Applied Network Science

RESEARCH Open Access

The interplay between communities and homophily in semi-supervised classification using graph neural networks

Hussain Hussain[1,2*], Tomislav Duricic[1,2], Elisabeth Lex[1], Denis Helic[1] and Roman Kern[1,2]

*Correspondence:
hussain@tugraz.at
[1] Institute of Interactive Systems and Data Science, Graz University of Technology, Inffeldgasse 13, 8010 Graz, Austria
Full list of author information is available at the end of the article

Abstract

Graph Neural Networks (GNNs) are effective in many applications. Still, there is a limited understanding of the effect of common graph structures on the learning process of GNNs. To fill this gap, we study the impact of community structure and homophily on the performance of GNNs in semi-supervised node classification on graphs. Our methodology consists of systematically manipulating the structure of eight datasets, and measuring the performance of GNNs on the original graphs and the change in performance in the presence and the absence of community structure and/or homophily. Our results show the major impact of both homophily and communities on the classification accuracy of GNNs, and provide insights on their interplay. In particular, by analyzing community structure and its correlation with node labels, we are able to make informed predictions on the suitability of GNNs for classification on a given graph. Using an information-theoretic metric for community-label correlation, we devise a guideline for model selection based on graph structure. With our work, we provide insights on the abilities of GNNs and the impact of common network phenomena on their performance. Our work improves model selection for node classification in semi-supervised settings.

Keywords: Graph neural networks, Community structure, Homophily, Semi-supervised learning

Introduction

Graphs are ubiquitous forms of data, which are encountered in many domains such as social networks, the web, citation networks, molecule interaction networks, and knowledge bases. For many years, machine learning on graphs has been an essential area in research. The developments in this area have resulted in solving or elevating the state of the art of many graph-related tasks, such as node classification (Kipf and Welling 2017), link prediction (Zhang and Chen 2018), and graph classification (Hamilton et al. 2017). One central task in this context is semi-supervised node classification, which consists of predicting unknown node labels from node features and a few known node labels. The state-of-the-art models for solving this task have predominantly been graph neural networks (GNNs) (Wu et al. 2019). However, recent works have shown that GNNs are

typically limited to work only on homophilic graphs (Zhu et al. 2020), i.e., where adjacent nodes are more likely to share the same label and have similar features. Homophily, which is a common phenomenon in many real-world graphs (Peixoto 2021), yields the rise of community structure in these graphs (Dev 2016). Communities have an impact on information propagation in graphs. For example, community structure can form barriers for information propagation in graphs, which forms the basis of processing with GNNs (Hasani-Mavriqi et al. 2018). However, neither community structure nor its relationship with homophily is well-studied in the context of GNNs.

Objective

We aim to extend our previous investigations (Hussain et al. 2020), in which we studied the impact of communities on GNNs. In the present paper, we go beyond community structure to study the impact of homophily, which plays a major role in forming communities in real-world graphs. Hence, we study the relationship between homophily and community structure, and their impact on the performance of GNNs in semi-supervised node classification.

Approach

We design an evaluation methodology to study the impact of communities and homophily.[1] Our methodology is based on manipulating a diverse set of real-world graph datasets. We start by defining contributing factors to GNN performance: homophily and community mixing. We quantify these factors ("Contributing factors" section) using measures on structure and node properties. We apply data manipulation (in "Structure manipulation" section) to modify homophily and/or community structure in the graph datasets. To understand the impact of communities and homophily on GNNs, we study the change in classification accuracy of trained GNNs on the original and manipulated graphs.

Based on our observations, we derive a measure to quantify the correlation between communities and node labels, using the uncertainty coefficient (Press et al. 2007). We use this measure to predict whether GNNs are suitable for semi-supervised node classification on a given graph dataset.

Findings

Our results show that homophily is the main contributing factor to the high accuracy of GNNs. Moreover, when community structure exists, it contributes to the performance of GNNs as it accelerates propagation between nodes of the same label when homophily exists. Furthermore, our results also suggest that small community structures (or subcommunities) act as barriers for propagation among nodes of the same class, and limit the GNN performance when homophily exists.

[1] Our code is on https://github.com/sqrhussain/structure-in-gnn.

Contributions

Our contributions are two-fold: (1) we provide an extensive study on the interplay between community structure and homophily and their impact on classification with GNNs, and (2) we propose to quantify the correlation between communities and labels using the uncertainty coefficient (Press et al. 2007), and show that this measure predicts the suitability of GNNs for graph data.

Our paper explains why GNNs do not work for heterophilic datasets and offers a guideline on the applicability of GNNs based on graph structure and node properties in the target graph. With our work highlighting the limitations of GNNs from the point of view of data (that is, homophily and communities), we hope to set the grounds for improved GNN models that account for the properties of the given data.

Communities and Homophily in GNNs

Preliminaries

Notations

Let $G = (V, E)$ be a graph with a set of nodes V and a set of edges E. Let each node $u \in V$ have a feature vector $x_u \in \mathbb{R}^d$, where d is the feature vector dimension; and a label $y_u \in \mathcal{L}$, where $\mathcal{L}$ is the set of labels.

Graph neural networks

GNNs are multi-layer machine learning models, that operate on graphs. They follow a message passing and aggregation scheme where nodes aggregate the messages that are received from their neighbors and update their representation on this basis. In a GNN with K hidden layers (with the input layer denoted as layer 0), each node u has a vector representation $h_u^{(k)}$ at a certain layer $k \leq K$ of dimension d_k. The transformation from a layer k to the next layer $k+1$ (as defined in Xu et al. 2019) is performed by updating the representation of each node u as follows:

$$\begin{aligned} a_u^{(k)} &:= \text{AGGREGATE}_{v \in \mathcal{N}(u)}(h_v^{(k)}), \\ h_u^{(k+1)} &:= \text{COMBINE}(h_u^{(k)}, a_u^{(k)}), \end{aligned} \tag{1}$$

where $\mathcal{N}(u)$ is the set of neighbors of u. The AGGREGATE function takes an unordered set of vectors, each of dimension d_k, as an input and returns a single vector of the same dimension d_k, e.g., element-wise mean or max. The COMBINE function combines the representation of u in layer k with the aggregated representation of its neighbors, e.g., a concatenation followed by ReLU of a linear transformation $\text{COMBINE}(h, a) = \text{ReLU}(W.[h, a])$. We set the representation of u in the input layer to the input features: $h_u^{(0)} := x_u$. In classification problems, the dimension of the last layer d_K equals the number of labels in the graph $|\mathcal{L}|$.

Semi-supervised learning on graphs

Semi-supervised learning aims to exploit unlabeled data in order to generate predictions given few labeled examples. In node classification, where the ground-truth classes of few nodes are given, semi-supervised learning exploits the unlabeled nodes as well as the

network structure to obtain accurate predictions. Many semi-supervised classification methods on graphs assume that connected nodes are more likely to share their label and features (Li et al. 2018; Zhu et al. 2020), which is referred to as homophily. Based on this assumption, approaches to solving this task usually aim to propagate node information along the edges. Earlier related approaches (Sen et al. 2008; Zhu et al. 2003) focused on propagating label information from labeled nodes to their neighbors. In many applications, however, graph nodes are associated with feature vectors, which GNNs utilize. GNNs achieved a significant improvement over the state of the art since they can effectively harness the unlabeled data, i.e., graph structure and node features.

Homophily

Homophily in graphs with labeled nodes is the tendency of adjacent nodes to have the same label. The opposite case, *heterophily*, is the tendency of adjacent nodes to have different labels. We use the terms *homophilic graphs* and *heterophilic graphs* to describe graphs, respectively, where the homophily holds and where heterophily holds.

Mutual interplay

Homophily and GNNs

The GNN update rule in Eq. 1 is seen as a form of *(Laplacian) feature smoothing* (Li et al. 2018) as it combines the feature vector of a node with the feature vectors of its neighbors. Feature smoothing results in neighboring nodes having similar vector representations. Therefore, for a homophilic graph, feature smoothing potentially results in node vector representation that matches the node labels. However, for a heterophilic graph, the propagation in Eq. 1 can cause nodes to have similar vector representations even if their labels are different. It is widely accepted that classifiers achieve better accuracy when similar vector representations tend to have the same labels.

Cluster assumption

Communities are densely connected subgraphs, and they are common in empirical graphs including social, citation, or web graphs. The existence of communities directly affects information propagation in graphs (Cherifi et al. 2019; Hasani-Mavriqi et al. 2018). As communities are densely connected, the feature smoothing performed by the update rule in Eq. 1 tends to make nodes within the same community have similar vector representations. The dense intra-community connectivity also causes the homophily to generalize to the community level, which means that nodes within the same community tend to share the same label. This is usually referred to as the *cluster assumption* (Chapelle et al. 2009) in semi-supervised learning. As a result, for a homophilic graph, GNNs cause nodes with the same label to have similar vector representations simplifying the classification task on the resulting vector representations. Li et al. (2018) hypothesize that this alignment of communities and node labels is the main reason why GNNs elevate state-of-the-art performance on the classification task. In this paper, we aim to experimentally test this hypothesis on eight datasets from different domains and with varying characteristics.

