Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many widely used datasets for graph machine learning tasks have generally been homophilous, where nodes with similar labels connect to each other. Recently, new Graph Neural Networks (GNNs) have been developed that move beyond the homophily regime; however, their evaluation has often been conducted on small graphs with limited application domains. We collect and introduce diverse non-homophilous datasets from a variety of application areas that have up to 384x more nodes and 1398x more edges than prior datasets. We further show that existing scalable graph learning and graph minibatching techniques lead to performance degradation on these non-homophilous datasets, thus highlighting the need for further work on scalable non-homophilous methods. To address these concerns, we introduce LINKX — a strong simple method that admits straightforward minibatch training and inference. Extensive experimental results with representative simple methods and GNNs across our proposed datasets show that LINKX achieves state-of-the-art performance for learning on non-homophilous graphs.

1 Introduction

Graph learning methods generate predictions by leveraging complex inductive biases captured in the topology of the graph [6]. A large volume of work in this area, including graph neural networks (GNNs), exploits *homophily* as a strong inductive bias, where connected nodes tend to be similar to each other in terms of labels [39, 3]. Such assumptions of homophily, however, do not always hold true. For example, malicious node detection, a key application of graph machine learning, is known to be non-homophilous in many settings [43, 10, 22]. In particular, fraudsters in the Facebook Social Graph may seek to fool homophilous algorithms by deliberately connecting to normal users [9].

Further, while new GNNs that work better in these non-homophilous settings have been developed [66, 37, 65, 14, 12, 57, 30, 29, 7, 42], their evaluation is limited to a few graph datasets introduced by Pei et al. [46] that have certain undesirable properties such as small size, narrow range of application areas, and high variance between different train/test splits [66]. Consequently, method scalability has not been thoroughly studied in non-homophilous graph learning. In fact, many non-homophilous techniques frequently require more parameters and computational resources [66, 1, 14], which is neither evident nor detrimental when they are evaluated on very small datasets. Even though scalable graph learning techniques do exist, these methods generally cannot be directly applied to the non-homophilous setting, as they oftentimes assume homophily in their construction [55, 26, 17, 8].

Non-homophily in graphs also degrades proven graph learning techniques that have been instrumental to strong performance in scalable graph learning. For instance, label propagation, personalized PageRank, and low-pass graph filtering have been used for scalable graph representation learning models, but these methods all assume homophily [55, 26, 17, 8]. Moreover, we give empirical evidence that existing minibatching techniques in graph learning [13, 61] significantly degrade

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

performance in non-homophilous settings. In response, we develop a novel model, LINKX, that addresses these concerns; LINKX outperforms existing graph learning methods on large-scale non-homophilous datasets and admits a simple minibatching procedure that maintains strong performance.

To summarize, we demonstrate three key areas of deficiency as mentioned above, namely: (1) that there is a lack of large, high-quality datasets covering different non-homophilous applications, (2) that current graph minibatching techniques and scalable methods do not work well in non-homophilous settings, and (3) that prior non-homophilous methods are not scalable. To these ends, this paper makes the following contributions:

Dataset Collection and Benchmarking. We collect a diverse series of large, non-homophilous graph datasets and define new node features and tasks for classification. These datasets are substantially larger than previous non-homophilous datasets, span wider application areas, and capture different types of complex label-topology relationships. With these proposed datasets, we conduct extensive experiments with 14 graph learning methods and 3 graph minibatching techniques that are broadly representative of the graph machine learning model space.

Analyzing Scalable Methods and Minibatching. We analyze current graph minibatching techniques like GraphSAINT [61] in non-homophilous settings, showing that they substantially degrade performance in empirical settings. Also, we show empirically that scalable methods for graph learning like SGC and C&S [55, 26] do not perform well in non-homophilous settings — even though they achieve state-of-the-art results on many homophilous graph benchmarks. Finally, we demonstrate that existing non-homophilous methods often suffer from issues with scalability and/or performance in large non-homophilous graphs, to a great extent due to a lack of study of large-scale non-homophilous graph learning.

