# Understanding how explainers work in graph neural networks

Antonio Longa[1,2][0000−0003−0337−1838], Steve Azzolin[2], Gabriele Santin[1][0000−0001−6959−1070], Giulia Cencetti[1][0000−0002−6946−3666], Pietro Liò[3][0000−0002−0540−5053], Bruno Lepri[1][0000−0003−1275−2333], and Andrea Passerini[2][0000−0002−2765−5395]

[1] Fondazione Bruno Kessler, Trento, Italy
`{alonga,gsantin,gcencetti,lepri}@fbk.eu`
[2] University of Trento, Italy `andrea.passerini@unitn.it`
`steve.azzolin@studenti.unitn.it`
[3] University of Cambridge, Cambridge, United Kingdom `pl219@cam.ac.uk`

**Abstract.** Following a fast initial breakthrough in graph based learning, Graph Neural Networks (GNNs) have reached a widespread application in many science and engineering fields, prompting the need for methods to understand their decision process. GNN explainers have started to emerge in recent years, with a multitude of methods both novel or adapted from other domains. To sort out this plethora of alternative approaches, several studies have benchmarked the performance of different explainers in terms of various explainability metrics. However, these earlier works make no attempts at providing insights into why different GNN architectures are more or less explainable, or which explainer should be preferred in a given setting. In this work, we fill these gaps by devising a systematic experimental study, which tests ten explainers on eight representative architectures trained on one carefully designed graph classification dataset. With our results we provide key insights on the choice and applicability of GNN explainers, we isolate key components that make them usable and successful and provide recommendations on how to avoid common interpretation pitfalls. We conclude by highlighting open questions and directions of possible future research.

**Keywords:** Graph Neural Networks · Explainability.

## 1 Introduction and Motivation

Graph Neural Networks (GNNs) have emerged as the de-facto standard for graph-based learning tasks. Regardless of their apparent simplicity, that allows most GNN architectures to be expressed as variants of Message Passing [28], i.e., exchanging messages between nodes, GNNs have proved extremely effective in preserving the natural symmetries present in many real-world physical systems [81, 37, 57, 14, 22, 8]. The versatility of GNNs allowed them to be also applied to emulate classical algorithms [13], addressing tasks like bipartite matching [27], graph coloring [48] or the Traveling Salesperson Problem [56], and approximate symbolic reasoning tasks like propositional satisfiability [70, 69, 82] and probabilistic logic reasoning [98]. Despite recent works trying to adapt the Transformer architecture, made popular by a wide success first in language [77, 63, 58] and then in vision applications [59, 49], to the graph domain [41, 60, 93, 20, 99], the natural inductive bias of GNNs remains at the basis of the current success of GNNs. A major drawback of GNNs is the opacity of their predictive mechanism, which they share with most deep-learning based architectures. This severely limits the applicability of these technologies to safety-critical scenarios. The need to provide insights into the decision process of the network, and the need to provide explanations for automatic decisions affecting human's life [17, 40], have stimulated research in techniques for shading light into the black box nature of deep architectures [62, 71, 84, 94, 76, 74, 75, 55, 30]. The approaches have also been adapted to generate explanations for GNN models [74, 75, 55, 6]. However, networked data have peculiarities that pose specific challenges that explainers developed for tensor data struggle to address. The main challenge comes from the lack of a regular structure, as nodes have variable number of edges, which requires ad-hoc strategies to be properly addressed. Indeed, a number of approaches have been recently developed that are specifically tailored to explain GNN architectures. Yuan et al. [91] proposed a categorization of explainers into four categories: *gradient-based* which exploit gradients of the input neural network [75, 76, 55]; *perturbation-based* where perturbations of the input graphs are aimed at obtaining explainable subgraphs [87, 52, 23, 66]; *decomposition-based* which try to decompose the input identifying the explanations [6, 55, 67]; and *surrogate-based* where a simple interpretable surrogate model is used to explain the original neural network [36, 97, 80].

It is often the case, however, that each work proposes a new set of benchmarks or metrics, making the comparison across works complicated. We thereby stress the need for a comprehensive evaluation that can fairly benchmark the explainers under a unified lens. One of the first attempts to provide such a comparative analysis is the up mentioned work by Yuan et al. [91], where a taxonomy of the available explainers was proposed. In addition to this, the authors reported a detailed overview of the most common datasets used to benchmark explainers, along with the adopted evaluation metrics.

However, despite the wide coverage of explainers, datasets, and evaluation metrics, only a single GNN architecture, namely a simple Graph Convolutional Network [42], was evaluated, so that nothing can be said about the impact of different architectures in the resulting explanations. A similar limitation affects the works of Zhao et al. [100] and Agarwal et al. [1, 2] that, despite presenting interesting insights in terms of consistency of explainers, desired properties of explanation metrics and even introducing a generator for synthetic graph benchmarks, focus their analysis to a single GNN architecture. Li et al. [47] conducted the first empirical study comparing different GNN architectures. However, their study is limited to node classification and the three explainers under analysis [87, 52, 66] are not well representative of the diversity of explanation strategies that have been proposed, as summarized in the aforementioned taxonomy [91]. The most comprehensive study to date is the recent work by Rathee at al. [61], that evaluated four GNN architectures over nine explainers for both node and graph classification. However, the main goal of this study is proposing a benchmarking suite to quantitatively evaluate explainers, with no attempts at providing insights into why different GNN architectures behave differently in terms of explainability, or which explainer should be preferred in a given setting.

In spite of the aforementioned recent studies benchmarking explainability methods for GNNs, no investigation has been done in characterizing the typical explanation patterns associated to the topological concepts learned by the network and how different architectures affect the explanation. In our survey[50], we address these issues, with an extensive experimental study on six different synthetic datasets. Here, for reasons of space, we focus on a specific dataset answering the following research questions:

– **RQ1:** How does the architecture affect the explanations?
– **RQ2:** How do explainers affect the explanations?

Overall, our survey[50] aims to go beyond a merely quantitative evaluation of the performance of explainer-GNN pairs and to make a significant step towards *explaining explainability*. We run an unprecedented number of experiments involving eight GNN architectures, ten instance-based explainers, and six graph calssification dataset and enrich the quantitative results we obtain by providing a deep understanding of the reasons behind the observed behaviours, together with a set of recommendations on how to select and best use the most appropriate explainer for the task under investigation while avoiding common pitfalls, as well as a number of open problems in GNN explainability that we believe deserve further investigation.

## 2 Benchmark datasets

In this section we present the graph benchmark dataset employed in our work. In designing the new benchmarks, we took inspiration from Faber et al. [21].

**Grid:** Inspired by the benchmarks presented in Ying et al. [87], the Grid dataset is composed by 1000 Barabási-Albert (BA) graphs [7]. To half of these 1000 graphs we attach a $3 \times 3$ grid, and the resulting graphs are assigned to the positive class, while the ones without grid are the negative class The number of nodes in the BA graph is a uniformly distributed random number between 15 and 30 (for the negative class) and between 6 and 21 (for the positive class). This guarantees that when adding the grid, the average number of nodes in the positive class matches the one in the negative class. It is worth mentioning that in the experiments done by Ying et al. [87], the total number of nodes is fixed. This benchmark evaluates the ability of the explainers to identify explanations consisting of a simple connected pattern.

## 3 Assessment of the explanation quality

Evaluating GNNs' explanations is a challenging task that requires to verify if and how the explainer is effective in capturing the behaviour of the model. There are two main strategies to evaluate explanation quality. The first is a *supervised* strategy [65, 87, 24], that measures the similarity of the extracted explanation with an existing ground-truth, which is assumed to be known. The second strategy is an *unsupervised* one, that measures how much the prediction of a GNN on the full graph resembles the prediction computed on the extracted explanation only, and this does not require to have a ground-truth explanation available.

We consider a metric for each of these two strategies, in order to capture different aspects of the quality of an explanation: the *plausibility* of the explanation with respect to a ground-truth concept that an accurate GNN is expected to have learned, and the *fidelity* of the explanation with respect to the prediction of the GNN to be explained. Namely, with plausibility we quantify the consistency between the explainer mask and a human-level intuition of what a plausible explanation looks like. On the other hand, fidelity measures the consistency between the model prediction on the full graph and the on the explanation subgraph, and thus it works with a sort of model-based instead of human-based ground truth.

In the following we detail the metrics we employ, namely *plausibility* ($P$) and *fidelity* ($F$), the latter further divided into its *comprehensiveness* ($F_{com}$) and *sufficiency* ($F_{suf}$) components presented in the supplementary3.

*Plausibility* Let $\overline{G}_{\text{exp}}$ be the expected ground truth for class $c \in \{0, 1, \ldots, n_c - 1\}$, represented by a copy of the original graph $G$ with an hard mask highlighting the ground truth nodes. Following [61], the plausibility $P$ of the explanation is defined as

$$P = AucROC(G_{\text{exp}}, \overline{G}_{\text{exp}}),$$

i.e., the area under the ROC curve between the computed soft mask and the ground truth hard mask.

It is clear that this metric can only be computed on benchmarks in which the ground truth explanation can be defined, and it is completely dependent on this definition. For each dataset, the ground truths that we are using to compute $P$ are defined in Section 2.

