Relating explanations with the inductive biases of Deep Graph Networks

Michele Fontanesi[1 [0009−0004−7566−903X]], Alessio Micheli[1 [0000−0001−5764−5238]], and Marco Podda[1 [0000−0003−1497−9515]]

University of Pisa, Department of Computer Science
Largo B. Pontecorvo 3, 56127 Pisa, Italy

Abstract. Deep Graph Networks (DGNs), i.e. neural networks able to process graphs directly, feature an iterative message passing (MP) step that implements the node embeddings computation. However, the inductive and architectural biases of different DGNs in relation to the type and number of MP iterations are yet to be unveiled. Here, we investigate this important topic using eXplainable Artificial Intelligence (XAI) techniques for graphs. Specifically, we use the XAI metric of plausibility to detect explanatory patterns and to relate this information to the biases exploited by the underlying DGN to correctly learn graph classification tasks. We use this method to gather evidence on the rich diversity of DGN biases in relation to the type and number of iterations of MP when applied to XAI benchmarks. In addition, we show that when the MP conditions are fixed, the learned explanatory pattern may change based on the norm of the learned weights, signifying that the training procedure, in particular cases, influences the generalization dynamics.

Keywords: Deep Graph Networks · Explainable Artificial Intelligence · Inductive Bias

1 Introduction

The increasing use of graph structures to model complex relational phenomena coupled with the latest peak of interest in Machine Learning have made neural networks able to handle graph data directly, a.k.a. Deep Graph Networks (DGNs) [2], the *de facto* technology to tackle classification and regression tasks on graphs. Since their introduction [11, 17], different types of DGNs have been developed, most commonly based on the message passing (MP) paradigm [6]. Loosely speaking, MP is a generic three-step iterative procedure whose objective is to update node embeddings based on the graph topology, which each DGN implements internally in different flavors.

While DGNs have been repeatedly shown to generalize to unseen graphs, the inductive biases [13], i.e., the set of assumptions implicitly made by the model that make this generalization possible, are still not completely characterized, since they are encoded into a large set of learned parameters. Nevertheless, knowing which inductive bias DGNs exploit is paramount to understanding and

interpreting their decisions, enhancing their trustworthiness [15] and as a consequence, their widespread adoption in safety-critical applications.

In this work, we contribute to this topic by applying eXplainable Artificial Intelligence (XAI) techniques for graphs [1, 7, 8, 20] to analyze the behavior of DGNs in synthetic graph classification tasks, deriving insights about the rich landscape of the inductive biases that they implicitly exploit. By using the Class Activation Mapping (CAM) [16,21] explainer, we first notice that different DGNs (characterized by different MP variants) achieve almost perfect generalization despite showing two different explanatory patterns (i.e., importance scores) at the node level. This suggests that there are different learning "routes", all valid, that a DGN can take to achieve the ultimate goal of generalization. We then use the XAI metric of plausibility [9] to detect which explanatory pattern the DGN has picked up, and show how it relates to the specific MP implementation. More precisely, we show experimentally that the explanatory patterns align with known properties of certain MP variants, and we show that by varying the MP configuration the model shifts from one explanatory pattern to the other.

Our results shed light on how the different inductive biases that DGNs use are leveraged to learn graph classification tasks. Moreover, they suggest that all kinds of inductive biases are equally important and play a prominent role in the end goal of generalization.

2 Background

In this section, we provide the notions to understand the proposed analysis.

2.1 Preliminary notions on graphs

A graph $G = (V_G, E_G)$ [3] is defined as a tuple consisting of a set of vertices $V_G = \{v_1, ..., v_n\}$ and a set of edges $E_G \subseteq (V_G \times V_G) = \{(u, v) \mid u, v \in V\}$. The graph connectivity is usually formalized as an *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $\mathbf{A}_{u,v} = \mathbb{1}[(u, v) \in E_G]$. The neighborhood of a node v is the set $\mathcal{N}_v = \{u \in V \mid (u, v) \in E_G\}$ of nodes connected to v by an edge. Graph nodes are associated with feature vectors $\mathbf{x}_v \in \mathbb{R}^d$ for some $d \in \mathbb{N}$, and we use the notation $\mathbf{X} \in \mathbb{R}^{n \times d}$ to indicate the matrix of node features stacked row-wise. Therefore, for the purposes of this work, we encode a graph with the tuple $(\mathbf{A}, \mathbf{X})$. Two graphs G and G' are said to be (structurally) isomorphic if there exists a bijection $\phi : V_G \to V_{G'}$ such that $\forall u, v \in V_G$, $(u, v) \in E_G \iff (\phi(u), \phi(v)) \in E_{G'}$.

