What Does My GNN Really Capture? On Exploring Internal GNN Representations

Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, Céline Robardet

What Does My GNN Really Capture? On Exploring Internal GNN Representations

Luca Veyrin-Forrer[1] , **Ataollah Kamal**[1] , **Stefan Duffner**[1] ,
Marc Plantevit[2] and **Céline Robardet**[1]

[1]Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS, UMR5205, FR-69621 Villeurbanne
[2]EPITA Research and Development Laboratory (LRDE), FR-94276 Le Kremlin-Bicêtre

Abstract

GNNs are efficient for classifying graphs but their internal workings is opaque which limits their field of application. Existing methods for explaining GNN focus on disclosing the relationships between input graphs and the model's decision. In contrary, the method we propose isolates internal features, hidden in the network layers, which are automatically identified by the GNN to classify graphs. We show that this method makes it possible to know the parts of the input graphs used by GNN with much less bias than the SOTA methods and therefore to provide confidence in the decision process.

1 Introduction

Graphs are a powerful and popular data structure used to represent relational data. One of their specificity is that their underlying structure does not form a Euclidean space, characteristic that facilitates the direct use of generic machine learning techniques. To overcome this difficulty, Graph Neural Networks (GNNs) learn embedding vectors $\mathbf{h}_v \in \mathbb{R}^K$ to represent each node v as a vector of fixed dimension K that eases comparison between similar nodes. GNN methods employ a message propagation strategy that recursively aggregates information from nodes to neighbouring nodes and produce vector representations of ego-graphs centered in a node v with radii equal to the recursion index so that to optimize a classification task based on these vectors.

Related work. Although GNNs achieve exceptional performance in many tasks, a major drawback is their lack of interpretability. The last five years have witnessed a huge growth in the definition of techniques for explaining deep neural networks [Burkart and Huber, 2021; Molnar, 2020], especially for image and text data. However, these methods cannot be directly used for explaining GNN due to the none grid-like format of graphs [Yuan *et al.*, 2020b]. Nevertheless, a few proposals have been made to explain GNNs according to two distinct approaches and have gained visibility.

Instance-level methods aim to learn a mask seen as an explanation of the model decision for a graph instance. These methods provide explanations specific to an input graph by identifying its important characteristics on which the model builds its prediction. We can identify four different families of methods. The *gradient/feature-based methods* [Baldassarre and Azizpour, 2019; Pope *et al.*, 2019] – directly adapted from dedicated image and text solutions – use the gradients or hidden feature map values to compute the importance of the input features. The *perturbation-based methods* [Ying *et al.*, 2019; Luo *et al.*, 2020] aim at learning a graph mask by studying the prediction changes when perturbing the input graphs. The *surrogate methods* [Huang *et al.*, 2020; Vu and Thai, 2020] explain an input graph by sampling its neighborhood and learning an interpretable model. The *decomposition-based methods* [Pope *et al.*, 2019; Schnake *et al.*, 2020] start by decomposing the prediction score to the neurons in the last hidden layer. Then, they back-propagate these scores layer by layer until reaching the input space. On top of that,

GraphSVX [Duval and Malliaros, 2021] falls into these 4 categories by learning a surrogate explanation model on a perturbed dataset that decomposes the explained prediction among input nodes and features based on their contribution. These methods perform well on metrics evaluating the relationships between explanations and model decisions. However, it appears that these masks can lead to unreliable explanations, and most importantly, can lead to misleading interpretations for the end-user. One can be tempted to interpret all the nodes or features of the mask as responsible for the prediction leading to wrong assumptions. For instance, a node feature may be perceived as important for the GNN prediction, whereas there is no difference between its distribution within and outside the graphs validating the mask.

Model-level methods. The only existing model-level method is XGNN [Yuan *et al.*, 2020a] which aims at training a graph generator to maximize the predicted probability for a class and uses such graphs to explain this class. However, it is based on the strong assumption that each class can be explained by a single graph, which is unrealistic when considering complex phenomena.

Most of the aforementioned methods aim at either explaining the final decision of a GNN or generating a representative graph for a given decision. We believe that focusing only on the model decision does not allow to fully understand how the model behaves and builds its decision. One can provide additional insights about the GNN by not only looking at the output of the model, but also by trying to characterize some representation subspaces that the model has built in the different layers.

