Investigating the Interplay between Features and Structures in Graph Learning

Daniele Castellana[1] and Federico Errica[2]

[1] University of Florence, Florence, Italy
daniele.castellana@unifi.it
[2] NEC Laboratories Europe, Heidelberg, Germany
federico.errica@neclab.eu

Abstract. In the past, the dichotomy between homophily and heterophily has inspired research contributions toward a better understanding of Deep Graph Networks' inductive bias. In particular, it was believed that homophily strongly correlates with better node classification predictions of message-passing methods. More recently, however, researchers pointed out that such dichotomy is too simplistic as we can construct node classification tasks where graphs are completely heterophilic but the performances remain high. Most of these works have also proposed new quantitative metrics to understand when a graph structure is useful, which implicitly or explicitly assume the correlation between node features and target labels. Our work empirically investigates what happens when this strong assumption does not hold, by formalising two generative processes for node classification tasks that allow us to build and study ad-hoc problems. To quantitatively measure the influence of the node features on the target labels, we also use a metric we call Feature Informativeness. We construct six synthetic tasks and evaluate the performance of six models, including structure-agnostic ones. Our findings reveal that previously defined metrics are not adequate when we relax the above assumption. Our contribution to the workshop aims at presenting novel research findings that could help advance our understanding of the field.

Keywords: Graph Representation Learning · Task Complexity · Deep Graph Networks.

1 Introduction

In recent years we have witnessed an exponentially increasing interest in machine learning techniques that can process graph-structured data [8,22,50,3]. This is due to the wide range of applications that these methods can deal with, as graphs are a natural abstraction in many scientific fields: chemistry [45], physics [9], and recommender systems [17] to name a few.

In the context of Deep Graph Networks (DGNs) [3], based on neural [46,36], probabilistic [2], or even untrained message passing [19], researchers have empirically observed that such models perform favourably on node classification tasks when the graph structure is homophilic [35], meaning that adjacent nodes in the graph share similar features or target labels to be predicted. In contrast, heterophilic graphs exhibit an opposite behaviour, and structure-agnostic baselines like a Multi-Layer Perceptron (MLP) proved to be better or very competitive compared to DGNs at classifying nodes under heterophily [54,32]. There has been a lot of effort in the literature to mitigate the (apparent) detrimental effects of heterophily, but a separate line of works recently showed that it is possible to construct completely heterophilic graphs where DGNs achieve an almost perfect test performance [31,10][3]. There, it is claimed that the distribution of the neighbouring class labels is a decisive factor to determine whether a DGN's inductive bias is suitable for the task, meaning that it leads to better performances compared to an MLP. These results, however, rely on a specific, and often implicit, assumption about a good correlation between the node features and the target.

The goal of this contribution is to continue the discussion on how to determine what are the components that make a node classification task difficult to solve, without being necessarily bound to message-passing architectures. At present time, we know that homophily alone is not a good metric to gauge the difficulty of the task, and in this paper we aim at showing that if we relax the assumptions of [31], the distribution of the neighbouring class labels does not tell the whole story either. We achieve this by evaluating different models on a few but carefully designed synthetic datasets for node classification that control specific properties of the graph, similarly to what is done in [40]. In particular, we give examples of cases where an unfavourable distribution of neighbouring class labels can be associated with very good DGNs performances.

Our observations open new questions that we would like to discuss in the context of the workshop: is it possible to create a taxonomy of the characteristics that make a task easy or hard to solve (e.g., for message-passing architectures)? Can we find new measures to quantitatively assess the presence or absence of such characteristics? Is it possible to find criteria to determine the intrinsic difficulty of tasks when the assumptions of [31] do not hold?

Addressing these questions can be critical for many reasons. First of all, it would enhance our understanding of DGNs' inductive bias (which is related but orthogonal to their expressiveness in distinguishing substructures [14]), thus allowing us to make sensible choices of model architectures for specific tasks. Second, it could give some practical guidelines to identify whether the structure of the graph gives a practical advantage compared to just using the node features, which (as of today) remains a purely empirical question [16]. In turn, this would allow people to successfully apply DGNs to problems where no structure is available, by creating ad-hoc structures with well-defined characteristics. So far, this has been done with nearest-neighbour graphs, although it remains unclear whether this brings a significant advantage compared to an MLP [25,4,33,44,53,29,48].

[3] We found some mistakes in the proofs of this paper, which however do not undermine its empirical results. We contacted the authors who are working on a new version.