In the heterophilic case, which is often overlooked in the literature, a community has a variety of labels. The feature propagation between nodes of the same community would

therefore result in feature smoothing for nodes with different labels. This eventually makes the classification task harder since representation similarity does not imply label similarity in this case.

In summary, theory and literature show that homophily, as well as community structure, have a direct impact on learning with GNNs. In this paper, we set out to study the impact of these factors[2] and the relationships between them.

Methods

With the discussion above, we showed two main factors in the graph data that contribute to the performance of GNNs. In our study, we aim to investigate these factors and their impact on classification with GNNs. Our methodology is based on manipulating these factors in the graph structure and observing the GNN response to this manipulation.

We first show how we quantify these factors, and we subsequently describe how we manipulate graph data.

Contributing factors

Homophily.

We measure the homophily in a labeled graph as the fraction of edges that link nodes of the same label together. It is computed as follows

$$h = \frac{|\{(u,v) \in E : y_u = y_v\}|}{|E|}. \tag{2}$$

Community structure

To quantify this factor, we first perform community detection using the widely-used Louvain method (Blondel et al. 2008)[3] for community detection. Then we compute the community mixing parameter on the resulting partition. Community detection partitions the graph into a set of disjoint communities $\mathcal{C}$. The mixing parameter of a community partition is defined as the fraction of edges that connect nodes from different communities

$$\mu = \frac{|\{(u,v) \in E : C(u) \neq C(v)\}|}{|E|}, \tag{3}$$

where $C(u) \in \mathcal{C}$ is the community of node $u \in V$. For a typical community partition, the lower this measure is, the more pronounced the community structure is. For example, $\mu = 0$ indicates that each community corresponds to one or more connected components. If the value of μ is greater than 0.5, the majority of links are inter-community links, which is unrealistic for a modular community partition of a real-world network.

[2] Other factors in the graph have an impact on learning with GNNs. In our work, we build the motivation for homophily and community structure, and leave the investigation of other factors for future work.

[3] In our experiments, we stick to Louvain method for community detection as it is a widely used method. There is no algorithmic reason why we choose this method over another method except for the brevity of the work. Investigating other community detection methods is an interesting direction for future work.

By looking at Eqs. 2 and 3, we can notice that mixing resembles the heterophily of community memberships. In other words, if community memberships were completely analogous to node labels, then $h + \mu = 1$.

Structure manipulation

After establishing the intuitions behind the role of communities and homophily, we aim to show their impact experimentally. To achieve this, we evaluate five popular state-of-the-art GNN models on eight empirical datasets. Subsequently, we re-evaluate these GNN models on the same datasets after manipulating their structures. In particular, we perform the following types of manipulation where we intend to increase/decrease the homophily/mixing of the original graphs.

Increasing homophily (Hom$^+$)

To increase the homophily in a given graph, we build a new graph given nodes labels in the original graph. In this newly built graph, nodes of the same label are likely to link together regardless of their features or position in the original graph. To achieve this, we build a graph using a stochastic block model (SBM) (Holland et al. 1983), where blocks correspond to labels. In the stochastic block matrix, we set the values in the diagonal to be (relatively) higher than other values in the matrix (we provide details on building this matrix in "Appendix 7"). This implies a high edge density among nodes with the same label and low edge density among nodes of different labels, i.e., high homophily. The resulting graphs also have a defined community structure (stemming from the high density within labels), but it does not necessarily match the original community structure. Since homophily is intentionally high in the resulting graph, we expect GNNs to have very high accuracy on Hom^+ graphs.

Decreasing homophily (Hom$^-$)

To decrease the homophily in a given graph, we perform random edge rewiring. We achieve this with the configuration model (CM) (Newman 2003), where we rewire the graph using all nodes together. Since the rewiring is completely random, we expect the resulting graph to be completely useless for GNN classification. In addition to decreasing homophily, this manipulation also affects the community structure, causing a major increase in the mixing parameter. This graph preserves the degree sequence of the original graph.

Increasing mixing (Mix$^+$)

In this manipulation, we intend to loosen the community structure—eventually increasing the mixing parameter—without significantly changing homophily. To achieve this, we first partition the nodes based on their labels into disjoint sets. We take each induced subgraph from each of the resulting sets, and we rewire its edges randomly using the configuration model (CM) (Newman 2003). With this process, we perform rewiring within labels, preserving homophily and destroying sub-communities formed among nodes with the same label in the original graph. As a result, the propagation becomes faster between nodes of the same label, and hence we expect the impact of high homophily to be stronger on GNNs. For example, if the original graph is homophilic, this

Table 1 Summary of the data manipulation and the desired effect on the graph data

Manipulation	Homophily	Mixing	Side-effects
Hom^+	Increased	Preserved	Binomial degree distribution
Hom^-	Decreased	Increased	Destroys community structure
Mix^-	Preserved(*)	Decreased	Binomial degree distribution
Mix^+	Preserved	Increased	Destroys sub-communities

(*)Mix^- should result in a "homophilic" graph if the original was "homophilic", and vice versa

manipulation will help nodes of the same label to gain similar representations, and hence a better GNN performance. Since obtaining better performance by destroying some communities sounds counter-intuitive, we look more into that later in "Destroying sub-community structure" section.

Decreasing mixing (Mix^-)

In this manipulation, we intend to tighten the existing community structure, eventually decreasing the mixing parameter. We perform graph reconstruction based on communities and regardless of each node's label or feature vector. To achieve this (more details in "Appendix 7"), we use SBM (Holland et al. 1983) where blocks refer to communities in the original graph. As a first step, we perform community detection on the original graph using Louvain method (Blondel et al. 2008), which partitions the graph into sets of disjoint communities $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$. Second, we build a stochastic block matrix where each element i, j represents the edge density between nodes from c_i and c_j in the original graph. We artificially increase the density of each community by increasing the values in elements i, i by 50%, and decreasing the elements $i, j : i \neq j$ by 50%. Finally, we build a graph from this matrix using the stochastic block model. The resulting graph should strengthen (or tighten) the existing community structure in the original graph but should result in a binomial degree distribution (Karrer and Newman 2011). These graphs should not significantly change the level of homophily in the original graph, i.e., we should obtain a "homophilic" graph if the original is "homophilic", and vice versa. We expect the GNN performance on these graphs to be close to their performance on the original ones (regardless of whether the original are homophilic or not).

Lastly, we summarize the effect of each type of manipulation on data in Table 1. It is not possible to manipulate individual properties in the graph without affecting other properties. In our study, we try to monitor the change of these properties to guarantee the effect is minimal.

Evaluation

Datasets

To provide a better understanding of the roles of the studied factors, we aim for a diverse selection of datasets concerning homophily. Having this in mind, we use the datasets summarized in Table 2, divided into homophilic and non-homophilic datasets. The nodes in these datasets represent bodies of text, such as web pages, Wikipedia pages, and scientific papers. Since GNNs depend on feature propagation (as shown in Eq. 1), we restrict our experiments to graphs with node feature vectors. The feature vectors

Table 2 Dataset statistics after preprocessing (similar to Shchur et al. 2018)

	Dataset	Labels	Features	Nodes	Edges	Homophily	Mixing
Homophilic	CORA-ML	7	1433	2485	5209	0.81	0.09
	CiteSeer	6	3703	2110	3705	0.74	0.06
	PubMed	3	500	19,717	44,335	0.80	0.09
	CORA-Full	67	8710	18,703	64,259	0.57	0.10
Non-homophilic	Squirrel	5	2089	5201	216,933	0.22	0.22
	Actor	5	932	7600	29,926	0.22	0.21
	Texas	4	1703	182	307	0.06	0.16
	Wisconsin	5	1703	251	499	0.17	0.16

in all studied datasets are binary vectors indicating the presence/absence of a word in the respective node. We do not assume any change in the graph structure (e.g., adding/removing edges/nodes) between the training phase and the testing phase.

Homophilic datasets

As graphs with high homophily, we use four well known citation datasets: *CORA-ML, CiteSeer, PubMed* and *CORA-Full*. The label of a node in a citation graph represents the topic of the paper. Citations are expected to be denser among papers with similar topics than they would be between papers of different topics. For example, a publication about natural language processing more likely cites other natural language processing papers than human-computer interaction papers. Therefore, one could intuitively expect that adjacent papers tend to share the same label.

Non-homophilic datasets

For low homophily, we use a topic graph from Wikipedia (*Squirrel*), a Wikipedia co-occurrence network (*Actor*), and web pages graphs (*Texas* and *Wisconsin*).