*Fidelity* To aggregate $F_{com}$ and $F_{suf}$ (formally defined in the supplementary 3) into a unique fidelity metric, for graph-classification tasks we compute what we call $f1$-*fidelity* ($F_{f1}$), which is defined by

$$F_{f1} = 2\frac{(1 - F_{suf}) \cdot F_{com}}{(1 - F_{suf}) + F_{com}}.$$

This is indeed the $f1$ score [38] between $F_{com}$ and $(1 - F_{suf})$. We use this metric in place of $F_{com}$ and $F_{suf}$.

### 3.1 Aggregation

After we evaluate any of these metrics on each (class, model, explainer)-configuration, we need an aggregation mechanisms to assign a unique score to the models and the explainers over all classes and datasets. This permits to avoid visualizing the detailed metrics over the entire set of configurations, and make the results easier to be interpreted.

To define these aggregate metrics we proceed as follows, where the same procedure are repeated for both plausibility and fidelity: *(1)* For each (class, model, explainer)-configuration we keep only the class with the highest value of the metric, i.e., the best explained class. *(2)* For a given dataset, we rank the model-explainer pairs according to the values selected in point (1). The aggregated score of each pair is the ranking number $1, 2, \dots$. *(3)* The dataset-level scoring of an architecture, of an explainer, or of a category of explainers (e.g. grad-based, or edge-based) is the average of the scores of point (2) over all the corresponding pairs.

To assess instead the stability the explanations over an entire dataset, we propose a qualitative visualization of the masks which is discussed for each experimental setting.

## 4 Experimental setting

Any explainer provides an explanation of the prediction of a given instance of a model, as it is obtained after an optimization process on a specific dataset. It is thus of paramount importance to identify the choices made in the training of the networks that will be analyzed in the following.

In the supplementary section 2 and 1, we provide an overview of the primary architectures and explainer utilized in our experiments, along with the rationale behind our selection.

We report in Table 1 the details of the networks used, the parameters used for their optimization, and the resulting train and test accuracies. For each architecture, the table shows the dimensions of the hidden layers of GNN type (column *GNN*) and of fully connected type (column *Fully conn.*), and any additional parameter used for the definition of the architecture (see Section 1 for a definition of these hyperparameters). For example, in the first row the numbers $30-30-30$ and $10-2$ mean that three GCN layers are applied, each mapping to a target dimension of 30, followed by two fully connected layers with target dimensions 10 and 2.

The table additionally reports the learning rate (column *LR*) and number of epochs used in the training, where an ADAM optimizer has been used in each case. We remark that these configurations have been chosen with the guiding principle of obtaining the simplest configuration achieving a target 0.95% train accuracy.

The last two columns of Table 1 show the resulting train and test accuracies obtained by the models trained according to these specifications. We remark that for some configurations it was not possible to achieve the desired target accuracy, so the table reports an "X" in place of the accuracy values, and it reports the values corresponding to the largest architecture which has been tested. Being not sufficiently accurate, these models have been removed from the successive analysis.

| Architecture | GNN | Fully conn. | HyperParams | LR | Epochs | Train Acc | Test Acc |
|---|---|---|---|---|---|---|---|
| GCN | 30-30-30 | 10-2 | - | 0.001 | 1500 | 0.994 | 0.998 |
| GRAPHSAGE | 30-30-30 | 10-2 | - | 0.01 | 3000 | X | X |
| GAT | 30-30-30 | 10-2 | heads = 1 | 0.01 | 3000 | X | X |
| GIN | 30-30 | 30-2 | - | 0.001 | 1000 | 1.0 | 1.0 |
| CHEB | 30-30 | 30-2 | - | 0.001 | 1000 | 1.0 | 1.0 |
| MINCUTPOOL | 32-32-32 | 32-2 | - | 0.001 | 700 | 0.92 | 0.93 |
| SET2SET | 30-30-30 | 10-2 | - | 0.001 | 1500 | 0.97 | 0.97 |
| GRAPHCONV | 30-30 | 30-2 | - | 0.001 | 500 | 1.0 | 1.0 |

**Table 1.** Configuration of the graph-classification models, and corresponding accuracies. The table reports for each dataset and each architecture the dimension, number, and hyperparameters defining the hidden layers, together with the optimization parameters, and the obtained train and test accuracies. The configuration-dataset pairs which did not reach the target 95% train accuracy are marked with an "X", and are not further analyzed in this work.

## 5 Results

### 5.1 Research questions

The comparative analysis of the behavior of the explainers is developed along the following research questions, which will apply to both node and graph classification tasks.

– **RQ1: How does the architecture affect the explanations?** This research question can be naturally divided into the following three subquestions:
  - ***RQ1.1:*** *Which is the architecture that has the best explanation?* With this question we would like to understand which is the architecture that achieves the best score, either in terms of f1-fidelity or plausibility.
  - ***RQ1.2:*** *Which is the easiest architecture to explain?* This question aims at finding what is the architecture that is well explained by the greatest number of explainers.
  - ***RQ1.3:*** *Which is the hardest architecture to explain?* In this case we want to search for an architecture that achieves the lowest score.

- **RQ2: How do explainers affect the explanations?** Even this question can be divided into subquestions, which try to cover different open problems related to state of the art GNN explainers. We identify them as follows:
  - ***RQ2.1****: Which is the explainer that explains in the best way?* Here we are interested in finding the explainer able to obtain the highest Plausibility or Fidelity.
  - ***RQ2.2****: Which is the explainer that explains the maximum number of architectures?* This aspect is particular important because we need explainers which are robust with respect to different GNN architectures.
  - ***RQ2.3****: Which is the category of explainers that provides the best explanations?* The subquestion searches for the best category of explainers. As defined by [91], we consider three macro-categories, namely gradient based (Grad), perturbation based (Pert), and decomposition based (Dec).
  - ***RQ2.4****: Which is the best mask type between node and edge?* By answering this question we investigate if there is an advantage for explainers based on node or edge importances.

  We would like to remark that **RQ2.3** and **RQ2.4** are particularly relevant for future research in GNN explainability, since they may provide actionable guidelines for the development of new explainers.

**RQ1: How does the architecture affect the explanations?** Table 2 visualizes in a compact form the answer to research questions **RQ1**, where the aggregation mechanism described in Section 3.1 has been used to identify a ranking of the architectures across all explainers, both in terms of plausibility and fidelity, and the highest ranking architecture is reported for each research question.
The architecture with the best explanation (**RQ1.1**) is GraphConv for plausibility (paired with IgEdge) and Cheb for fidelity (paired with PgExpl).
The easiest architecture to explain (**RQ1.2**) is Cheb for plausibility and Gcn for fidelity. We remark that Gcn and Set2Set work with the same underlying GNN layers (Gcn), and they differ only in the final aggregation operation (a sum in Gcn, and an LSTM in Set2Set). The better performances of Gcn are thus hinting to the fact that a different global aggregation alone is responsible for changing a network's explainability, and that linear aggregations (Gcn) are, perhaps unsurprisingly, easier to explain than non-linear ones (Set2Set). In general terms, the role of the global aggregation function and its stability across different tasks is yet to be fully understood, and has not received great attention in the explanation literature. Thus, we believe that a systematic study in this direction may be an interesting future direction of research.
Finally, from the figure it is easy to see that Gin is the most difficult network to explain (**RQ1.3**), for both plausibility and fidelity.
This stark difficulty in explaining Gin is not directly understandable, especially because it is implemented with a single-layer MLP, which does not

introduces stronger non-linearity than a simpler GCN. This aspect could be interesting to be addressed by future studies.

Plausibility

| | Grid |
|---|---|
| RQ1.1 | GraphConv |
| RQ1.2 | Cheb |
| RQ1.2 | Gin |

Fidelity

| | Grid |
|---|---|
| RQ1.1 | Cheb |
| RQ1.2 | Gcn |
| RQ1.2 | Gin |

**Table 2.** Experimental answer to **RQ1** for graph classification. The table shows the top-ranking architecture with respect to each subquestion **RQ1.1**, **RQ1.2**, **RQ1.3**. The rankings are computed with respect to the Plausibility and Fidelity metrics.

To offer an additional insight into this fine-grained behavior of Gcn, which is the easiest architecture to explain according to RQ1.2 in terms of fidelity, we pick a random element and analyze the mask provided by each explainer. Figure 1 reports these masks, where each column corresponds to a different explainer. The first five explainers return a node importance, while the last four are edge-based. In both cases, the node or edge importance is rendered by a different color intensity. We stress once again that only one random graph is shown in the figure, and thus the following discussion is of a rather qualitative nature, to be complemented with the metrics discussed in the first part of this section.
Gcn manages to achieve good results in terms of both plausibility (ranked as second) and fidelity, meaning that its explanations are both close to the expected ground truth and to the actual one used by the models to realize their prediction. This duality can be observed across the examples of Figure 1. Indeed, there are cases where the masks identify clearly the grids in Grid (Cam, GradCam, IgEdge, GradExplEdge, PgExpl). On the other hand, for many other explainer, the mask is less localized and interpretable by a human eye, but the overall high scores of this explainer suggest these explanations could still be good, at least in terms of fidelity and thus from a model perspective, even if they deviate from the expected ground truth.