2 Related Works

The body of literature that critically investigates the apparent harm of heterophilic graphs has recently been growing. Some authors have tried to define different "node behaviours" by analyzing the degree of a node and its local level of heterophily, arguing that they can explain whether a DGN will perform well or badly on a task [52]. These results rely on the assumption that nodes from different classes exhibit distinct (to some extent) distributions of their features. This is precisely the assumption we want to challenge in our contribution because, when put to the extreme, it means a structure-agnostic baseline would already be able to correctly separate all nodes. Another contribution introduced a new metric that measures how different node embeddings of different classes look after an aggregation step, showing a good correlation between high values of the metric and good performances of a DGN [30]. Others exploited the homophily assumption on the node features to rewire the graph by connecting homophilic but distant neighbours, under the assumption that homophily helps DGNs to perform better [27].

In the following, we will build on some recently defined metrics such as Label Informativeness (LI) [42] and Cross-Class Neighborhood Similarity (CCNS) [31] to create synthetic tasks. These are metrics that look more at the class information rather than the features, and so far they have shown to correlate well with the performance of a DGN when features of nodes of different classes are generated by distinct (i.e. well-enough separated) distributions.

In time, researchers have also tried to empirically assess the quality of DGNs compared to structure-agnostic baselines on a plethora of different real and synthetic tasks [47,16,15,23,40]. Their results are mostly quantitative, but they might act as a "certificate of goodness" of newly developed measures of tasks' difficulty. Recently, an attempt has been made in this direction by comparing the performance of models across a range of node and graph properties [32], but some of the conclusions do not explicitly mention all assumptions made in the process; for example, it appears that DGNs works better in the homophilic setup, but we know from other works that this is not necessarily the case.

Finally, most theoretical research on message-passing methods deals with the issues of over-smoothing [26,12,11,6], which is concerned with the flattening of the graph signal after repeated applications of graph convolutions, and over-squashing [1,49,20] of learned representations, meaning that there is a bottleneck of information when predictions have to be made. In addition, there is a very active research direction on the expressive power that comes with each message-passing model [38,51,18,5]. These works are orthogonal to the current discussion, as they talk about how difficult it can be for DGNs to process particular graphs regardless of the specific task under consideration.

Overall, the landscape of contributions is vast and it is easy to misinterpret some of the results when important assumptions are not explicitly formalized. In the next sections, we will consider the scenario where the assumption of the correlation between node features and target labels is not valid.

3 Method

In this section, we formalise a generative process to create synthetic node classification tasks, each defined by a graph. We will focus on undirected graphs $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges, respectively. The edges in $\mathcal{E}$ define the structure of G: two nodes $u \in \mathcal{V}$ and $v \in \mathcal{V}$ are connected, or adjacent, if an edge $\{u, v\}$ belongs to $\mathcal{E}$. The set of neighbours of node $u \in \mathcal{V}$ is denoted by $\mathcal{N}_u$, that is, $\mathcal{N}_u = \{v \in \mathcal{V} \mid \{u, v\} \in \mathcal{E}\}$; also, the degree of a node is defined as $d_u = |\mathcal{N}_u|$. In this paper, each node has a feature vector and a target class label attached to it. For a node u, we denote its input feature and output label as x_u and y_u, respectively. Without loss of generality, each input feature is a d-dimensional vector, meaning $x_u \in \mathbb{R}^d$, and each class label y_u belongs to a finite set of labels $\mathcal{C}$.

Before introducing the new generative process that we use in our experiments, we review the most common metrics used in the literature to quantify homophily, heterophily, and other relevant properties in graphs. A subsequent analysis of their limitations will motivate our systematic study of different node classification tasks by explicitly defining their generative processes and thus the underlying assumptions.

Homophily/Heterophily Quantities Quantitative metrics for measuring homophily in graphs rely on the class similarity between adjacent node pairs. In particular, a graph is considered homophilic if most of the adjacent nodes share the same class label. In contrast, in an heterophilic graph nodes of the same class tend not to connect together. The most common definition is possibly the "edge-homophily" [54]:

$$h_e = \frac{\sum_{\{u,v\} \in \mathcal{E}} \mathbb{I}[y_u = y_v]}{|\mathcal{E}|}, \tag{1}$$

where $\mathbb{I}[pred] = 1$ if and only if the predicate *pred* is true. A purely heterophilic graph has $h_e = 0$, whereas a purely homophilic graph satisfies $h_e = 1$.