The *Squirrel* (Pei et al. 2019) graph is a Wikipedia topic subgraph, where nodes represent pages, and edges represent links between pages. The five classes in this graph are based on average monthly traffic, a property that does not particularly imply homophily.

The *Actor* (Pei et al. 2019) dataset (induced from Tang et al. 2009) is a co-occurrence graph. Nodes correspond to actors, while edges correspond to their co-occurrence in Wikipedia pages. The nodes are assigned to five categories in terms of words of the actor's Wikipedia. The classes, therefore, do not imply homophily.

Texas and *Wisconsin* graphs are two components from the *WebKB* (Craven et al. 1998) graph dataset. In these graphs nodes are web pages, edges are links, and labels indicate the type of the web page, i.e., course, faculty, project, staff, or student. In this case, one cannot intuitively assume that nodes within a graph community are expected to share a label. For example, a web page of a staff member more likely links to projects on which this staff member is working than to other staff members' web pages.

GNN models

In our experiments, we study six GNN architectures that are widely used for semi-supervised classification on graphs (a) Graph Convolutional Networks (GCN) (Kipf

and Welling 2017), (b) Graph Sample and Aggregate (SAGE) (Hamilton et al. 2017), (c) Graph Attention Networks (GAT) (Veličković et al. 2018), (d) Simple Graph Convolutions (SGC) (Wu et al. 2019), (e) Approximate Personalized Propagation of Neural Predictions (APPNP) (Klicpera et al. 2019), and (f) Cluster Graph Neural Networks (CGCN) (Chiang et al. 2019).

We additionally compare these approaches to a simple feature-only baseline, namely, logistic regression, which ignores the graph structure and only uses the node features. GNNs benefit of both the graph structure and node features, while the feature-only baseline can only benefit of node features. For example, in datasets with a high number of features (comparing to nodes), it is expected that the feature-only baseline should have a high accuracy since it has access to a rich set of features. However, GNNs also have access to graph structure in addition to that rich set of features. Therefore, the comparison to this baseline can indicate whether a GNN model is useful for the classification task on the respective datasets, that is, whether GNNs can exploit the graph structure to increase the basleine perfromance. We show hyperparameter settings in "Appendix 7".

Evaluation setup

While some original graphs in our dataset selection are directed, we treat all graphs as undirected by ignoring the edge direction (which is in the line with previous research Kipf and Welling 2017; Hamilton et al. 2017; Shchur et al. 2018). All of these graphs are preprocessed similarly as Shchur et al. (2018), i.e., removing nodes with rare labels and selecting the largest weakly connected component. Following the train/validation/test split strategy as in Shchur et al. (2018), each random split consists of 50 labeled examples per class (20 for training and 30 for validation), and the rest are considered test examples. This applies to all of our datasets except *Texas* and *Wisconsin*, where we use 10 training examples and 15 validation examples per class due to the fewer number of nodes. To evaluate the GNN models on the original graphs, we follow the evaluation setup close to Shchur et al. (2018) by having 10 random splits and randomly initializing model weights 10 times for each split. The same process is carried out to evaluate the feature-only baseline (logistic regression) model.

The evaluation is slightly different for the manipulated graphs since they include an additional level of randomization, and it goes as follows. For a manipulation type on one dataset, we generate 10 manipulated graphs with different random seeds. We evaluate each of these generated graphs through 5 different random splits and 5 times initializing model weights. As a result, the reported accuracy for a GNN architecture on the *original graph* is presented after training the GNN 100 times. Meanwhile, the reported accuracy of a GNN architecture on one of the *manipulation types* is presented for training the GNN 250 times per dataset, i.e., 10 randomly generated graphs $\times$ 5 random splits $\times$ 5 times of weight initialization.

Results

Manipulation impact on graphs

We introduced intuitions about the datasets and their properties previously in "Datasets" section. Next, we show the measurements of the two contributing factors we introduced in "Contributing factors" section on the original datasets, and on their

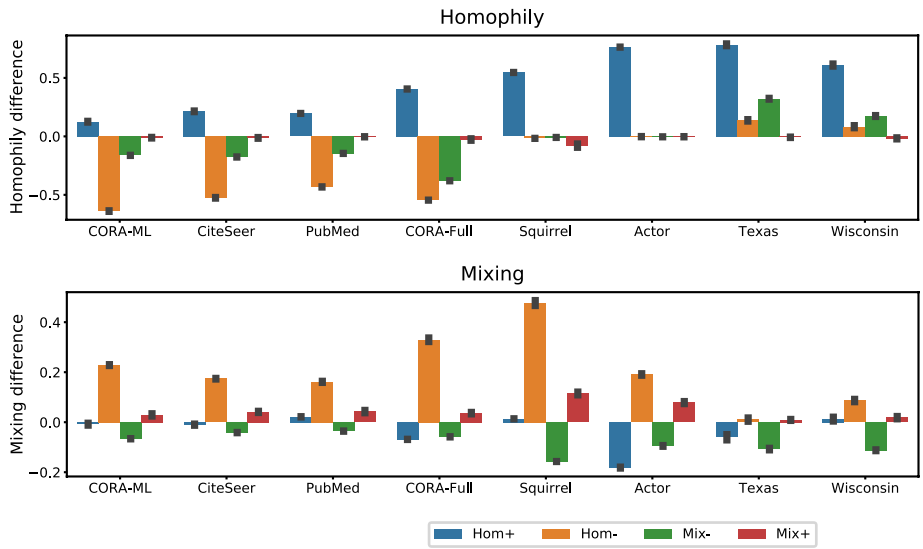

Fig. 1 Homophily and mixing measurements of the manipulated graphs. Here we show the change in homophily and mixing of the manipulated graphs comparing to the original graphs of each dataset. We can see that the intuitions we mention in our discussion in "Datasets" section are confirmed

manipulated versions. We present these measurements in Fig. 1. Comparing the original graphs to the manipulated ones in this figure, we notice the following.

Hom^+ increases homophily for all datasets (as expected). For most datasets, it does not significantly change the mixing parameter. There are some cases where Hom^+ decreases mixing, e.g., Actor.

Hom^- decreases homophily for homophilic datasets to a relatively very low level. However, these graphs increase the mixing of the datasets thereby effectively destroying community structure. The rewiring does not decrease homophily for datasets, where the homophily is already low. This can be explained by some of these graphs being heterophilic (or anti-homophilic). In this case, our manipulation decreases heterophily, e.g., Texas and Wisconsin. Other graphs where homophily does not change, that are Actor and Squirrel, can be neutral with respect to homophily. In other words, they are neither homophilic nor heterophilic, and random rewiring just results in a similar level of homophily.

Mix^+ increases the mixing parameter, loosening the community structure and potentially causing faster feature propagation. It also keeps a very close level of homophily in general.

Mix^- decreases the mixing parameter, tightening the community structure. It does change the homophily but not as significantly as Hom^+ or Hom^-. Mix^- decreases the homophily for the first four homophilic datasets, and slightly increases it for the other four non-homophilic datasets.

All in all, the manipulations generally achieve the changes which we intended for each of them with few exceptions. Since the effects of our manipulations depends on the nature of the original graphs, we will present the results of GNN accuracy on the basis of homophily and mixing of the input graph in "Impact of homophily and

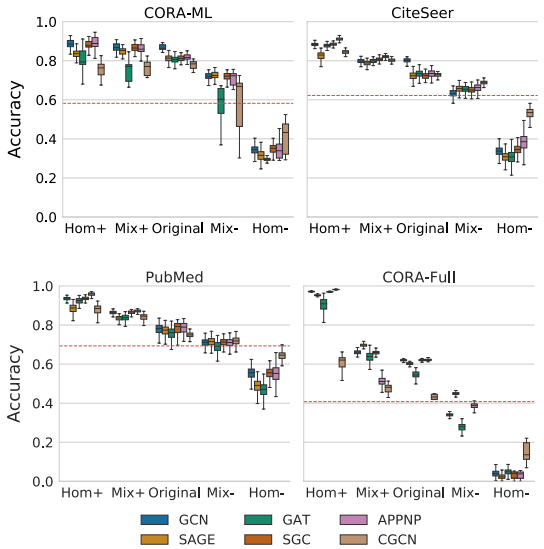

Fig. 2 GNN accuracy for homophilic datasets. This figure presents the accuracy of GNN models on the *homophilic* datasets (original and manipulated). The red dashed line represents the median accuracy of the feature-only baseline. The performance on the original graphs is consistently higher than that of the baseline. Increasing homophily results in a higher accuracy for all datasets. Decreasing community mixing causes a performance drop, which still outperforms the baseline in most cases. Increasing mixing (by destroying sub-communities) while preserving homophily generally causes an increase in performance. Destroying homophily and community structure (with Hom^{-}) causes a major performance drop

mixing on GNN performance" section. Next, we evaluate and compare different GNN models on original and manipulated graphs.

