Prototype-Based Interpretable Graph Neural Networks

Alessio Ragno, Biagio La Rosa, and Roberto Capobianco

***Abstract*—Graph neural networks have proved to be a key tool for dealing with many problems and domains, such as chemistry, natural language processing, and social networks. While the structure of the layers is simple, it is difficult to identify the patterns learned by the graph neural network. Several works propose post hoc methods to explain graph predictions, but few of them try to generate interpretable models. Conversely, the topic of the interpretable models is highly investigated in image recognition. Given the similarity between image and graph domains, we analyze the adaptability of prototype-based neural networks for graph and node classification. In particular, we investigate the use of two interpretable networks, ProtoPNet and TesNet, in the graph domain. We show that the adapted networks manage to reach better or higher accuracy scores than their respective black-box models and comparable performances with state-of-the-art self-explainable models. Showing how to extract ProtoPNet and TesNet explanations on graph neural networks, we further study how to obtain global and local explanations for the trained models. We then evaluate the explanations of the interpretable models by comparing them with post hoc approaches and self-explainable models. Our findings show that the application of TesNet and ProtoPNet to the graph domain produces qualitative predictions while improving their reliability and transparency.**

***Impact Statement*—Explainability works for graph neural networks mainly focus on post-hoc explanations rather than developing self-explainable models. By adapting two self-interpretable prototype-based networks from the image to the graph domain, we analyze a self-explainable graph technique that enables both graph and node classification. We show that the explainable methods are able to obtain equal or higher accuracy scores than their respective black box versions while providing the possibility to obtain insights about their reasoning process.We set the basis for future works in the explainability of the graph neural networks.We also provide an open source implementation that can be employed for approaching several scientific problems, such as computational chemistry and natural language processing.**

***Index Terms*—Artificial neural networks, case-based reasoning, classification and regression, deep learning, explainable artificial intelligence, interpretable artificial intelligence.**

Manuscript received 31 May 2022; revised 5 September 2022; accepted 9 November 2022. Date of publication 16 November 2022; date of current version 9 April 2024. This paper was recommended for publication by Associate Editor Koduvayur P. Subbalakshmi upon evaluation of the reviewers' comments. *(Corresponding author: Alessio Ragno.)*

Alessio Ragno and Biagio La Rosa are with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy (e-mail: ragno@diag.uniroma1.it; larosa@diag.uniroma1.it).

Roberto Capobianco is with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy, and also with Sony AI, 8952 Zurich, Switzerland (e-mail: capobianco@diag.uniroma1.it).

Digital Object Identifier 10.1109/TAI.2022.3222618

I. Introduction

THE ability of graph neural networks (GNNs) to adapt to several domains, such as social networks [1], molecules [2], [3], and textual data [4] have made these architectures palatable. Although GNNs succeed in solving many problems, it is difficult to understand the motivations behind their predictions or to identify the learned patterns.

The field of explainable artificial intelligence (XAI) addresses this issue by proposing ways to understand the inner working of the models and provide interpretations of their predictions. XAI approaches can be categorized into two main groups: methods that interpret already trained models, known as *post hoc* methods; and the ones that aim at defining models that are interpretable by design, i.e., *intrinsic methods* [5]. In this study, we focus on the latter.

The field of image recognition is one of the most active on this topic. Some recent works of this field focus on prototype-based networks that learn representations able to encode entire samples, or common parts of the input, called prototypes, which are then used during the reasoning process. Among them, ProtoPNet [6] and TesNet [7] use similarity distances to identify relevant portions of the images close to the learned prototypes and use these to classify the images. Additionally, the case-based reasoning behind these models makes it easier to interpret their predictions.

The aim of this work is to take a step in this direction by investigating the application of part-based prototype models to the graph domain for both graph and node classification tasks. The idea is to look at images as a particular type of graph and replace 2D-convolution layers with graph ones. In particular, we focus on adapting ProtoPNet and TesNet to learn prototypes that represent node embeddings and that are able to identify relevant class-aware motifs. At inference time, the models actively use the prototypes to generate the prediction, and we can use them to extract explanations about the model's behavior too.

More in detail, since the prediction is based on the similarity between the node prototypes and the input graph/node, we inspect the subgraphs that most activate the prototypes of a certain class to understand the reasoning process of the model. When using n graph convolution layers, in facts, the node embeddings contain a latent representation of the *k-hop* surrounding subgraph. This information allows us to explain the models both locally, i.e., for a certain prediction, and globally, i.e., visualizing the learned patterns from the network.

We then train such networks to perform graph and node classification on seven datasets and compare their performance

against their black-box variants. We further investigate the explanations by comparing different ways to compute them and by providing a qualitative and quantitative comparison against alternative methods, i.e., post hoc.

Summarizing, the purpose of this work is to show and analyze the benefits of the adaptation of ProtoPNet and TesNet to the graph domain. Indeed, we note that they are often overlooked by current research, despite their capabilities. We highlight the changes in the architecture needed for the adaptation, and propose a different procedure tailored for the graph domain to extract the explanations. More in details, the contributions can be summarized as follows.

1) We analyze the adaptation of ProtoPNet and TesNet to GNNs and study the elements that impact the performance and explanations.
2) We show how to explain the obtained models through the inspection of the prototype activations.
3) We analyze the performance of these approaches by comparing the models' classification and explanation performances with black-box architectures over seven graph and node classification datasets:
4) We compare the performance of these approaches with other state-of-the-art self-explainable architectures over three classification datasets.
5) We analyze the quality of the explanations provided by the interpretable models by comparing them with the ones of post hoc methods and self-explainable models.
6) We provide an open-source implementation[1] useful as a baseline for future work.

The rest of this work is organized as follows. Section II presents the related work for GNNs and interpretability; Section III describes prototype-based neural networks and how they can be translated to the graph domain; in Section IV, we analyze the results for both performances and explanations. Finally, Section V concludes this article.

II. Related Work

A. Prototype-Based Networks

The idea of prototype-based neural networks is to use an embedding for clustering points of the datasets around specific ones called prototypes. Snell et al. [8] first propose this framework for performing few-shot classification, where a classifier has to be able to generalize to new classes not seen during the training process. Having some representative instances that resemble a big portion of the dataset, prototypes can also be used by post hoc methods, in particular those that generate explanations by example. These methods explain the prediction by identifying samples similar to the input that the model associates with either a similar prediction (i.e., factuals) or an opposite one (i.e., counterfactuals). When it is possible to assign a semantic meaning to the set of samples identified by a prototype, we refer to them as concepts. For instance, Kim et al. [9] identify concepts by using concept activation vectors, which are vectors in the direction of the neuron activations of a concept's set of examples.

Explainable architectures can learn and use concepts too, both in a supervised [10] and an unsupervised fashion [6], [7], [11]. They usually use embedding layers to represent concepts and feed them into a final layer to make the prediction.