A related quantity is the node-homophily [41], which computes the average proportion of neighbours that have the same class across all nodes in the graph. These two quantities are simple and intuitive, but they are sensitive to the number of classes and their imbalance, making the numbers hard to interpret and not very comparable across different datasets [28,42]. To overcome this issue, researchers have introduced the notion of "adjusted homophily" [42]:

$$h_{adj} = \frac{h_e - \sum_{k \in \mathcal{C}} \bar{p}(k)^2}{1 - \sum_{k \in \mathcal{C}} \bar{p}(k)^2}, \tag{2}$$

where $\bar{p}(k) \propto \sum_{u \in \mathcal{V}_k} d_u$, and $\mathcal{V}_k = \{u \in \mathcal{V} \mid y_u = k\}$ is the subset of nodes with the node class label equal to k. We will consider both edge and adjusted homophily in our experimental settings to see how different they can get.

Recent Quantities More recently, researchers realised that the dichotomy between homophily and heterophily does not tell the whole story about how "good" a graph structure is, where good means that said structure allows to better separate nodes of different classes with the right inductive bias [31,43,42]. Consequently, they started focusing on capturing neighbouring patterns across nodes of the same or different classes.

The Cross-Class Neighbourhood Similarity (CCNS) [31] computes the similarity between two classes k, k' using the neighbourhood class label distribution:

$$\mathrm{CCNS}(k, k') = \frac{1}{|\mathcal{V}_k||\mathcal{V}_{k'}|} \sum_{u \in \mathcal{V}_k} \sum_{v \in \mathcal{V}_{k'}} \cos(D_u, D_v), \tag{3}$$

where D_u is the empirical histogram (over $|\mathcal{C}|$ classes) of node u's neighbouring class labels, that is, $D_u(t) = \sum_{w \in \mathcal{N}_u} \mathbb{I}[y_w = t]$ counts how many neighbours of u have the class label t. The function $\cos(\cdot, \cdot)$ computes the cosine similarity between two histograms, but any other similarity function can be used. Importantly, the elements on the diagonal measure the neighbourhood similarity among nodes with the same class label; instead, the off-diagonal elements measure the neighbourhood similarity among nodes with different class labels. The CCNS has empirically proven to be a good certificate of difficulty for a task: when there is high intra-class similarity and low inter-class similarity, a Graph Convolutional Network (GCN) [24] typically performs better compared to an MLP [31].

The Label Informativeness (LI) [42] is another metric that measures the impact of neighbours' class labels to determine a particular node's class label:

$$\mathrm{LI} = 2 - \frac{\sum_{k \in \mathcal{C}} \sum_{k' \in \mathcal{C}} p(k, k') \log p(k, k')}{\sum_{k \in \mathcal{C}} \bar{p}(k) \log \bar{p}(k)}, \tag{4}$$

where $p(k, k') \propto \sum_{\{u,v\} \in \mathcal{E}} \mathbb{I}[y_u = k \wedge y_v = k']$ is the empirical joint distribution of class labels along the edges, and $\bar{p}(k) \propto \sum_{u \in \mathcal{V}_k} d_u$. Given an edge $\{u, v\} \in \mathcal{E}$, LI measures the amount of knowledge the label y_v gives for predicting y_u. Its value is equal to 0 when the node label does not depend on the neighbours' labels, and it is equal to 1 when the node label is completely determined by one of its neighbours' labels. Similarly to the CCNS, the LI empirically correlates with the performances of DGNs better than other homophily metrics [42].

Limitations of Current Approaches The CCNS and LI metrics seem to give more insights into what is a helpful graph structure, in the sense that the numbers they provide correlate with the performance of a graph machine learning classifier. However, these encouraging results rely on a strong assumption: the input features x_u are generated from a distribution F_{y_u}, and the expectation of the different distributions F_k are assumed to be distinct from each other [31]. While this assumption might seem reasonable (especially for tabular datasets), it is also too simplistic for at least two reasons. First, it assumes that there is already a good degree of separability between the nodes by just looking at their feature values. Secondly, we can build tasks where the node label depends solely on the structural property of the input graph and not on their feature values; in this case, the node features might even be missing. Generally speaking, a task can be solved even though the relationship between node features and target labels is too complex to show a direct correlation.

