When Comparing to Ground Truth is Wrong: On Evaluating GNN Explanation Methods

Lukas Faber, Amin K. Moghaddam, Roger Wattenhofer*
{lfaber,khashkhm,wattenhofer}@ethz.ch
ETH Zurich
Switzerland

ABSTRACT

We study the evaluation of graph explanation methods. The state of the art to evaluate explanation methods is to first train a GNN, then generate explanations, and finally compare those explanations with the ground truth. We show five pitfalls that sabotage this pipeline because the GNN does not use the ground-truth edges. Thus, the explanation method cannot detect the ground truth. We propose three novel benchmarks: (i) pattern detection, (ii) community detection, and (iii) handling negative evidence and gradient saturation. In a re-evaluation of state-of-the-art explanation methods, we show paths for improving existing methods and highlight further paths for GNN explanation research.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

graph neural networks, explainable ML, interpretability

ACM Reference Format:
Lukas Faber, Amin K. Moghaddam, Roger Wattenhofer. 2021. When Comparing to Ground Truth is Wrong: On Evaluating GNN Explanation Methods. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Neural networks are generally black boxes. Given some input, a model will produce a certain output, but *why* the model produces a particular output is often beyond our understanding. This is problematic, for safety-critical applications and for our general trust in a learned model [2]. We can get a better understanding of certain neural networks by learning which parts of an input are important. For graphs and graph neural networks (GNNs), we want to understand which edges and nodes are significant in the classification process. As we will discuss in the next section, there exist several explanation methods for GNNs. We evaluate these methods in pipelines as in Figure 1. We start from a problem that has some ground truth, train a GNN, run an explanation method on the GNN, and then compare the explanation against the ground truth. We emphasize that in this paper, ground truth does *not* refer to the prediction label, but the underlying evidence that allows to derive the correct label that we hope the explanations methods to find.

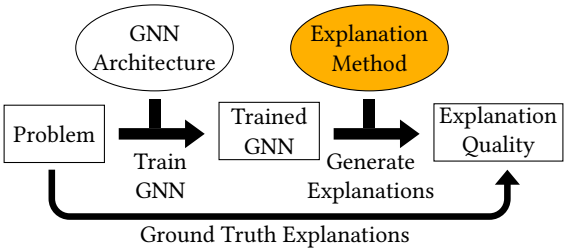


Figure 1: State-of-the-art evaluation pipeline for explanation methods (highlighted yellow). We choose a graph problem and a GNN architecture to train. Then we train a GNN and use the explanation method on its predictions. Cross-checking the predictions with the problem-given ground truth yields a quality score for the explanation method.

Problems arise when there is a mismatch between the ground truth and the GNN. Assume that the ground truth evidence are some edges $\mathcal{T}$, but the GNN uses other edges $\mathcal{M}$ for classification (even if the classification is correct). The explanation method should follow the GNN and output those edges. If we compare the explanation $\mathcal{M}$ with the ground truth evidence $\mathcal{T}$, we would wrongly conclude that the explanation is incorrect. However, the actual problem is the GNN not using the edges in $\mathcal{T}$. In this paper, we show that such mismatches between the ground truth and GNNs are a frequent problem. We suggest new benchmarks to fix the evidence mismatch. We make the following contributions:

- We present five pitfalls that can arise when creating an evaluation setup for GNN explanation methods. Then, we show how each pitfall can cause a mismatch between the GNN edges and the ground-truth edges and discuss the extent to which these pitfalls apply to current benchmarks.
- We design three benchmarks that avoid these pitfalls. Popular GNN problems (pattern detection, community detection) inspire the first two benchmarks. The third benchmark evaluates two properties for explanation methods, i.e., whether they can explain negative evidence and if they suffer from the saturation effect (more evidence causes worse explanations)
- We run state-of-the-art GNN explanation methods on these new benchmarks. We obtain mixed results but generally,

*alphabetical ordering

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467283

integrated gradients on edges seems like a sensible choice while also leaving room for improvement.

2 RELATED WORK

2.1 GNN Explanation

Baldassarre and Azizpour [4] and Pope et al. [23] are among the first to tackle the problem of explaining GNN predictions. They both adapt several standard neural network explanation methods to GNNs. For evaluation, they use simple molecular graphs, scene graphs, and a simple infection scenario, which we use as a basis for our infection benchmark we present later. Ying et al. [36] introduce GNNExplainer. They explain predictions through mutual information between edges or node features and the target labels. They introduce new datasets detecting motifs attached to Barabási-Albert graphs or balanced trees. Follow-up works on GNN explanation pick up these datasets [13, 16, 18, 25, 31] or vary them slightly [7]. However, we show later that some of the datasets have deficiencies, allowing misleading conclusions about the explanation method.

Sanchez-Lengeling et al. [25] investigate the problem of evaluating these explanation methods more systematically, defining several properties an explanation method should meet: accuracy (matching the ground-truth), consistency (across different well-performing model architectures), faithfulness (to the model performance), stability (against small input changes). Yuan et al. [38] survey GNN explanation methods, datasets, and metrics. They proposed a new graph dataset based on text sentiment data and human evaluation. The authors suggest accuracy, stability, sparsity (explaining with few edges), and fidelity (close to faithfulness) as metrics for explanation methods. In this work, we do not look at requirements for an explanation method but at requirements for evaluating an explanation method.

2.2 Evaluating Explanations

In computer vision, Sundararajan et al. [29] discussed desirable properties for explanation methods: sensitivity to features and invariance to the model implementation. They also propose Integrated Gradients. Adebayo et al. [3] introduce another important aspect for evaluation: a model should pass sanity checks, meaning that explanations should be sensitive to model weights (as faithfulness or fidelity in graph works). Thus, explanations should change when model weights are randomized. They also show how simple baselines such as edge detectors produce plausible "explanations". Another line of research investigates how to evaluate explanations quantitatively. Hooker et al. [10] argue that removing the explanation part, rerunning the model, and measuring prediction changes is not a good solution since the images post-removal are out of the training distribution (a similar argument as in Faber et al. [7]). Thus, there is a need for a controlled benchmark where we know the ground-truth evidence for target labels. Yang and Kim [35] and Johnson et al. [14] propose several such benchmarks with ground truth for the vision domain. We believe this is the right approach; thanks to the discrete nature of graphs, such benchmarks are also easy to create in the graph domain.