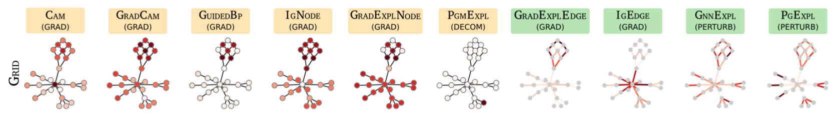

**Fig. 1.** Explanation masks (node- or edge-based) computed by the different explainers on the predictions of Gcn. Each row visualizes the mask computed for a given random graph from each dataset.

**RQ2: How do explainers affect the explanations?** The answers to this question are summarized in Table 3, where we used again the aggregation strategies defined in Section 3.1 to establish a ranking of the explainers and select the best one.
Interestingly, the overall best explainer is different for plausibility and fidelity, but it is the same when looking at the best performing one in absolute terms (**RQ2.1**), and on average across all architectures (**RQ2.2**). In fact, the highest plausibility is achieved by IgEdge, while the highest fidelity is achieved by PgExpl. Although different, they both produce an edge importance mask.
In terms of average performances (**RQ2.3**), perturbation based explainers are those that best explain all the models for both plausibility and fidelity, even if the single best ones are edge- and gradient-based (RQ2.1). The answer to this question is particularly remarkable, since perturbation-based explainers are selected as the best ones for both metrics.
When looking at the average over the entire groups (**RQ2.4**), edge-mask based explainers are clearly overperforming node-based ones, in accordance with RQ2.1 and RQ2.2. We argue that this may be due to the fact that edge-based explainers have been developed specifically for graph-explanation tasks, while node-based ones are all adaptations of existing explainers, introduced for other settings. We remark once again that this is the case only for graph classification, while for node-based tasks node-based explainers appear to be superior.[50]

Plausibility

| | Grid |
|---|---|
| RQ2.1 | IgEdge |
| RQ2.2 | GradExplEdge |
| RQ2.3 | Pert |
| RQ2.4 | Edge |

Fidelity

| | Grid |
|---|---|
| RQ2.1 | PgExpl |
| RQ2.2 | IgEdge |
| RQ2.3 | Pert |
| RQ2.4 | Edge |

**Table 3.** Experimental answer to **RQ2** for graph classification. The table reports the top-ranking explainer with respect to each subquestion **RQ2.1**-**RQ2.4**. The rankings are computed with respect to the Plausibility and the Fidelity metrics.

Similarly to the previous section, Figure 2 zooms into GradExplEdge, which is the best-ranking explainer according to RQ2.1 and with respect to plausibility. We show the results for each GNN for which GradExplEdge works, i.e., those that accept edge weights in the training phase. The high plausibility of the explainer means that it is effective in identifying the human-expected explanations in the graphs, and this is clearly visible in the examples of Figure 2: with a few exceptions, the dark red edges identify the grid in Grid.

**Grid** In the Grid dataset, the concept is a grid attached to a random Barabási–Albert (BA) network. Since the BA component is identical in the

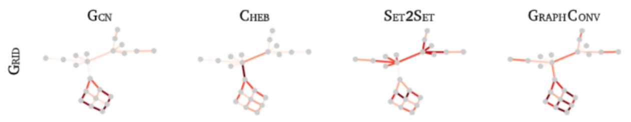

**Fig. 2.** Explanation masks computed by GradExplEdge on the predictions of the different models. Each cell visualizes the mask computed for a given random graph.

positive and negative classes, the only discriminative subgraph is the grid (or part of it). In fact, the minimal discriminant subgraph for this dataset is a square, because the BA component does not contain it.

Left panel of figure 3 visualizes the performance of each model-explanation pair when applied to this dataset. Each pair is located according to the two-dimensional coordinate given by the resulting Fidelity (horizontal axis) and Plausibility (vertical axis), and it is identified by the model name and by a color representing the explainer. We use warm colors for node-based explainers, and cold colors for edge-based ones.

This visualization permits to identify those model-explanation pairs which strike the best balance between the two scores, namely, models that maximize both plausibility and fidelity are in the top right corner of the figure. It is first relevant to observe that a clear positive correlation emerges for the top-performing pairs, in the sense that there are no cases where a high fidelity is achieved without a correspondingly high plausibility, and viceversa. Moreover, the highest plausibility is achieved by GraphConv with IgEdge, while Cheb with PgExpl obtain the highest fidelity. These are also the two Pareto-optimal pairs, i.e., any other pair reaches either a smaller fidelity or a smaller accuracy. In this sense, they are the best ones according to this evaluation.

Moreover, it is remarkable to observe that the three best performing pairs (thus including also the Gcn-GradExplEdge pair) have all edge-based explainers. This fact is in perfect accordance to the answer to RQ2.4 (Section 5.1), which identifies this type of explainers as superior to node-based ones. Observe however that Grid has no node features, and this may bias this aspect.

For these three top-performing pairs (GraphConv-IgEdge, Cheb-PgExpl, Gcn-GradExplEdge) we further investigate the quality of the explanations by quantifying their stability. Namely, we are interested in understanding how the different instances of graphs in the dataset are explained, and if there is any recurring pattern in these explanations.

This stability is shown in the right panel of Figure 3, where each edge in the grid motif is colored according to its importance averaged over all the networks in Grid, with a color scale ranging from white (for importance 0) to dark red (for importance 1). The width of each edge is instead proportional to the standard deviation of the explanation across the dataset, such that

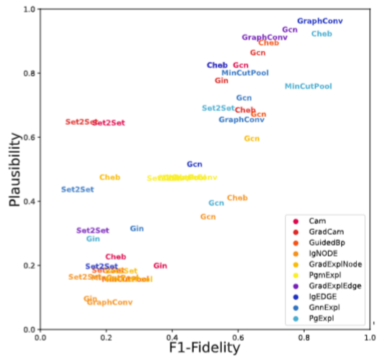
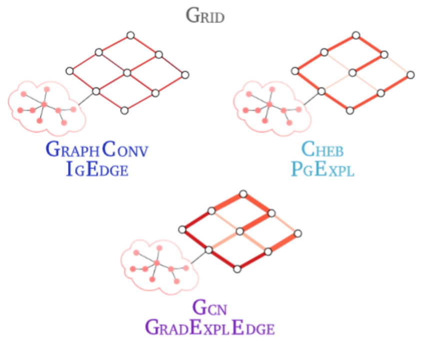

**Fig. 3.** Left: fidelity and plausibility achieved by all the model-explainer pairs when applied to Grid. In each pair the name refers to the model, while the color identifies the explainer. Right: stability of the explanations for the three top-performing model-explanation pairs. The colors identify important edges (dark red), and the edge thickness the variability of the importance in the dataset.

thicker edges describe a larger deviation, hence a smaller stability, and vice-versa.
We notice that the most stable explanation is given by GraphConv with IgEdge, since the entire grid motif is dark red and with thin edges. On the other hand, the most unstable one is Gcn with GradExplEdge, where not all edges are important, and a considerable deviation is found across the dataset (thick edges). To conclude the analysis on Grid, Figure 4 shows a prototypical explanation for each GNN, paired with its best explainer as identified by highest combination of the two metrics. Overall, we can assert that each GNN can be explained fairly well if the concept is a simple subgraph into the network. As anticipated in subsection 5.1, Gin is the hardest to explain, while the best explanations are obtained with grad and perturbation based explanations producing edge masks.

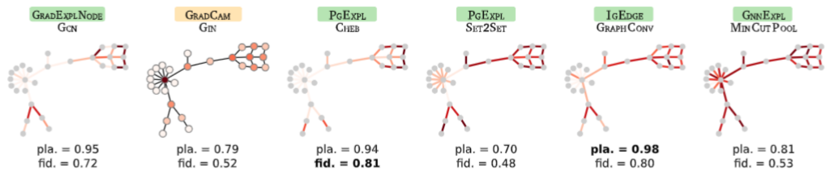

**Fig. 4.** Examples of explanations provided for each model and its highest plausibility explainer, when applied to a random sample from Grid. The plausibility and fidelity values are those of the entire dataset, as reported in Figure 3

## 6 Discussion

The main objective of this work is to experimentally study the effectiveness of explainers on different GNNs and types of data, identifying current pitfalls and formulating possible future directions in the field of GNN explainability. Given that GNNs may learn different concepts, possibly less intuitive, than those expected by humans, defining ground truths is not a trivial task and may be prone to human biases. A first simple remark is that GNNs, as neural networks in general, tend to be lazy and learn a "default" option for one of the classes. For this reason, a high accuracy does not necessarily imply having learned the ground-truth concept for the class. A useful insight that emerged from our analysis is that these human biases can often be detected by comparing plausibility and fidelity. Indeed, an explainer with high fidelity and low plausibility (or vice-versa) clearly indicates a discrepancy between what is considered to be the ground truth and the concept learned by the GNN.
Identifying a general category of explainers working consistently better than others is challenging. However, the category of explainers that best explain GNNs for graph classification are those that focus on edges, be it by perturbation or gradient. In general, edge-based explainers outperform node-based ones.
Concerning GNN architectures, there is a substantial difference in their explainability, regardless of the explainer that best suits each of them. We believe that this results is surprising yet not fully understood, given that explainers are usually aimed to be model agnostic. Given the importance of explaining predictions, it would be advisable to include explainability as a metric to be optimized when designing novel GNN architectures.