GNN performance

We summarize the evaluation results of the GNN models with respect to accuracy for the original graph and its corresponding manipulations of homophilic datasets in Fig. 2 and non-homophilic datasets in Fig. 3.

Original graphs

We first compare the performance of GNNs on the original graphs to the performance of the feature-only baseline. We easily notice that GNNs outperform the feature-only baseline on all homophilic datasets in Fig. 2, and do not outperform the baseline on non-homophilic datasets in Fig. 3. This makes the impact of homophily very clear: the higher the homophily is the more likely GNNs are to outperform feature-only baselines. For non-homophilic datasets, the graph structure seems irrelevant to the learning process or is even hindering it.

Next we test the statistical significance for each approach on each dataset against the corresponding baseline. We compute the non-parametric *Mann-Whitney U test* for unpaired data with the significance level $\alpha = 0.01$ (Bonferroni corrected). We compare the performance of all GNN models on the original graphs to the performance of the feature-only baseline. The significance test shows that the performance of GNNs on the original homophilic graphs is significantly higher than the performance of the feature-only baseline. For datasets with low homophily, the baseline significantly outperforms

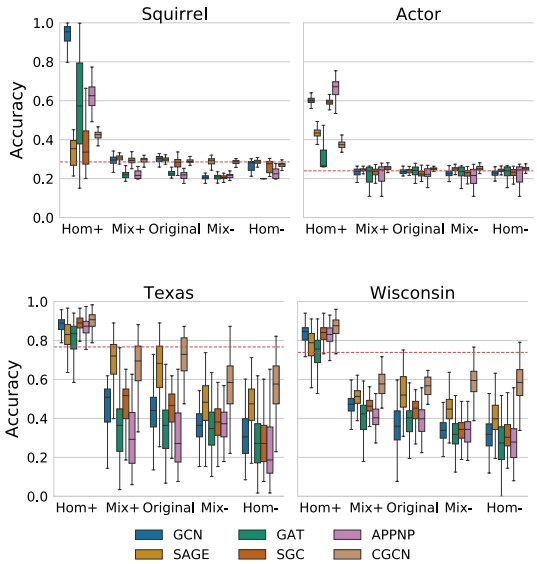

Fig. 3 GNN accuracy for non-homophilic datasets. This figure presents the accuracy of GNN models on the *non-homophilic* datasets (original and manipulated). GNNs cannot outperform the feature-only baseline on the original graphs, and achieve a similar accuracy to the Hom^- graphs, where the structure is destroyed. This suggests that the graph structure is irrelevant to GNN accuracy. When we enforce homophily with Hom^+ graphs, GNNs outperform the baseline on all datasets. In other words, homophily makes the graph structure useful for classification with GNNs. Destroying any aspect of the network does not improve the GNN performance over the baseline. Comparing these results to those in Fig. 2, we see that these aspects (degree distribution, community structures and sub-communities within labels) do not concern GNNs when homophily does not exist

GNN models on WebKB graphs (i.e., Texas and Wisconsin). On the other hand, there is an overlap of the performance on Actor and Squirrel datasets, where the homophily is not as low. The significance test on these two datasets shows a non-significant difference for some GNN models (namely, SAGE and GAT for Actor dataset, and GCN and SGC for Squirrel dataset).

Manipulated graphs

On all datasets, GNNs performance on Hom^+ (both for homophilic and non-homophilic data) outperforms their performance on the original graphs and the performance of the feature-only baseline. By artificially adding high homophily, GNNs improve their performance regardless of other aspects. Even when the mixing decreases for CORA-Full, Actor and Texas (as pointed out in "Manipulation impact on graphs" section), the GNN still outperforms the baseline and the performance on the original graphs. This indicates that the change of mixing does not affect the general conclusion here, that is, higher homophily causes better GNN accuracy.

For Hom^- graphs, the GNN performance is generally lower than the feature-only baseline. These graphs cause a major drop in performance for homophilic data compared to the original graphs. For non-homophilic data, the performance mainly does not drop with Hom^- graphs. These observations lead to the conclusion that low homophily decreases the performance of GNNs. We should, however, bear in mind that this manipulation also increases mixing significantly. We dedicate "Impact of homophily and

mixing on GNN performance" section to look at homophily and mixing separately to test against confounding factors.

Let's look at Mix^+ graphs now. We can see an increase of GNN accuracy on all homophilic datasets. Mix^+ graphs destroy the sub-community structure within labels. One explanation for the increased accuracy is that the new structure causes a faster feature propagation between nodes of the same label. This fast propagation helps making the GNN vector representation of same-label nodes more similar than it is in the original graphs. We further discuss this interpretation in "Destroying sub-community structure" section.

With Mix^- graphs for homophilic data, the GNNs show a drop in accuracy, but they still outperform the feature-only baseline. This observation shows a noticeable impact of communities on node classification in homophilic data since Mix^- rebuilds the graph only based on community memberships, and still maintains a higher performance than the baseline. On the contrary, for non-homophilic data, the performance on Mix^- graphs is not very different from destroying communities and homophily with Hom^-. In other words, community structure becomes irrelevant when homophily is low. Since Mix^- manipulation (as well as Hom^- and Hom^+) has an effect on both homophily and mixing, we need to take a deeper look into the results, which we address in "Impact of homophily and mixing on GNN performance" section.

GNN models comparison

Figures 2 and 3 show that GNN models generally behave in a similar manner. Most models achieve a high performance when the homophily is high and vice versa. We notice that CGCN does not perform as well as the other GNN models when the homophily is high. CGCN outperforms other GNN models when the homophily is low showing resilience to low homophily.

Impact of homophily and mixing on GNN performance

In this section, we report the accuracy of the GCN model[4] on original and manipulated graphs. Figure 4 presents accuracy as a function of homophily and mixing.

We notice that the main factor that affects the accuracy is the homophily of the given graph, with higher homophily correlating with higher GCN accuracy in all datasets. The changes in mixing do not show a strong impact on GCN accuracy. However, we do notice a slight increase in accuracy when the mixing (slightly) increases where the homophily is originally high (top row). We do not notice a similar behaviour when increasing mixing, and the homophily is originally low (bottom row)—observation discussed in "Destroying sub-community structure" section. In fact, changing the mixing when the homophily is low does not cause a noticeable change in GCN accuracy, stressing the finding from "GNN performance" section that community structure becomes irrelevant when homophily is low.

The conclusion of this section is that regardless of the manipulation applied to the graph, we find the homophily as the main contributor to the accuracy. We notice that

[4] We restrict the results in the main body of the paper to GCN and report the results of the rest of the models in "Appendix 7"

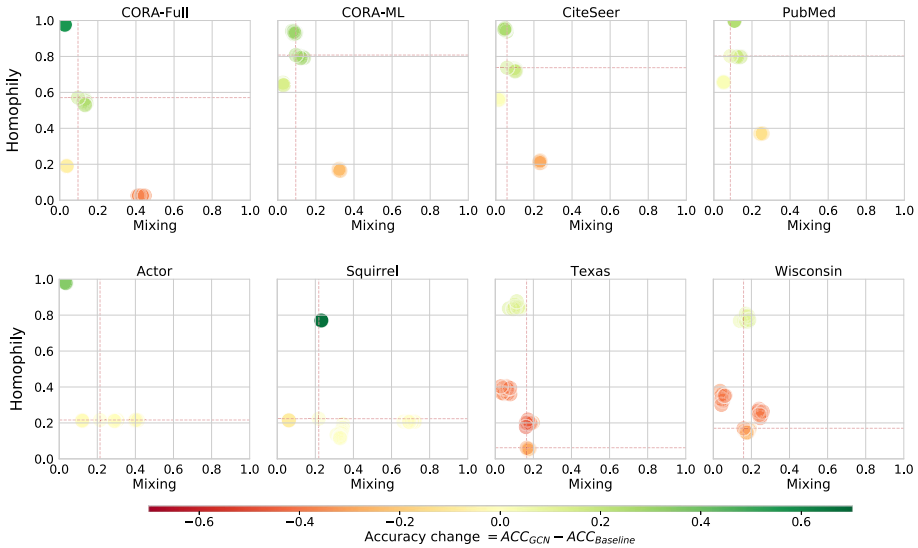

Fig. 4 Homophily and mixing impact on GCN accuracy. In this figure, we present the accuracy gain of GCN over the feature-only baseline (with hue) as a function of mixing (x-axis) and homophily (y-axis). The red dashed lines represent the mixing and the accuracy of the original graph (the point at their intersection in each sub-figure is the original graph). We notice a change in hue along the y-axis, and not the x-axis. This shows a strong impact of homophily on the accuracy gain, while mixing does not have such a strong impact. Note, high mixing and high homophily cannot be achieved with our manipulations at the same time, e.g., cases of pair-wise homophilic relations without any community structure.

the role of communities is less trivial to interpret but is relevant only when the homophily is high.

