exact details of the architecture and training parameters for each model is shown in Table A.2. All models are trained using cross-entropy loss.

| | MUTAG | PROTEINS | IMDB-B | REDDIT-B |
|---|---|---|---|---|
| $N$ | 3 | 3 | 2 | 3 |
| $D$ | 64 | 64 | 64 | 64 |
| Global pooling | Add | Add | Add | Add |
| Layer type | GIN | GIN | GIN | GCN |
| Output logits | 2 | 2 | 2 | 2 |
| Learning rate | 10e-4 | 10e-4 | 10-e4 | 10-4 |
| Epochs | 2000 | 2000 | 1000 | 1000 |
| L2 decay | 10e-4 | 10e-4 | 10-e4 | 10-4 |
| Early stop | N/A | 600 | 500 | 400 |
| Batch size | 32 | 32 | 32 | 32 |

Table A.2: Exact details of GNN models used to produce our experimental results.

## A.3 Atomic concepts

The set of atomic base concepts for each dataset is shown in Table A.3. I provide short explanations of the notation.

| **Task** | **Atomic concepts** |
|---|---|
| MUTAG | is(C) is(N) is(O) is(Cl) is(Br) is(I) is(F) next-to(C) next-to(N) next-to(O) next-to(Cl) next-to(Br) next-to(I) next-to(F) next-to(C,C) next-to(C,C,C) degree-is(1) degree-is(2) degree-is(3) degree-is(4) nb-degree-is(1) nb-degree-is(2) nb-degree-is(3) |
| Proteins | is(A) is(B) is(C) next-to(A) next-to(A,A) next-to(A,A,A) next-to(B) next-to(B,B) next-to(B,B,B) next-to(C) next-to(C,C) next-to(C,C,C) next-to(C,C,C,C) next-to(C,C,C,C,C) nb-next-to(A) nb-next-to(A,A) nb-next-to(A,A,A) nb-next-to(B) nb-next-to(B,B) nb-next-to(B,B,B) nb-next-to(C) nb-next-to(C,C) nb-next-to(C,C,C) |
| IMDB-B | degree-greater(1), degree-greater(3), . . . , degree-greater(99) nb-degree-greater(10,1) nb-degree-greater(10,2) nb-degree-greater(10,3) nb-degree-greater(10,4) nb-degree-greater(10,5) nb-degree-greater(30,1) nb-degree-greater(30,2) nb-degree-greater(30,3) nb-degree-greater(30,4) nb-degree-greater(30,5) |
| Reddit-Binary | degree-greater(1), degree-greater(4), . . . , degree-greater(99) nb-degree-greater(5,1) nb-degree-greater(5,2) nb-degree-greater(10,1) nb-degree-greater(10,2) nb-degree-greater(30,1) nb-degree-greater(30,2) nb-degree-equal(1,1) nb-degree-equal(1,2) nb-degree-equal(1,3) nb-degree-equal(1,4) nb-degree-equal(1,10) |

Table A.3: Atomic concepts.

## A.4 Configurations used to make fair comparisons against existing work

For GNNExplainer, I use the official torch-geometric implementation. Since GNNExplainer has a tuneable learning rate, I try my best to choose a learning rate that ensures

GNNExplainer performs the best on the evaluation benchmarks.

For XGNN, I use the official implementation from DIG ('Dive Into Graphs') which is an open-source library containing recent state-of-the-art methods related to GNNs. The library is created by the same research group that authored XGNN. The default code contains model architectures for the reinforcement learning agent used to perform graph generation, which I do not modify. The code by default supports only a certain type of model to explain, so to make it compatible with my own models, I perform slight modification of the code. I ensure that the base method is not changed.

# Appendix B

# Additional explanations

## B.1 Comparison to GNNExplainer on ThreeHops

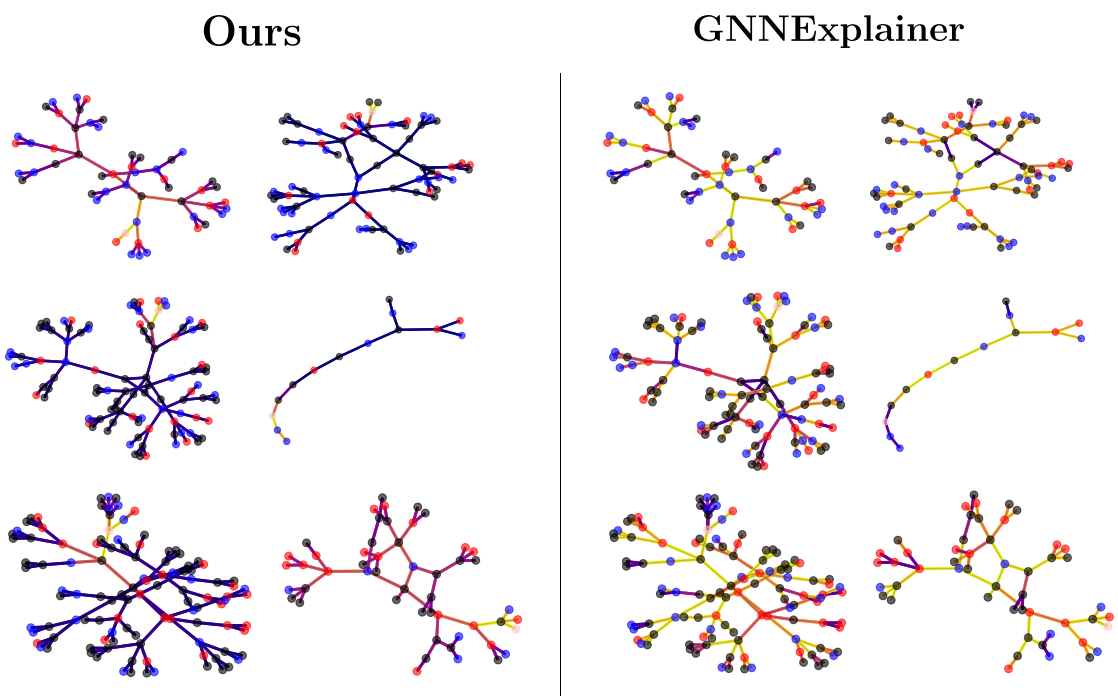


Figure B.1: Additional results on the ThreeHops sythetic dataset.