2.2 Deep Graph Networks

In this work, we focus on graph classification tasks consisting of learning an unknown function f that maps graphs $G \in \mathcal{G}$ to class labels $y_G \in \mathcal{C}$, where $\mathcal{G}$ and $\mathcal{C}$ indicate a set of graphs and the discrete set of possible labels, respectively. In this setting, a DGN implements a parameterized function f_Θ which assigns a predicted label $\hat{y}_G$ to input graphs. During training, the parameters Θ are

adjusted such that f_Θ well approximates f. In this work, we specify a DGN as the following function composition:

$$f_\Theta = f^{\text{trans}}_{\theta_1} \circ f^{\text{pool}}_{\theta_2} \circ f^{\text{out}}_{\theta_3}, \tag{1}$$

where

- $f^{\text{trans}}_{\theta_1}$ implements an isomorphic transduction of the input graph which maps the node features $\mathbf{x}_v$ to node embeddings $\mathbf{h}^L_v \in \mathbb{R}^{d'}$, $d' \in \mathbb{N}$, by applying MP for $L \geq 1$ iterations.
- $f^{\text{pool}}_{\theta_2}$ is a pooling operator that sums up the embeddings into a single graph representation vector
$$\sum_{v \in V_G} \mathbf{h}^L_v \in \mathbb{R}^{d'}.$$
- $f^{\text{out}}_{\theta_3}$ is a downstream classifier (in this work, it is a logistic regression model) that outputs a $|\mathcal{C}|$-dimensional vector of class probabilities based on the graph representation:
$$\hat{\mathbf{y}}_G = \text{softmax}\left(\theta^{\text{out}} \sum_{v \in V_G} \mathbf{h}^L_v\right).$$

The output of a DGN is the most likely class according to the output vector:

$$\hat{y}_G = \arg\max \hat{\mathbf{y}}_G.$$

2.3 Message passing variants

Usually, DGNs realize $f^{\text{trans}}_{\theta_1}$ with some form of MP. Generally speaking, MP is a blueprint defined at the node level as follows:

$$\mathbf{h}^{l+1}_v = \text{Upd}\left(\mathbf{h}^l_v, \text{Agg}(\{\text{Msg}(\mathbf{h}^l_v, \mathbf{h}^l_u) \mid u \in \mathcal{N}_v\})\right), \tag{2}$$

where the Msg function computes a message between every node and its neighbors; Agg combines all the messages received by each node from its neighborhood in a permutation-invariant fashion; and Upd updates every node embedding by combining the current node embedding and the aggregated messages. Each DGN implements its own version of MP; broadly speaking, MP implementations can be categorized as convolutional or recursive. Convolutional DGNs [2] implement $f^{\text{trans}}_{\theta_1}$ by stacking $L \geq 1$ MP layers, creating deep, end-to-end trainable architectures that progressively expand the receptive field of the node embeddings [11]. Instead, recursive DGNs implement $f^{\text{trans}}_{\theta_1}$ as a recursive contractive dynamical system. This process is executed until convergence, with each iteration corresponding to a single MP computation. This study focuses on three convolutional DGNs and one recursive DGN architecture, as described in the following. Notice that l indexes the layer for convolutional MP, while for recursive MP, it indexes the iterations.

Graph Isomorphism Network (GIN) [19] convolutional layers implement MP as follows (with $1 \leq l \leq L$, where L is a hyper-parameter):

$$\mathbf{h}_v^0 = \mathbf{x}_v, \tag{3}$$

$$\mathbf{h}_v^l = \mathrm{MLP}_{\boldsymbol{\Theta}^{l-1}}\left((1+\epsilon^{l-1})\mathbf{h}_v^{l-1} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{l-1}\right), \tag{4}$$

where ϵ is a learnable or fixed parameter, Msg returns the neighboring embedding, Agg is the sum function, and Upd is a sum function followed by a multilayer perceptron (MLP) parameterized with layer-dependent weights $\boldsymbol{\Theta}$.