Destroying sub-community structure

For Mix^+ graphs, we rewire each subgraph of nodes with the same label. This preserves the homophily level of the graph, but destroys the inner sub-community structure for nodes of the same label. In "Structure manipulation" section we hypothesized that this causes faster propagation, and in Fig. 2 we observed an increase of GNN performance on these graphs. Now we intend to empirically show that the feature propagation becomes faster.

We take a closer look into the feature propagation in Mix^+ and original graphs by extracting node representations in the first layer of a trained GCN model. The representation in the first layer gives an intuition on how fast feature propagation occurs.

As an example, we first visualize the vector representations of original and Mix^+ for CORA-ML. We reduce the dimensionality of these vectors to two dimensions using T-SNE, and plot the 2D points with their labels in Fig. 5. The two plots of original and Mix^+ graphs show that nodes of the same label agglomerate already in the first layer for Mix^+ graph, while they still show more overlap for the original graphs. This suggests the faster propagation among nodes of the same label for Mix^+ graphs in this example.

To study the overlap of nodes from different classes, we measure the average silhouette coefficient for the representations of the original graphs and the Mix^+ graphs (Table 3). We notice an increase of the average silhouette coefficient on the Mix^+ graphs especially for homophilic datasets, indicating the reduced overlap between labels. This reduced

Table 3 Average silhouette coefficient for original and Mix^+ graphs. After training a GCN model on each graph (initializing the weights 10 times), we calculate the average silhouette coefficient of the intermediate representation in the first GCN layer. We report the mean of these values over each dataset, together with the difference

	CORA-ML	CiteSeer	PubMed	CORA-Full	Squirrel	Actor	Texas	Wisconsin
Original	0.188	0.147	0.141	− 0.048	− 0.119	− 0.039	0.102	− 0.027
Mix^+	0.266	0.227	0.226	− 0.014	− 0.092	− 0.037	0.102	− 0.016
Silhouette Δ	0.078	0.080	0.085	0.034	0.027	0.002	0	0.011

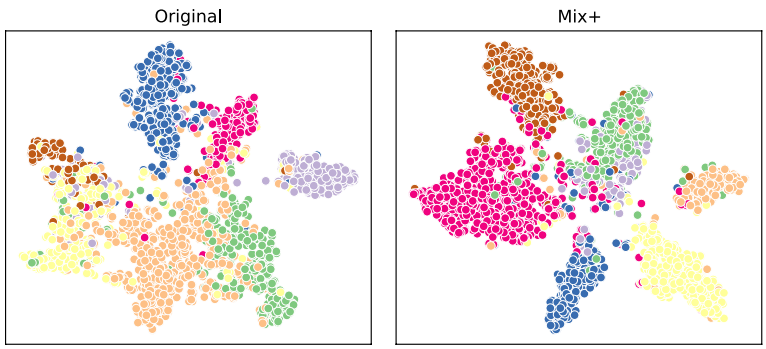

Fig. 5 Feature propagation in original and Mix^+ graphs. We visualize the embeddings of nodes in the first hidden layer of a trained GCN model for Cora-ML dataset projected to two dimensions with T-SNE. The embeddings in the original graph (on the left) show an overlap of the areas of different labels. The embeddings in the Mix^+ graph (on the right) show less overlap and are more linearly separable already in the first layer. As a result, we see an increase of the performance with Mix^+ networks

overlap indicates a faster intra-label feature propagation in Mix^+ graphs compared to original graphs.

Community-label correlation

The results we obtained so far show deeper insights on the impact of homophily and communities in node classification with GNNs. Based on these results, we intend to devise a method that helps to determine whether to use a GNN model for semi-supervised node classification on a given graph.

We start by looking at what homophily indicates. In Fig. 1, we can see that homophily estimates whether GNNs work on a given graph very well, i.e., homophily is high when GNNs have high accuracy (in Fig. 2) and vice versa (in Fig. 3). In other words, homophily correlates really well with GNN performance, and is a good predictor of whether GNNs can be applied to a graph dataset for node classification.

So why not estimate homophily directly?

Although homophily is a good predictor, estimating homophily requires having enough *adjacent labeled* nodes. This is not easily achievable in semi-supervised learning settings, where only very few nodes are labeled. Therefore, we need an alternative measure that can function with very few labeled nodes.

Intuitions from the results

The results in Fig. 2 show that GNNs could still outperform the baseline on Mix^{-} graphs, where the graph structure is built merely from community memberships. This observation highlights that communities encode label information in homophilic graphs, i.e., where GNNs should work. In other words, when the community membership correlates with node labels, GNNs can outperform a feature-only baseline. Thus we can require to approximate homophily not on pair-wise level, but on aggregated (or community) level. Next we describe a measure for correlation between communities and labels to use as a predictor for GNNs applicability.

Community-label correlation measure

To quantify the correlation between communities and labels, we introduce a measure for *community-label correlation*. This measure should quantify how much information a node's community reveals about its label. Let L be a random variable taking values in the set of labels $\mathcal{L}$, i.e., $L(u)$ is the label of node $u \in V$. Assuming the graph is partitioned into a set of disjoint communities $\mathcal{C}$, we define another random variable C taking values in $\mathcal{C}$, i.e., $C(u)$ is the community of node $u \in V$.

To measure how much the (fraction of) uncertainty about L is reduced knowing C, we use the uncertainty coefficient (Press et al. 2007) of L given C. This coefficient can be written as

$$\rho = U(L|C) = \frac{I(L;C)}{H(L)} \in [0,1], \tag{4}$$

where $H(L)$ is the entropy of L, and $I(L;C)$ is the mutual information between L and C.

When $\rho = 1$, all nodes within each community share the same label, and thus knowing the node's community means also that we know the node's label. On the other hand, when $\rho = 0$, the label distribution is identical in all communities, so knowing the community of a node does not contribute to knowing its label. In general, the higher the eliminated uncertainty about the labels when knowing communities is (i.e., the closer ρ is to 1), the more likely it is that GNNs can exploit the graph structure, and vice versa.

Unlike homophily, this correlation measure does not require adjacent labeled nodes. This measure requires enough labeled nodes belonging to the same community, which is easier to have in semi-supervised settings. To calculate ρ for a set of nodes, the labels of these nodes must be available. Therefore, for each dataset in the next experiments, we compute ρ using the labeled nodes from the training and validation sets.[5]

Can community-label correlation predict GNN applicability?

Figure 6 shows the value of the community-label correlation ρ on the manipulated graphs. The increase/decrease in the community-label correlation follows a similar pattern to the increase/decrease of homophily value in Fig. 1, such as, increased homophily with Hom^{+} and decreased with Hom^{-}. Since this measure helps to indicate homophily,

[5] We are aware that this correlation measure is just one choice of estimating homophily. Therefore, we plan on doing a broader search for better metrics and guidelines to estimate homophily, and assess whether GNNs are suitable for node classification in semi-supervised settings.

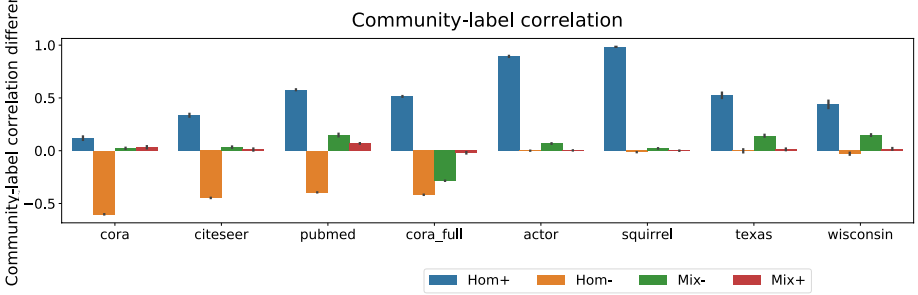

Fig. 6 Community-label correlation (ρ) on manipulated graphs. The bars represent the change in this correlation measure. We see that the change in correlation is high for homophilic graphs and low for non-homophilic graphs, following a pattern similar to homophily in Fig. 1. This measure was computed for the labeled nodes (train and validation sets) of each graph (original and manipulated) with ten different random splits

we expect that it can determine whether GNNs are suitable for classification on a given graph dataset. In the following experiments, we empirically show viability of this measure, and we use it to build a guideline that helps determining GNNs applicability.