3 METHOD

3.1 Preliminaries

Let $G = (V, E, F)$ denote a graph with nodes V and edges E. $F : V \to \mathbb{R}^d$ assigns every node a $d-$ dimensional feature vector. Edges could also have attributes, but we focus on unlabeled edges for the scope of this paper. In graph classification, we have a target label y per graph. In node classification we have one label l_v per node $v \in V$. Labels come from a label space Y We consider graph neural networks as prediction functions on graphs $\mathcal{GNN} : (\mathcal{G}, I) \to Y$. Most GNN architectures follow the message passing paradigm [9]: Every node has an embedding and sends a learnable message based on the embedding to its neighbors. Then, every node aggregates the received messages and learns to update its local embedding. This process corresponds to one GNN layer and is usually repeated several times. GNNs take one graph $G = (V, E, F) \in \mathcal{G}$ and a prediction task $i \in I$ (i would be a node out of V for node classification or the graph G for graph classification) and predict the label for i based on the embeddings.

3.2 Explaining Edges

For a trained model and a particular prediction, we expect the explanation method to highlight which information the model used to predict. For example, for node classification, the GNN can use local node features, connected edges, and features of the node's neighbors. There are two principal ways to highlight this information: highlight important nodes or highlight important edges. An intuitive way to capture feature information is to assign importance for each node. This importance can reflect this node has important edges or that its features are important. Many works [25, 31] follow this approach. However, we argue not to highlight nodes but edges instead, which comes with the main advantage that edges have more fine-grained information than nodes. Consider a node with many (several hundred) edges but only one important edge. On node-level we would identify this node as important and might conclude all edges are relevant. If we assume only a subset of edges is important, we could not find out which edges. On the other hand, by explaining with edges, we can identify this one single important edge. The drawback of highlighting edges is that we cannot explain local node features of the node itself. For explaining local node features, we can use existing techniques that work on features without graphs structure, such as [17, 24]. Graph explanation methods should tackle the additional problem arising from explaining the important graph structure. Formally, we consider an explanation method as a function $\mathcal{ATT}$ which tasks a trained model $\mathcal{GNN}$, a graph $G = (V, E, F)$, and a prediction task i and returns a mapping $E \to \mathbb{R}$ assigning every edge in the graph an importance score with respect to prediction i —also called attribution.

3.3 Evaluation Explanations

The follow-up question is how to evaluate a given set of edge attributions. Usually, most edges are not important for the prediction, meaning these edges have attributions of 0 or close to 0. Therefore, a possible approach is choosing a subset of edges based on attribution (as sparsity in Yuan et al. [38]) through thresholding attributions or selecting the k edges with the largest attributions.

We refer to the obtained set of edges with high attributions as $\mathcal{A}$. There are two approaches to evaluate $\mathcal{A}$.

The first approach is occlusion, where we remove edges in $e \in \mathcal{A}$ from G (for example, one edge or all edges) and rerun the model. If the model confidence for the correct label decreases, we consider e important and having e in $\mathcal{A}$ correct. However, Hooker et al. [10] shows that this evaluation can lead to out of distribution inputs in images. In these cases, model behavior might differ not because of "removing" important information but because of evaluating a sample outside the training distribution. Faber et al. [7] argue that the risk for out of distribution inputs is even more relevant for graphs due to their discrete nature. They show that occlusion first highlights an edge that is not in a clique but whose removal causes an isolated node. For a further shortcoming of occlusion-based evaluation, consider this simple example of redundant evidence: Mutag is a molecule graph dataset with the task of predicting molecules' mutagenicity as a binary label. Both NO_2 and NH_2 substructures are strong evidence for the molecule mutagenicity [6]. Therefore, a properly trained GNN model is sensitive to both of these substructures. For a molecule where both NO_2 and NH_2 are present, an explanation method may highlight only one of them. But occluding either of them does not change the GNN prediction. Thus the explanation is regarded as "incorrect" according to the metric. Therefore, we argue not to evaluate based on occlusion and instead follow the second approach.

The second approach assumes that we have ground truth for which edges are important for making a prediction. The existing graph explanation methods mostly follow this approach [25, 31, 36]. For every prediction task $i \in I$, we have a set of edges $\mathcal{T}$ that we know are important for the prediction. Now we can measure the quality of $\mathcal{A}$ as the overlap with $\mathcal{T}$ (accuracy [25, 38]). However, we argue that this idea does not consider one important aspect: the set of edges $\mathcal{M}$ that the model $\mathcal{GNN}$ actually uses. If the overlap between $\mathcal{M}$ and $\mathcal{T}$ is low and the explanation method is faithful to the model, then the overlap of $\mathcal{A}$ and $\mathcal{T}$ is also low. Thus the explanation method—even if it works well—is doomed to appear weak. From this perspective, we also explore whether the consistency property [25] is desirable. In general, consistency measures whether an explanation method returns stable explanations when applied to different models. However, it is unclear in the general case that $\mathcal{M}$ will be constant across different models. Hence expecting a constant explanation $\mathcal{A}$ seems unreasonable. We also revisit stability [25, 38]. Stability measures whether an explanation method returns stable explanations when varying the input graph slightly. But if we cannot rule out that these modified inputs alter $\mathcal{M}$, we cannot expect $\mathcal{A}$ to stay constant. The weakness of GNNs to adversarial attacks [5, 40] suggests, it is well possible that even slight perturbations of the input cause drastic changes in the GNN's predictions and thus changes in $\mathcal{M}$. We could consider stability a desirable property for the GNN, but we wish explanation methods to be very sensitive to the model they are supposed to explain.