The supervised process aligns the latent representation to specific labeled concepts. Even if it produces a meaningful representation, this approach requires a labeled concept dataset. On the other hand, unsupervised methods aim to extract relevant patterns for predicting the desired output in the embedding layer.

Among the unsupervised methods, there are different solutions for tackling various tasks. For time series, for example, Ni et al. [12] proposed to split input sequences into relevant segments that represent with prototypes. Alvarez Melis and Jaakkola [13], instead propose a general form of a self-explainable neural network (SENN), based on prototypes that consists of the following three components: a concept encoder that transforms input features into basis concepts; an input-dependent parametrizer that calculates relevance scores for the basis concepts; and an aggregation function that combines the first two to produce the prediction. Starting from the SENN architecture, Li et al. [14] proposed a model without the input-dependent parametrizer, where the prediction is performed on the similarity between the input and the basis concepts.

Chen et al. [6] proposed ProtoPNet, a slight modification of SENN, where the encoder consists of an embedding layer, trained using a clustering and a separation losses based on the L^2-distance, and the parametrizer does not depend on the input. Since the L^2-distance used by ProtoPNet cannot ensure the disentanglement of the prototypes, Wang et al. [7] proposed TesNet, replacing the L^2-distance with the inner product similarity. In this architecture, the prototypes are trained to produce an orthonormal space, which minimizes and maximizes the similarity of same-class and different-class concepts, respectively.

We contribute to this literature by investigating the application of ProtoPNet and TesNet to the graph domain. We substitute 2D-convolutional layers with graph ones and analyze the modifications needed to adapt such models.

B. Graph Neural Networks

GNNs are based on a message passing paradigm that consists of a neighborhood aggregation scheme, where the latent representation of a node is computed by recursively aggregating and updating the ones of its adjacent nodes [15].

We can define different variants of GNNs by changing the aggregate and update functions.

In this work, we employ the following three popular GNN architectures: graph convolutional networks (GCNs) [16], graph attention networks (GATs) [17], and graph isomorphism networks (GINs) [18]. With GCN, Kipf and Welling [16] proposed a layer that updates the node representations by means of the average latent representations of neighbor nodes, as in

[1]The code is available at https://github.com/KRLGroup/PrototypeGNN

2D-convolution. While GCN assumes equal contribution between the connections of different nodes, GAT [17] changes the convolution scheme by weighting the aggregation with attention scores between couples of nodes. Finally, using a theoretical approach based on the Weisfeiler–Lehman algorithm [19], Xu et al. [18] find that previous message passing neural networks lack in distinguishing different subgraphs based on the generated graph embedding and, to address this problem, they propose GIN, which adjust the weight of the central node with a trainable parameter.

While this article focuses on these three architectures, its findings can be easily translated to other architectures that share the same update and aggregate paradigm.

C. GNN Interpretability

As stated before, the approaches for interpretability can be categorized in post hoc and intrinsic methods [20]. Post hoc ones can be gradient-based, such as CAM, Grad-CAM [21], and DeepLIFT [22], optimization-based, such as GNNExplainer [23] and PGExplainer [24], and decomposition-based methods, such as LRP-based ones [4].

A first drawback of these approaches, especially the ones based on permutations, is the time needed to compute a reasonable explanation. At the same time, Kindermans et al. [25] and Adebayo et al.[26] show that some of these methods are independent of both the training data and the model parameters, a crucial problem when they are used to debug the models. Other works propose an alternative to post hoc methods in the form of neural networks interpretable by design, thus enforcing reasoning processes easier to explain.

To the best of the authors' knowledge, little effort has been done for building interpretable GNNs. The closest work to ours is the one by Zhang et al. [27], who propose ProtGNN. In this case, the prototypes are computed on graph embedding and the classification is based on the similarity between them and the current input graph embedding. ProtGNN+ is an evolution that uses a subsampling layer to identify substructures in the graph. Differently from them, we do not compare the similarity at a graph level but at a node level and focus on learning graph-prototypical parts. An other related approach is the one proposed by Dai and Wang [28]: SEGNN is a node classification network that uses subgraph similarity to find K-nearest nodes for performing the prediction.

While they represent a promising direction, the current research seems to overlook at similar methods proposed for the image counterparts. Indeed, none of the so far published self-explainable architectures for GNNs compares against ProtopNet, TesNet, or similar architectures. Thus, one of the goal of our work is to encourage future research to compare with these kind of models by providing an implementation ready to be used.

Several proxy metrics have been proposed in order to quantitatively evaluate the explanations. As most of XAI for GNNs methods aim to find structural motifs that are relevant for a certain prediction, many metrics focus on input attribution approaches. Among them, we consider the Fidelity+ [20], defined as the average of the difference of accuracy between the prediction of the graphs and the predictions after masking out relevant nodes and edges.

III. Unsupervised Prototype-Based Graph Networks

This section analyzes the different strategies needed to adapt prototype-based image recognition networks to graphs. First, we present the details of ProtoPNet and TesNet, and then we provide the mathematical formulation for their graph application.

A. Prototype-Based Neural Networks

1) ProtoPNet: ProtoPNet's architecture consists of a convolutional neural network f, followed by a prototype layer g_P and a final linear layer l without bias.

Given an input image $\mathbf{x}$, f extracts features organized as a matrix of dimensions $H \times W \times D$

$$\mathbf{z} = f(\mathbf{x}) \in \mathbb{R}^{H \times W \times D}. \tag{1}$$

g_P then computes squared L^2-distances between the m prototypes and all the possible patches $\mathbf{z}_i \in \mathbb{R}^{H_P \times W_P \times D}$ of $\mathbf{z}$ and returns the minimum distance

$$\mathbf{d} = g_P(\mathbf{z}) = \left[\min_{\mathbf{z}_i} L^2(\mathbf{p}_j, \mathbf{z}_i),\ j \in \{1, \ldots, m\} \right] \tag{2}$$

where $\mathbf{p}_j$ is the jth prototype learned by the layer g_P

$$\mathbf{p}_j \in \mathbb{R}^{H_P \times W_P \times D}, \quad H_P \leq H,\ W_P \leq W,\ j \in \{1, \ldots, m\}. \tag{3}$$

Finally, the distances are converted into similarity scores, using a logarithmic activation function, and the last layer l takes them as input and returns the prediction

$$\hat{y} = l\left(\frac{\log(1 + \mathbf{d})}{\mathbf{d}} \right). \tag{4}$$

The prototypes are allocated such that there exist m_k prototypes for each class $k \in \{1, \ldots, K\}$. The subset of allocated prototypes to the class k are defined as $\mathbf{P}_k \in \mathbf{P}$.

Each epoch consists of the following three phases:

1) stochastic gradient descent of layers before the last layer;
2) projection of prototypes;
3) convex optimization of last layer.