LINKX: a strong, simple method. We propose a simple method LINKX that achieves excellent results for non-homophilous graphs while overcoming the above-mentioned minibatching issues. LINKX works by separately embedding the adjacency $\mathbf{A}$ and node features $\mathbf{X}$, then combining them with multilayer perceptrons and simple transformations, as illustrated in Figure 1. It generalizes node feature MLP and LINK regression [63], two baselines that often work well on non-homophilous graphs. This method is simple to train and evaluate in a minibatched fashion, and does not face the performance degradation that other methods do in the minibatch setting. We develop the model and give more details in Section 4.

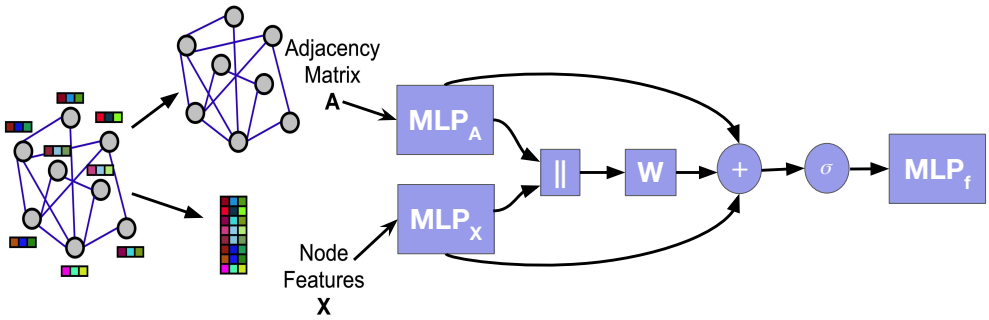

Figure 1: Our model LINKX separately embeds node features and adjacency information with MLPs, combines the embeddings together by concatenation, then uses a final MLP to generate predictions.

2 Prior Work

Graph Representation Learning. Graph neural networks [24, 32, 54] have demonstrated their utility on a variety of graph machine learning tasks. Most GNNs are constructed by stacking layers that propagate transformed node features, which are then aggregated via different mechanisms. The neighborhood aggregation used in many existing GNNs implicitly leverage homophily, so they often fail to generalize on non-homophilous graphs [66, 5]. Indeed, a wide range of GNNs operate as low-pass graph filters [41, 55, 5] that smooth features over the graph topology, which produces similar representations and thus similar predictions for neighboring nodes.

Scalable methods. A variety of scalable graph learning methods have been developed for efficient computation in larger datasets [61, 13, 59, 24, 55, 26, 17, 8]. Many of these methods explicitly make use of an assumption of homophily in the data [55, 26, 17, 8]. By leveraging this assumption, several simple, inexpensive models are able to achieve state-of-the-art performance on homophilic datasets [55, 26]. However, these methods are unable to achieve comparable performance in non-homophilous settings, as we show empirically in Section 5.

Graph sampling. As node representations depend on other nodes in the graph, there are no simple minibatching techniques in graph learning as there are for i.i.d. data. To scale to large graphs, one line of work samples nodes that are used in each layer of a graph neural network [24, 59, 11]. Another family of methods samples subgraphs of an input graph, then passes each subgraph through a GNN to make a prediction for each node of the subgraph [13, 60, 61]. While these methods are useful for scalable graph learning, we show that they substantially degrade performance in our non-homphilous experiments (see Section 5).

Non-Homophilous methods. Various GNNs have been proposed to achieve higher performance in low-homophily settings [66, 37, 65, 14, 12, 57, 30, 29]. Geom-GCN [46] introduces a geometric aggregation scheme, MixHop [1] proposes a graph convolutional layer that mixes powers of the adjacency matrix, GPR-GNN [14] features learnable weights that can be positive and negative in feature propagation, GCNII [12] allows deep graph convolutional networks with relieved oversmoothing, which empirically performs better in non-homophilous settings, and H_2GCN [66] shows that separation of ego and neighbor embeddings, aggregation in higher-order neighborhoods, and the combination of intermediate representations improves GNN performance in low-homophily.

There are several recurring design decisions across these methods that appear to strengthen performance in non-homophilous settings: using higher-order neighborhoods, decoupling neighbor information from ego information, and combining graph information at different scales [66]. Many of these design choices require additional overhead (see Section 4.3), thus reducing their scalability.