GraphConv (GC) [14] convolutional layers implement MP as follows (with $1 \leq l \leq L$, where L is a hyper-parameter):

$$\mathbf{h}_v^0 = \mathbf{x}_v, \tag{5}$$

$$\mathbf{h}_v^l = \mathrm{ReLU}\left(\boldsymbol{\Theta}_1^{l-1}\mathbf{h}_v^{l-1} + \boldsymbol{\Theta}_2^{l-1}\sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{l-1}\right), \tag{6}$$

where Msg returns the neighboring embedding, Agg is the sum function, and Upd is a sum function followed by a ReLU nonlinearity. Notice that the current node embedding and the aggregated neighborhood embedding are weighted with layer-dependent parameters $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$.

The **Principal Neighbourhood Aggregation** (PNA) [4] convolutional layer implements MP as follows (with $1 \leq l \leq L$, where L is a hyper-parameter):

$$\mathbf{h}_v^0 = \mathbf{x}_v, \tag{7}$$

$$\mathbf{h}_v^l = \mathrm{MLP}_{\boldsymbol{\Theta}_1^{l-1}}\left(\mathbf{h}_v^{l-1}, \bigoplus_{u \in \mathcal{N}_v} \mathrm{MLP}_{\boldsymbol{\Theta}_2^{l-1}}(\mathbf{h}_v^{l-1}, \mathbf{h}_u^{l-1})\right) \tag{8}$$

with

$$\bigoplus = \begin{bmatrix} 1 \\ S(\mathbf{D}, \alpha = 1) \\ S(\mathbf{D}, \alpha = -1) \end{bmatrix} \otimes \begin{bmatrix} mean \\ std \\ min \\ max \end{bmatrix} \tag{9}$$

where Msg and Upd are MLPs with layer-dependent parameters $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$. Similarly, the function $\bigoplus$ realizes Agg, with $S(\mathbf{D}, \alpha)$ being a degree-based scaler, 1 being an identity scaler, $\otimes$ being a tensor product and *mean*, *std*, *min*, *max* being the mean, standard deviation, minimum and maximum functions to aggregate neighborhood messages, respectively.

Lastly, **Graph Echo State Networks** (GESN) [5] provides the following efficient recursive message passing variant based on Reservoir Computing:

$$\mathbf{h}_v^0 = \mathbf{0}, \tag{10}$$

$$\mathbf{h}_v^l = \tanh\left(\bar{\boldsymbol{\Theta}}\,\mathbf{x}_v + \boldsymbol{\Theta}\sum_{j \in \mathcal{N}_v} \mathbf{h}_j^{l-1}\right) \tag{11}$$

where $\bar{\mathbf{\Theta}}$ is a weight matrix introducing residual connections, and $\mathbf{\Theta}$ is the recursive weight matrix shared across all L iterations. The efficiency and further specific bias of this architecture come from the fact that both weight matrices are untrained but carefully initialized, creating a contractive/Markovian dynamical system that provides meaningful node embeddings to solve a task at convergence. Specifically, the recursive matrix $\mathbf{\Theta}$ must be initialized such that for every input $\mathbf{x}_v$ and every initial state of the system, as the number of iterations grows to infinity each node embedding reaches a stable fixed point. This is known as the Graph Embedding Stability property for which one sufficient and one necessary condition have been identified. The sufficient condition requires $\|\mathbf{\Theta}\|\|\mathbf{A}\| < 1$ while the necessary condition requires $\rho(\mathbf{\Theta}) < 1/\alpha$, where ρ indicates the spectral radius and α is the graph spectral radius [12]. The matrix $\bar{\mathbf{\Theta}}$, instead, is randomly initialized sapling values from the interval $[-\omega, \omega]$. It should be noted that both ρ and ω constitute hyperparameters of the architecture. However, L is not a hyperparameter as it is for the convolutional variants. In GESN, L is a stopping criterion in the form of the maximum number of allowed iterations, and therefore MP operations, for the dynamical system to reach the required convergence and provide meaningful node embeddings. As a consequence, L is usually large as the convergence of the system is a prerequisite to solving tasks with GESN.