First phase: The first phase optimizes the following problem:

$$\min_{w,P} \text{Class} + \lambda_1 \text{Clst} + \lambda_2 \text{Sep} \tag{5}$$

where

$$\text{Class} = \frac{1}{n} \sum_{i=1}^{n} \text{CrsEnt}(\hat{y}_i, y_i) \tag{6}$$

$$\text{Clst} = \frac{1}{n} \sum_{i=1}^{n} \min_{j:\mathbf{p}_j \in P_{y_i}} \min_{\mathbf{z} \in \text{patches}(f(x_i))} ||\mathbf{z} - \mathbf{p}_j||_2^2 \tag{7}$$

$$\text{Sep} = -\frac{1}{n} \sum_{i=1}^{n} \min_{j:\mathbf{p}_j \notin P_{y_i}} \min_{\mathbf{z} \in \text{patches}(f(x_i))} ||\mathbf{z} - \mathbf{p}_j||_2^2. \tag{8}$$

w are the weights of the convolutional layers f and λ_1 and λ_2 are two penalty terms set to 0.8 and 0.08, as proposed by Chen et al.

[6]. CrsEnt is the cross entropy loss for the classification task, and Clst and Sep are two cost functions, which encourage and penalize same-class and different-class prototypes similarity, respectively.

Second phase: The second phase consists of replacing the jth prototype of the class k $\mathbf{p}_j \in \mathbf{P}_k$ with the closest point $\text{argmin}_{\mathbf{z}_i} L^2(\mathbf{p}_j, \mathbf{z}_i)$ over all the patches z_i of the features extracted from images of the class k.

Third phase: In the last phase, the weights w_l of the last layer are optimized starting from the following initialization:

$$w_l^{(k,j)} = \begin{cases} +1.0 & \forall j \;\; s.t. \;\; \mathbf{p}_j \in \mathbf{P}_k \\ -0.5 & \forall j \;\; s.t. \;\; \mathbf{p}_j \notin \mathbf{P}_k \end{cases} \tag{9}$$

where the (k, j)th entry of w_l is the weight between the output g_{P_j} and the logit of the class k. This setting induces the model to learn the weights such that the similarity to a class k prototype should increase the predicted probability that the image belongs to class k, whereas the similarity to a nonclass k prototype should decrease class ks predicted probability.

2) TesNet: TesNet shares the same learning paradigm as ProtoPNet, organized in three phases, but it uses an alternative formulation for the optimization problem:

$$\min_{P,w} \text{Class} + \lambda_1 \text{Clst} + \lambda_2 \text{Sep} + \lambda_3 \text{Orth} + \lambda_4 \text{ClassSep} \tag{10}$$

where

$$\text{Clst} = \frac{1}{n} \sum_{i=1}^{n} \min_{j:\mathbf{p}_j \in P_{y_i}} \min_{\mathbf{z} \in \text{patches}(f(x_i))} - \frac{\mathbf{z}^T \mathbf{p}_j}{||\mathbf{z}||} \tag{11}$$

$$\text{Sep} = \frac{1}{n} \sum_{i=1}^{n} \min_{j:\mathbf{p}_j \notin P_{y_i}} \min_{\mathbf{z} \in \text{patches}(f(x_i))} \frac{\mathbf{z}^T \mathbf{p}_j}{||\mathbf{z}||} \tag{12}$$

$$\text{Orth} = \sum_{k=1}^{K} ||P_k P_k^T - \mathbb{I}||_F^2 \tag{13}$$

$$\text{ClassSep} = -\frac{1}{\sqrt{2}} \sum_{k_1=1}^{K-1} \sum_{k_2=k_1+1}^{K} ||P_{k_1}^T P_{k_1} - P_{k_2}^T P_{k_2}||_F. \tag{14}$$

In this case, ClassSep encourages the similarity between the basis concepts of different classes and Orth enforces the orthonormality among the basis concepts of a class.

1) Prototype-Based Explanations: ProtoPNet and TesNet allow producing both global and local explanations. The former can be obtained by inspecting the images represented by prototypes, while the latter is obtained by looking at the similarities of a certain input to the various prototypes. Additionally, we can visualize the image patches that activate the most a prototype by plotting its activations (similarities) on the different pixels in a heatmap. In particular, given an image $\mathbf{x}$, we can extract the activations of the (i, j)th patch of $\mathbf{x}$ with respect to the kth prototype as

$$\text{act}(\mathbf{x})_{(i,j)} = s(f(\mathbf{x})_{(i,j)}, \mathbf{p}_k) \tag{15}$$

where $s : (\mathbb{R}^D, \mathbb{R}^D) \to [0, 1]$ is a similarity function between the embedding and the prototypes, such as the logarithmic activation for ProtoPNet and the cosine similarity for TesNet.

B. Graph Neural Networks

GNNs are a particular kind of neural networks designed to work with graph domains. Most of them use a message passing schema formed by two basic operations: aggregation and update. The generic kth layer of a graph network can be defined in the following way:

$$a_v^{(k)} = \text{AGGREGATE}^{(k)} \left(\left\{ h_u^{(k-1)} : u \in \mathcal{N}(v) \right\} \right) \tag{16}$$

$$h_v^{(k)} = \text{UPDATE}^{(k)} \left(h_v^{(k-1)}, a_v^{(k)} \right) \tag{17}$$

where v is the current node, $h_v^{(k)}$ is the feature vector of v, and $\mathcal{N}(v)$ is the set of the neighbor nodes of v.

GCNs [16] update the representation by aggregating an average contribution from the neighboring nodes, GATs [17], instead, weight the aggregation mean by learning self-attention scores between couples of node. Finally, GINs [18] base aggregation and update on the Weisfeiler–Lehman test. The three aggregate functions are summarized as follows:

$$h_{v,\,\text{GCN}}^{(k)} = \text{W}^{(k)} \cdot \text{MEAN} \left(\left\{ h_u^{(k-1)} : u \in \mathcal{N}(v) \cup \{v\} \right\} \right) \tag{18}$$

$$h_{v,\,\text{GAT}}^{(k)} = \text{SUM} \left(\left\{ \alpha_{vu} \text{W}^{(k)} \cdot h_u^{(k-1)} : u \in \mathcal{N}(v) \right\} \right) \tag{19}$$

$$h_{v,\,\text{GIN}}^{(k)} = \text{MLP}^{(k)} \left(\left(1 + \epsilon^{(k)} \right) h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right) \tag{20}$$

where $\text{W}^{(k)}$ are the learnable weights of the convolutional layers. The structure of a GNN is composed of a sequence of convolutional layers that generate a latent representation of the nodes. For node classification tasks, a final feed-forward layer takes this representation and produces the logits. For graph classification, a graph representation is obtained by means of a readout layer, such as a global pooling layer, followed by a final classification layer that makes the prediction [15].