In the following, we highlight five pitfalls that can cause little overlap between $\mathcal{M}$ and $\mathcal{T}$. Thus each pitfall can sabotage our goal to evaluate explanation methods fairly. Then, we will present three new evaluation benchmarks that avoid these pitfalls.

4 PITFALLS IN EVALUATING EXPLANATIONS

4.1 Pitfall 1: Bias Terms

Wang et al. [32] show that bias terms are often overlooked in attribution and explanation of neural networks. Ignoring biases is problematic because a GNN can learn to predict one class using only bias terms, in which case $\mathcal{M} = \emptyset$. It does not matter that the problem has a valid explanation $\mathcal{T}$; the model can decide to ignore it in favor of using biases. In these cases, any faithful explanation method will fail when we compare against the non-empty set $\mathcal{T}$. Consider for example the CYCLIQ dataset from Faber et al. [7]. The task is graph classification, whether we append a cycle or a clique to a tree base structure. In our experiments on this dataset, the GNN models never learn what cycles are: the model bias is a cycle unless the model finds enough evidence for a clique (see Figure 2). Thus, when we evaluate the explanation method for highlighting the cycle edges, we get poor performance. This performance is consistent with the authors' results showing worse accuracy for cycles than for cliques. Unfortunately, only removing bias parameters is not sufficient for eliminating this pitfall because the model can "create" its own bias terms. For example, if one feature is one-hot encoded, summing up all the encoding dimensions creates a constant one that can be a bias in later layers. Therefore, we try to add an explicit class where $\mathcal{T} = \emptyset$, so the model must use the bias for this class or take care that the bias term cannot solve one class by itself.

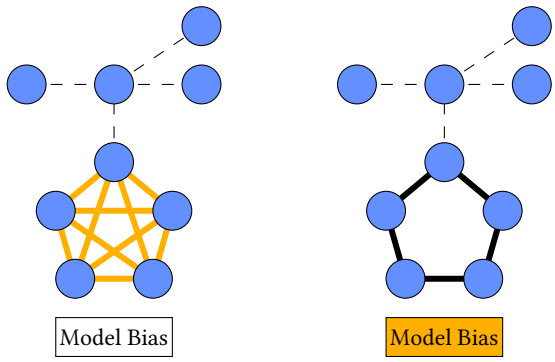


Figure 2: Binary classification whether a graph is a cycle (right) or a clique (left). Solid lines show the explanation ground truth $\mathcal{T}$, yellow-colored parts show the edges $\mathcal{M}$ the model uses. Here, the GNN uses bias terms to detect cycles ($\mathcal{M} = \emptyset$), so all edge explanations for cycles are wrong.

4.2 Pitfall 2: Redundant Evidence

Another problem arises when $\mathcal{T}$ is not unique. That could be because already a subset $\mathcal{T}'$ is a valid explanation, or a partially disjoint set $\mathcal{T}''$ is also a valid explanation. We cannot control that $\mathcal{GNN}$ learns to exploit the edges in $\mathcal{T}$ and not those from its alternatives. A strict evaluation only measuring overlap against $\mathcal{T}$ might yield wrong weak results for explanation methods. Consider for example the `Tree-Cycles` dataset introduced by Ying et al. [36]. The graph base structure is a balanced binary tree where six-node cycles are attached to it as motifs. The GNN task is to detect whether a node belongs to the base tree or cycles. The authors train a 3–

layer GNN for classification. However, we observe that a 2–layer GNN achieves the same accuracy (around 95%). This suggests that the GNN does not need to look further than 2 hops, which does not contain the entire cycle motif. Therefore, we argue that expecting the entire cycle from the explanation method is not correct. Thus, it is necessary to evaluate methods on benchmarks where $\mathcal{T}$ is unique, and all edges are needed. Thanks to the discrete structure of graphs, such benchmarks are rather easy to create. Alternatively, having more than one set $\mathcal{T}$ is feasible if we relax the evaluation to compare the explanation $\mathcal{A}$ to all of them.

4.3 Pitfall 3: Trivial Correct Explanations

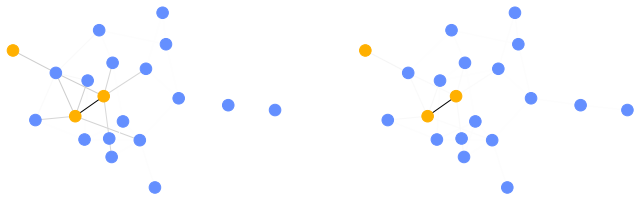
This pitfall is related to an observation from Adebayo et al. [3]. They found that a simple edge detector on images produces "good explanations". Of course, these are not explanations since they are completely model agnostic. In a similar spirit, we argue to create benchmarks, where simple baselines cannot detect $\mathcal{T}$. Having only "easy" edges in a benchmark can give false positive results on explanation performance. For example, consider the `Tree-Grid` dataset from Ying et al. [36] where nodes have to detect if they are in a grid. In this benchmark, a model-agnostic nearest neighbor explanation method seems very strong, as will all methods having a bias to explaining through direct neighbors. As another example, consider a simple community detection benchmark, where the graph has strong homophily. The nearest nodes will naturally hit many same-community nodes, and appear as a strong explanation method. Therefore, we argue to ensure during benchmark creation that a simple baseline cannot solve $\mathcal{T}$; we later validate our benchmarks with a random, a nearest neighbor, and a page-rank baseline.