3.1 Generative Processes for Graphs

As mentioned in Section 1, our investigation stems from a few simple questions: what happens when the assumption on features we just described does not hold anymore? Can we still use quantitative metrics such as the CCNS and LI to determine how "good" a structure is? How badly can these quantities fail at recognising a helpful structure? To investigate this interplay between features and structures in graph learning, we introduce two generative processes to create synthetic node classification tasks with relatively straightforward characteristics. In both cases, we will be able to enforce a low correlation between the features and the target, which we measure with a quantity called Feature Informativeness (FI) (also known as the coefficient of determination in previous works [34]). The idea of FI is similar to LI: given that we model the input features $\mathbf{x}$ and the class labels $\mathbf{y}$ as random variables, we measure the amount of knowledge the feature $\mathbf{x}$ gives for predicting $\mathbf{y}$ as:

$$\mathrm{FI} = \frac{I(\mathbf{x}, \mathbf{y})}{H(\mathbf{x})}, \tag{5}$$

where $I(\cdot, \cdot)$ is the (empirical) mutual information and $H(\cdot)$ is the (empirical) entropy, with $0 \leq I(\mathbf{x}, \mathbf{y}) \leq H(\mathbf{x})$. When the input features are not categorical, we discretise them using a histogram with a bin size equal to 1. The value of FI ranges in the interval $[0, 1]$, where 0 (resp. 1) denotes minimal (resp. maximal) informativeness between node features and target labels.

Neighbours-Based Generative Process. The first generative process builds tasks where the class label of a node depends solely on its neighbours:

$$\begin{aligned} &h_u \sim \mathrm{Cat}(1/K, \ldots, 1/K), \qquad x_u \mid \mathbf{h} \sim \mathcal{N}(\mu_{h_u}, \sigma_{h_u}), \qquad \mathcal{E} \mid \mathbf{h} \sim E(\mathbf{h}), \\ &H_{uk} = \sum_{v \in \mathcal{N}_u} \mathbb{I}[h_v = k], \qquad y_u \mid \mathcal{E}, \mathbf{h} = \phi(H_{u1}, \ldots, H_{uK}) \end{aligned} \tag{6}$$

where all h_u are hidden factors modelled as categorical variables with K states and the node input feature x_u is generated from a Gaussian distribution whose parameters depend on h_u. The node class label y_u is generated using a deterministic function ϕ which takes as input K values $H_{u1}, \ldots, H_{uK}$, where H_{uk} counts the number of neighbours whose hidden variables are in the k-th state. Thanks to the introduction of the hidden variables and the choice of ϕ, we can highlight failure cases of the known metrics and argue for the creation of new ones that are effective when the value of FI is low.

Structural-Based Generative Process. In the second generative process, the class label of a node depends solely on some structural properties and not necessarily on neighbouring information. We define such process as:

$$y_u \sim \mathrm{Cat}(1/K, \dots, 1/K), \quad \mathcal{E} \mid \mathbf{y} \sim E(\mathbf{y}), \quad x_u \mid \mathcal{E} = d_u, \tag{7}$$

where all y_u are categorical variables with K states and the node input feature x_u is the node degree. The distribution E generates a structure whose edges are dependent on the specific node label assignments $\mathbf{y}$; we refer the reader to the next section for concrete examples. By construction, the node features are not required at all to solve the task. Here, we use the node degree as the only node feature similar to previous works [16] to obtain a low FI score.

4 Experiments

In the following, we report the details to reproduce our results, from the task generation to the training of the different baselines.[4]

4.1 Task Generation

We create six different node classification tasks: the first three are generated by the neighbour-based generative process (symbol N_i) while the last three are generated by the structural one (symbol S_i).

Most Common Hidden State (N_1) In the first neighbour-based task N_1, we fix $K = 4$ and generate a random structure (i.e. for each node we randomly select its neighbours). A continuous node feature is generated using 4 different Gaussian, one for each class. The function ϕ is defined as follows:

$$y_u = \phi_1(H_{u1}, H_{u2}, H_{u3}, H_{u4}) = \arg\max_{k \in [1,4]} H_{uk}. \tag{8}$$

Least Common Hidden State (N_2) The second task N_2 is obtained by just replacing the $\arg\max$ of Equation (8) with the $\arg\min$. To simplify the task from a “structural” viewpoint, we ensure that, for each node, it exists a hidden state that never occurs in its neighbourhood (i.e. $\forall u \in \mathcal{V}.\, \exists k \in [1,4].\, H_k = 0$).

Parity (N_3) The last neighbour-based task uses yet another different ϕ that generates the class node label as follows:

$$\phi_3(H_{u1}, H_{u2}) = H_{u2} \bmod 2. \tag{9}$$

This means we consider the parity of the number of neighbours of class 2 to determine the class of each node. In N_3, the number of hidden states (and thus the number of node class labels) is equal to 2.