Contribution. We introduce a new method, called DISCERN (DISClosing the IntERnal workings of gNns with graphs), that aims at characterizing interesting internal representations of the GNN with graphs as illustrated in Figure 1. In each hidden layer of the GNN, we identify sets of neurons that are differently activated according to the output variable. Such activation rules capture specific configurations in the embedding space of a given layer that is discriminant for the GNN decision. We believe that such activation rules also catch hidden features of input graphs. However, these activation rules cannot be easily interpreted by human beings. Our goal is then to explain each activation rule by generating a graph that fully embeds in the related

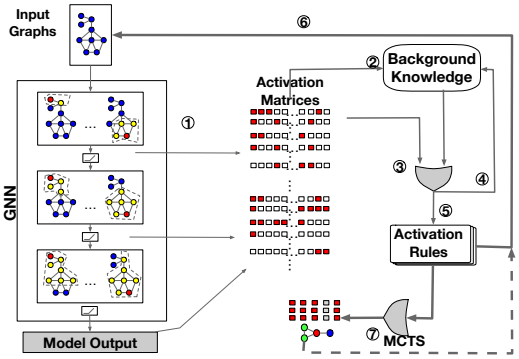

Figure 1: DISCERN overview. For each layer (1), a background model captures the activation distribution (2) used to assess the interest of activation rules (3). The most relevant rule is added to the pattern set (5) and used to update the background model. Steps (2-5) are repeated until no more informative activation rule is retrieved. Activation rules directly support instance level explanations (6) or are transformed into graphs (7).

subspace identified by the rule. To this end, we define a proximity measure to assess how close a graph is to an activation rule and optimize this measure using a Monte Carlo Tree Search (MCTS).

Our main contributions are as follows. We introduce the novel problem of identifying activation rules and their characterization with subgraphs (Section 2). We report an empirical evaluation on three real-world datasets (Section 3) where we show that our activation rules identify masks of higher quality than SOTA methods and that DISCERN provide good explanations with realistic graphs compared to baselines. The code source of the method is made available: https://github.com/luvf/inside-gnn.

Targeted audience. Imagine you are a scientist who has a GNN that accurately predicts the graphs you are studying. This means that the GNN is able to capture interesting features and combine them. Therefore, understanding how the GNN constructs its internal representation would shed new light on your field of research. This is our goal.

2 Method

We propose an introspective method to explain GNN decisions in a post-hoc manner. The method identifies sets of neurons that work together in the decision-making process, called activation rules,

and characterizes them with subgraphs.

2.1 GNN Embedding Vectors

We consider a set of graphs $\mathcal{G}$ with labels, $G = (V, E, L)$ with V a set of vertices, E a set of edges in $V \times V$, and L a mapping between vertices and labels of T: $L \subseteq V \times T$. We study GNNs that classify each graph of $\mathcal{G}$ into two categories $\{0, 1\}$: GNN : $\mathcal{G} \rightarrow \{0, 1\}$. The GNN takes decisions at the level of each graph on the basis of vectors, the neurons, computed at the level of the nodes of each graph. For each node, ego-graphs of increasing radii are embedded in a Euclidean space in such a way that similar ego-graphs are associated to similar vectors. Indeed, we consider Graph Convolutional Networks (GCN) [Kipf and Welling, 2017] that compute vectors $\mathbf{h}_v^\ell$ associated to the ego-graph centered in vertex v with radius ℓ, recursively by the following formula: $\mathbf{h}_v^\ell = ReLU\left(\mathbf{W}_\ell \cdot \sum_{w \in \mathcal{N}(v)} \frac{e_{w,v}}{\sqrt{d_v d_w}} \mathbf{h}_v^{\ell-1}\right)$, $e_{v,w}$ is the weight of the edge between nodes v and w, $\mathcal{N}(v)$ is the set of neighboring nodes of v including v, ReLU is the rectified linear activation function, and $\mathbf{W}_\ell$ are the parameters learnt during the training phase of the model. We also have $d_v = \sum_{w \in \mathcal{N}(v)} e_{v,w}$ and $\mathbf{h}_v^0$ is the initial feature vector for node v with the one-hot encoding of its label from T. Each vector is of size K and ℓ varies from 0 up to L (the maximum number of layers in the GNN), two hyperparameters of the GNN.