2.4 The CAM attribution method

The class activation mapping (CAM) technique is a local post-hoc XAI method that assigns an *importance score* to each node in a graph. Specifically, given a graph G and a DGN f_{Θ}, the CAM method computes importance scores for each node by exploiting the following equivalence in the $f^{\text{out}}_{\theta_3}$ module:

$$logit_{(y)} = \theta^{\text{out}}_{(y)} \sum_{v \in V_G} \mathbf{h}^L_v = \sum_{v \in V_G} \theta^{\text{out}}_{(y)} \mathbf{h}^L_v. \tag{12}$$

where $logit_{(y)}$ identifies the score associated to class y by its readout unit, $\theta^{\text{out}}_{(y)}$ is the weight vector of the readout unit associated with class y, and $\mathbf{h}^L_v$ are the node embeddings computed by the model at the last layer L. In particular, CAM exploits the observation that the final logits of a DGN, usually computed as a linear transformation of the graph embedding $\sum_{v \in V} \mathbf{h}^L_v$, can be seen as the sum of a weighted contribution of each node $\theta^{\text{out}}_{(y)} \mathbf{h}^L_v$ to the logit.

Ultimately, the CAM method returns a vector $\hat{\mathbf{t}}_G \in \mathbb{R}^n$ where the i-th position stores the contribution (i.e., the importance score) of the i-th node to the overall graph prediction. Among the many explainers for DGN architectures available in the literature (see e.g., [18, 20]), CAM was chosen since it does not require the specification of hyperparameters, making it a suitable choice to compare different MP-based DGNs on equal terms.

3 Method

We develop our methodology on multiple XAI graph classification datasets of the form $\mathcal{D} = \{(G, y_G, \mathcal{T}) \mid G \in \mathcal{G}, y \in \mathcal{C}\}$ where graphs G are associated to target classes y as well as to sets of *ground truth explanations* $\mathcal{T} = \{\mathbf{t}_G^p \in \{0,1\}^n \mid p \in \mathcal{P}\}$ collecting a diverse *ground truth* (GT) for a given explanatory pattern p in the set of explanatory patterns $\mathcal{P}$. Specifically, the ground truth explanation is a binary vector encoding the relevance (1) or irrelevance (0) of each node to the graph prediction, while the explanatory pattern refers to the properties of the substructure identified by the relevant nodes within the graph. Our objective is to detect which explanatory pattern p is learned by different MP configurations – number of layers for the convolutional variants. Moreover, at fixed MP conditions, we also analyze the effect of training on the learned policy by studying the 2-norms of the weights of the convolutional MP layers.

3.1 Explanatory pattern detection

The identification of the explanatory pattern proceeds as follows:

1. First, the dataset $\mathcal{D}$ is partitioned into a training set $\mathcal{D}_{\text{train}}$ and a hold out test set $\mathcal{D}_{\text{test}}$. The training set is used to train and select the DGNs f_Θ, while the test set $\mathcal{D}_{\text{test}}$ is used for model assessment.
2. Once f_Θ has been learned and its generalization is assessed, graphs in $\mathcal{D}_{\text{test}}$ are processed by the CAM explainer, generating a set of explanations $\{\hat{\mathbf{t}}_G = \text{CAM}(G, f_\Theta) \mid G \in \mathcal{D}_{\text{test}}\}$;
3. Finally, given an explanatory pattern p, we compute its plausibility $\bar{Plis}_p$, which consists of the sample-wise Area Under the Receiver Operating Characteristic curve (AUROC) between the explanations provided by CAM $\hat{\mathbf{t}}_G$ with the corresponding ground truth explanations $\mathbf{t}_G^p$:

$$\bar{Plis}_p = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{G \in \mathcal{D}_{\text{test}}} \text{AUROC}(\hat{\mathbf{t}}_G, \mathbf{t}_G^p), \tag{13}$$

We compute $\bar{Plis}_p$ for each explanatory pattern $p \in \mathcal{P}$, and we determine the detected explanatory pattern as the one that maximizes $\bar{Plis}_p$. The choice of plausibility to establish the explanatory pattern is due to the fact that its computation does not require the definition of a threshold to transform explanations into binary versions, which is mandatory for metrics like accuracy.

3.2 Analysis procedure to detect the training effect

Given a set of DGN models, characterized by the same MP type and iteration number we compute the norms of all the learned weights as an indicator of their differences due to the convergence of the training procedure to different local minima. Then, we relate the average plausibility values of all models $\bar{Plis}_p$, $\forall p \in \mathcal{P}$, to their corresponding total norm values to understand the relationship between a learned policy and the models' characteristics determined by the training process at fixed MP conditions.