[4] https://github.com/danielecastellana22/feature-structure-interplay-graph-learning

	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Avg. d_u	$K = \|\mathcal{C}\|$	h_e	h_{adj}	LI	FI
N_1	1911	32418	16.9639	4	0.2566	-0.0053	0.0003	0.0093
N_2	1600	33416	20.8850	4	0.2475	-0.0037	0.0003	0.0112
N_3	1600	31790	19.8687	2	0.5014	0.0025	0.0000	0.0029
S_1	1600	33524	20.9525	4	0.0000	-0.3333	1.0000	0.0182
S_2	1600	33398	20.8738	4	0.0000	-0.3333	0.2076	0.0164
S_3	1580	9288	5.8785	4	0.3402	0.1152	0.0464	0.2704

Table 1. We report the statistics of the tasks considered. All our tasks exhibit a low Feature Informativeness.

Easy Multipartite (S_1) The first task purely based on structural properties is task S_1, where we generate a multipartite graph with $K = 4$ clusters. The label y_u indicates the cluster of a node. To facilitate the task, the connectivity pattern among clusters is further simplified: nodes in cluster 1 are connected with nodes in cluster 4 and nodes in cluster 2 are linked to nodes in cluster 3. Thus, each node has neighbours with the same label.

Random Multipartite (S_2) The structural task S_2 is again a multipartite graph with $K = 4$ clusters. However, this time the structure is generated randomly. Given a node u and its cluster y_u, we randomly select its neighbours from all the nodes outside its cluster, i.e. the set $\{v \mid y_v \neq y_u\}$. Thus, each node has neighbours with all labels different from its own.

Count Triangles (S_3) In the last structural task, the node class label y_u indicates the number of triangles in which the node is involved. At first, we sample the labels $y_u \in [1, 4]$ for all nodes ($K = 4$). Then, we generate a structure such that, for all nodes, the value of y_u effectively indicate the number of triangles the node belongs to. There exist various structures that can satisfy these constraints; in our experiment, we simply use the algorithm proposed in [37,39] to generate a valid structure. We obtain the final structure by removing self-loops and parallel edges.

4.2 Task Statistics

In Table 1, we report some statistics of the generated tasks. First, we observe that the FI scores are always close to zero except for S_3, and all graphs appear to be quite heterophilic since the maximal values of h_{adj} is 0.11. All generated graphs exhibit different label connectivity patterns: in the neighbour-based tasks, there is no information gain in observing the label of a neighbour in isolation (i.e. all tasks have LI = 0); instead, in the structure-based tasks, we have three different and higher values for LI.

We also show the CCNS matrix for each task in Figure 1. All neighbour-based tasks (first row) have an almost uniform CCNS matrix, meaning that not

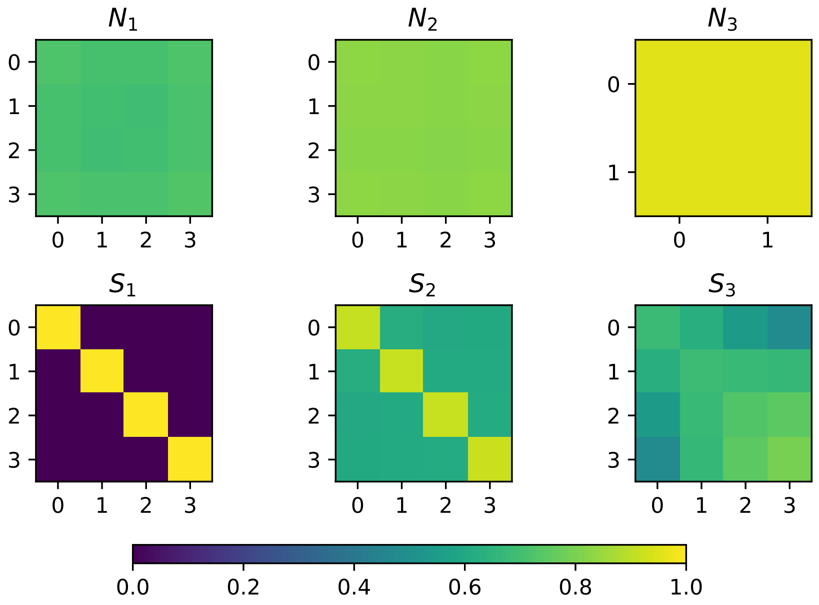

Fig. 1. We show the CCNS matrices for every task considered. Our tasks exhibit a broad spectrum of CCNs matrices with low FI.

even the neighbouring label distribution of a node is informative. In this case, one can easily see a relation to the very low values of LI in Table 1. Instead, in the first two structural tasks (second row) the CCNS exhibits a clear diagonal pattern, whereas in S_3 the pattern is less obvious; this indicates that the neighbouring label distribution of a node contains some information about its label. Interestingly, here there is less agreement between the CCNS and the LI: while on S_1 the value of LI is equal to 1 (i.e. the label of a node is completely determined by a neighbouring label), it is only 0.2 on S_2. As pointed out in [42], this happens because LI only considers the effect of picking one label from the neighbourhood rather than considering the whole neighbouring label distribution (as CCNS does). As a side note, we notice that the CCNS matrix in S_1 is different from the one in S_2 even if in both tasks each node can be uniquely classified by looking at the neighbouring class label distribution. This behaviour is likely due to the limited expressiveness of the cosine similarity, which cannot capture all differences between probability distributions.