Once the GNN learnt, the vectors $\mathbf{h}_v^\ell$ capture the key characteristics of the corresponding graphs on which the classification decision is made. When one of the vector components is of high value, it plays a role in the decision process. More precisely, activated components of the vectors – the indices k such that $(\mathbf{h}_v^\ell)_k > 0$ – are combined by the neural network in a path leading to the decision, either 0 or 1. For a given layer ℓ, the activated components of the embedding $\mathbf{h}_v^\ell$ correspond to the part of the ego-graph centered in v and of radius ℓ that triggers the decision.

2.2 Activation Rules

To shed light on the inner workings of GNN, we seek to identify sets of neurons that are co-activated for a given class. Activation rules are groups of vector components that are mostly activated together in graphs having the same GNN decision. We say that a rule $R \equiv \mathbf{A}^\ell \rightarrow c$, with $\mathbf{A}^\ell$ a binary vector of size K and $c \in \{0, 1\}$, is activated for a graph $g_i = (V_i, E_i, L_i) \in \mathcal{G}$ if there exists a node v in V_i whose components in layer ℓ that correspond to the activated components of the rule are also co-activated, that is iff $\exists v \in V_i$ such that if $\mathbf{A}_k^\ell = 1$ then $(\mathbf{h}_v^\ell)_k > 0$. The graphs of decision c for which $\mathbf{A}^\ell$ is activated constitute the support of R: $\mathrm{supp(R)} = \{\mathrm{g_i} \mid \exists \mathrm{v} \in \mathrm{V_i}\ \mathrm{s.t.}\ \forall \mathrm{k}\left(\mathbf{A}_\mathrm{k}^\ell = 1\right) \Rightarrow \left((\mathbf{h}_\mathrm{v}^\ell)_\mathrm{k} > 0\right)\ \mathrm{and}\ \mathrm{GNN(g_i)} = \mathrm{c}\}$.

We evaluate the interest of a rule using an interestingness measure based on the FORSIED framework [De Bie, 2011] that makes possible to measure the subjective interest of a rule using information theory to quantify both its informativeness and its complexity. Suppose that knowledge of the distribution of activated neurons over the nodes is formalized by a probability distribution P, called background distribution. From this distribution, we can define an optimal code to transmit a pattern, a set of activated neurons, to a user. This optimal code is specified as $-\log(P(x))$ for data x. The more probable the user judges the data to be, the shorter the code describing it would be. Thus, by estimating the probability $P((h^\ell)_k, v)$ that the component $(h^\ell)_k$ is activated for a node v, we can evaluate the interest of a rule by the length of the code for communicating the rule to the user using the sum of $-\log(P((h^\ell)_k, v))$ over all $(h^\ell)_k$ of the rule and v, an activated node in its support graphs. The more probable the pattern – and therefore the less interesting – the shorter the code. As there may exist several nodes activated in a single graph, we choose the one that maximizes the above value and define the Information Content as: $IC(R) = \sum_{g=(V,E) \in supp(R)} \max_{v \in V} -\sum_{x \in A^\ell} \log(P(x, v))$.

The initial background distribution P is estimated using the Maximum Entropy Principle (other distributions introducing additional knowledge unduly [De Bie, 2011]) coercing the distribution by the frequency of activation of each component on the nodes of the graphs and by the average number of activated components per nodes. The explicit mathematical MaxEnt model solution can be found in [De Bie, 2009].

A rule with a large IC is more informative, but it may be more difficult for the user to assimilate it, especially when its description is complex. To avoid this drawback, the IC measure is contrasted by the description length which measures the com-

plexity of communicating the rule to the user. The higher the number of components in A^ℓ, the more difficult to communicate it to the user. Therefore, we propose to measure the description length of an activation rule by $DL(A^\ell) = a(|A^\ell|) + b$ with a the cost for the user to assimilate each component and b a fixed cost for the rule. We set $b = 1$ and $a = 0.6$, as the constant parameter b does not influence the relative ranking of the rules, and with a value of 1, it ensures that the DL value is greater than 1. With $a = 0.6$, we express a slight preference toward shorter rules. Hence, the subjective interestingness measure of an activation rule is $SI(R) = \frac{IC(R)}{DL(A^\ell)}$.