Community perturbations

To shed more light on the correlation between ρ and the classification performance of GNNs, we now study the change of both GNN accuracy and the measured community-label correlation ρ on the given datasets after applying additional community perturbations. Particularly, we start with the Mix^{-} networks for each dataset and we perform the following perturbations. We randomly select a fraction of nodes and assign them to different communities by simply swapping the nodes position in the network. Then we gradually increase the fraction[6] of the selected nodes to obtain a spectrum of community-label correlation ρ. Finally, we compute ρ and the accuracy of the GCN model[7] on each of the obtained graphs. We show ρ as a function of the swapping fraction in Fig. 7 and visualize the correlation between ρ and the GCN accuracy in Fig. 8. We choose the Mix^{-} networks for this experiment to guarantee that the node swapping only changes nodes' communities and not their importance (i.e., degree). We expect that these perturbations gradually reduce ρ when the cluster assumption holds, i.e., for homophilic datasets.

We show that ρ decreases and then converges for the homophilic datasets (Fig. 7—top), supporting that the community perturbations decrease the correlation between communities and labels. In these cases, the GNN accuracy has a positive correlation with ρ (Fig. 8—top). However, when community perturbations do not reduce the correlation between communities and labels, ρ is already at a convergence level (Fig. 7—bottom). For these datasets, there is no strong correlation between ρ and the GNN accuracy (see Fig. 8—bottom). That is the case of non-homophilic datasets where GNNs were not

[6] We iterate over fractions in steps of 0.1. For each fraction, we repeat the swapping ten times for different random seeds.

[7] We train the GCN model five times on each obtained graph and report the mean accuracy.

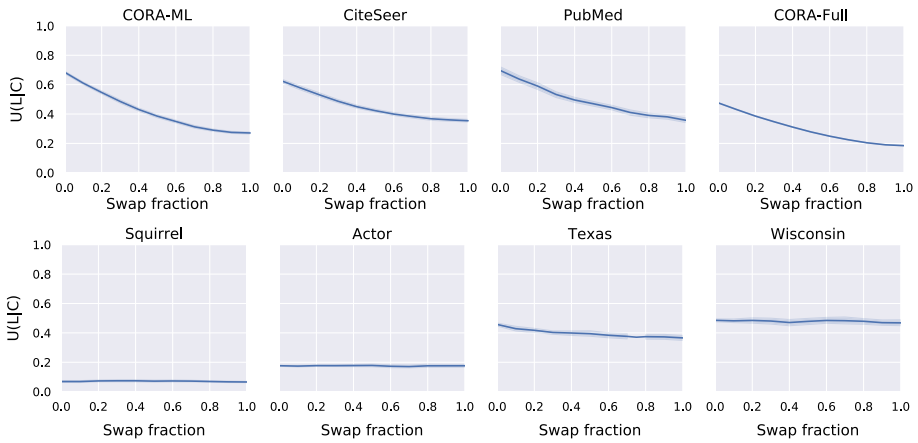

Fig. 7 Community-label correlation with gradual community swapping. The figure shows the calculated ρ for each dataset. The swap fraction represents the fraction of nodes which changed their community. We see a decline in the coefficient with increasing swapping fraction for homophilic datasets (top row). This line is already constant for non-homophilic datasets (bottom row) as the measure is already low in the original graphs

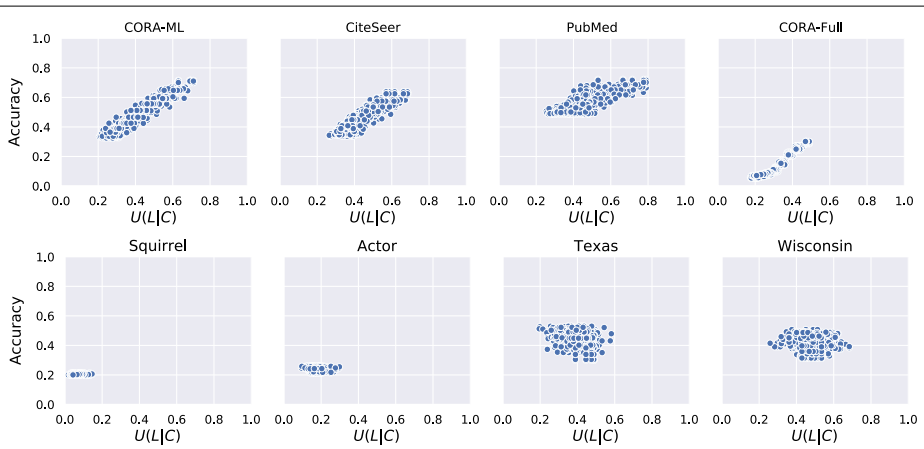

Fig. 8 Community-label correlation impact on GCN accuracy. The figure shows GCN test accuracy on all eight datasets with changing ρ. Each point represents a graph, where each graph is a Mix^{-} graph after applying swapping for a fraction of nodes with a certain random seed for swapping. We see a positive correlation between ρ and the GNN accuracy for homophilic datasets (top row), and no obvious correlation for non-homophilic datasets (bottom row)

able to outperform the feature-only baseline. Our measure shows to reflect the suitability of GNNs for different datasets.

Guideline for application of GNNs

To verify whether a GNN model is applicable on a certain dataset, we suggest use the community-label correlation[8] based on the previous observations. The first step is to

[8] Notice that this measure is only useful to estimate homophily when rare nodes are labeled, e.g., in semi-supervised node classification. When a considerable amount of nodes is labeled, e.g., 20% of nodes, then it is advised to use the homophily measure as explained in "Contributing factors" section

perform community detection on the dataset. Then we suggest to perform gradual community perturbations (as explained above) and inspect the respective community-label correlation. If the value of the coefficient decreases with more perturbations, this supports that the original graph is homophilic, and GNNs are applicable. Otherwise, if it does not decrease, the original graph is most likely non-homophilic in the first place, and feature-only methods are more advisable than GNNs. Computing the correlation should not be an expensive procedure in semi-supervised settings since only a small fraction of nodes are labeled.

Conclusion

In this work, we analyzed the impact of community structures and homophily in graphs on the performance of GNNs in semi-supervised node classification. Homophily and community structure are significant factors that influence how graph data is generated. By studying their impact on GNNs, we gain a deeper understanding of the extents and limitations of state-of-the-art GNN models. By systematically manipulating eight graph datasets, we showed that GNNs outperform a feature-only baseline on this task in case the given graphs are homophilic. Otherwise, GNNs cannot effectively exploit the graph structure for classification. Additionally, we show an analysis on the relation between labels and graph communities. With our analysis, we suggest that when community information does not contribute to classification, it is not advisable to use GNNs for this task. In particular, we show that our measure for the community-label correlation (computed through the uncertainty coefficient) can indicate whether a dataset is homophilic. We further formalize a guideline to select where to apply GNNs based on community-label correlation. Our work serves as a contributing factor to intrinsic validation of the applicability of GNN models, and gives insights on how plausible are current GNNs models. Future work can also investigate how the generation of graphs (broader than homophily) influences GNN performance, and how to build GNN architectures which are intrinsically adaptive to different levels of homophily.

Appendix A: Stochastic block matrix computation for Hom^+

To create L-SBM graph graphs, we use a stochastic block model which is a generative model that produces random graphs with communities identified by a stochastic block matrix and with a Poisson degree distribution. To build the stochastic block matrix, we first partition the nodes into disjoint sets based on their labels $\{V_1, V_2, \ldots, V_{|\mathcal{L}|}\}$, where $\mathcal{L}$ is the set of labels. We build a vector of block sizes reciprocals $\eta = [\frac{1}{|V_1|}, \frac{1}{|V_2|}, \ldots, \frac{1}{|V_{|\mathcal{L}|}|}]^T$ to use for normalization. We build the stochastic block matrix M as follows

$$M := \frac{1}{2}\bar{d_{in}}\text{diag}(\eta) + \frac{1}{n}\eta \cdot \eta^T, \tag{5}$$

where $\bar{d_{in}}$ is the average node in-degree in the graph. We use M with the block sizes to build a synthetic graph using the stochastic block model.

Appendix B: Stochastic block matrix computation for *Mix*$^-$

The aim of this manipulation is to preserve community structure while eliminating the degree distribution. To create these graphs, we use a stochastic block model after performing community detection as follows.

Community detection

In order to achieve this, we first assigned a community id to each node using the Louvain method for community detection. With the resulting communities we were able to produce a stochastic matrix which served as an input to the SBM.

Building the stochastic block matrix

We first calculate edge density within each community and then between each pair of different communities. These edge densities serve as edge probabilities between each pair of nodes in the resulting stochastic matrix M. To increase the intra-community edge density, we multiply each value $M_{i,i}$ in M by 1.5 (clipped from the top so no value exceeds 1). To decrease the inter-community edge density, we multiply each value $M_{i,j} : i \neq j$ in M by 0.5. We finally use matrix M for the stochastic block model, which results in a graph with reduced community mixing.