4.4 Pitfall 4: Weak GNN Model

The GNN model itself can also cause problems, particularly when the model performance is far from the maximum. Subpar model performance is a strong indicator that the model does not use the edges in $\mathcal{T}$ in many cases (since these would lead to the correct label), but other edges instead. Even for the correctly classified labels, it remains unclear that the GNN used the edges in $\mathcal{T}$ correctly. If the model understood the systematicity underlying $\mathcal{T}$, its performance would be higher. Therefore, we aim for benchmarks where we can train GNNs close to their maximum possible performance. In many cases, this is close to 100%. But for some example graphs with an underlying stochastic block model, the stochasticity of the graph will not allow for 100% classification accuracy. In such cases, we choose to ignore incorrectly classified examples. While incorrect predictions are most definitely interesting for troubleshooting and debugging when we deploy GNNs, they are unfair in evaluating explanation methods.

4.5 Pitfall 5: Misaligned GNN Architecture

We observe that different GNNs following different architectures can have vastly differing explanation accuracy—even when each model's classification accuracy is quasi-perfect (not a problem of pitfall 4). You et al. [37] discuss a design space for GNNs, laying out the choices for layers, hyperparameters and others. Consider the example of the following graph classification example. We are given random graphs that have blue or yellow colored nodes. Graphs have class 1 if they contain two adjacent yellow nodes, 0 otherwise. We deliberately encode the node color as a scalar (not one-hot) with 1 corresponding to blue and 2 to yellow. We train one 3–layer GNN without pre-processing layers and one 3–layer GNN with 2 pre-processing layers. Both models reach close to 100% accuracy. Now we explain their predictions using a version of integrated gradients [29] that is adopted for graphs. Figure 3 shows the results. Even though both models achieve the same accuracy they use different edges for prediction. Because of our "bad" choice to model the input features, both model need to spend one processing step to transform the color information into a usable format. The GNN from figure 3b can use the preprocessing layers for this task, while the GNN in Figure 3a has to realize this through graph convolution layers. Thus the edges to blue nodes are also important for the latter GNN, making those edges appear in $\mathcal{M}$. As a consequence, the explanation method picks up on this and assigns high attribution values. However, this is wrong according to the ground truth $\mathcal{T}$ since only edges between yellow nodes are important. This simple example shows that different GNN architectures—even with similar accuracy—can have strongly different edges $\mathcal{M}$ they use. We see this example as further evidence that the property consistency [25] might be undesirable. Consistency strives for constant explanations across different models, even though different models—with the same accuracy—use different edges.



(a) Without pre-processing **(b) With pre-processing**

Figure 3: The impact of pre-processing layers on explanation accuracy for detecting if graphs have two adjacent yellow nodes. Plots show explanations for two different architectures, each reaching quasi-perfect accuracy. The bolder an edge, the higher its attribution. The left model has to preprocess node features via graph convolutions, which causes wrong explanations on edges.

Whether or not to add pre-processing layers is one of many design choices to make when creating a GNN architecture. You et al. [37] present 12 design dimensions, for example, pre-processing layers, post-processing layers, batch normalization, skip connections, or the choice of graph convolutions. Each of these choices can have unintended side effects on $\mathcal{M}$ and the explanation accuracy. Unfortunately, we did not find a principled approach for ensuring that GNN architecture choices do not cause mismatches between $\mathcal{M}$ and $\mathcal{T}$. We report the steps taken to avoid this problem when presenting the benchmark.

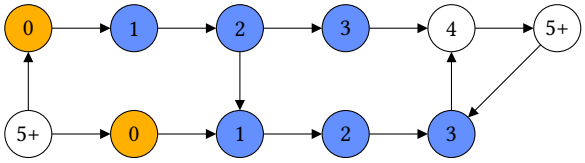


Figure 4: Infection Benchmark. Nodes predict their distance from infected nodes (yellow). Not reachable nodes and those with at least distance 5 are collapsed into one class. Nodes with non-unique shortest paths to infected nodes are excluded (as node 4).

5 BENCHMARKS FOR EVALUATING EXPLANATIONS

5.1 Infection Benchmark

One class of graph problems requires a GNN to identify whether a graph contains a specific pattern of (labeled) nodes. This kind of setting is prominent with biological and chemical problems, where particular substructures can predict properties for molecules, as in the Mutag [20] or the HIV [34] datasets, or predict properties of proteins, as in the Enzymes dataset [20].

Here, we present a dataset based on detecting paths inspired by the infection dataset from Baldassarre and Azizpour [4]. We create a random directed graph where each node can be healthy or infected. We define a node classification task to predict the length of the shortest directed path from an infected node. Possible labels reach from 0 (the node is infected) to 5 (the distance is 5, more than 5, or not reachable from an infected node).

Pitfall 1 Nodes not reachable from an infected node have label 5; these nodes have no ground-truth edges ($\mathcal{T} = \emptyset$). The only way the model can (and does) predict these nodes is through bias terms. This means the bias terms do not solve the other labels, for which we want to evaluate explanations.

Pitfall 2 We exclude nodes with multiple shortest paths to have unique paths as explanations. By removing any edge from the shortest path, the node label will change. Therefore, the $\mathcal{T}$ should include edge e if and only if it belongs to the unique shortest path. If the node's label is k, we assemble $\mathcal{A}$ as the set of the k edges with the highest attribution values.

Pitfall 3 The correct explanation is a path between the target node and its nearest infected node. This path does not have a strong bias towards the node's neighbors. Edges of nodes with high centrality could be overrepresented in the ground-truth, being part of many shortest paths. However, we generate graphs with the Erdős-Renyi random graph model, so we can expect there are no hub nodes in expectation (for example, in comparison to scale-free networks).

Pitfall 4 Our experiments show that the GNNs can solve this task with perfect accuracy.

Pitfall 5 We employ a four-layer GNN since four is the furthest infected distance we want to measure. We provide skip connections from every layer to the output to allow the model to skip convolution layers.