C. Adapting Prototype Networks to the Graph Domain

Recall that a graph $G = (N, E)$ is a tuple of nodes and edges that connect them. We can represent graphs by using two matrices: the node features matrix $X \in \mathbb{R}^{|N| \times F}$, where F is the number of node features, and the adjacency matrix $A \in \{0, 1\}^{N \times N}$. Therefore, a prototype-based GNN is composed of the following three functions.

1) f, is a GNN feature extractor formed of graph convolutional layers that return node embeddings:
$$\mathbf{z} = f(X, A). \tag{21}$$
2) g_P, is a prototype layer that projects the node embeddings and calculates the similarities between them and the

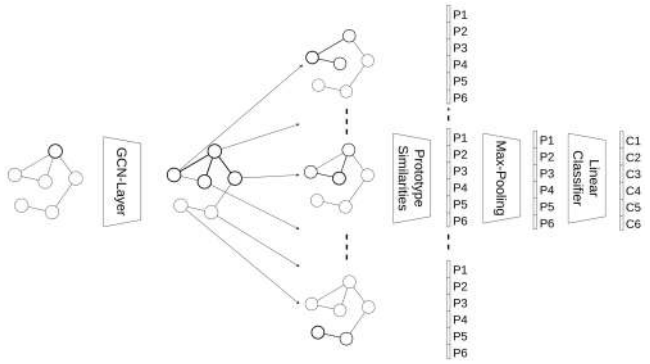

Fig. 1. Prototype-based GNN architecture for graph classification using one GCN-layer.

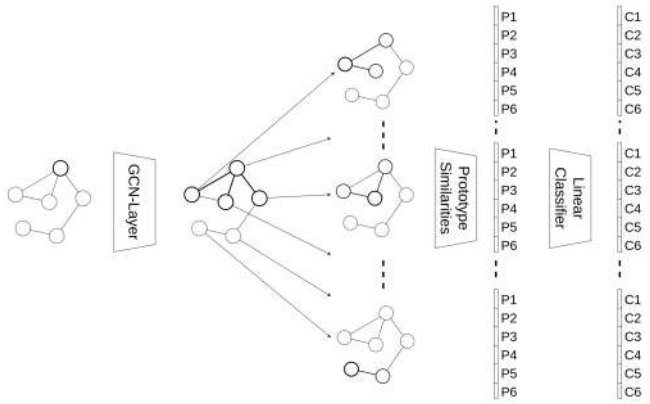

Fig. 2. Prototype-based GNN architecture for node classification using one GCN-layer.

learned prototypes

$$\mathbf{d} = g_P(\mathbf{z}). \tag{22}$$

3) l, is a classification layer that takes the prototype similarities and outputs the probabilities of a graph, or a node, to belong to a certain class

$$\hat{y} = l(\mathbf{d}). \tag{23}$$

Additionally, in the case of graph classification, as g_P returns the similarities between prototypes and projected node embeddings,

we need to add a global max-pooling readout layer after g_P, which selects the highest similarities between the nodes and each prototypes. The resulting similarity of the graph to the jth prototype $\mathbf{d}_j$ is calculated as follows:

$$\mathbf{d}_j = \max_{n \in N} \mathbf{d}_{nj}. \tag{24}$$

In this way, l classifies using prototype similarities at graph level.

Summarizing, since node embeddings encode information of a subgraph as big as the number of GNN layers, the prototype layer compares the presence of particular patterns inside the graph for predicting the graph or node properties. Figs. 1 and 2 show a schema of the described architecture for performing graph and node classification, respectively.

Note that while in the original TesNet and ProtoPNet the prototypes identify patches of the images, here the prototypes are obtained from the node embeddings, and thus, their meaning is different. In fact, since the node representation over k convolutions contain information of the k-hop subgraphs around the nodes, we are actually embedding the information of subgraphs of radius k inside the prototypes.

1) Prototype-Based Explanations: Inspecting prototypes is a key procedure for understanding the model's reasoning process. Chen et al. [6] and Wang et al. [7] provide prototype visualization by means of a heat-map that shows the prototype activations over different image patches. This procedure allows them to identify the areas in the image that mostly activate a certain prototype. Similarly, we can identify the most important subgraphs learned by the model.

While images usually have thousands of pixels, in most datasets graphs might be composed of less than a hundred nodes. This difference would result in a less sparse visualization of the prototype activations. Instead of showing the prototype activations over all the nodes, we propose to highlight only the *k-hop* subgraph around the node that is mostly similar to the specific prototype, where k is the number of GNN layers. This is justified by the fact that by using k GNN layers, each node embedding contains information about the node and the neighbor nodes within a radius of k edges.

For node classification, instead, as the datasets are composed of only one graph, where nodes are masked throughout training and validation procedures, we propose to extract the *(k+2)-hop* subgraphs around the most similar node and only highlight the *k-hop* subgraphs with the prototype activations. This allows us to extract a small subgraph from the original one, without the need of plotting all the nodes.

The proposed technique allows dropping unnecessary attributions derived from prototype activations that are not actually used for the prediction, by only selecting them that are within a k-hop subgraph from the activated node.

IV. Experiments

In this section, we analyze the classification and explanation performances of ProtoPNet and TesNet applied to the graph domain: first, we present the experimental setting; subsequently we compare the classification performances on the presented datasets with black-box and self-explainable models, including an analysis of the impact of the number of prototypes and transfer learning; finally, we study the explanations of the trained models providing visual and quantitative results.

A. Experimental Setup

We train and evaluate the models on the following graph and node classification datasets:

1) MUTAG [29], BBBP [30], and BACE [31], which are molecule datasets for graph classification, where atoms and bonds are encoded as nodes and edges, respectively, and the labels classify molecules as either active or inactive against some targets.

TABLE I
CLASSIFICATION PERFORMANCES

Model	BBBP	MUTAG	BACE	BA-2Motifs	BA-Community	BA-Shapes	Tree-Grid
GCN	85.75 ± 0.61	77.00 ± 6.78	76.32 ± 1.74	87.20 ± 4.19	97.34 ± 0.14	91.23 ± 1.08	84.79 ± 1.76
– TesNet	**87.61 ± 1.18**	80.50 ± 5.68	79.28 ± 1.67	59.00 ± 4.12	97.31 ± 0.29	**96.30 ± 0.42**	97.17 ± 0.59
– ProtoPNet	85.41 ± 1.47	79.50 ± 5.68	78.55 ± 2.44	**99.40 ± 0.80**	95.74 ± 0.55	85.10 ± 0.62	76.93 ± 2.59
GAT	**87.32 ± 1.20**	77.50 ± 8.14	78.22 ± 2.66	51.20 ± 2.32	86.61 ± 1.76	76.54 ± 1.18	58.49 ± 0.00
– TesNet	**87.02 ± 0.73**	79.00 ± 6.24	77.57 ± 2.44	50.10 ± 3.08	87.94 ± 1.66	42.86 ± 0.00	58.49 ± 0.00
– ProtoPNet	85.71 ± 1.05	72.00 ± 7.14	78.16 ± 1.83	83.30 ± 5.92	83.85 ± 1.08	59.83 ± 1.47	84.80 ± 3.06
GIN	85.61 ± 1.56	82.50 ± 7.50	75.40 ± 1.45	**99.50 ± 0.67**	92.34 ± 1.95	89.79 ± 3.89	87.03 ± 1.93
– TesNet	**87.17 ± 1.05**	**85.50 ± 3.50**	79.34 ± 2.68	**99.80 ± 0.40**	**98.36 ± 0.34**	**96.96 ± 0.14**	**98.67 ± 0.28**
– ProtoPNet	85.66 ± 1.74	83.50 ± 5.50	**80.86 ± 3.12**	98.90 ± 1.14	96.65 ± 0.74	85.11 ± 1.41	82.42 ± 6.82
Graph type	Molecule	Molecule	Molecule	Synthetic	Synthetic	Synthetic	Synthetic