## 7 Conclusion

In this work, we proposed an extensive experimental study to quantify the effectiveness of the existing explainers and to obtain actionable recommendations to select the optimal method for a given task. For this comparison we evaluated ten explainers on eight different GNN architectures, all chosen to represent the most commonly utilized instances in a vast taxonomy of existing solutions. These methods have been tested on graph classification. As a result of our experimental study, we were able to describe significant criticalities in the common explainer evaluation methods, and to identify recurring patterns that make some category of explainers preferable in certain situations. Our findings naturally point to promising future research directions, and especially highlight once more that much has yet to be understood to achieve a satisfactory GNN explainability.

## References

1. Agarwal, C., Queen, O., Lakkaraju, H., Zitnik, M.: Evaluating explainability for graph neural networks. arXiv preprint arXiv:2208.09339 (2022)

2. Agarwal, C., Zitnik, M., Lakkaraju, H.: Probing gnn explainers: A rigorous theoretical and empirical analysis of gnn explanation methods. In: International Conference on Artificial Intelligence and Statistics. pp. 8969–8996. PMLR (2022)
3. Azzolin, S., Longa, A., Barbiero, P., Liò, P., Passerini, A.: Global explainability of gnns via logic combination of learned concepts (2022). https://doi.org/10.48550/ARXIV.2210.07147, https://arxiv.org/abs/2210.07147
4. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.R.: How to explain individual classification decisions. The Journal of Machine Learning Research **11**, 1803–1831 (2010)
5. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
6. Baldassarre, F., Azizpour, H.: Explainability techniques for graph convolutional networks. arXiv preprint arXiv:1905.13686 (2019)
7. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science **286**(5439), 509–512 (1999)
8. Basu, S., Gallego-Posada, J., Viganò, F., Rowbottom, J., Cohen, T.: Equivariant mesh attention networks. arXiv preprint arXiv:2205.10662 (2022)
9. Bianchi, F.M., Grattarola, D., Alippi, C.: Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning. pp. 874–883. PMLR (2020)
10. Bianchi, F.M., Lachi, V.: The expressive power of pooling in graph neural networks. arXiv preprint arXiv:2304.01575 (2023)
11. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21**(suppl_1), i47–i56 (2005)
12. Cangea, C., Veličković, P., Jovanović, N., Kipf, T., Liò, P.: Towards sparse hierarchical graph classifiers. arXiv preprint arXiv:1811.01287 (2018)
13. Cappart, Q., Chételat, D., Khalil, E.B., Lodi, A., Morris, C., Veličković, P.: Combinatorial optimization and reasoning with graph neural networks. In: Zhou, Z.H. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21. pp. 4348–4355. International Joint Conferences on Artificial Intelligence Organization (8 2021). https://doi.org/10.24963/ijcai.2021/595, https://doi.org/10.24963/ijcai.2021/595, survey Track
14. Cardia, M., Luca, M., Pappalardo, L.: Enhancing crowd flow prediction in various spatial and temporal granularities. In: Companion Proceedings of the Web Conference 2022. pp. 1251–1259 (2022)
15. Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: High-performance neural networks for visual object classification. arXiv preprint arXiv:1102.0183 (2011)
16. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555 (2020)
17. Dai, E., Zhao, T., Zhu, H., Xu, J., Guo, Z., Liu, H., Tang, J., Wang, S.: A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. ArXiv **abs/2204.08570** (2022)
18. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems **29** (2016)

19. Duval, A., Malliaros, F.: Higher-order clustering and pooling for graph neural networks. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 426–435 (2022)
20. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699 (2020)
21. Faber, L., K. Moghaddam, A., Wattenhofer, R.: When comparing to ground truth is wrong: On evaluating gnn explanation methods. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 332–341 (2021)
22. Fu, X., Xie, T., Rebello, N.J., Olsen, B.D., Jaakkola, T.: Simulate time-integrated coarse-grained molecular dynamics with geometric machine learning. arXiv preprint arXiv:2204.10348 (2022)
23. Funke, T., Khosla, M., Anand, A.: Hard masking for explaining graph neural networks (2020)
24. Funke, T., Khosla, M., Rathee, M., Anand, A.: Z orro: Valid, sparse, and stable explanations in graph neural networks. IEEE Transactions on Knowledge and Data Engineering (2022)
25. Gao, H., Ji, S.: Graph u-nets. In: international conference on machine learning. pp. 2083–2092. PMLR (2019)
26. Gao, J., Gao, J., Ying, X., Lu, M., Wang, J.: Higher-order interaction goes neural: a substructure assembling graph attention network for graph classification. IEEE Transactions on Knowledge and Data Engineering (2021)
27. Georgiev, D., Liò, P.: Neural bipartite matching (2020). https://doi.org/10.48550/ARXIV.2005.11304, https://arxiv.org/abs/2005.11304
28. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. p. 1263–1272 (2017)
29. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International conference on machine learning. pp. 1263–1272. PMLR (2017)
30. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 1–42 (2018)
31. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
32. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis **30**(2), 129–150 (2011)
33. Hastie, T., Tibshirani, R., Friedman, J.: Neural Networks, pp. 389–416. Springer New York, New York, NY (2009). https://doi.org/10.1007/978-0-387-84858-7_11
34. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
35. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. Advances in neural information processing systems **33**, 22118–22133 (2020)
36. Huang, Q., Yamada, M., Tian, Y., Singh, D., Chang, Y.: Graphlime: Local interpretable model explanations for graph neural networks. IEEE Transactions on Knowledge and Data Engineering (2022)

37. Jaini, P., Holdijk, L., Welling, M.: Learning equivariant energy based models with equivariant stein variational gradient descent. Advances in Neural Information Processing Systems **34**, 16727–16737 (2021)
38. Jones, K.S., Van Rijsbergen, C.J.: Information retrieval test collections. Journal of documentation (1976)
39. Keshavarzi Arshadi, A., Salem, M., Firouzbakht, A., Yuan, J.: Moldata, a molecular benchmark for disease and target based machine learning. Journal of Cheminformatics **14** (03 2022). https://doi.org/10.1186/s13321-022-00590-y
40. Khosla, M.: Privacy and transparency in graph machine learning: A unified perspective. arXiv preprint arXiv:2207.10896 (2022)
41. Kim, J., Nguyen, T.D., Min, S., Cho, S., Lee, M., Lee, H., Hong, S.: Pure transformers are powerful graph learners. arXiv preprint arXiv:2207.02505 (2022)
42. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
43. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2017)
44. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. In: International conference on machine learning. pp. 3734–3743. PMLR (2019)
45. Li, B., Fan, Y., Sataer, Y., Gao, Z., Gui, Y.: Improving semantic dependency parsing with higher-order information encoded by graph neural networks. Applied Sciences **12**(8), 4089 (2022)
46. Li, J., Peng, H., Cao, Y., Dou, Y., Zhang, H., Yu, P., He, L.: Higher-order attribute-enhancing heterogeneous graph neural networks. IEEE Transactions on Knowledge and Data Engineering (2021)
47. Li, P., Yang, Y., Pagnucco, M., Song, Y.: Explainability in graph neural networks: An experimental survey. arXiv preprint arXiv:2203.09258 (2022)
48. Li, W., Li, R., Ma, Y., Chan, S.O., Pan, D., Yu, B.: Rethinking graph neural networks for the graph coloring problem (2022). https://doi.org/10.48550/ARXIV.2208.06975, https://arxiv.org/abs/2208.06975
49. Liu, Y., Zhang, Y., Wang, Y., Hou, F., Yuan, J., Tian, J., Zhang, Y., Shi, Z., Fan, J., He, Z.: A survey of visual transformers. arXiv preprint arXiv:2111.06091 (2021)
50. Longa, A., Azzolin, S., Santin, G., Cencetti, G., Liò, P., Lepri, B., Passerini, A.: Explaining the explainers in graph neural networks: a comparative study. arXiv preprint arXiv:2210.15304 (2022)
51. Lucic, A., Ter Hoeve, M.A., Tolomei, G., De Rijke, M., Silvestri, F.: Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In: Camps-Valls, G., Ruiz, F.J.R., Valera, I. (eds.) Proceedings of The 25th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 151, pp. 4499–4511. PMLR (28–30 Mar 2022), https://proceedings.mlr.press/v151/lucic22a.html
52. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. Advances in neural information processing systems **33**, 19620–19631 (2020)
53. Magister, L.C., Kazhdan, D., Singh, V., Liò, P.: Gcexplainer: Human-in-the-loop concept-based explanations for graph neu-

ral networks (2021). https://doi.org/10.48550/ARXIV.2107.11889, https://arxiv.org/abs/2107.11889

54. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 4602–4609 (2019)
55. Pope, P.E., Kolouri, S., Rostami, M., Martin, C.E., Hoffmann, H.: Explainability methods for graph convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10772–10781 (2019)
56. Prates, M., Avelar, P.H.C., Lemos, H., Lamb, L.C., Vardi, M.Y.: Learning to solve np-complete problems: A graph neural network for decision tsp. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (2019)
57. Puny, O., Atzmon, M., Ben-Hamu, H., Smith, E.J., Misra, I., Grover, A., Lipman, Y.: Frame averaging for invariant and equivariant network design. arXiv preprint arXiv:2110.03336 (2021)
58. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
59. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)
60. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. arXiv preprint arXiv:2205.12454 (2022)
61. Rathee, M., Funke, T., Anand, A., Khosla, M.: Bagel: A benchmark for assessing graph neural network explanations. arXiv preprint arXiv:2206.13983 (2022)
62. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
63. Roccabruna, G., Azzolin, S., Riccardi, G.: Multi-source multi-domain sentiment analysis with BERT-based models. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference. pp. 581–589. European Language Resources Association, Marseille, France (Jun 2022), https://aclanthology.org/2022.lrec-1.62
64. Roddenberry, T.M., Segarra, S.: Hodgenet: Graph neural networks for edge data. In: 2019 53rd Asilomar Conference on Signals, Systems, and Computers. pp. 220–224. IEEE (2019)
65. Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L., Wiltschko, A.: Evaluating attribution for graph neural networks. Advances in neural information processing systems **33**, 5898–5910 (2020)
66. Schlichtkrull, M.S., De Cao, N., Titov, I.: Interpreting graph neural networks for nlp with differentiable edge masking. arXiv preprint arXiv:2010.00577 (2020)

67. Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schutt, K.T., Müller, K.R., Montavon, G.: Higher-order explanations of graph neural networks via relevant walks. IEEE transactions on pattern analysis and machine intelligence (2021)
68. Schwarzenberg, R., Hübner, M., Harbecke, D., Alt, C., Hennig, L.: Layerwise relevance visualization in convolutional text graph classifiers. arXiv preprint arXiv:1909.10911 (2019)
69. Selsam, D., Bjørner, N.: Guiding high-performance sat solvers with unsat-core predictions (2019)
70. Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., Dill, D.L.: Learning a SAT solver from single-bit supervision. In: International Conference on Learning Representations (2019)
71. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
72. Shan, C., Shen, Y., Zhang, Y., Li, X., Li, D.: Reinforcement learning enhanced explainer for graph neural networks. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 22523–22533. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/file/be26abe76fb5c8a4921cf9d3e865b454-Paper.pdf
73. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE signal processing magazine **30**(3), 83–98 (2013)
74. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
75. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
76. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International conference on machine learning. pp. 3319–3328. PMLR (2017)
77. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
78. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
79. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391 (2015)
80. Vu, M., Thai, M.T.: Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. Advances in neural information processing systems **33**, 12225–12235 (2020)
81. Wang, R., Walters, R., Yu, R.: Approximately equivariant networks for imperfectly symmetric dynamics. arXiv preprint arXiv:2201.11969 (2022)
82. Wang, W., Hu, Y., Tiwari, M., Khurshid, S., McMillan, K., Miikkulainen, R.: Neurocomb: Improving sat solving with graph neural networks. arXiv:2110.14053 (2021), http://www.cs.utexas.edu/users/ai-lab?wang:arxiv21

83. Weisfeiler, B., Leman, A.: The reduction of a graph to canonical form and the algebra which appears therein. NTI, Series **2**(9), 12–16 (1968)
84. Wu, W., Su, Y., Chen, X., Zhao, S., King, I., Lyu, M.R., Tai, Y.W.: Towards global explanations of convolutional neural networks with concept attribution. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8649–8658 (2020). https://doi.org/10.1109/CVPR42600.2020.00868
85. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning. pp. 2048–2057. PMLR (2015)
86. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
87. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. Advances in neural information processing systems **32** (2019)
88. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems **31** (2018)
89. Yuan, H., Ji, S.: Structpool: Structured graph pooling via conditional random fields. In: International Conference on Learning Representations (2020)
90. Yuan, H., Tang, J., Hu, X., Ji, S.: Xgnn: Towards model-level explanations of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 430–438 (2020)
91. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: A taxonomic survey. arXiv preprint arXiv:2012.15445 (2020)
92. Yuan, H., Yu, H., Wang, J., Li, K., Ji, S.: On explainability of graph neural networks via subgraph explorations. In: International Conference on Machine Learning. pp. 12241–12252. PMLR (2021)
93. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. Advances in neural information processing systems **32** (2019)
94. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)
95. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
96. Zhang, R., Zou, Y., Ma, J.: Hyper-sagnn: a self-attention based graph neural network for hypergraphs. arXiv preprint arXiv:1911.02613 (2019)
97. Zhang, Y., Defazio, D., Ramesh, A.: Relex: A model-agnostic relational model explainer. In: Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society. pp. 1042–1049 (2021)
98. Zhang, Y., Chen, X., Yang, Y., Ramamurthy, A., Li, B., Qi, Y., Song, L.: Efficient probabilistic logic reasoning with graph neural networks. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rJg76kStwH
99. Zhang, Z., Liu, Q., Hu, Q., Lee, C.: Hierarchical graph transformer with adaptive node sampling. ArXiv **abs/2210.03930** (2022)
100. Zhao, T., Luo, D., Zhang, X., Wang, S.: On consistency in graph neural network interpretation. arXiv preprint arXiv:2205.13733 (2022)

101. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)
102. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI Open **1**, 57–81 (2020)

# Supplementary for: "Understanding how explainers work in graph neural networks"

Antonio Longa$^{1,2[0000-0003-0337-1838]}$, Steve Azzolin$^{2[1111-2222-3333-4444]}$, Gabriele Santin$^{1[0000-0001-6959-1070]}$, Giulia Cencetti$^{1[0000-0002-6946-3666]}$, Pietro Liò$^{3[0000-0002-0540-5053]}$, Bruno Lepri$^{1[0000-0003-1275-2333]}$, and Andrea Passerini$^{2[0000-0002-2765-5395]}$

$^1$ Fondazione Bruno Kessler, Trento, Italy
{alonga,gsantin,gcencetti,lepri}@fbk.eu
$^2$ University of Trento, Italy andrea.passerini@unitn.it
steve.azzolin@studenti.unitn.it
$^3$ University of Cambridge, Cambridge, United Kingdom pl219@cam.ac.uk

## 1 Graph Neural Networks

In this section we first introduce the notation to deal with the GNN formalism, then we review the GNN architectures explicitly used in our study. We consider a graph $G := (V, E, \boldsymbol{X})$, with $n_V \in \mathbb{N}$ nodes $V := \{1, \ldots, n_V\}$, $n_E \in \mathbb{N}$ edges $E \subset V \times V$, and a matrix of $d$-dimensional node features $\boldsymbol{X} \in \mathbb{R}^{n_V \times d}$, where the $i$-th row of $\boldsymbol{X}$ is the vector of $d \in \mathbb{N}$ features of the $i$-th node. We use the matrices $\boldsymbol{A}, \boldsymbol{L}, \boldsymbol{I}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{D}}, \tilde{\boldsymbol{L}} \in \mathbb{R}^{n_V \times n_V}$, where $\boldsymbol{A}$ and $\boldsymbol{L}$ are the adjacency and Laplacian matrices of $G$, $\boldsymbol{I}$ is the $n_V$-dimensional identity matrix, $\tilde{\boldsymbol{A}} := \boldsymbol{A} + \boldsymbol{I}$, $\tilde{\boldsymbol{D}}$ is its diagonal matrix, and $\tilde{\boldsymbol{L}} := \frac{2}{\lambda_{\max}(\boldsymbol{L})}\boldsymbol{L} - \boldsymbol{I}$ is the scaled and normalized Laplacian, where $\lambda_{\max}(\boldsymbol{L})$ is the largest eigenvalue of $\boldsymbol{L}$. Furthermore, $N(i) := \{j \in V : (i, j) \in E\}$ is the first order neighborhood of the node $i \in V$.
Each GNN layer takes as input the graph $G$, and maps the node features $\boldsymbol{X} \in \mathbb{R}^{n_V \times d}$ to updated node features $\boldsymbol{X}' \in \mathbb{R}^{n_V \times d'}$ for a given $d' \in \mathbb{N}$. Some specific GNN layers, like Hierarchical pooling layers [102, 88, 44, 12], instead of refining the embedding for each input node aggregate nodes in order to coarsen the graph in a similar way as done by pooling methods for vision models [15, 75, 43], thus resulting in a node feature matrix $\boldsymbol{X}' \in \mathbb{R}^{n' \times d'}$ where $n' < n_V$. Overall, this new feature matrix $\boldsymbol{X}'$ represents the embedding or representation of the nodes after the application of one layer of the network. When needed, we denote as $\boldsymbol{X}_i, \boldsymbol{X}'_i$ the original and transformed feature vector of the $i$-node, i.e., the transpose of the $i$-th row of the matrices $\boldsymbol{X}, \boldsymbol{X}'$. Specifying the map $\boldsymbol{X} \to \boldsymbol{X}'$ is thus sufficient to provide a full specification of the different layers. These transformations are parametric, and they depend on trainable weights that are learned during the optimization of the network. We represent these weights as matrices $\boldsymbol{W}$. Additional terms specific to single layers are defined in the following.
After an arbitrary number $t$ of GNN layers stacked in sequence, the node embedding matrix $\boldsymbol{X}^{(t)}$ is further processed in a way that depends on the