To measure how a rule R is specific to a GNN decision, we compare $SI(\mathbf{A}^\ell \rightarrow c)$ with $SI(\mathbf{A}^\ell \rightarrow 1 - c)$: $SI\_SG(\mathbf{A}^\ell \rightarrow c) = \omega_c SI(\mathbf{A}^\ell \rightarrow c) - \omega_{1-c} SI(\mathbf{A}^\ell \rightarrow 1 - c)$. The weights ω_0 and ω_1 are used to counterbalance the measure in unbalanced decision problems. The rational is to reduce the SI values of the majority class. We set $\omega_0 = \max(1, \frac{|GNN(g_i)=1|}{|GNN(g_i)=0|})$ and $\omega_1 = \max(1, \frac{|GNN(g_i)=0|}{|GNN(g_i)=|})$.

Activation rules are computed in an iterative way. First, the activation rule that maximizes SI_SG is computed with an enumerate-and-rank approach. Then, the knowledge bring by the rule is integrated into the background distribution P to force the extraction of diversified rules. Once the rule is known, its subjective interest falls down to 0. This consists in setting the probabilities corresponding to the rule components and its supporting nodes to 1. Then, the process is iterated a fixed number of times or until there is no more rule with a positive SI_SG value. Details can be found in [Veyrin-Forrer *et al.*, 2021].

2.3 Characterizing Activation Rules With Subgraphs

For each activation rule, we seek to identify a subgraph whose presence in an input graph triggers the rule. For this, we are looking for a subgraph whose GNN embedding in layer ℓ is as close as possible to the activation rule. This requires defining a measure of proximity between embedding and activation rule and finding the subgraph that maximizes it.

We propose two measures to evaluate the proximity between an ego-graph embedding $\mathbf{h}_v^\ell$ and an activation rule $\mathbf{A}^\ell \rightarrow c$, with $\mathbf{A}^\ell = \{a_1, \ldots, a_K\}$, $a_i \in \{0, 1\}$. We denote by $\mathbf{E}_g = \{\epsilon_1, \ldots, \epsilon_K\}$ the ego-graph embedding truncated to fit the interval $[0, 1]$: $\epsilon_k = \min\big(\max(0, (\mathbf{h}_v^\ell)_k), 1\big)$. The truncation avoids distorting our measurements by extreme values. The cosine measure evaluates the similarity between $\mathbf{E}_g$ and $\mathbf{A}^\ell$: $\text{Cos}(\mathbf{E}_g, \mathbf{A}^\ell) = \frac{\mathbf{E}_g \,.\, \mathbf{A}^\ell}{\|\mathbf{E}_g\| \, \|\mathbf{A}^\ell\|} = \frac{\sum a_i \epsilon_i}{\sqrt{\sum a_i^2} \sqrt{\sum \epsilon_i^2}}$. It is equal to the cosine of the angle between the two vectors, or identically be the inner product of the vectors normalized to length 1. We can also use the cross-entropy measure (equivalently the log-likelihood) to evaluate this similarity: $\text{CE}(\mathbf{E}_g, \mathbf{A}^\ell) = \sum a_i \log(\epsilon_i)$. It increases with the number of components that have a large value in the embedding of the ego-graph for the components activated in the activation rule.

We also define a realism score that depends both on the probability that two vertices are connected according to their type. This score is added to the similarity measure to form the final score used to evaluate the adequacy of a graph to represent an activation rule.