4 Experiments

4.1 Datasets and prediction policies

We tested our approach on three different synthetic binary classification graph datasets originally associated with a single explanatory pattern based on the detection of diverse motifs in the graphs [9, 10]. In particular, the BA2Motif dataset consists of 960 graphs with 25 nodes on average and assigns class 1 to Barabási-Albert (BA) graphs linked to a house motif and class 0 to BA graphs linked to a 5-node cycle motif; the BA2grid dataset consists of 2000 graphs with 22 nodes on average and assigns class 1 to BA graphs linked to a 3x3 grid motif and class 0 to plain BA graphs; the GridHouse dataset consists of 2000 graphs with 24 nodes on average and assigns class 1 to BA graphs linked to a 3x3 grid and a house motif, class 0 to BA graphs linked to either the grid or the house. However, each of these datasets admits a second explanatory pattern based on the degree of each node. Specifically, perfect classifiers can be constructed using only the average degree of the input graphs.

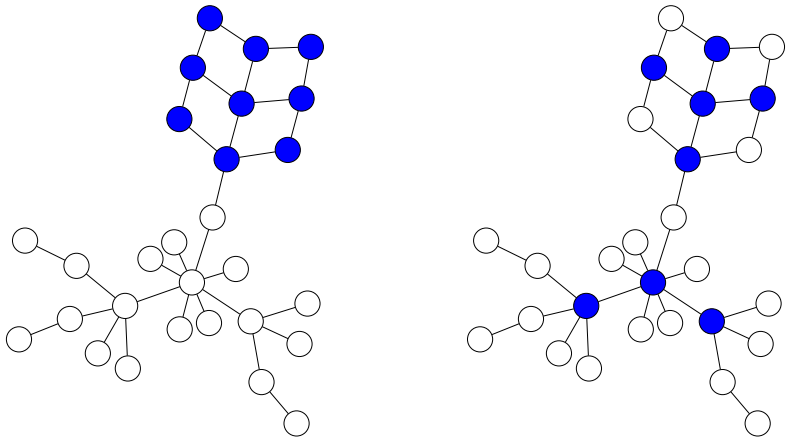
	Class 0		Class 1	
	min	max	min	max
BA2grid	1.87	1.93	2.20	2.4
BA2Motif	2	2	2.08	2.08
GridHouse	2.06	2.3	2.34	2.5

Table 1: Minimum and maximum average degrees by target class for each dataset.

In Table 1, we show that the maximum average degree characterizing class 0 graphs is always lower than the minimum average degree characterizing class 1 graphs for each dataset. Therefore, a DGN that learned this threshold would be able to separate the two classes. Moreover, we also constructed the set of associated ground truth explanations for the degree explanatory pattern, exploiting the characteristic that the minimum average degree for class 1 graphs is always above 2. Specifically, the associated ground truth vector marks nodes with a degree ≥ 3 as relevant (since they move the average towards class 1) and all other nodes as irrelevant. Thus, in our experiments, $\mathcal{P} = \{Degree, Motif\}$. Figure 1 shows the different explanatory patterns that can be picked up by the different DGNs during training.

4.2 Model selection

The models used in this study have similar hyper-parameters, allowing for the comparison between different MP variants. In particular, all convolutional models comprise a varying number of layers (up to a maximum of 5) followed by



(a) Motif-based (b) Degree-based

Fig. 1: We show the two possible explanatory patterns associated with the data employed in this study, using the BA2Grid dataset as an example. In both cases, the graph is assigned class 1 if the graph contains the pattern (i.e., the subset of nodes in blue). For the motif-based explanatory pattern (a), this subset identifies a 3x3 grid motif. For the degree-based explanatory pattern (b), the subset includes all nodes with degree ≥ 3.

a sum pooling operation to generate the graph embedding and a single layer MLP to map graph embeddings into class probabilities. The only exception to the shared model scheme is GESN, as recursive architectures do not map MP iterations to different layers. Consequently, for this MP variant, we only tested a one-layer configuration while keeping the number of iterations large enough (up to a maximum of 50) for the node embeddings to reach a stable fixed point useful for classification purposes. The model selection procedure adheres to the following scheme:

1. We split each dataset into training (80%) and test (20%) sets stratifying the splits following the class distributions.
2. We perform a 5-fold cross-validation technique over the training set, testing multiple hyperparameter configurations. In particular, for convolutional MP-DGNs, we used a grid search approach testing multiple values of the learning rate, weight decay, number of units inside each layer, and number of layers. As this latter hyperparameter is a key target of the analysis we kept its grid-search range fixed across all convolutional MP variants (from 1 up to a maximum of 5 layers). Other hyperparameter ranges instead have been tuned to increase the number of models achieving high enough performances to be kept as subjects of the analysis. Concerning, GESN, instead, the number of layers has been kept fixed to 1 but we tested different values for the ρ hyperparameter, the σ input scaler, the size of the untrained reservoir and

the l2 regularization coefficient of the trained readout. It is important to stress that we kept the range of studied ρ values in the range between 0 and 1 to satisfy the Graph Embedding Stability property and keep GESN in a contractive regime.
3. For the analysis purpose, we collected all models whose hyperparameter configurations exceed an average Accuracy value across all folds of 90%.

Once we identified all suitable models we applied, to all of them, the procedures to identify the learned explanatory pattern as well as the procedure to gather information concerning the influence of training in the learned policy through the computation of the norm of the weights.

5 Results

In Table 2, we show the average $\bar{Pls}_{Degree}$ and the $\bar{Pls}_{Motif}$ values across the selected hyperparameters configurations for each MP variant while grouping results based on the number of MP iterations. From the table, it is possible to see that both explanatory patterns have been picked up by the different DGNs examined. This highlights that both patterns are useful and allow generalization, despite their very different nature. In general, we observe that the preferred explanatory pattern changes across MP variants as well as across different layers (for convolutional MPs).

Delving into the results, two interesting phenomena can be observed. First, on some convolutional MP variants, the explanatory pattern changes as the number of MP layers increases. Specifically, GIN and GC with a low number of MP layers (1-2) pick up the degree explanatory pattern; however, they switch to the motif pattern at higher iterations. This result agrees with the expected inductive bias of these models, which are able to capture low-order structures such as degree at the lowest layers, and tend to capture more complex substructures such as motifs with a large number of layers. Interestingly enough, while GIN with two layers picks up the motif explanatory pattern, GC still prefers to pick up the degree explanatory pattern with the same number of layers. This indicates that even if the two MP variants have similar shifts from low-order to high-order explanatory patterns, their learning behavior, dictated by their respective inductive biases, is slightly different.

Second, there are MP variants whose explanatory pattern does not change when varying the iteration number. Specifically, PNA constantly picks up the degree explanatory pattern regardless of the number of MP layers; a behavior that we associate with the model's direct and easy access to the degree information via the scaler $S(\mathbf{D}, \alpha)$. Similarly, GESN, which is based on recursive MP, picks up the degree explanatory pattern, independently by the explored hyperparameter configuration space. In the latter case, this behavior is in agreement with the nature of recursive MP, since GESN matrices initialization imposes a contractive/Markovian dynamics on the model, which leads the model to favor localized information such as the degree. Overall, these results show that we can

		BA2grid					BA2Motif					GridHouse				
	Iterations	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
GIN	Motif	0.79	**0.95**	**0.99**	**1.00**	**0.99**	0.78	**0.83**	**0.92**	**0.95**	**0.96**	0.76	**0.90**	**0.98**	**0.99**	**0.99**
	Degree	**1.00**	0.84	0.83	0.77	0.75	**1.00**	0.74	0.80	0.69	0.61	**1.00**	0.83	0.85	0.80	0.79
GC	Motif	0.79	0.85	**0.98**	**1.00**	**1.00**	0.78	0.83	**0.92**	**0.95**	**0.95**	0.76	0.85	**0.96**	**0.99**	**0.99**
	Degree	**1.00**	**0.99**	0.87	0.78	0.76	**1.00**	**0.96**	0.81	0.70	0.67	**1.00**	**0.96**	0.87	0.82	0.80
PNA	Motif	0.79	0.83	0.84	0.84	0.84	0.77	0.78	0.77	0.78	0.77	0.76	0.76	0.75	0.75	0.75
	Degree	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.98**	**0.96**	**0.97**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GESN	Motif	0.80					0.78					0.70				
	Degree	**1.00**					**1.00**					**0.99**				

Table 2: Average values across the selected hyperparameter configurations of $\bar{Pls}_{Degree}$ and $\bar{Pls}_{Motif}$, here indicated with Motif and Degree tags, respectively, while grouped by the number of MP iterations (1-5). As GESN features one layer, only a single value is reported. Higher values across MP types, iterations, and prediction policies are shown in **bold**.

trace back to the inductive biases exploited by the DGNs by detecting which explanatory patterns they have picked up in relation to their architecture.