task to perform. In node classification settings [42, 35], where the aim is predicting one or more node properties, a Multi-Layer Perceptron (MLP) [33] (with shared parameters across nodes) is applied to each node's embedding independently in order to output its predicted class. For graph classification settings [35] instead, where the goal is predicting a label for the entire graph, a permutation invariant aggregation function (like mean, max, or sum) is applied over nodes' embedding to compress $\boldsymbol{X}^{(t)}$ into a single vector which is then mapped to the final prediction via a standard MLP.
With this notation settled, we can now fully define the architectures that we are going to consider. In selecting the architectures to be included in our study, we relied on the comprehensive taxonomy of GNN methods published by Zhou et al. [102]. Since our goal is to provide an extensive overview of explainability methods for GNNs, we selected the models to benchmark aiming at covering as much as possible the different categories of the taxonomy. The specific methods are also selected depending on their popularity, their ease of training, their performances on our benchmark datasets, and their code availability. Overall, we analyzed the following categories: *Convolutional* whose computation can be roughly intended as a generalization of the convolution operation on the image domain. Such convolution can either be *Spectral* [18, 42], theoretically grounded in graph signal processing [73], or *Spatial* [78, 31, 86, 29], where the operations are usually defined in terms of graph topology; The *Pooling* category contains all approaches that aggregate node representations in order to perform graph-level tasks. They can be further differentiated into *Direct* [79, 95], where nodes can be aggregated with different aggregation strategies, often called readout functions, and *Hierarchical* [88, 12, 89, 44, 9], where nodes are progressively hierarchically aggregated based on their similarity. The latter methods often allow one to cluster nodes both based on their features and their topological neighborhood [88, 9]. Despite covering the major aspects of GNN architectures, the aforementioned taxonomy lacks some of the fundamental works that we will analyze in our study. Particularly, to compensate that, we decided to respectively include the Graph Isomorphism Network (Gin) [86] and the GraphConv Higher Order Network (GraphConv) [54] as *Spatial Convolution* and *Higher Order*, the latter being a new category added to the taxonomy. A summary of such categorization can be found in Figure 1. In the following, we broadly describe each GNN model used in this work, reporting for each one the category as identified by Zhou et al. [102]. Although the following definitions are often given as global functions of the graph for notational convenience, we remark that all the layers of a GNN can be efficiently implemented as local operations by means of a message-passing or sparse matrix multiplications.

**Chebyshev Spectral Graph Convolution** (Cheb, Spectral Convolution) [18]: The Chebyshev spectral graph convolutional operator [18] was aimed to generalize the convolution operation from the image to the graph domain. In doing so, it approximates the convolution of the node features with a trainable filter where such approximation is defined in the Fourier

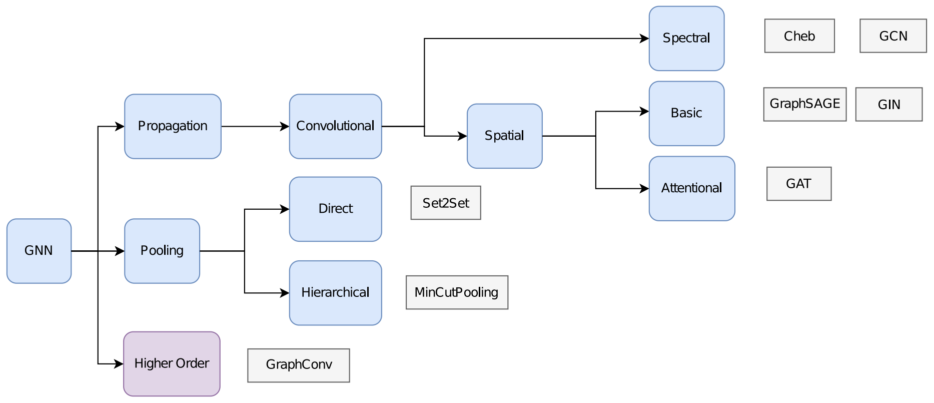

**Fig. 1.** An overview of the adopted GNN architectures structured in a taxonomy as defined by Zhou et al. [102]. In particular, blue boxes represent the categories as defined by the aforementioned work, while the purple box corresponds to the newly introduced Higher Order category.

domain by means of a Chebyshev polynomial. Since explicitly computing the convolution in the Fourier domain is very computationally expensive, to this end Cheb adopts the truncated recursive Chebyshev expansion [32], where the Chebyshev polynomial $T_k(x)$ of order $k$ can be computed by the recurrence $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_1 = 1$, $T_2 = x$. Given a degree $K \in \mathbb{N}$, we thus have:

$$\boldsymbol{X}' = \sum_{k=0}^{K} \boldsymbol{Z}^{(k)} \cdot \boldsymbol{W}^{(k)}, \tag{1}$$

where $\boldsymbol{Z}^{(k)}$ is computed recursively as:

$$\boldsymbol{Z}^{(1)} = \boldsymbol{X}, \quad \boldsymbol{Z}^{(2)} = \tilde{\boldsymbol{L}} \cdot \boldsymbol{X}, \quad \boldsymbol{Z}^{(k)} = 2\tilde{\boldsymbol{L}} \cdot \boldsymbol{Z}^{(k-1)} - \boldsymbol{Z}^{(k-2)}.$$

In our analysis we kept $K = 5$.

**Graph Convolutional Network** (Gcn, Spectral Convolution) [42]: Gcn [42] represents a first-order approximation of localized spectral filters on graphs [18]. For a single layer, the node representation is computed as:

$$\boldsymbol{X}' = \sigma(\tilde{\boldsymbol{D}}^{-1/2} \cdot \tilde{\boldsymbol{A}} \cdot \tilde{\boldsymbol{D}}^{-1/2} \cdot \boldsymbol{X} \cdot \boldsymbol{W}), \tag{2}$$

with $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$, and where $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function that is applied entry-wise. The normalization applied to $\tilde{\boldsymbol{A}}$ is in place to avoid numerical instabilities after successive applications of the above propagation rule. It effectively normalizes the node-wise aggregation by a weighted sum where each neighbor is weighted by the incident edge weight normalized by the degree of the two nodes. The connection with Cheb introduced above can be seen by keeping the number of convolutions per layer to 1, thus by setting $K = 1$. After some further simplifications detailed in [42], and here omitted for brevity, we can rewrite the spectral convolution as $\boldsymbol{W}^{(k)} \cdot (\boldsymbol{I} + \tilde{\boldsymbol{D}}^{-1/2} \cdot \tilde{\boldsymbol{A}} \cdot \tilde{\boldsymbol{D}}^{-1/2}) \cdot x$. After applying the renormalization trick introduced in [42] and by generalizing the formulation to a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n_V \times d}$ we get the formulation of Eq 2.

**Graph SAmple and aggreGatE** (GraphSage, Spatial Convolution) [31]: This work was proposed as an extension of Gcn. Contrary to Gcn, which inherently implements a weighted mean neighborhood aggregation, GraphSAGE generalizes to different kinds of aggregations like mean, max or Long Short-Term Memory (LSTM) aggregations [31, 34].

$$\boldsymbol{X}'_i = \boldsymbol{W}_1 \boldsymbol{X}_i + \boldsymbol{W}_2 \cdot aggregation_{j \in N(i)} \boldsymbol{X}_j, \tag{3}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2$ have both dimension $n_V \times d'$ and the function aggregation can be any permutation invariant function. Note that in the case of the LSTM aggregation, the authors adapted LSTMs to operate on sets by simply applying the LSTMs to a random permutation of the nodes. In our study we used the mean of node features as aggregator.

**Graph Isomorphism Network** (Gin, Spatial Convolution) [86]: The work of Xu et al. [86] was among the first ones to study the expressive power of GNNs in relation to the Weisfeiler-Lehman (WL) test of isomorphism [83]. The key insight is that a GNN can have as large discriminative power as the WL test if the GNN's aggregation scheme is highly expressive and can model injective functions, like the aggregation of the WL test. They studied the conditions for which a GNN has the same discriminative power as the WL test, and then they proposed the Graph Isomorphism Network (Gin) that provably satisfies such conditions [86]. The Gin computes node representations as:

$$\boldsymbol{X}' = h_{\boldsymbol{W}'}((\boldsymbol{A} + (1 + \epsilon) + \boldsymbol{I}) \cdot \boldsymbol{X}), \tag{4}$$

where $\epsilon \in \mathbb{R}$ is a small number, and $h_{\boldsymbol{W}'}$ is a neural network whose weights $\boldsymbol{W}'$ are the trainable part of the Gin layer. This model generalizes the WL test and hence achieves maximum discriminative power with respect to the WL test [86]. In our work, to limit the complexity of the network, we limited $h_{\boldsymbol{W}'}$ to have a single layer.

**Graph Attention Network** (Gat, Spatial Attentional Convolution) [78]: Following the successes of attention mechanism in natural language processing [77, 58, 63] and in computer vision [85, 59], Veličković et al. [78] proposed a GNN layer which computes each node's representation as a weighted sum of neighborhood features, where the weights are computed via an attention mechanism. Such attention mechanism helps the layer to focus on neighbors which are considered to be important for the current node, instead of treating each neighbor equally importantly [5].