We use Monte Carlo Tree Search (MCTS) to find an ego-graph g_t that maximizes the above mentioned measure. Each node of the tree is obtained by adding an edge to the graph of its parent node. This process stops when either (1) the diameter of the graph is greater than ℓ, or (2) the number of edges is greater than $min\text{-}edges$, or (3) the number of vertices is greater than $min\text{-}vertices$. The tree is partially explored using the classical the Upper Confidence Bound UCB1 to guide the selection of the node to be expanded [Auer *et al.*, 2002]. The possible actions to extend a non-terminal ego-graph g_t are the different edges that it is possible to add to the graph so that it remains connected: (1) those with an endpoint in V_t, the other one being a new node ($v \notin V_t$) labeled with one label of T, or (2) the ones whose two endpoints are in V_t. Thus, there are $\#V_t \times (\#V_t + \#T)$ possible actions. To improve the search, we cut down this set of available actions. The goal is to reduce the breadth of the tree, to avoid the generation of some isomorphic graphs, while obtaining more homogeneous values of measure $score$ on the graphs obtained in the subtree.

3 Experiments

We evaluate DISCERN through several experiments. We first describe the datasets and the experimental setup. Then, we compare our method against sev-

eral instance-level and model-level baselines. Finally, we discuss some examples of subgraphs generated by our method and the baselines. Experiments are performed on three graph classification datasets (Aids [Morris *et al.*, 2020], BBBP [Wu *et al.*, 2017], Mutagen [Morris *et al.*, 2020]) depicting molecules and important properties in Chemistry or Drug Discovery (class). A 3-convolutional layer GNN (with $K = 20$) is trained on each dataset.

We mine 10 activation rules per layer and for each class. This experimental study aims to answer the following questions: Are the activation rules good? How does DISCERN behave against baselines? To that end we compare our method against both instance-level and model-level methods. For instance-level methods, we consider 3 SOTA methods: **GNNExplainer** [Ying *et al.*, 2019], **PGExplainer** [Luo *et al.*, 2020] and **PGM-Explainer** [Vu and Thai, 2020]. We also examine 3 model-level baselines: **Random** generates graphs randomly by calling the Roll-out function 250x. **XGNN++** is an extension of XGNN [Yuan *et al.*, 2020a] to our problem. We integrate both Cos and CE metrics as function optimized by XGNN. We set a budget that corresponds to 5000 calls to the GNN to do a fair comparison with DISCERN. **DISC-GSPAN** is a sound and complete method that aims at discovering discriminant subgraphs within a collection based on GSPAN enumeration [Yan and Han, 2002] while exploiting some tight upper bounds on the WRAcc measure. The input dataset contains the set of ego-network of nodes that support the activation rule as the "positive class" and the ego-network of nodes not involved in the support of the rule as the negative class. Then, DISC-GSPAN consists in computing the top-k subgraphs that are discriminant for the positive class.

Comparison to instance-level methods. We consider a ground-truth free metric to compare the methods. We opt for the *Fidelity* [Pope *et al.*, 2019] which is defined as the difference of predicted probability between the predictions on the original graph and the one obtained when masking part of the graph based on the explanations: $Fid. = \frac{1}{N} \times \sum_{i=1}^{N} (f(g_i)_{y_i} - f(g_i \setminus m_i)_{y_i})$, where m_i is the mask, $g_i \setminus m_i$ is the complementary mask and $f(g)_{y_i}$ is the prediction score for class y_i. Similarly, we study the prediction change by keeping important features (i.e., the mask) and removing the others as Infidelity measure does: Infid. $= \frac{1}{N} \times \sum_{i=1}^{N} (f(g_i)_{y_i} - f(m_i)_{y_i})$. The higher the fidelity, the lower the infidelity, the better the explainer.

Model	Aids	BBBP	Mutagen
	(a) Fidelity		
AR(node)	**0.175**	**0.362**	**0.582**
DISCERN(Cos)	0.025	0.175	0.311
DISCERN(CE)	0.027	0.181	0.288
GnnExplainer	0.036	0.100	0.177
PGExplainer	0.032	0.098	0.157
PGM-Explainer	0.089	0.212	0.260
	(b) Infidelity		
AR(node)	0.767	0.374	0.237
DISCERN(Cos)	0.079	0.149	0.214
DISCERN(CE)	**0.031**	0.438	0.191
GnnExplainer	0.036	0.099	**0.140**
PGExplainer	0.038	**0.098**	0.157
PGM-Explainer	0.765	0.392	0.354
	(c) Sparsity		
AR(node)	**0.897**	0.870	0.731
DISCERN(Cos)	0.765	0.666	0.717
DISCERN(CE)	0.731	0.639	0.754
GnnExplainer	0.501	0.501	0.505
PGExplainer	0.547	0.534	0.515
PGM-Explainer	0.855	**0.884**	**0.956**

Table 1: Assessing the explanations with several metrics. A better explainer achieves higher fidelity, lower infidelity while keeping a sparsity close to 1.