5.2 Community Benchmark

GNNs are widely used on graphs with an underlying community structure between nodes. Being able to sort nodes into communities is an important node classification task, especially graphs with homophily such as social networks. Therefore, we find many benchmarks on this topic, for example classifying in academic networks [11, 28], co-purchasing graphs [19]. In contrast to the previous task, the (non)-existence of distinct edges is not important. Rather, the GNN has to identify the community despite noisy edges and detect tightly connected groups of nodes.

We base our community benchmark on the stochastic block model (SBM) [1]. We have n nodes, split into k equally sized distinct communities with intra-community edge probability p and inter-community edge probability q such that $p > q$ and $p < q(k-1)$. All edges are directed; we consider the probabilities for each direction independently. We associate each community with a distinct color. Every node receives its community's color, then we recolor a ratio f of nodes to a new random color (might again be the correct color). We now define the node classification task to predict the community to which every node belongs. Ground truth explanation is a difficult concept in this problem domain because community detection is about finding the communities in an underlying noisy graph. Some of the underlying noise will look helpful (such as a node from a different community with a correct color) or not helpful (nodes from the same community but with a wrong color). Additionally, communities are redundant structures where removing a few edges does not substantially change the results, which conflicts with pitfall 2. Therefore, we need a relaxed evaluation mechanism.

We follow the general idea of relative feature importance [35]. If we modify the graph slightly, an increase in model confidence should correlate with increasing attribution for the correct edges. For explaining node v, we create two modified graphs following two rewiring schemes, in the part of the graph v can see through the GNN. In the first rewiring scheme, we rewire edges ending in v's community but start in another community (inter-community edges) so that both endpoints lie in v's community. Thus, the community structure around v increases. If this results in a more confident prediction of the correct label for v, we verify that the rewired edges receive on average higher attribution than before. For the second rewiring, we rewire edges starting from other communities to end in v's community. This weakens v's community structure because of more misleading edges. Thus, the model likely predicts v's correct label with less confidence, in which case we expect the hurtful rewired edges to have a lower attribution compared to before. Figure 5 illustrates these rewiring ideas for a node in the blue community. We now define the explanation accuracy as the ratio of rewirings where the change in edge attribution correctly follows the change in model certainty.

Pitfall 1 This benchmark is different in that it does not have a catch class for bias terms. Fortunately, this is not a problem since we only measure on nodes where the confidence increases with rewiring. The bias terms occur before and after rewiring, thus they cancel each other when we measure the increase through rewiring edges. Therefore, we can attribute changes in confidence to the edges.

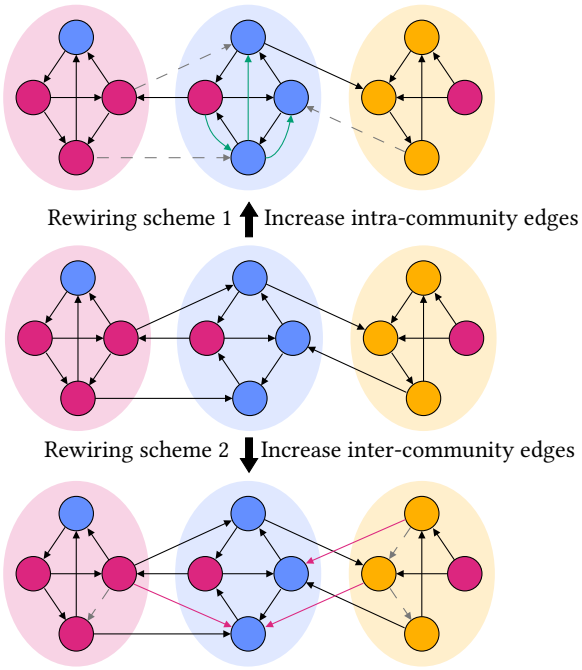


Figure 5: Community Benchmark. Nodes have to determine the correct community (shown by background color) they belong to. Initially, some nodes have the wrong color. Using community structure, nodes can identify their correct community. Rewirings add green edges and remove gray dashed edges for a node in the blue community.

Pitfall 2 Since there is a lot of redundant evidence for each prediction, rewiring a single edge will not change prediction nor attributions. On the other hand, rewiring too many edges might push the graph out of the training distribution. Therefore, we experimentally tuned the number of edges to rewire: we found 50 to be a good number.

Pitfall 3 Nearby nodes are not helpful in this benchmark (the majority of direct neighbors are from a different community), nor do high-centrality nodes help (these rather would hurt because they link between communities).

Pitfall 4 Depending on the choice of p, q, and f, we can vary the difficulty of this task from being easy to hard. If the task is too easy ($p > (k-1)q$), the model can resort to majority counting, never needing to detect communities. On the other hand, the task can be too hard. We selected $p = 0.05$, $q = 0.007$, and $f = 0.5$ where the model can reach an accuracy of 0.81. Full accuracy is not expected since some nodes will look as if they belong to a different community because of stochasticity in graph creation.

Pitfall 5 We employ a 4– layer GNN for this task. On average, we found with experiments that 4– layers are the least amount needed (for this particular p) for most nodes to reach every other node in their community. We also use skip connections to allow the model to preserve each layer's information for the final prediction. We also see that the model behaves as expected when we try different values for f. Increasing f leads to decreasing accuracy, until we reach random guessing for $f = 1.0$.

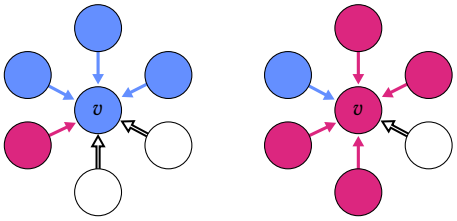


Figure 6: Negative Evidence Benchmark: Node v is initially a white node. Based on the colored edges, it receives a red or blue label. We expect white edges to be attributed differently from minority edges.