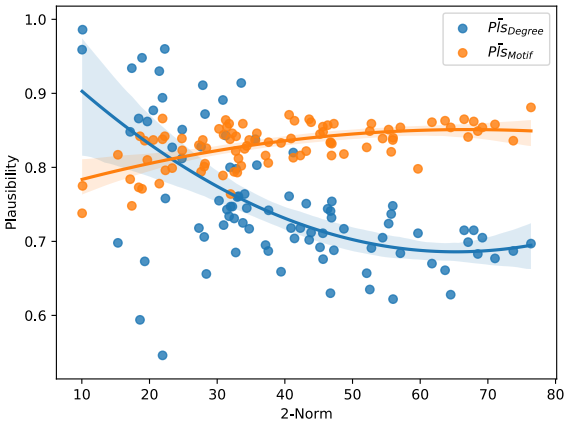


Fig. 2: Plausibility trends of the degree-based pattern and the motif-based pattern with respect to the 2-norm of the learned weights for a soft biased MP variant (2-layers GIN) on the BA2Motif dataset. A single model generates two points as its explanations are scored against both available policies.

Figure 2 summarizes the analysis of the influence of training on the learned explanatory patterns. The figure displays how the plausibility of the two ex-

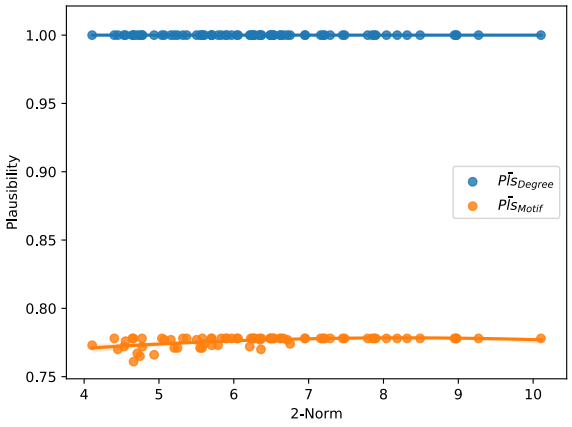


Fig. 3: Plausibility trends of the degree-based pattern and the motif-based pattern with respect to the 2-norm of the learned weights for a hard biased MP variant (1-layer GIN) on the BA2Motif dataset. A single model generates two points as its explanations are scored against both available policies.

planatory patterns picked up by a 2-layered GIN model varies as the 2-norm of the weights increases. Intuitively, smaller norms indicate that the learning dynamics are more similar to a 1-layered variant (i.e., one that is only capable of detecting low-order structures such as the degree) than larger norms. As can be seen, when the 2-norm is smaller, the degree explanatory pattern has the highest plausibility, while the setting slowly reverses and the motif explanatory pattern has higher plausibility as the norm increases. This plot shows that in certain MP configurations, the training dynamics (i.e., the local minima to which the training procedure has converged, which itself is related to the norm of the weights) play a role in determining which inductive bias is exploited by the DGN to generalize. Contrast this finding with Figure 3, where a 1-layered GIN variant is depicted. In this case, the training dynamics are not influential, as the weights norm does not determine a shift of explanatory pattern. In general, we observe that when the inductive bias favors the degree explanatory pattern (i.e., 1-layered convolutional variants or recursive variants with strong Markovianity), the training procedure has little to no influence in determining the generalization dynamics.

6 Conclusions

In this work, we studied the inductive biases of DGNs under the lens of XAI. Starting from the fact that DGNs achieve generalization according to different

mechanisms that are relatable to their inductive biases, we have shown that we can use tools from the XAI literature to unveil these mechanisms, drawing a connection between the downstream explanatory pattern (which itself relates to how the graph prediction is formed) and the upstream inductive bias of the MP implementation.