Specifically, the propagation rule for each node can be defined by:

$$\boldsymbol{X}'_i = \sum_{j \in N(i) \cup \{i\}} \alpha_{i,j} \boldsymbol{W} \cdot \boldsymbol{X}_j, \tag{5}$$

where the $\alpha_{i,j}$ are attention coefficients, which are the output of a single-layer feed forward neural network applied to $\boldsymbol{W} \cdot \boldsymbol{X}_i, \boldsymbol{W} \cdot \boldsymbol{X}_j$. Namely, if $[x||y]$ denotes the concatenation of the vectors $x, y$, and if $\boldsymbol{W}' \in \mathbb{R}^{2d' \times 1}$ is a vector of trainable weights, then $\alpha_{i,j}$ is defined by

$$\alpha_{i,j} := \frac{\exp(\mathit{LeakyReLU}((\boldsymbol{W}')^T [\boldsymbol{W}\boldsymbol{X}_i || \boldsymbol{W}\boldsymbol{X}_j]))}{\sum_{k \in N(i) \cup \{i\}} \exp(\mathit{LeakyReLU}((\boldsymbol{W}')^T [\boldsymbol{W}\boldsymbol{X}_i || \boldsymbol{W}\boldsymbol{X}_k]))}, \tag{6}$$

where *LeakyReLU* has usually a slope $\alpha := 0.2$ if $x \leq 0$. Similarly as previous works on attention [77, 16, 58, 63, 59], Gat can implement multi-head attention in order to increase the attention's expressive power in modelling different aspects of node features.

**MinCutPooling** (MinCutPool, Hierarchical Pooling) [9]: Hierarchical pooling methods can be categorized into dense methods[9] and sparse methods[25], with the former being shown to be more expressive than the latter[10]. Similarly to other graph pooling mechanisms [88, 12, 89, 44], MinCutPool is able to hierarchically aggregate nodes into coarser representations to summarize local components and remove redundant information. Graph pooling follows a similar idea as pooling in standard architectures for vision applications [15, 75, 43], where it helps to reduce the memory and computation footprint of the model. However, the structured graph domain poses new challenges which required ad-hoc techniques for achieving good performing pooling methods. MinCutPool [9] achieves so by formulating a relaxation of the MinCut problem and by training a GNN to compute cluster assignment by jointly optimizing the downstream supervised loss and the additional unsupervised *mincut* loss. In particular, a soft cluster assignment matrix $\boldsymbol{S} \in \mathbb{R}^{n_V \times n'_V}$ is computed with a MLP with softmax activation over a refined set of node features (e.g. after one or more layers of message passing). Then, both the adjacency matrix and the node features' matrix are updated accordingly:

$$\begin{aligned}
\boldsymbol{X}' &= \boldsymbol{S}^T \cdot \boldsymbol{X}, \\
\tilde{\mathbf{A}} &= \boldsymbol{S}^T \cdot \boldsymbol{A} \cdot \boldsymbol{S} - \boldsymbol{I}_{\boldsymbol{n}'_V} \mathit{diag}(\boldsymbol{S}^T \cdot \boldsymbol{A} \cdot \boldsymbol{S}), \\
\boldsymbol{A}' &= \tilde{D}^{\frac{1}{2}} \cdot \tilde{\mathbf{A}} \cdot \tilde{D}^{\frac{1}{2}}.
\end{aligned}$$

We refer the interested reader to [9] for the details about the mathematical formulation of the differentiable MinCut problem. Since pooling methods modify the graph structure, they are not typically suitable for node classification and link predictions tasks [88, 12, 89, 44]. We will henceforth refer to as MinCutPool for an architecture implemented with Gcn layers interleaved with the MinCutPool operator.

**Set2Set** (Direct Pooling) [79]: The Set2Set model [79] is a specific approach for global graph pooling which takes as input the node-level representations, as computed by any GNN layer, and outputs a single representation for the entire graph suitable for graph-level tasks (e.g. graph classification [35], molecular property prediction [39, 11], etc.). Whilst traditional approaches simply taking the mean/max/sum of every node's representations, Set2Set can be seen as learning the aggregation via an LSTM. Thus, the node features are treated as sequences, which are processed through an LSTM network to obtain an embedding of the entire graph, represented as a vector of length $2d$. To enforce permutation equivariance, the ordering of the nodes can be either learned or randomized during training. The underlying LSTM works as follows:

$$\mathbf{q}_t = \text{LSTM}(\mathbf{q}^*_{t-1}), \quad \alpha_{i,t} = softmax(\boldsymbol{X}_i \cdot \mathbf{q}_t),$$
$$\mathbf{r}_t = \sum_{i=1}^{n_V} \alpha_{i,t}\boldsymbol{X}_i, \quad \mathbf{q}^*_t = [\mathbf{q}_t || \mathbf{r}_t] .$$

where $\mathbf{q}^*_t$ is the output of the layer. The subscript $t \in \{1, \ldots, T\}$ indicates that the process may be repeated a number $T \in \mathbb{N}$ of times. In our work we refer to Set2Set as a GNN architecture implemented as a number of Gcn layers with the Set2Set global pooling operator and with a value of $T = 7$.

**GraphConv** (Higher Order) [54]: To increase the expressivity of GNNs, researchers developed techniques to capture not only 1-hop connections, i.e., between the node neighborhood, but also to capture higher order connections [54, 46, 19, 64, 96, 45, 26]. This network captures higher order connections of a graph (up to order two) by aggregating information from neighbourhood nodes and incident edges. It is defined as:

$$\boldsymbol{X}'_i = \boldsymbol{W}_1\boldsymbol{X}_i + \boldsymbol{W}_2 \sum_{j \in N(i)} e_{i,j} \cdot \boldsymbol{X}_j, \tag{7}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are learnable weight matrices, while $e_{i,j}$ is the edge weight from node $i$ to node $j$.

## 2 GNN Explainability

To analyze and understand the strengths and weakness of graph explanation algorithms, we selected instances of GNN explainers which are representative of the current state of the art. To this end, we follow the systematization proposed by Yuan et al. [91], and choose to investigate *instance-based* explainers[101, 71, 74, 76, 75, 80, 87, 6, 68, 36, 62, 92, 53, 72, 51, 97, 67, 23], i.e., those which aim at identifying components of the input that are responsible for the model's output. This is in contrast with *model-based* explainers, which rather try to provide a global understanding of a trained model [3, 90]. Since the available model-based explainers are very heterogeneous (i.e. it is not available a unified evaluation setting), and since previous works on benchmarking graph explainers have focused on instance-based methods [91, 100, 1, 2, 47, 61], we thereby omit model-based explainers. In particular, Yuan et al. [91] identifies four macro categories of instance-based explainers, namely *gradient-*, *perturbation-*, *decomposition-* and *surrogate-based* models. Roughly speaking, gradient-based explainers exploit gradients of the input neural network [75, 76, 55], perturbation-based models perturb the input aiming to obtain explainable subgraphs[87, 52, 23, 66], decomposition-based models try to decompose the input identifying the explanations [6, 55, 67], while surrogate-based models use a simple interpretable surrogate to explain the original neural network [36, 97, 80].

Independently from this categorization, a further fundamental distinction is among explainers providing explanations in terms of edge [87, 52, 66, 92] or node masks [75, 76, 55, 6, 55, 67]. We refer to edge mask every time the explainer gives as output any sort of likelihood for each edge in the input graph. Conversely, a node mask represents likelihoods for every node in the graph. Few explainers allow also to extract a node feature mask [87, 74], which highlights the contribution of each single feature in the input node feature mask. However since single node features are not representative of the underlying topological structure which we are interested in, and in line with most previous works [91, 100, 1, 2, 47, 61], we do not consider single node features' explanations.

Below we report a brief overview of our benchmark explainers. Despite the existence of other works proposing explainers, which occasionally fall outside the aforementioned categorization [92, 53, 72, 51, 36, 97, 67], we limited our analysis on a subset. More specifically, the criteria for selecting a given explainer can be roughly summarized by *i)* representativity of a specific category as outlined before; *ii)* code availability; and *iii)* feasibility of usage, i.e., whether the explainer is not too computationally heavy to be used. Unfortunately, for the *decomposition* category, we did not find an explainer with a working codebase that could be promptly included in our study.

Given a GNN $g$ to be explained, let $g(e)^c = y^c = (w^c)^T e$ be the prediction of the model where $e$ corresponds to the final graph-level or node-level embedding. The vector $w^c \in \mathbb{R}^{d'}$ contains instead the learned Fully Connected weights for class $c$ to perform the final classification, and $\mathrm{H}^c[n]$ represents the importance of node $n$ for the prediction of class $c$. A brief summarization of the methods is available in Table 1.

**GradExplNode** (Gradient) [74]: Inspired by the previous work computing explanation in the context of Bayesian classification [4], GradExplNode computes the attribution for each input by backpropagation to the input space. The general idea is that the magnitude of the derivative gives insights into the most influential features that, if perturbed, give the highest difference in the output space:

$$\mathrm{H}^{c}_{\textsc{GradExplNode}}[n] = \frac{\partial y^c}{\partial \boldsymbol{X}_n} \tag{8}$$

To derive a single value for each input node, it is possible to take the maximum magnitude along the feature channel. Similarly, if the GNN to explain supports edge weights, then this method can be applied on edges, thus producing an edge mask. In the rest of the paper we will refer to this method with GradExplNode and GradExplEdge, whether it is applied to node or edges, respectively.