Appendix C: Model settings

Common settings

For all the GNN architectures, we use a model of two hidden layer, with the second layer as the output layer. The first hidden layer for each GNN architecture uses the activation that was used in the corresponding paper. The second hidden layer has the size of the number of labels in the respective dataset, and uses softmax as an activation function. For the training process, we use the negative log likelihood as our loss function. We use Adam optimizer for 200 epochs, with early stopping after the validation accuracy has not improved for 10 consecutive epochs.[9] Then we select the state of the model at the epoch where the highest validation accuracy was achieved.[10]

Hyperparameter search

We perform the following grid search to find the hyperparameter setting which maximizes the mean validation accuracy over 10 random splits with 10 random model initialization.

- First hidden layer size: [12, 24, 48, 96]
- Learning rate: [0.001, 0.005, 0.01]
- Dropout probability: [0.2, 0.4, 0.6, 0.8]
- Regularization weight: [0.0001, 0.001, 0.01, 0.1]
- Attention dropout probability for GAT: [0.2, 0.3, 0.4, 0.6, 0.8]
- Attention heads for GAT's first hidden layer: [2, 4, 8]

[9] The training typically stops before the 50-th epoch.

[10] We implement our models and evaluations using PyTorch Geometric https://pytorch-geometric.readthedocs.io/.

After grid search, the chosen hidden layer size for all models turned out to be 96. Table 5 shows the hyperparameters used after grid search except for GAT which is shown in Table 4.

Table 4 Chosen hyperparameters for GAT model

Dataset	Dropout	Learning rate	Regularization weight	Heads	Attention dropout
CORA-ML	0.4	0.01	0.0100	4	0.2
Citeseer	0.4	0.01	0.0100	4	0.2
Pubmed	0.2	0.01	0.0001	8	0.3
CORA-Full	0.2	0.01	0.0001	8	0.3
Actor	0.4	0.01	0.0100	4	0.3
Squirrel	0.4	0.01	0.0001	2	0.2
Texas	0.4	0.01	0.0100	4	0.3
Wisconsin	0.4	0.01	0.0100	4	0.3

Table 5 Chosen hyperparameters for all GNN models except GAT

	Dataset	Dropout	Learning rate	Regularization weight
GCN	CORA-ML	0.8	0.010	0.0100
	Citeseer	0.8	0.010	0.0100
	Pubmed	0.8	0.010	0.0100
	CORA-Full	0.2	0.005	0.0010
	Actor	0.8	0.010	0.0100
	Squirrel	0.4	0.010	0.0100
	Texas	0.8	0.010	0.0100
	Wisconsin	0.8	0.010	0.0100
SAGE	CORA-ML	0.8	0.010	0.0100
	Citeseer	0.8	0.010	0.0100
	Pubmed	0.8	0.010	0.0100
	CORA-Full	0.8	0.005	0.0010
	Actor	0.8	0.010	0.0100
	Squirrel	0.4	0.005	0.1000
	Texas	0.8	0.010	0.0100
	Wisconsin	0.8	0.010	0.0100
APPNP	CORA-ML	0.8	0.010	0.0100
	Citeseer	0.8	0.010	0.0100
	Pubmed	0.8	0.010	0.0100
	CORA-Full	0.2	0.005	0.0001
	Actor	0.8	0.010	0.0100
	Squirrel	0.2	0.010	0.0010
	Texas	0.8	0.010	0.0100
	Wisconsin	0.8	0.010	0.0100

Table 5 (continued)

	Dataset	Dropout	Learning rate	Regularization weight
SGC	CORA-ML	0.8	0.010	0.0100
	Citeseer	0.8	0.010	0.0100
	Pubmed	0.8	0.010	0.0100
	CORA-Full	0.2	0.005	0.0010
	Actor	0.8	0.010	0.0100
	Squirrel	0.4	0.010	0.0100
	Texas	0.8	0.010	0.0100
	Wisconsin	0.8	0.010	0.0100
CGCN	CORA-ML	0.8	0.001	0.1000
	Citeseer	0.8	0.001	0.1000
	Pubmed	0.6	0.010	0.0001
	CORA-Full	0.8	0.001	0.1000
	Actor	0.6	0.010	0.1000
	Squirrel	0.6	0.005	0.1000
	Texas	0.8	0.010	0.1000
	Wisconsin	0.6	0.005	0.1000

Appendix D: Homophily and mixing impact on GNN accuracy

We visualize the accuracy of different GNN models as a function of homophily and mixing in Figs. 9, 10, 11, 12, and 13 in the same manner done for GCN in Fig. 4.