For completeness, Figure 2 depicts the histograms of the node feature for each class and each task. In all the neighbour-based tasks (first row) we can clearly distinguish the different Gaussians used to generate the feature, but there is almost no correlation with the target labels thanks to our generative process.
Similarly, in the structural tasks (second row), the target labels have been generated in a completely independent way from the node features. The higher FI

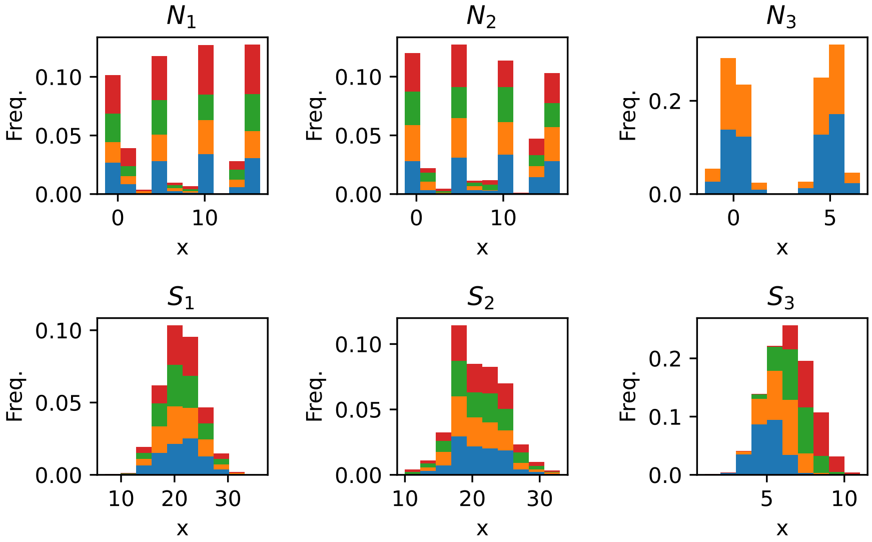

Fig. 2. We visualise the (discretised) distributions of the continuous features for the different tasks, divided by class value. Except for S3, the distribution of the feature values is approximately the same for each class.

for S_3 is merely an artefact due to our arbitrary choice of the degree as a node feature.

These analyses reveal that *when the target labels do not directly correlate with the node features*, but instead depend on the underlying generative process in complex ways, metrics such as LI and CCNS suggest that DGNs might perform on par or worse than an MLP on the first three tasks (N_1, N_2, N_3), whereas the opposite should be true for the last three (S_1, S_2, S_3). In the next sections, we will empirically show that this is indeed not the case, thus questioning the effectiveness of these metrics in discriminating the "goodness" of the graph structure when the FI is low.

4.3 Experimental Settings

For each node classification task, we consider an experimental setup where we randomly split 70% of the nodes for training, 10% for validation, and 20% for testing. To assess the model performances, we first carry out model selection by selecting the best configuration on the validation set, where accuracy is the metric of interest. After the best configuration is selected, we estimate the trained model's empirical risk on the test nodes. This process is repeated 10 times for different splits and the test results are averaged together.

The models considered in our experiments are GCN [24], GATv2 [7], Graph-SAGE [21], PNA [13], and an MLP. In addition, we consider another baseline called "Mode" that always predicts the most frequent label in the training data. In the model selection, we tried the following hyper-parameters: latent dimension $\in \{2, 5, 10, 20\}$, learning rate $\in \{0.01\}$, number of epochs $\in \{1000\}$, early-stopping patience $\in \{100\}$, number of layers $\in \{2, \ldots, 5\}$ for the neighbour-based tasks and $\in \{2, \ldots, 9\}$ for the structure-based ones, number of heads $\in \{1, 2, 3\}$ for GAT and aggregation function $\in \{\text{mean}, \text{max}\}$ for Graph-SAGE. For all DGNs, we also validated: graph augmentation with self-loops, input skip-connections and the ego- and neighbour-embedding separation (as suggested in previous works for heterophilic graphs [54]).