5.3 Negative Evidence and Saturation

With this last benchmark, we aim to test properties that are, to the best of our knowledge, not yet discussed in the context of graph explanation methods. The first important property is to detect negative evidence. Negative evidence actively reduces the model confidence for a certain prediction. It is important to distinguish between negative and uninformative edges because it allows us to understand the model behavior more clearly. The second property is the robustness of the explanation method to the saturation effect. Shrikumar et al. [27] discusses this effect in the context of images. Intuitively, saturation happens when *more* evidence for a certain classes leads to *less* attribution for all of that evidence. Thus, despite the model becoming more certain of a class, the explanations become less certain— a counter-intuitive and undesirable behavior. Thanks to the discrete nature of graphs, we can create a synthetic benchmark to test both properties.

We create a graph with ten blue nodes, ten red nodes, and 1980 white nodes. The objective is to classify the white nodes whether they have more red or blue neighbors. We refer to edges with a red source as red edges, edges with a blue source as blue edges, and edges with a white source as white edges. Therefore, only red and blue edges are relevant for this classification task, while white edges carry no information. We define the majority color (equal to the node label) as the more frequent color in a node's 1–hop neighborhood, the other being the minority color. Blue and red nodes have no label. We connect each white node to r red nodes and b blue nodes such that $1 \leq b, r \leq 10$, $b \neq r$, and every combination of (b, r) appears exactly 22 times. We choose nodes where $|b - r| \geq 2$ for testing the explanation so that occlusion does not encounter graphs with an equal number of red and blue edges. We add edges from white nodes with probability 0.015 such that each white node has 30 white edges on average. We show an example graph in Figure 6.

For a node having more red edges, blue edges are negative evidence and vice versa. At the same time, white edges do not carry any information. Depending on the semantics of an explanation method, we could imagine higher attribution for a white edge (a white edge is more "helpful" than a minority edge) or a lower attribution (a white edge is less "informative" than a negative edge).

However, white edges should be attributed differently from minority edges. We verify that their attribution values' distributions differ by running a two-sided Kolmogorov-Smirnov-Test with a significance level of 0.99. Saturation happens in this benchmark when the majority color clearly outnumbers the minority color. Let δ be the difference between nodes of the majority minus nodes of the minority color. The evidence for the correct class becomes stronger with increasing δ. However, more nodes of the majority color do not add new information. Therefore, the model saturates its prediction after a certain δ. An explanation method suffers from this saturation if we see the majority color's median of attributions drop. Thus, the explanations become less certain when the model has better evidence.

Pitfall 1 This benchmark has no class that would suit to catch model biases. However, we find that the model learns this task to 100% accuracy. Because of the task's symmetry, the model cannot learn one class with biases since both classes need both red and blue edges.

Pitfall 2 The benchmark contains redundant evidence by design since we need redundancy for the saturation effect. Therefore, we cannot measure accuracy as the overlap between $\mathcal{M}$ and $\mathcal{T}$. But we can measure that generally, the attribution of edges in $\mathcal{T}$ should not decrease with more evidence. Also, we check that the model can distinguish between no-information white edges and minority edges.

Pitfall 3 The relevant nodes are a vast minority. Therefore, the majority of nodes will be white and have no impact on classification. Having just a 1− layer GNN does also mean explaining through nearest neighbors does not make sense.

Pitfall 4 Our experiments show that the model can solve this task perfectly, i.e., reaching 100% classification accuracy.

Pitfall 5 A 1−layer GNN can solve this task perfectly, removing the need for skip connections, plus there is no need for pre-processing any features, as colors are given in an easy to use one-hot encoding.

6 EXPERIMENTS

6.1 Experiment Setup

As our last contribution, we re-evaluate several state-of-the-art methods and baselines (to sanity check that our benchmarks do not have trivial solution—pitfall 3) on our benchmarks[1]:

Random This method attributes every edge a random attribution score between 0 and 1.

Nearest Neighbor A simple baseline that assign higher values to edges that have a lower distance to the target node.

PageRank Attributions are defined through personalized (by node to explain) page rank values of the adjacent nodes[21].

Node Gradients By calculating the gradient of the output with respect to node features, we can measure each node's importance. We calculate the importance of each edge as the average value of its endpoints.

Edge Gradients We define weights for each edge and multiply weights with the messages in each graph convolution. Thus, a weight of 1 for all edges is equal to the original GNN. We calculate the gradient of the output with respect to edge weights and use gradients as attribution.

Integrated Gradients This gradient-based method accumulates gradients while interpolating between a baseline and the current instance [29]. We use it for both nodes and edges with zero vector as the baseline.

GradCAM GradCAM [26] is a generalization of class activation maps (CAM). In this method, each node's attribution value is the sum of its activation values on each convolution layer multiplied by the gradient value of the output with respect to that layer. We did not use GradCAM only on the last layer since that would result in attributions where only the first-hop neighborhood values are non-zero. The last-layer variant is more suitable for graph classification tasks.

PGMExplainer PGMExplainer uses probabilistic graphical models on perturbed instances of the input to find the dependencies between the nodes and the output.[38]

Occlusion We adopt occlusion [39] for graphs by removing each edge and rerunning the model. Occlusion uses the differences in prediction certainty as attribution strength.

GNNExplainer GNNExplainer [36] relaxes the edges and features to continuous strengths and measures importance based on the mutual information of their strength and the target class. We used an implementation based on the torch geometric library [8] (see Appendix A)

We do not include attention-weight explanations, for example, with a GAT architecture [30]. Unlike attention, all the above approaches are model agnostic (some require differentiability) and do not require an attention-based GNN. Furthermore, previous research in graphs tried attention-based explanation with weak results [18, 36] and other domains still investigate whether attention yields promising explanations without a definite answer yet. [12, 22, 33].