Obviously, masking all the input graph would have important impact to the model prediction. Therefore, the former measures should not be studied without considering the *Sparsity* metric that aims to measure the fraction of graph selected as mask by the explainer: $Spars. = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{|m_i|}{|g_i|}\right)$, where $|m_i|$ denotes the size of the mask m_i and $|g_i|$ is the size of g_i (the size includes the number of nodes, of edges and the attributes associated to them). Based on these measures, a better explainability method achieves higher fidelity, lower infidelity while keeping a sparsity close to 1.

Several policies to build a mask from an activation rule are possible. We opt for the simplest policy **AR(node)** which takes as a mask only the nodes that are covered by the activation rule and the edges adjacent to these nodes. This policy allows to assess how relevant are the activation rules. We also consider masks built thanks to DISCERN, i.e., that associates a unique interpretable graph to each activation rule. To this end, for an instance graph, we select among all the rules that are activated, the related generated graph that maximizes the trade-off between Fidelity and Infidelity. The selected subgraph is used as a mask for explanation.

	Aids	BBBP	Mutagen
Random	4.60	4.37	4.13
DISC-GSPAN	3.69	3.82	4.17
XGNN++(Cos)	4.13	4.42	4.57
XGNN++(CE)	4.06	4.25	4.48
DISCERN(Cos)	**3.12**	**3.08**	**3.39**
DISCERN(CE)	3.29	3.32	3.68

Table 2: Average L2 norm between the GNN embeddings of graphs provided by each method and the activation rules. The lower the value, the better.

Table 1(a) outlines the performance of the explainers based on the Fidelity measures. Results show that **AR(node)** outperforms the baselines. These results must be analysed while considering the sparsity (see Table 1(c)). **AR(node)** provides sparser explanation than the baselines. The quality of the explanations are also assessed with the Infidelity metrics in Table 1(b). **AR(node)** is outperformed by GNNExplainer and PGExplainer. This suggest that a rule taken in isolation does not allow a correct classification of a graph. It is undoubtedly necessary to consider combinations of graphs to explain a decision. Interestingly, DISCERN that builds on activation rules provides better results in term of Indfidelity than **AR(node)**, achieving score that are similar to the SOTA methods while having better fidelity and sparsity scores than these methods in most of the cases. Finally, the generated graph brings further interpretability on activation rules without altering too much the performance of the explainer directly built from these rules. All together, these results suggest that the activation rules allow to identify relevant representation space within the GNNs.

Comparison against model-level baselines. The three baselines (i.e., Random, DISC-GSPAN, and XGNN++) optimizes either Cos or CE metrics. We compare them against DISCERN. For each activation rule, we study the L2 norm between the GNN embedding of the best graph generated by each method and the activation rule $\mathbf{A}^{\ell}$. Results are reported in Table 2. For both CE and Cos, DISCERN provides graphs that better embed within the target space than any other method.

Examples. We report in Figure 2 the best graphs for each method for two activation rules on Mutagen. These rules are highly correlated to the decision "Mutagenicity". Graphs generated by Random are not shown as they are unrealistic. For the first

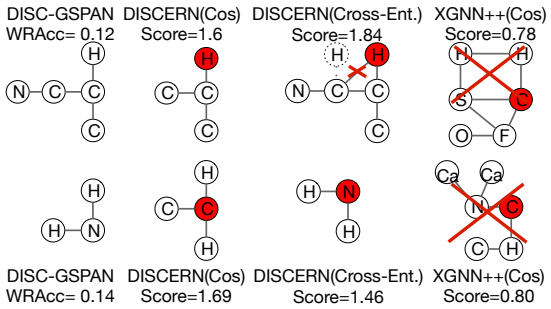

Figure 2: Graphs generated by each method for two activation rules (rows) that are highly correlated to mutagenicity. Red cross highlights unrealistic bonds or molecules. Red nodes activate the rule. XGNN++(CE) returns only a carbon atom.