For each architecture, three versions are presented: the base, the TesNet and the ProtoPNet model.
The values between the standard deviation limits of the best model are highlighted in bold.

2) BA-shapes [23] and BA-Community [23], which are two synthetic node classification datasets that contain graphs obtained by the *Barabási-Albert* model with the house-like five node motifs.
3) BA-2Motifs [24], which is a synthetic graph classification datasets where graphs are discriminated by the presence of a house-like motif between their nodes.
4) Tree-Grid [23], which is a synthetic node classification datasets where nodes are labeled by their presence in a grid motif inside some tree-shaped graphs.
5) Cora, Citeseer, and Pubmed [32] are three widely used real-world benchmark datasets. They consist of graphs where nodes represent scientific publications labeled in several classes while links represent the citations. While for Cora and Pubmed we use the publicly available splits, for Citeseer we refer to the version from [28] and [33] for comparison purposes.

For each dataset, we compare three black-box models composed of 3 GCN, GAT, and GIN layers, respectively, against their ProtoPNet and TesNet variants, obtained by adding the prototype layers. Following the dive-into-graphs benchmark repository [23], for the molecular graph datasets, the GNN layers are formed of 128 units, and each GNN layer's output is activated by an ReLU unit. We set the number of prototypes per class to 10, as done by [6] and [7]. We repeat each experiment 15 times to compare different seeds for statistical significance.

B. Classification Performances

In this section, we test whether adding the prototype layers leads to a decrease of performances. Wang et al. [7] and Chen et al. [6] show that, for images, this is not necessarily true, and self-interpretable models sometimes manage to perform better than the black-box ones.

1) Small Datasets: In Table I, we report the accuracy scores for the graph and node classification tasks using GCN, GAT, and GIN networks as backbone. We observe that, in general, explainable models reach higher accuracy than the black-box counterparts. Furthermore, in most cases, the TesNet models reach the highest score, thus confirming the findings in the image domain. Our hypothesis is that by using basis concepts in place of the prototypes, we encourage learning a disentangled prototype space that is crucial for the graph domain too.

TABLE II
CLASSIFICATION PERFORMANCES ON BIG DATASETS

	Citeseer	Pubmed	Cora
Protop-GCN	70.12 ± 0.90	76.38 ± 1.07	78.62 ± 1.57
Tes-GCN	70.72 ± 1.01	**76.77 ± 0.86**	78.65 ± 0.80
SEGNN	**74.19 ± 0.51**	**76.92 ± 0.30**	**79.66 ± 0.56**
ProtGNN	53.07 ± 1.98	75.93 ± 1.14	59.73 ± 1.95
GCN	73.60 ± 0.35	74.52 ± 0.54	79.09 ± 0.81

The values between the standard deviation limits of the best model are highlighted in bold.

More in detail, we see that TesNet based GIN models achieve the highest performances for almost all the tasks. While for the molecules graph classification the difference between TesNet models and the baseline is subtle, for the synthetic datasets, and in particular for node classification, it gets sharper. We relate these findings to the observations by Xu et al. [18], where they show that GNNs suffer from not being able to distinguish between some graph structures and propose GIN as a remedy.

These results demonstrate that prototype-based models, such as TesNet and ProtoPNet can be adapted to the field of graphs, and they can also be employed for training self-interpretable models for the node classification task.

2) Big Datasets: Now, we compare the performances with other state-of-the-art models on bigger datasets. In particular, we use three big node classification datasets and train and compare GCN-TesNet and GCN-ProtoPNet models with two state-of-the-art self-explainable models: SEGNN [28] and ProtGNN[27]. We also show the accuracy score for the black-box GCN model for comparison purposes (see Table II).

We observe comparable performances on Cora and Pubmed for both ProtoPNet and TesNet with respect to the top performer (SEGNN). On Citeseer, the performances are lower than SEGNN but higher than ProtGNN. The lower performance is probably due to the difficulty of generalizing the publications with a limited set of prototypes. Moreover, this gap is mitigated by the higher explanation performances of TesNet and ProtoPNet (see Section IV-C).

3) Impact of the Number of Prototypes: Here, we investigate the impact of the number of prototypes per class on the accuracy of the transparent models. We test three different values (5,

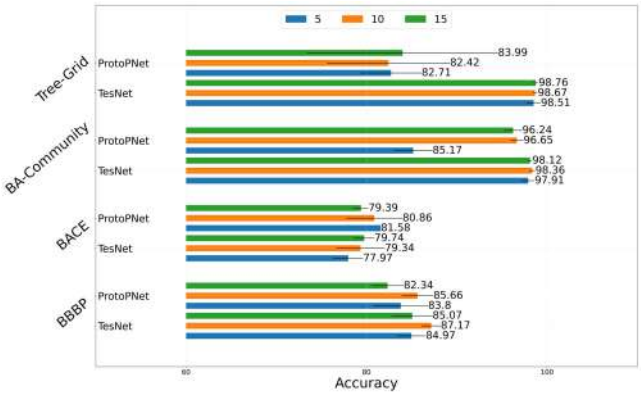

Fig. 3. Accuracy scores of the explainable GIN models varying the number of prototypes on the BBBP, BACE, BA-Community, and Tree-Grid datasets.

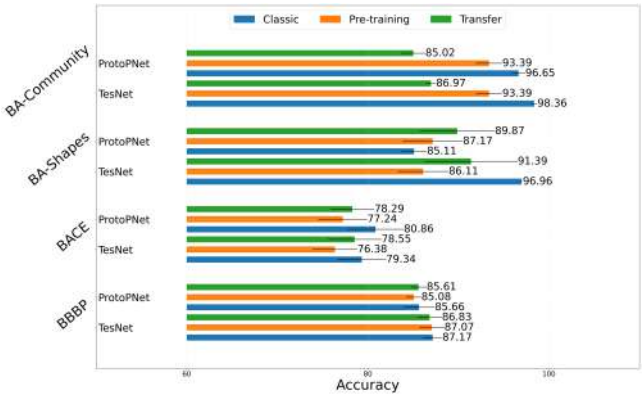

Fig. 4. Accuracy scores of the transfer learning models using the GIN architecture. BBBP and BACE share the same node features as well as BA-Shapes and BA-Community.