**GuidedBp:** (Gradient) [75] Guided Back Propagation (GuidedBp) follows a similar approach as GradExplNode, with the only difference that in the backpropagation it clips negative gradients which corresponds to features prone to decreasing the target activation, and thus considered as noise.

**IntegratedGradients:** (Gradient) [76] IntegratedGradients builds on top of simple Gradient-based methods, like GradExplNode, by integrating the gradient along a path. Specifically, given $x' \in \mathbb{R}^d$ a baseline input which represents a neutral input, often represented by a zero-vector, the resulting explanation is computed as:

$$\mathrm{H}^{c}_{\textsc{IntegratedGradients}}[n] = (\boldsymbol{X}_n - x') \int_0^1 \frac{\partial f(x' + \alpha(\boldsymbol{X}_n - x'))}{\partial \boldsymbol{X}_n}\, d\alpha \tag{9}$$

In short, the explanation corresponds to the integral computation of the gradients along the straight line path from an input baseline to the original value of the input. The adoption of this integral computation was shown to provide better theoretical guarantees then other approaches, as demonstrated in [76]. Nonetheless, for an efficient implementation, the integral in Eq 9 is substituted by a finite summation. This explainer can be applied even on edges if the GNN support edge weights. In the rest of the paper, we will refer to IgNode and IgEdge, whether it is applied to nodes or edges, respectively.

**Cam** (Gradient) [55]: Class Activation Map (Cam) maps backward the node features in the final layer to the input space for identifying important nodes. It represents a straightforward adaptation of Cam originally developed for Convolutional Neural Networks [101] to the graph domain. The Cam heat-map for node $n$ and class $c$ is defined as:

$$\mathrm{H}^{c}_{\textsc{Cam}}[n] = ReLU\left((w^c)^T \boldsymbol{X}_n^{(t)}\right) \tag{10}$$

**GradCam** (Gradient) [55]: GradCam represents an extension of Cam. Instead of using the weights $w^c$ to weight the contribution of each feature, GradCam uses the gradient of the output with respect to node features.

$$\mathrm{H}^{c}_{\textsc{GradCam}}[n] = ReLU\left((\alpha^c)^T \boldsymbol{X}_n^{(t)}\right) \tag{11}$$

$$\alpha_k^c = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial y^c}{\partial \boldsymbol{X}_{n,k}^{(t)}} \tag{12}$$

where $\alpha^c = [\alpha_0^c \dots \alpha_{d'}^c]$. Note GradCam allows also to compute the heatmap for each layer independently, by simply replacing the node feature matrix in the above equations with the features of any specific layer [71].

**GNNExplainer** (Perturbation) [87]: GnnExpl is able to provide an explanation both in terms of a subgraph of the input instance to explain, and a feature mask indicating the subset of input node features which is most responsible for the GNN's prediction. Overall, it is formulated as an optimization problem maximizing the Mutual Information between the GNN's predictions and distribution of possible subgraphs. However, in practical terms, since the optimization problem formulated in this way is intractable given the exponential number of subgraphs for a specific input graph, a relaxed version is actually computed, which can be interpreted as a variational approximation of the distribution of subgraphs. Nonetheless, despite the non-convex nature of the problem, the authors empirically observed that the aforementioned approximation together with a regularizer for promoting discreteness converges to good local minima.

**PgExpl:** (Perturbation) [52]: The Parametrized Explainer for GNNs (PgExpl) [52] adopts a very similar formulation of the explanation problem as GnnExpl where the two major differences are: *i)* PgExpl provides solely explanations in terms of subgraph structures, neglecting explanations in terms of node features; *ii)* instead of directly optimizing continuous edge and features masks as done by GnnExpl, it uses Gradient Descend to train a MLP which, given the two concatenated node embeddings $[X_i^{(t)} || X_j^{(t)}]$, predicts the likelihood of the edge $(i,\ j)$ being a relevant edge.

**PgmExpl** (Surrogate) [80]: PgmExpl builds a surrogate model of the GNN to explain by building a probabilistic graphical model. Random node features perturbations are applied to the given instance and, for each perturbation, the algorithm records the influence of the perturbation to the final prediction. After a number of perturbations, a dataset is generated and used to learn an interpretable Bayesian network which serves as final explanation [80].

## 3 Evaluation details and preprocessing

Each of the scores or metrics is computed for a specific instantiation of a dataset with $n_c \in \mathbb{N}$ classes, a class $c \in \{0, 1, \dots, n_c - 1\}$, a model, and an explainer. We thus assume that these four are fixed in the following, and we stress that the same computation has to be repeated for each of these configurations. Moreover, we remark that the metrics are computed on the training set alone, as we need access to the labels of the graphs or nodes.

| Name | Category | Task | Mask type |
|---|---|---|---|
| GradExplNode | Gradient | Graph/Node | Node |
| GradExplEdge | Gradient | Graph/Node | Edge |
| GuidedBp | Gradient | Graph/Node | Node |
| IgEdge | Gradient | Graph/Node | Edge |
| IgNode | Gradient | Graph/Node | Node |
| Cam | Gradient | Graph/Node | Node |
| GradCam | Gradient | Graph/Node | Node |
| GnnExpl | Perturbation | Graph/Node | Edge |
| PgExpl | Perturbation | Graph/Node | Edge |
| PgmExpl | Surrogate | Graph/Node | Node |

**Table 1.** Summary of explainers analyzed in this work. The columns *Task* represents to which downstream task the explainer can be applied to, while *Mask type* represents whether the explainer returns explanations in terms of entire node importance, single node features importance, or edge importance.

We assume to have a graph $G$ of class $y = c$, and denote as $g$ the trained GNN. We have GNNs which output a class probability prediction vector in form of a soft max, so that the predicted class probabilities sum to 1. Since we are considering one class at a time, in the following we assume to be working with only the output's entry corresponding to class $c$.
Only the graphs which are correctly classified by the trained GNN are considered further and run through the explainer, which returns a corresponding soft explanation mask $G_{\text{exp}}$, which is a copy of the original graph with associated node or edge weights (for node- or edge-based explainers).
Before computing the metrics, these soft-mask explanations are processed and filtered by means of three operations:

- *Conversion:* Edge masks are converted to node masks by assigning to each node the weight given by the average of the weights of its incident edges. This operation makes it easier to compare the scores of edge-based and node-based explainers, and we choose to use node masks since node-based explainers are more common in our taxonomy (see Section 1).
- *Filtering:* For each mask we check the difference between the largest and the smallest weight. If the difference is below a tolerance $\tau = 10^{-3}$, we discard the graph or node for the given combination of dataset, class, model, and explainer (the graph or node may still pass the filter for other settings). The goal of this filter is to discard poorly informative explanations.
- *Normalization:* The remaining explanation masks are normalized instance by instance, so that each explanation has weights in $[0, 1]$. This has the effect of making the computation of the metric uniform across the entire dataset, and comparing its values to those obtained with other settings.

After these operations have been applied, we compute the metrics as follows. We formalize each metric as it is computed on a single instance (a graph or a node), and remark that the overall values of plausibility or fidelity for the entire (dataset, class, model, explainer)-configuration is obtained by averaging over these single instances.

*Sufficiency* The fidelity sufficiency $F_{suf}$ [61] is the difference in the predicted probability when computed on the graph and on the explanation. Since the explanation is a soft mask, we fix a number of levels $N_t \in \mathbb{N}$ and apply an incremental thresholding with $N_t + 1$ threshold levels $t_k = k/N_t$, $k = 0, \dots, N_t$, where we define $G_{\exp}(t_k)$ to be the hard mask explanation derived from $G_{\exp}$ with threshold $t_k$.
Using $N_t = 100$, we define the metric by

$$F_{suf} = \frac{1}{N_t - 1} \sum_{k=1}^{N_t - 1} \left( g(G) - g(G_{\exp}(t_k)) \right),$$

i.e., the average change in prediction over all the possible hard masks.
This metric may possibly be negative, and a smaller value indicates a better result. This indeed may happen only if the explanation provides an higher probability for the correct class than the entire graph, and thus the explanation mask manages to filter unnecessary parts of the graph. For this reason this metric is harder to compare to other scores, so when used alone we transform it to a renormalized metric $F'_{suf}$, which has values in $[0, 1]$ and where $F'_{suf} = 1$ means a good quality of the explanation. The normalization takes into account the number of classes $n_c$, and is defined for $p = \frac{n_c - 1}{n_c}$ as

$$F'_{suf} = 1 - \frac{F_{suf} + p}{1 + p} = \frac{n_c}{2n_c - 1}(1 - F_{suf}).$$

*Comprehensiveness* The fidelity comprehensiveness $F_{com}$ [61] is instead the difference in the predicted probability when computed on the graph and on the complement of the explanation. Proceeding as in the computation of the sufficiency, we define

$$F_{com} = \frac{1}{N_t - 1} \sum_{k=1}^{N_t - 1} \left( g(G) - g(G \setminus G_{\exp}(t_k)) \right),$$

where now $G \setminus G_{\exp}(t_k)$ is the complement of the hard mask $G_{\exp}(t_k)$. This metric may as well assume negative values, but good explanation have in this case $F_{com}$ close to 1. (the complement of the explanation provides low probability).