[1]Code available at: https://github.com/m30m/gnn-explainability

	Infection	Community	Neg. Evidence
#Train/Test Graphs	6/4	45/5	6/4
#Nodes/graph	1000	1000	2000
Avg. #Edges/graph	3996	11250	103955
#Classes	6	10	2
Avg. #Test Nodes	491	1000	1232
GNN layers	4	4	1
Avg. GNN Acc.	**1.0**	**0.81**	**1.0**

Table 1: Experiment settings for each benchmark

6.2 Experiment Settings

We create the GNNs and datasets as described in the respective benchmark sections. We use the following graph convolution.

$$f^{(t)}(v) = \mathrm{ReLU}\Big(f^{(t-1)}(v) \cdot W_1^{(t)} + \sum_{w \in N(v)} f^{(t-1)}(w) \cdot W_2^{(t)}\Big)$$

Where $f^{(t)}(v)$ are the embeddings of node v on layer t and $f^{(0)}(v) = F(v)$ are the initial embeddings. $W_1^{(t)}$ and $W_2^{(t)}$ are the parameters for layer t. By learning different values for $W_1^{(t)}$ and $W_2^{(t)}$, the model can distinguish between the node' previous features and the

neighbor features. We run each experiment 10 times and report the average accuracy plus standard deviations. We train each model for 400 epochs with a learning rate of 0.0003 (0.005 for saturation). We test only generated graphs unseen in training. Detailed statistics is available in Table 1. We run explanations on all relevant nodes in the test graphs, except for PGMExplainer, for runtime reasons.

6.3 Results

Table 2 shows results for all benchmarks. We could train the underlying GNNs to close-to-perfect accuracy (compare last row in Table 1), which is 1 for Infection and Negative Evidence and a bit less for Community due to the randomness in the underlying Stochastic Block Model. Generally, we observe edge-based methods working better than node based values, which is an expected result given that edges are finer-grained and the ground truth is edges and not nodes. For the infection benchmark (first column), we report the ratio of edges in $\mathcal{A}$ that are also in $\mathcal{T}$. We observe that all gradient-based methods and occlusion beat the baselines, some even yield perfect results. PGMExplainer and GNNExplainer fall behind the Nearest Neighbor baseline, showing they struggle with this dataset. We find that these two methods also struggle for the community detection benchmark. Here, we report the fraction of an increased attribution value after rewiring 1 or decreased attribution value after rewiring 2 (conditional on the GNN prediction changing as expected). A score of 0.5 equals random guessing (see first row). Both methods only score marginally above random guessing. Most methods only marginally beat random guessing on this benchmark. The two well-performing methods are edge gradients and occlusion, while integrated gradients on edges does better than random. The third column of Table 2 shows the ratio of times we can reject the null hypothesis that minority and white edges come from the same distribution with 99% confidence. We do one Kolmogorov-Smirnov-Test per node we explain, provided the node has at least 2 minority edges. Otherwise, we found the test to provide unstable results. Here, Edge Gradients and Integrated Gradient variants achieve quasi-perfect solutions. We show the results for the saturation effect in Figure 6. The $x-$axis shows the evidence strength (the difference between majority and minority edges) and the $y-$axis the median of attribution values (mean was sensitive to outliers). A method not suffering from the saturation effect shows as an (almost) horizontal line. Edge gradients and occlusion suffer from the saturation effect, while the other methods do not.

6.4 Explanation Runtimes

We did another run of the benchmarks with an emphasis on measuring runtime. We run the infection benchmark on a TITAN Xp GPU with 12GB memory. We show the average time per explanation in figure 8. Generally, gradient-based methods run similarly fast. Integrated gradient methods take longer since interpolation requires computing multiple gradients (50 in our case) instead of one. GNNExplainer and Occlusion change edge weights and rerun the model multiple times, thus taking more time. PGMExplainer takes the most time since it needs many perturbations for each explanation. In practice, one could look into caching strategies to alleviate some of this overhead. This is not a final solution since we could not apply it to the rewiring procedures in the community dataset. We wrote our code in PyTorch Geometric [8] and used Captum [15] for gradient-based methods.

Method	Infection	Community	Neg. Evidence
Random	0.00 ± 0.00	0.50 ± 0.00	0.01 ± 0.00
NearestNeighbor	0.44 ± 0.01	0.57 ± 0.00	N/A
PageRank	0.05 ± 0.00	0.56 ± 0.00	N/A
Node Gradients	0.03 ± 0.03	0.40 ± 0.03	0.00 ± 0.00
Edge Gradients	0.74 ± 0.12	**0.92 ± 0.01**	**1.00 ± 0.00**
Node IG	0.53 ± 0.13	0.55 ± 0.02	**1.00 ± 0.00**
Edge IG	**1.00 ± 0.00**	0.71 ± 0.13	**1.00 ± 0.00**
GradCAM	0.50 ± 0.11	0.33 ± 0.11	0.00 ± 0.00
Occlusion	**1.00 ± 0.00**	0.89 ± 0.01	0.25 ± 0.16
PGMExplainer	0.38 ± 0.06	0.53 ± 0.01	0.01 ± 0.01
GNNExplainer	0.32 ± 0.09	0.53 ± 0.04	0.32 ± 0.05

Table 2: Results for explanation methods, averaged over 10 runs. Columns show 1) overlap of explanation edges with ground truth 2) ratio of attribution value changes consistent with model changes after rewiring 3) ratio of attributions where minority and white edges differ. Higher is better.

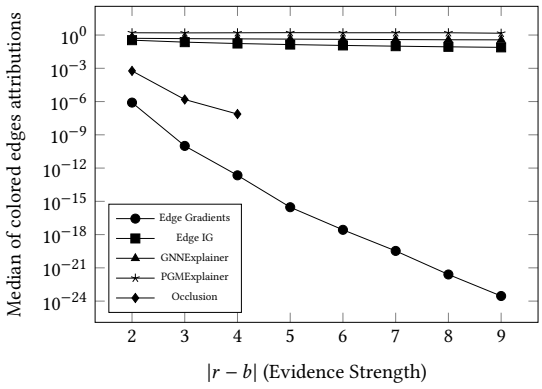


Figure 7: Saturation effect for explanation methods. $x-$axis shows evidence strength and $y-$axis median attribution strength in log-scale. To not suffer from the saturation effect, a method should stay almost parallel to the $x-$axis. Occlusion values become zero when $|r-b|>4$.