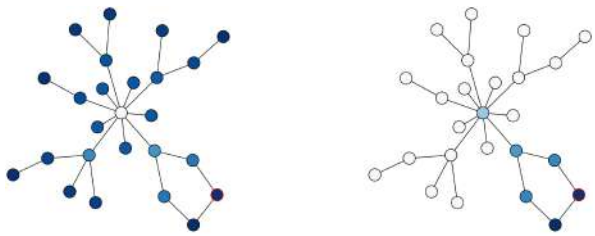

Fig. 5. Comparison between classic (a) and our (b) visualization of ProtoPNet explanations. The prototype activations are highlighted with a shade of colors from light blue, low activation, to dark blue, high activation. The node used for the prediction is marked with a red border.

10, and 15) on four datasets, using GIN as backbone model. Although in some cases using a varying number of prototypes determines an increase or decrease in the model's performances, we do not find a general pattern over the datasets (see Fig. 3). The reason is that, for most datasets, the discrimination between the classes is represented by few patterns and the models, even if they have different prototypes, end up learning the same subgraphs. On the other hand, the number of prototypes is crucial when we know a priori that there exists a specific number of patterns to look at.

Therefore, the number of prototypes is a hyperparameter to be tuned, possibly using the explanations returned by the model (see Section IV-C).

4) Impact of Transfer Learning: In this section, we investigate two ways for reproducing the transfer learning procedure in our setting. In fact, the models analyzed in the previous section are trained from scratch. This differs from the image domain, where the networks are obtained by means of transfer learning of models trained on large corpora. However, for graphs there is not any available library of pretrained GNNs. Moreover, while when dealing with images it is possible to preprocess the input to certain shapes and channels, on the other hand, with GNNs, node, and edge features might vary between different tasks and domains.

Therefore, we test pretraining on a dataset and then use the learned weights of the convolutional layers to initialize the explainable networks for a new training phase. In this way, we end up having a task-specific feature extraction network to which we add a prototype layer. In particular, we compare the performance when the network is pretrained both on the same dataset and on another dataset that share the same features. For instance, the datasets BBBP and BACE represent the molecules with the same features, hence, we can use the first for pretraining and the second for the standard training procedure.

Using pretrained models on the same dataset, both TesNet and ProtoPNet almost reproduce the same performance of the earlier results, although in general the values are slightly lower (see Fig. 4). Overall, transfer learning tests do not provide any advantage and the performances decrease when the pretrained network is built on another dataset. This issue represent the main dissimilarity between the application of ProtoPNet and TesNet to images and graphs. For the molecule dataset, we hypothesize that the datasets are particularly different, and the molecules belong to separate chemical spaces. This problem is noted as domain of applicability [34] in chemistry. Similarly, on the other datasets, the substructures in the graphs are dissimilar from each other, and, for this reason, the features learned on BA-Shapes might not be adequate for BA-Community.

C. Explanations

In this section, we analyze the explanations provided by ProtoPNet and TesNet on the graph domain. We test different ways for computing them, we compare their quality against some post hoc methods, and finally we provide some visual examples to show of how users can exploit this type of explanations to extract insights about the model behavior.

We start the analysis by comparing the explanations obtained with two different methods (see Section III-C1): the original method from ProtoPNet and TesNet, plotting the activation scores over the whole input; and our modification, where we restrict the explanation only to the nodes involved in the actual prediction. To compare them, we use the Fidelity+ score, defined as

$$\text{Fidelity+} = \frac{1}{N}\sum_{i=1}^{N}(f(G_i)_{y_i} - f(G_i^{1-m_i})_{y_i}). \quad (25)$$

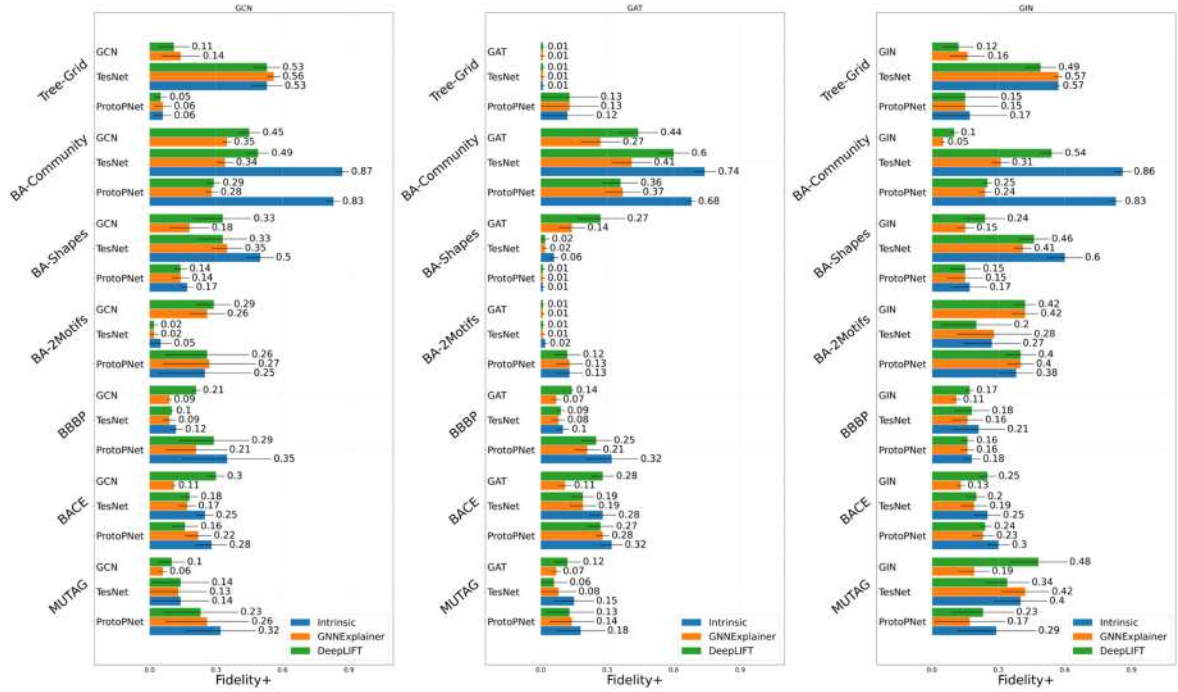

Fig. 6. Fidelity+ score comparison between explanations computed using post hoc (orange and green) and intrinsic methods (blue).