rule (left column), DISC-GSPAN and DISCERN identify parts of toxycophores (Bay-region, K-region) [Kazius *et al.*, 2005]. XGNN++ provides either unrealistic (i.e., Cos) or too general graphs (i.e., only one carbon). Note that the graph generated by DISCERN for CE is not entirely realistic since a hydrogen atom cannot have two bonds. Nevertheless, duplicating this atom and binding it to another carbon (dashed node and edge) leads to a similar representation which is realistic. For the second activation rule, both DISC-GSPAN and DISCERN(CE) depict a part of amine group (NH_2). These molecules are known to be toxicophore. DISCERN(Cos) generates a Vinylidene group also known to be toxic. Graphs generated by XGNN++ are unrealistic.

4 Conclusion

We have introduced a novel method for explaining internal representations of GNNs. Given some activation rules that define internal representations having a strong impact on the classification process, DISCERN generates, with a MCTS based approach, realistic graphs that fully embed in the related subspace identified by the rules. Experiments demonstrate that (i) the activation rules identify relevant representation spaces built by the GNN and (ii) DISCERN makes it possible to describe these activation rules with realistic graphs and then to capture insights about the internal representations built by the GNN. We believe that such method can support knowledge discovery from powerful GNNs and provide insights on object of study for scientists.

References

[Auer *et al.*, 2002] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[Baldassarre and Azizpour, 2019] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *CoRR*, abs/1905.13686, 2019.

[Burkart and Huber, 2021] Nadia Burkart and Marco F. Huber. A survey on the explainability of supervised machine learning. *J. Artif. Intell. Res.*, 70:245–317, 2021.

[De Bie, 2009] Tijl De Bie. Finding interesting itemsets using a probabilistic model for binary databases. Technical report, Univ. Bristol, 2009.

[De Bie, 2011] Tijl De Bie. An information theoretic framework for data mining. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *ACM SIGKDD*, pages 564–572. ACM, 2011.

[Duval and Malliaros, 2021] Alexandre Duval and Fragkiskos D. Malliaros. Graphsvx: Shapley value explanations for graph neural networks. In *ECML PKDD 2021*, pages 302–318, 2021.

[Huang *et al.*, 2020] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *CoRR*, abs/2001.06216, 2020.

[Kazius *et al.*, 2005] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Jour. med. chem.*, 48(1):312–320, 2005.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

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

[Molnar, 2020] C. Molnar. Interpretable machine learning. Lulu.com, 2020.

[Morris *et al.*, 2020] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset. *CoRR*, abs/2007.08663, 2020.

[Pope *et al.*, 2019] Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. Explainability methods for GCN. In *IEEE CVPR*, pages 10772–10781, 2019.

[Schnake *et al.*, 2020] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T. Schütt, Klaus-Robert Müller, and Grégoire Montavon. XAI for graphs. *CoRR*, abs/2006.03589, 2020.

[Veyrin-Forrer *et al.*, 2021] Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, and Céline Robardet. On GNN explanability with activation patterns. https://hal.archives-ouvertes.fr/hal-03367714, 2021.

[Vu and Thai, 2020] Minh N. Vu and My T. Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In *NeurIPS 2020*, 2020.

[Wu *et al.*, 2017] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet. *CoRR*, abs/1703.00564, 2017.

[Yan and Han, 2002] Xifeng Yan and Jiawei Han. GSPAN: Graph-based substructure pattern mining. In *(ICDM 2002)*, pages 721–724. IEEE Computer Society, 2002.

[Ying *et al.*, 2019] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *NeurIPS 2019*, pages 9240–9251, 2019.

[Yuan *et al.*, 2020a] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: towards model-level explanations of graph neural networks. In *ACM SIGKDD*, pages 430–438, 2020.

[Yuan *et al.*, 2020b] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *CoRR*, abs/2012.15445, 2020.