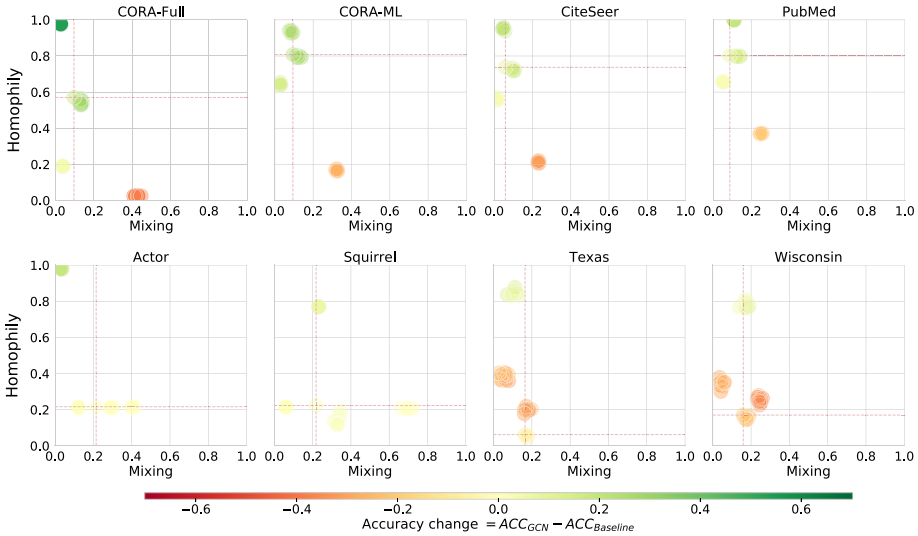

Fig. 9 Homophily and mixing impact on SAGE accuracy

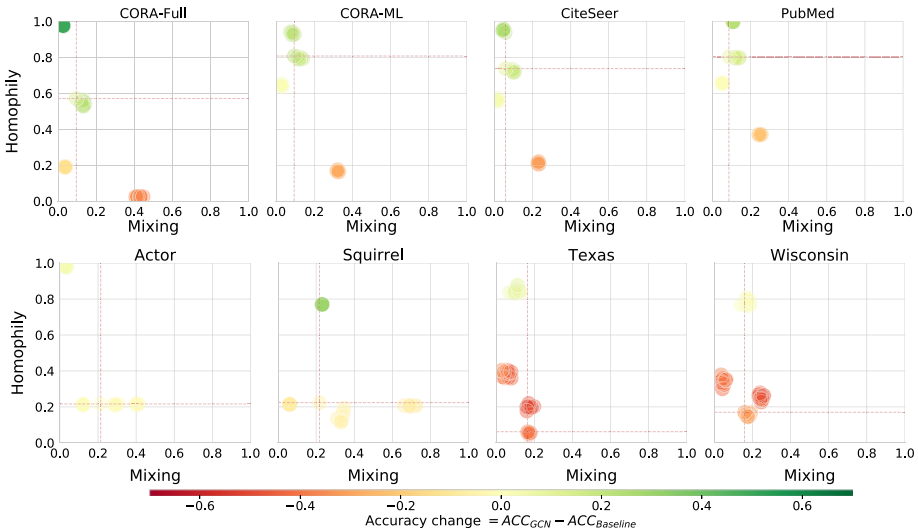

Fig. 10 Homophily and mixing impact on GAT accuracy

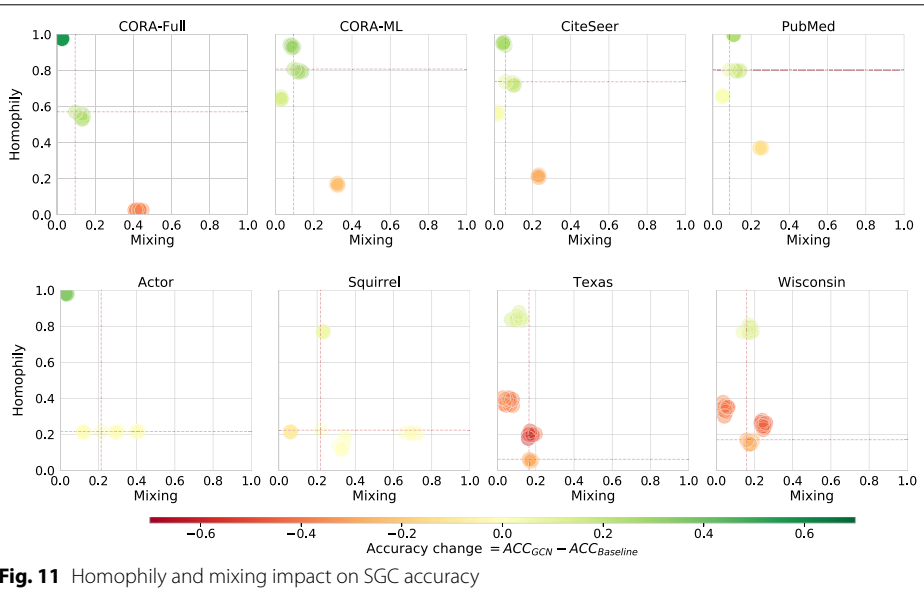

Fig. 11 Homophily and mixing impact on SGC accuracy

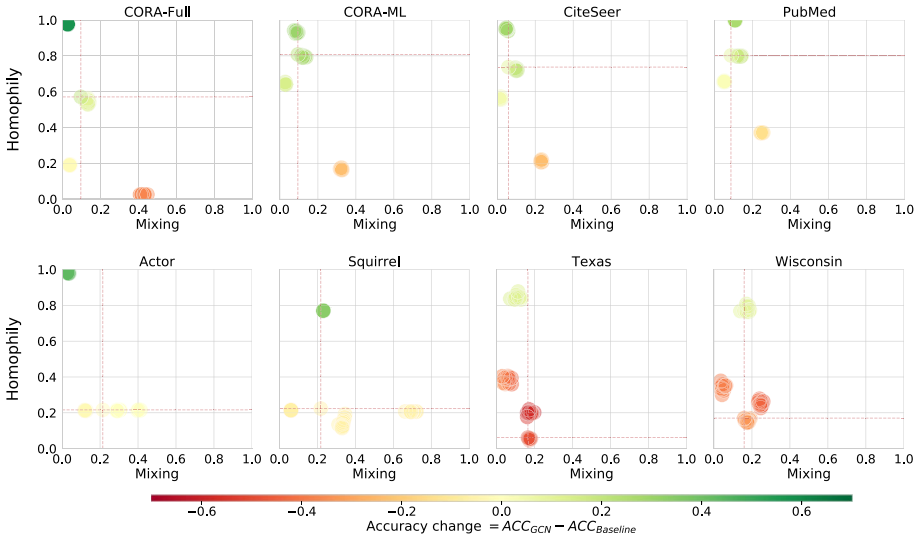

Fig. 12 Homophily and mixing impact on APPNP accuracy

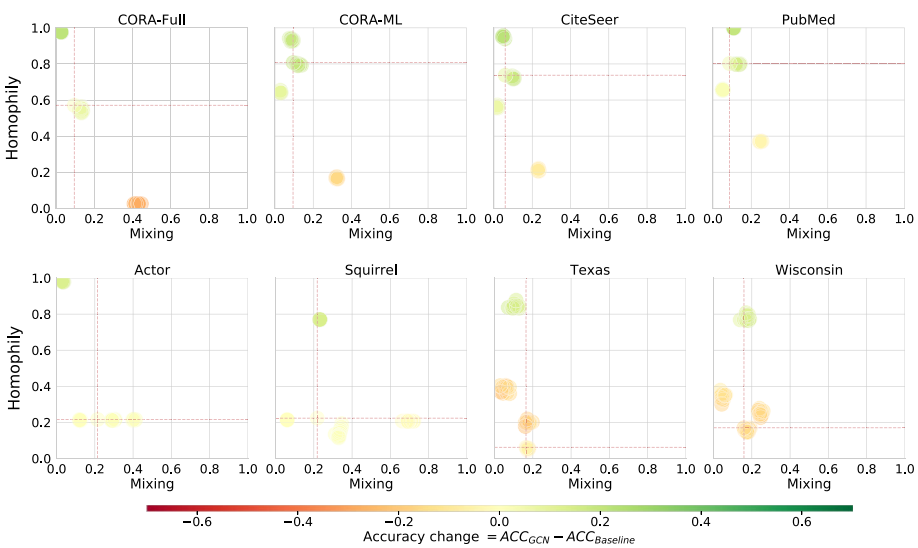

Fig. 13 Homophily and mixing impact on CGCN accuracy

Abbreviations
GNN: Graph neural network.

Acknowledgements
We thank the anonymous reviewers for their valuable feedback on the manuscript. This work is supported by the H2020 project TRUSTS (GA: 871481) and the "DDAI" COMET Module within the COMET Program, funded by the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry for Digital and Economic Affairs (bmdw), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia.

Authors' contributions
HH, TD, EL, DH and RK designed the study. HH and TD implemented the data processing, training and evaluation experiments. HH, TD, EL, DH and RK wrote, reviewed and approved the manuscript. All authors read and approved the final manuscript.

Funding
The Know-Center is funded within the Austrian COMET Program—Competence Centers for Excellent Technologies—under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

Availability of data and materials
The code repository for this paper can be found at https://github.com/sqrhussain/structure-in-gnn/. The datasets used in this study can be found through their corresponding links (included in the repository at https://github.com/sqrhussain/structure-in-gnn/blob/master/src/data/dataset_handle.py).

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Institute of Interactive Systems and Data Science, Graz University of Technology, Inffeldgasse 13, 8010 Graz, Austria.
[2]Know-Center GmbH, Graz, Austria.

Received: 12 April 2021 Accepted: 7 September 2021
Published online: 26 October 2021

References

Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech: Theory Exp 2008(10):10008
Chapelle O, Scholkopf B, Zien A (2009) Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. IEEE Trans Neural Netw 20(3):542
Cherifi H, Palla G, Szymanski BK, Lu X (2019) On community structure in complex networks: challenges and opportunities. Appl Netw Sci 4(1):1–35
Chiang W-L, Liu X, Si S, Li Y, Bengio S, Hsieh C-J (2019) Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, pp 257–266
Craven M, DiPasquo D, Freitag D, McCallum A, Mitchell T, Nigam K, Slattery S (1998) Learning to extract symbolic knowledge from the world wide web. In: Proceedings of the national conference on artificial intelligence, pp 509–516
Dev P (2016) Homophily and community structure in networks. J Public Econ Theory 18(2):268–290
Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp 1024–1034
Hasani-Mavriqi I, Kowald D, Helic D, Lex E (2018) Consensus dynamics in online collaboration systems. Comput Soc Netw 5(1):2
Holland PW, Laskey KB, Leinhardt S (1983) Stochastic blockmodels: first steps. Social networks 5(2):109–137
Hussain H, Duricic T, Lex E, Kern R, Helic D (2020) On the impact of communities on semi-supervised classification using graph neural networks. In: International conference on complex networks and their applications. Springer, Berlin, pp 15–26
Karrer B, Newman ME (2011) Stochastic blockmodels and community structure in networks. Phys Rev E 83(1):016107
Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations (ICLR)
Klicpera J, Bojchevski A, Günnemann S (2019) Predict then propagate: graph neural networks meet personalized PageRank. In: 7th international conference on learning representations, ICLR 2019. arXiv:1810.05997
Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: 32nd AAAI conference on artificial intelligence, AAAI 2018. arXiv:1801.07606
Newman ME (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256

Pei H, Wei B, Chang KC-C, Lei Y, Yang B (2019) Geom-GCN: geometric graph convolutional networks. In: International conference on learning representations
Peixoto TP (2021) Disentangling homophily, community structure and triadic closure in networks. arXiv preprint arXiv:2101.02510
Press WH, William H, Teukolsky SA, Vetterling WT, Saul A, Flannery BP (2007) Numerical recipes 3rd edition: the art of scientific computing. Cambridge University Press, Cambridge
Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93
Shchur O, Mumme M, Bojchevski A, Günnemann S (2018) Pitfalls of graph neural network evaluation. In: Relational representation learning workshop, NeurIPS 2018
Tang J, Sun J, Wang C, Yang Z (2009) Social influence analysis in large-scale networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 807–816
Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: International conference on learning representations
Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2019) A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596
Wu F, Zhang T, Souza Jr AHD, Fifty C, Yu T, Weinberger KQ (2019) Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153
Xu K, Jegelka S, Hu W, Leskovec J (2019) How powerful are graph neural networks? In: 7th International conference on learning representations, ICLR 2019. arXiv:1810.00826
Zhang M, Chen Y (2018) Link prediction based on graph neural networks. In: Advances in neural information processing systems, pp 5165–5175
Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 912–919
Zhu J, Yan Y, Zhao L, Heimann M, Akoglu L, Koutra D (2020) Beyond homophily in graph neural networks: current limitations and effective designs. Adv Neural Inform Process Syst 33

Publisher's Note