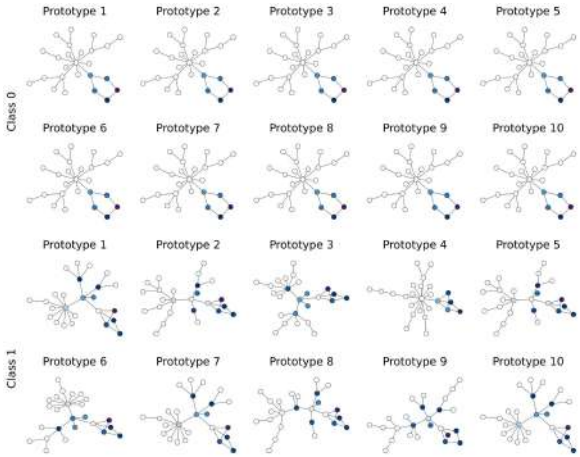

Fig. 7. Subgraphs learned by the ProtoPNet GCN model on the Ba-2Motifs dataset. The most activated node is marked with a red borderline and the subgraph nodes are highlighted in blue: the darker the node the higher the activation.

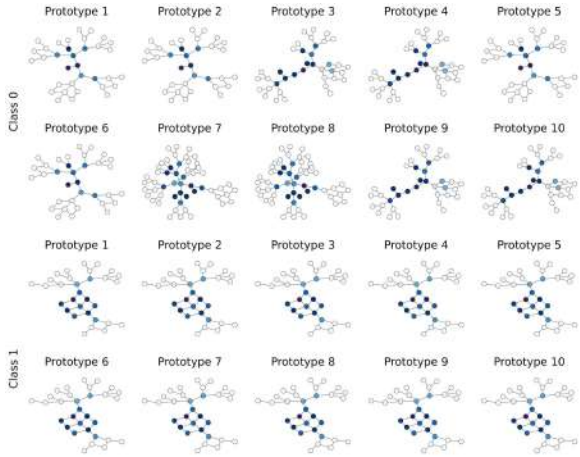

Fig. 8. Subgraphs learned by the TesNet GIN model on the tree-grid dataset. The most activated node is marked with a red borderline and the subgraph nodes are highlighted in blue: the darker the node the higher the activation.

TABLE III
COMPARISON BETWEEN VANILLA AND K-HOP METHOD FOR EXTRACTING EXPLANATIONS FROM PROTOPNET AND TESNET

	MUTAG	BACE	BBBP	Ba-2Motifs
k-hop	0.35 ± 0.10	0.25 ± 0.06	0.18 ± 0.07	0.16 ± 0.15
Vanilla	0.36 ± 0.13	0.24 ± 0.06	0.18 ± 0.05	0.16 ± 0.15

We report the Fidelity+ scores for the best model, i.e., TesNet GIN, over the classification datasets.

Although in terms of Fidelity+, the explanation produce the same results (see Table III), the visualization of our method appears clearer (see Fig. 5) and gives insights on the activations of a certain prototype: the model is recognizing the house-like motif with the chosen prototype. Conversely, the classic method does

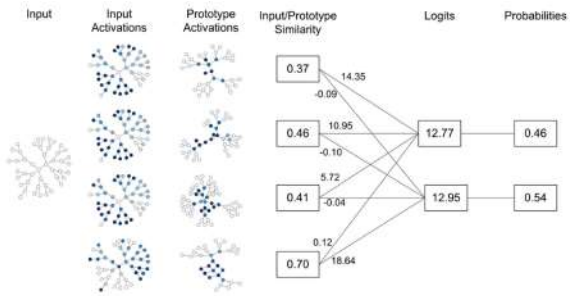

Fig. 9. Visualization of local explanations. We indicate with a red border the node we want to predict in the Input and Input Activation columns, and the node associated to the prototype in the prototype activation column.

TABLE IV
EXPLANATIONS PERFORMANCES ON BIG DATASETS

	Fidelity+ (↑)			ROC-AUC (↓)		
	Citeseer	**Pubmed**	**Cora**	**Citeseer**	**Pubmed**	**Cora**
ProtoPNet	**0.2943 ± 0.0161**	0.1988 ± 0.0118	0.3304 ± 0.0625	0.8641 ± 0.0074	0.9034 ± 0.0069	0.9396 ± 0.0039
TesNet	0.2869 ± 0.0322	**0.2409 ± 0.0163**	**0.4979 ± 0.0649**	0.8084 ± 0.0134	0.8708 ± 0.0077	0.8811 ± 0.0243
SEGNN	0.0276 ± 0.0032	0.0249 ± 0.0112	0.1027 ± 0.0092	0.8614 ± 0.0166	0.8981 ± 0.0261	0.9604 ± 0.0018
ProtGNN	0.0243 ± 0.0251	0.0588 ± 0.0196	0.0702 ± 0.0715	0.8139 ± 0.0275	0.8901 ± 0.0166	**0.8270 ± 0.0119**
GNNExplainer	0.1528 ± 0.0255	0.0636 ± 0.0059	0.1259 ± 0.0083	0.8122 ± 0.0041	0.8536 ± 0.0072	0.9148 ± 0.0053
DeepLIFT	0.2756 ± 0.0196	0.2152 ± 0.0030	0.3267 ± 0.0047	**0.7860 ± 0.0068**	**0.7209 ± 0.0108**	0.8476 ± 0.0068

Fidelity+ and ROC-AUC are reported for the self-interpretable models and the post- hoc methods on the black-box GCN model.
The values between the standard deviation limits of the best model are highlighted in bold.

not highlight the specific subgraph that is used in the reasoning process. These results show that neglecting the activations of the other nodes does not reduce the performance of the explanation and, at the same time, provides a more "human-readable" interpretation.

Once the method is chosen, we study the goodness of explanations provided by such models, by comparing them against explanations computed on both the interpretable and the black-box models using GNNExplainer and DeepLift (see Fig. 6).

White-box models' explanations report relatively higher scores than the post hoc ones. For the graph classification task, the explanations of ProtoPNet are more faithful than the ones of TesNet, while for the node classification this is inverted. In some cases, the black-box models reach higher Fidelity+ than the white box ones, but with a small difference. This is somehow compensated by the smaller amount of time required by ProtoPNet and TesNet for producing the explanations: while DeepLift, using the gradients, takes similar time to ProtoPNet and TesNet for explaining the prediction, GNNExplainer takes up to 20 times longer for optimizing the graph mask.

Regarding the self-explainable models, we observe higher fidelity scores on the explanations extracted by ProtoPNet and TesNet (see Table IV). Indeed, ProtoPNet reports the highest fidelity score on the Citeseer dataset, while TesNet is the leading one explainer on Pubmed and Cora. We justify these findings by the fact that masking out relevant nodes the has a high impact on the prototype activations and affects the prediction. On the other hand, SEGNN, not having a fixed number of prototypes, might relate on the similarity to other labeled nodes.