Our experiments highlighted that even in controlled environments like synthetic benchmarks, we can (i) witness the emergence of different explanatory patterns and (ii) gain insights into the inductive bias of diverse MP variants by identifying which pattern has been learned by each variant. Our analysis revealed several differences across the studied MP variants. First, there are convolutional variants that align to different explanatory patterns depending on the number of MP layers (specifically, GC and GIN). Second, for certain intermediate MP configurations (e.g., GIN with 2 layers), the 2-norm of the convolutional weights (which depends on the training procedure) impacts the generalization dynamics more than the MP specifics. Lastly, PNA (convolutional) and GESN (recursive) consistently align to the degree explanatory pattern, indicating that PNA's particular MP formulation and GESN's contractive dynamics characterize their inductive bias.

We believe our results are relevant since they present novel insights concerning the relationships between different DGNs, their MP implementation, and the inductive bias they embody. The undisclosed rich diversity of inductive biases grants Machine Learning practitioners multiple opportunities to solve their tasks and reach generalization capabilities by leveraging possibly diverse patterns. As a consequence of the inductive biases' effects on explanatory patterns, we encourage Machine Learning practitioners to test multiple message-passing variants to solve a given problem and use XAI techniques to check which explanatory patterns grant good generalization and to which inductive biases they can be related.

In subsequent works, our intention is to perform this analysis on a larger scale, considering different (possibly real-world) datasets, further architectures, and other XAI methods.

Acknowledgments. Research partly funded by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information fusion **58** (2020)

2. Bacciu, D., Errica, F., Micheli, A., Podda, M.: A gentle introduction to deep learning for graphs. Neural Networks **129** (2020). https://doi.org/10.1016/j.neunet.2020.06.006
3. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Elsevier, New York (1976)
4. Corso, G., Cavalleri, L., Beaini, D., Liò, P., Veličković, P.: Principal neighbourhood aggregation for graph nets. Advances in Neural Information Processing Systems **33** (2020)
5. Gallicchio, C., Micheli, A.: Graph echo state networks. In: The 2010 International Joint Conference on Neural Networks (IJCNN) (2010). https://doi.org/10.1109/IJCNN.2010.5596796
6. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70. PMLR (2017)
7. Gunning, D., Aha, D.: DARPA's explainable artificial intelligence (XAI) program. AI magazine **40**(2) (2019)
8. Kakkad, J., Jannu, J., Sharma, K., Aggarwal, C., Medya, S.: A survey on explainability of graph neural networks. arXiv:2306.01958 (2023)
9. Longa, A., Azzolin, S., Santin, G., Cencetti, G., et al., P.L.: Explaining the explainers in graph neural networks: a comparative study. arXiv:2210.15304 (2022)
10. Luo, D., Cheng, W., Xu, D., Yu, W., et al., B.Z.: Parameterized explainer for graph neural network. In: Advances in Neural Information Processing Systems. vol. 33. Curran Associates, Inc. (2020)
11. Micheli, A.: Neural network for graphs: A contextual constructive approach. IEEE Transactions on Neural Networks **20**(3) (2009). https://doi.org/10.1109/TNN.2008.2010350
12. Micheli, A., Tortorella, D.: Addressing heterophily in node classification with graph echo state networks. Neurocomputing **550** (2023). https://doi.org/10.1016/j.neucom.2023.126506
13. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
14. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., et al., J.E.L.: Weisfeiler and leman go neural: Higher-order graph neural networks. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01) (2019). https://doi.org/10.1609/aaai.v33i01.33014602
15. Oneto, L., Navarin, N., Biggio, B., Errica, F., et al., A.M.: Towards learning trustworthily, automatically, and with guarantees on graphs: An overview. Neurocomputing **493** (2022). https://doi.org/10.1016/j.neucom.2022.04.072
16. Pope, P.E., Kolouri, S., Rostami, M., Martin, C.E., Hoffmann, H.: Explainability methods for graph convolutional neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019). https://doi.org/10.1109/CVPR.2019.01103
17. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1) (2009). https://doi.org/10.1109/TNN.2008.2005605
18. Wu, Z., Pan, S., Chen, F., Long, G., et al., C.Z.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems **32**(1) (2021). https://doi.org/10.1109/TNNLS.2020.2978386
19. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)

20. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in Graph Neural Networks: A Taxonomic Survey. IEEE Transactions on Pattern Analysis & Machine Intelligence **45**(05) (2023). https://doi.org/10.1109/TPAMI.2022.3204236
21. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)