4.4 Results

	N_1	N_2	N_3	S_1	S_2	S_3
GCN	99.1 ± 0.4	72.9 ± 1.7	53.9 ± 3.0	99.5 ± 0.5	37.4 ± 1.8	54.9 ± 0.9
GATv2	99.0 ± 0.5	74.6 ± 1.5	58.0 ± 5.7	99.1 ± 0.5	59.2 ± 11.4	57.0 ± 2.0
GraphSAGE	98.9 ± 0.4	73.3 ± 1.4	53.3 ± 4.2	99.2 ± 0.7	70.6 ± 7.3	54.9 ± 1.5
PNA	98.9 ± 0.5	72.9 ± 1.0	53.4 ± 3.0	99.2 ± 0.4	49.0 ± 5.1	55.4 ± 1.2
MLP	32.0 ± 2.4	26.5 ± 1.8	51.6 ± 1.6	25.7 ± 1.7	25.8 ± 1.0	49.6 ± 1.0
Mode	32.7 ± 0.5	26.0 ± 0.6	50.9 ± 0.8	25.0 ± 0.6	24.8 ± 0.6	26.1 ± 0.6

Table 2. Mean and std test accuracy results obtained by models on the tasks.

In Table 2 we report the accuracy of all models under study on our synthetic tasks. On the neighbour-based tasks (N_1, N_2, N_3), the MLP has significantly poorer performances compared to the DGNs because it does not have access to the important information carried by the connectivity patterns, and it has comparable accuracy to the Mode baseline. This result comes as no surprise due to the absence of a correlation between features and target labels. The DGNs, on the other hand, are able to exploit the structural information and produce better scores. Nevertheless, the synthetic tasks are associated with different degrees of difficulty: while on N_1 all DGNs reach a test accuracy of almost 99%, they achieve only 74% and 58% on N_2 and N_3, respectively. Following up on the discussion of Section 4.2, the CCNS and LI are in clear disagreement with the empirical results of Table 2: a very low value of LI and an almost uniform CCNS matrix correspond to very good DGN scores on N_1.

In the structure-based tasks (S_1, S_2, S_3), the MLP still struggles. On S_1 and S_2, its performances are close to the Mode baseline, while on S_3 it is able to reach a higher accuracy of almost 50% while Mode is stuck at 26%. This behaviour is justified by the higher value of feature informativeness reported in Table 1. Also in this case, the DGNs always outperform the structure-agnostic baselines since the class labels depend on the structure, and as before the values of accuracy obtained by DGNs vary across the task. While all models solve S_1 by reaching an accuracy of 99%, the results on S_2 range from 37% (GCN) to 70% (GraphSAGE). Both S_1 and S_2 are multipartite graphs with the following difference: in S_1 the class of a node is completely determined by a label from its neighbourhood (since all neighbours have the same label), while in S_2 the node class label is determined by all the neighbours' label (it is the only missing labels in the neighbours). The different performances obtained are therefore due to the intrinsic difficulties in detecting the presence of neighbours rather than their absence. In fact, in S_1 it is sufficient to observe one of the neighbours' labels to predict the correct node class; instead, in S_2 we need to know all the neighbours' labels to detect the missing class. This is yet another difference that cannot be detected by metrics like the CCNS matrix since the neighbours' label distribution is different for each class (as shown in Figure 1). Finally, on S_3 all DGNs reach an accuracy that is slightly better than the MLP one. The hardness of this task arises from the complexity of counting the number of triangles in random graphs, and it cannot be captured by previous metrics that do not focus on the structural properties of the graph. For instance, the CCNS would seem to suggest that S_3 should have been simpler than N_1, but this is not what we have observed empirically.

In summary, these results show that class label-based metrics alone are not enough to capture the intrinsic difficulty of neighbour-based and structural tasks when the node features do not correlate with class labels, which is the main and strong assumption that previous works have (more or less implicitly) made. This calls for the development of new metrics and further investigation of data where the FI is low. We also argue that researchers have not devoted enough attention yet to the nature of the function ϕ. While we do not have access to this function in real-world problems, deepening our understanding of the relation between ϕ and DGNs' performances in a controlled setting can help us make considerable progress in the field.

5 Conclusions

In this paper, we identified potential limitations of known node classification metrics that estimate how useful the graph structure is in effectively distinguishing nodes belonging to different classes. We then presented two generative processes for node classification tasks where the main assumption of these metrics does not hold. Through our findings, it became evident that both the CCNS and LI are inadequate indicators when node features do not correlate with target labels. This work hopes to highlight the necessity for novel indicators to fill this gap and to emphasise the importance of a more systematic investigation into the underlying assumptions of these metrics. Such efforts are crucial if we want to make concrete progress in the area of graph representation learning.