We also report the ROC-AUC scores of the predictions of masked inputs, it is expected to have lower ROC-AUC scores when relevant edges are masked out. While DeepLIFT obtains the lowest values of ROC-AUC score, TesNet generally reports lower values compared to SEGNN and ProtGNN, which, in accordance with the Fidelity+ score, confirms the better explainability of the adapted models.

Finally, to inspect the model behavior, we can visually study which are the most activated prototypes for the elements of the dataset and plot the subgraphs that correspond to the learned prototypes, as shown in Section III-C1.

Fig. 7 shows the learned prototypes for a ProtoPNet GCN model trained on Ba-2Motifs. For the class 0, all the prototypes match the house motif. Instead, the prototypes of class 1 match a particular 3-node loop motif, captured in different positions on the graph. We further investigate this result by checking the activations of the class 1 graphs with respect to these prototypes, and we find that over 86% of the samples share an activation higher than 0.95, likely due to an artifact resulting from the automated generation of the dataset. Fig. 8 shows another example, where we identify the most important substructures of the best TesNet GIN model trained on the tree-grid dataset. In particular, we use the explanations to understand why the model was not able to correctly score mislabeled nodes. Similarly to the example of Ba-2Motifs, we can obtain global interpretation of the model behavior by analyzing the subgraphs that the model learns as the most important. The first two rows highlight the subgraphs matched for predicting the nodes that do not belong to a grid shape subgraph. Conversely, the last two rows clearly grasp the grid shape for predicting the nodes.

Additionally, Fig. 9 shows the local explanations for a mispredicted node. It reports the activations of the nodes for each prototype together with the similarities between the analyzed node (red border) and the prototypes. The reason for the misprediction is that the node shares a similarity of 0.7 with class 1 prototype due to the high degree of the node, which makes it more similar to a "grid node" rather than a "tree node."

These analyzes show that we can understand the reasoning process that led to a specific prediction, both at local and global level, by visualizing the activations of the nodes to the different prototypes.

V. CONCLUSION

This article analyzed the application of ProtoPNet and TesNet for performing graph and node classification. We found that, in contrast to the image domain, the best models are obtained by starting from randomized weights. We confirmed that these models reach equal or higher performances of black box ones. Additionally, we studied the interpretability power of the architectures, both in terms of global and local explanations, and we found that they produce more faithful explanations than post hoc methods and competitors. In particular, we showed that while maintaining comparable classification performances with state-of-the-art self-explainable networks, our models outperform the others in terms of explanation capability.

We think that the findings of this work could open the path for further development of interpretable graph architectures. Future works might investigate the use of metrics better tailored to the graph domain, or methods to automatically modulate the number of prototypes learned by the model. Additionally, we release the open source code to encourage further research.

REFERENCES

[1] W. Fan et al., "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426, doi: 10.1145/3308558.3313488.

[2] D. Jiang et al., "Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models," *J. Cheminformatics*, vol. 13, no. 1, pp. 1–23, Feb. 2021, doi: 10.1186/s13321-020-00479-8.
[3] P. Bongini, M. Bianchini, and F. Scarselli, "Molecular generative graph neural networks for drug discovery," *Neurocomputing*, vol. 450, pp. 242–252, Aug. 2021, doi: 10.1016/j.neucom.2021.04.039.
[4] R. Schwarzenberg, M. Hübner, D. Harbecke, C. Alt, and L. Hennig, "Layerwise relevance visualization in convolutional text graph classifiers," in *Proc. 13th Workshop Graph-Based Methods Natural Lang. Process.*, 2019, pp. 58–62. [Online]. Available: https://aclanthology.org/D19-5308
[5] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, pp. 68–77, Dec. 2019, doi: 10.1145/3359786.
[6] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: Deep learning for interpretable image recognition," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf
[7] J. Wang, H. Liu, X. Wang, and L. Jing, "Interpretable image recognition by constructing transparent embedding space," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 895–904.
[8] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon et al., Eds. Red Hook, NY, USA: Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf
[9] B. Kim et al., "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, vol. 80, pp. 2668–2677. [Online]. Available: https://proceedings.mlr.press/v80/kim18 d.html
[10] Z. Chen, Y. Bei, and C. Rudin, "Concept whitening for interpretable image recognition," *Nature Mach. Intell.*, vol. 2, no. 12, pp. 772–782, Dec. 2020, doi: 10.1038/s42256-020-00265-z.
[11] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 3530–3537.
[12] J. Ni et al., "Interpreting convolutional sequence model by learning local prototypes with adaptation regularization," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 1366–1375, doi: 10.1145/3459637.3482355.
[13] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf
[14] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Proc. 32nd AAAI Conf. Artif. Intell., 30th Innov. Appl. Artif. Intell., 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 3530–3537. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17082
[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/tnnls.2020.2978386.
[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ
[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km
[19] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Ser.*, vol. 2, no. 9, pp. 12–16, 1968.
[20] M. Liu et al., "DIG: A turnkey library for diving into graph deep learning research," *J. Mach. Learn. Res.*, vol. 22, no. 240, pp. 1–9, 2021. [Online]. Available: http://jmlr.org/papers/v22/21-0343.html
[21] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10764–10773.
[22] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3145–3153.
[23] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf
[24] D. Luo et al., "Parameterized explainer for graph neural network," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 19620–19631, 2020.
[25] P.-J. Kindermans et al., *The (Un)reliability of Saliency Methods*. Cham, Switzerland: Springer, 2019, pp. 267–280, doi: 10.1007/978-3-030-28954-6_14.
[26] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf
[27] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, "ProtGNN: Towards self-explaining graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 8, pp. 9127–9135, doi: 10.1609/aaai.v36i8.20898.
[28] E. Dai and S. Wang, "Towards self-explainable graph neural network," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 302–311, doi: 10.1145/3459637.3482306.
[29] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *J. Med. Chem.*, vol. 34, no. 2, pp. 786–797, Feb. 1991, doi: 10.1021/jm00106a046.
[30] I. F. Martins, A. L. Teixeira, L. Pinheiro, and A. O. Falcao, "A Bayesian approach to in silico blood-brain barrier penetration modeling," *J. Chem. Inf. Model.*, vol. 52, no. 6, pp. 1686–1697, Jun. 2012, doi: 10.1021/%2Fci300124c.
[31] G. Subramanian, B. Ramsundar, V. Pande, and R. A. Denny, "Computational modeling of β-secretase 1 (BACE-1) inhibitors using ligand based approaches," *J. Chem. Inf. Model.*, vol. 56, no. 10, pp. 1936–1949, Oct. 2016, doi: 10.1021/%2Facs.jcim.6b00290.
[32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: 10.1007/s11263-019-01228-7.
[33] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 66–74, doi: 10.1145/3394486.3403049.
[34] S. Weaver and M. P. Gleeson, "The importance of the domain of applicability in QSAR modeling," *J. Mol. Graph. Model.*, vol. 26, no. 8, pp. 1315–1326, Jun. 2008, doi: 10.1016/j.jmgm.2008.01.002.