7 CONCLUSION

In this paper, we reinvestigated the current state of the art for evaluating GNN explanation methods. The current setup in comparing explanation method edges to ground-truth edges is sensible. However, we identify five pitfalls that can render the evaluation setup meaningless: when the GNN does not use the ground truth's edges. In turn, an accurate, faithful explanation method does neither. We propose two benchmarks that circumvent these pitfalls and evaluate explanation methods on pattern and community detection tasks. Furthermore, we add one benchmark that validates if an explanation method can detect negative evidence and is not prone to the saturation effect. In a re-evaluation of several state-of-the-art

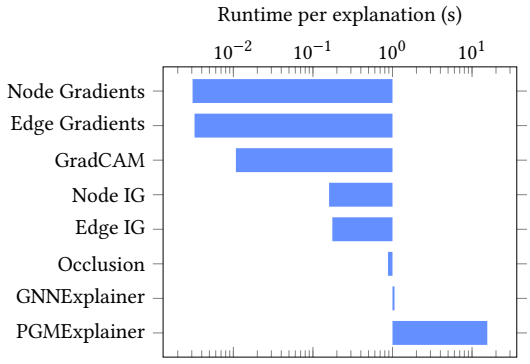


Figure 8: Runtime for several explanation methods

methods, we find that existing GNN explanation methods perform well. Integrated gradients on edges works generally well but leaves room for improvement, especially in community detection tasks.

We expect rising GNN capability in the future, requiring more complex explanation tasks and more sophisticated explanation methods. Our benchmark tasks can be scaled in difficulty, for example, detecting larger patterns further away in the infection benchmark. New benchmarks are another good choice provided they avoid the presented pitfalls.

REFERENCES

[1] Emmanuel Abbe. 2017. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research* 18, 1 (2017), 6446–6531.

[2] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access* 6 (2018).

[3] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *Proceedings of 32nd Conference on Neural Information Processing Systems*.

[4] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).

[5] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[6] A. Debnath et al. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry* 34, 2 (1991), 786–797.

[7] Lukas Faber, Amin K Moghaddam, and Roger Wattenhofer. 2020. Contrastive Graph Neural Network Explanation. *arXiv preprint arXiv:2010.13663* (2020).

[8] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*. PMLR, 1263–1272.

[10] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019. A benchmark for interpretability methods in deep neural networks. In *Proceedings of 33th Conference on Neural Information Processing Systems*. 9737–9748.

[11] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687* (2020).

[12] Sarthak Jain and Byron C Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 3543–3556.

[13] Chaojie Ji, Ruxin Wang, and Hongyan Wu. 2020. Perturb More, Trap More: Understanding Behaviors of Graph Neural Networks. arXiv:2004.09808 [cs.LG]

[14] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2901–2910.

[15] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for PyTorch. arXiv:2009.07896 [cs.LG]

[16] Ana Lucic, Maartje ter Hoeve, Gabriele Tolomei, Maarten de Rijke, and Fabrizio Silvestri. 2021. CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. arXiv:2102.03322 [cs.LG]

[17] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of 31th Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA.

[18] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized Explainer for Graph Neural Network. In *Proceedings of 34th Conference on Neural Information Processing Systems*.

[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Santiago, Chile*.

[20] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).

[21] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report.

[22] Damian Pascual, Gino Brunner, and Roger Wattenhofer. 2020. Telling BERT's full story: from Local Attention to Global Aggregation. *arXiv preprint arXiv:2004.05916* (2020).

[23] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " 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*.

[25] Benjamin Sanchez-Lengeling, Jennifer Wei, Brian Lee, Emily Reif, Peter Wang, Wesley Wei Qian, Kevin McCloskey, Lucy Colwell, and Alexander Wiltschko. 2020. Evaluating Attribution for Graph Neural Networks. *Proceedings of 34th Conference on Neural Information Processing Systems* 33 (2020).

[26] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[27] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685* (2017).

[28] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *International Conference on World Wide Web (WWW), Florence, Italy*.

[29] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*. 3319–3328.

[30] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[31] Minh N. Vu and My T. Thai. 2020. PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks. arXiv:2010.05788 [cs.LG]

[32] Shengjie Wang, Tianyi Zhou, and Jeff Bilmes. 2019. Bias also matters: Bias attribution for deep neural network explanation. In *International Conference on Machine Learning*. 6659–6667.

[33] Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not Explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 11–20.

[34] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.

[35] Mengjiao Yang and Been Kim. 2019. Benchmarking Attribution Methods with Relative Feature Importance. *arXiv* (2019), arXiv–1907.

[36] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Proceedings of 33rd Conference on Neural Information Processing Systems*.

[37] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design space for graph neural networks. In *Proceedings of 34th Conference on Neural Information Processing Systems*.

[38] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in Graph Neural Networks: A Taxonomic Survey. arXiv:2012.15445 [cs.LG]

[39] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[40] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.

A CHANGES TO EXPLANATION METHODS

Our implementation on GNNExplainer is based on PyTorch Geometric version 1.6.1. Since then, there was an update, but its changes only add configuration options for edge and node reductions (see commit [2]). We create a version of GNNExplainer where we can specify the target class we want to explain—the reference implementation always does a forward pass of the GNN and explains the class with highest probability.

We modified PGMExplainer's original implementation to be compatible with PyTorch Geometric. We only use single instance explanations in this benchmark. We also modified this method to work with a specified target class we want to explain.

[2] https://github.com/rusty1s/pytorch_geometric/commit/7a29dd48121e3c62d5de323869828bd723e3168f