References

1. Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *9th International Conference on Learning Representations (ICLR)*, 2021.
2. Davide Bacciu, Federico Errica, and Alessio Micheli. Probabilistic learning on graphs via contextual architectures. *Journal of Machine Learning Research*, 21(134):1–39, 2020.
3. Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 9 2020.
4. Adrien Benamira, Benjamin Devillers, Etienne Lesot, Ayush K Ray, Manal Saadi, and Fragkiskos D Malliaros. Semi-supervised learning and graph neural networks for fake news detection. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2019.
5. Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
6. Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Lio, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
7. Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *10th International Conference on Learning Representations*, 2022.
8. Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
9. Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.
10. Andrea Cavallo, Claas Grohnfeldt, Michele Russo, Giulio Lovisotto, and Luca Vassio. 2-hop neighbor class similarity (2ncs): A graph structural metric indicative of graph neural network performance. *Workshop on Graphs and more Complex structures for Learning and Reasoning (AAAI)*, 2023.
11. Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. PMLR, 2021.
12. Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 3438–3445, 2020.
13. Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, pages 13260–13271, 2020.
14. Giuseppe Alessio D'Inverno, Monica Bianchini, Maria Lucia Sampoli, and Franco Scarselli. A new perspective on the approximation capability of gnns. *arXiv preprint arXiv:2106.08992*, 2021.
15. Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

16. Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations (ICLR)*, 2020.
17. Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.
18. Floris Geerts, Filip Mazowiecki, and Guillermo Perez. Let's agree to degree: Comparing graph convolutional networks in the message-passing framework. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.
19. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
20. Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-symmetric dgn: a stable architecture for deep graph networks. In *11th International Conference on Learning Representations (ICLR)*, 2023.
21. Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017.
22. William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74, 2017.
23. Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, pages 22118–22133, 2020.
24. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.
25. Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 2019.
26. Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
27. Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, pages 13242–13256. PMLR, 2022.
28. Derek Lim, Felix Matthew Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Prasad Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
29. Siyuan Lu, Ziquan Zhu, Juan Manuel Gorriz, Shui-Hua Wang, and Yu-Dong Zhang. Nagnn: classification of covid-19 based on neighboring aware representation from deep graph neural network. *International Journal of Intelligent Systems*, 37(2):1572–1598, 2022.
30. Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*, 2022.

31. Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *10th International Conference on Learning Representations (ICLR)*, 2022.
32. Seiji Maekawa, Koki Noda, Yuya Sasaki, and makoto onizuka. Beyond real-world benchmark datasets: An empirical study of node classification with gnns. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
33. Brandon Malone, Alberto Garcia-Duran, and Mathias Niepert. Learning representations of missing data for predicting patient outcomes. In *Workshop on Deep Learning on Graphs: Method and Applications (AAAI)*, 2021.
34. Francesco M Malvestuto. Statistical treatment of the information content of a database. *Information Systems*, 11(3):211–223, 1986.
35. Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.
36. Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.
37. Joel C. Miller. Percolation and epidemics in random clustered networks. *Physical Review E*, 80(2):020901, August 2009. Publisher: American Physical Society.
38. Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, 2019.
39. M. E. J. Newman. Random Graphs with Clustering. *Physical Review Letters*, 103(5):058701, July 2009. Publisher: American Physical Society.
40. John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. Graphworld: Fake graphs bring real insights for gnns. In *Proceedings of the 28th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2022.
41. Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.
42. Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *arXiv preprint arXiv:2209.06177*, 2022.
43. Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *11th International Conference on Learning Representations (ICLR)*, 2023.
44. Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. Hetegcn: heterogeneous graph convolutional networks for text classification. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM)*, 2021.
45. Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.
46. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.
47. Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Workshop on Relational Representation Learning, Neural Information Processing Systems (NeurIPS)*, 2018.

48. Catherine Tong, Emma Rocheteau, Petar Veličković, Nicholas Lane, and Pietro Liò. Predicting patient outcomes with graph representation learning. In *AI for Disease Surveillance and Pandemic Intelligence: Intelligent Disease Detection in Action*, pages 281–293. Springer, 2022.
49. Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *10th International Conference on Learning Representations (ICLR)*, 2022.
50. Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.
51. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations (ICLR)*, 2019.
52. Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1287–1292. IEEE, 2022.
53. Xiang Yu, Siyuan Lu, Lili Guo, Shui-Hua Wang, and Yu-Dong Zhang. Resgnet-c: A graph convolutional neural network for detection of covid-19. *Neurocomputing*, 452:592–605, 2021.
54. Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.