Datasets. The widely used citation networks Cora, Citeseer, and Pubmed [49, 58] are highly homophilous (see Appendix A) [66]. Recently, the Open Graph Benchmark [25] has provided a series of datasets and leaderboards that improve the quality of evaluation in graph representation learning; however, most of the node classification datasets tend to be homophilous, as noted in past work [66] and expanded upon in Appendix A.2. A comparable set of high-quality benchmarks to evaluate non-homophilous methods does not currently exist.

3 Datasets for Non-Homophilous Graph Learning

3.1 Currently Used Datasets

The most widely used datasets to evaluate non-homophilous graph representation learning methods were presented by Pei et al. [46] (see our Table 1); however, these datasets have fundamental issues. First, they are very small — the Cornell, Texas, and Wisconsin datasets have between 180-250 nodes, and the largest dataset Actor has 7,600 nodes. In analogy to certain pitfalls of graph neural network evaluation on small (homophilic) datasets discussed in [50], evaluation on the datasets of Pei et al. [46] is plagued by high variance across different train/test splits (see results in [66]). The small size of these datasets may tend to create models that are more prone to overfitting [18], which prevents the scaling up of GNNs designed for non-homophilous settings.

Peel [45] also studies node classification on network datasets with various types of relationships between edges and labels. However, they only study methods that act on node labels, and thus their datasets do not necessarily have node features. We take inspiration from their work, by testing on Pokec and Facebook networks with node features that we define, and by introducing other year-prediction tasks on citation networks that have node features.

3.2 Proposed Datasets

Here, we detail the non-homophilous datasets that we propose for graph machine learning evaluation. These datasets come from a variety of contexts, and some have been used for evaluation of graph machine learning models in past work; we make adjustments such as modifying node labels and adding node features that allow for evaluation of GNNs in non-homophilous settings. We define

node features for Pokec, genius, and snap-patents, and we also define node labels for arXiv-year, snap-patents, and genius. Additionally, we crawl and clean the large-scale wiki dataset — a new Wikipedia dataset where the task is to predict page views, which is non-homophilous with respect to the graph of articles connected by links between articles (see Appendix D.1). This wiki dataset has 1,925,342 nodes and 303,434,860 edges, so training and inference require a large amount of computational resources and scalable algorithmic techniques like minibatching.

Basic dataset statistics are given in Table 2. Note the substantial difference between the size of our datasets and those of Pei et al. [46] in Table 1; our datasets have up to 384x more nodes and 1398x more edges. We report the standard edge homophily measure and a new measure $\hat{h}$ of the presence of homophily that we define to alleviate certain weaknesses of edge homophily; further details of the measures are given in Appendix Section A.1. Both measures take values in $[0, 1]$ and are higher if the graph is homophilous. These measures along with the lower empirical performance of homophily-assuming models (Section 5) and examination of compatibility matrices (Appendix A) show that our datasets are indeed non-homophilous. Our proposed datasets are as follows:

Penn94 [53] is a friendship network from the Facebook 100 networks of university students from 2005, where nodes represent students. Each node is labeled with the reported gender of the user. The node features are major, second major/minor, dorm/house, year, and high school.

Pokec [34] is the friendship graph of a Slovak online social network, where nodes are users and edges are directed friendship relations. Nodes are labeled with reported gender. We derive node features from profile information, such as geographical region, registration time, and age.

arXiv-year [25] is the ogbn-arXiv network with different labels. Our contribution is to set the class labels to be the year that the paper is posted, instead of paper subject area. The nodes are arXiv papers, and directed edges connect a paper to other papers that it cites. The node features are averaged word2vec token features of both the title and abstract of the paper. The five classes are chosen by partitioning the posting dates so that class ratios are approximately balanced.

snap-patents [35, 34] is a dataset of utility patents in the US. Each node is a patent, and edges connect patents that cite each other. Node features are derived from patent metadata. Our contribution is to set the task to predict the time at which a patent was granted, resulting in five classes.

genius [36] is a subset of the social network on genius.com — a site for crowdsourced annotations of song lyrics. Nodes are users, and edges connect users that follow each other on the site. This social network has not been used for node classification in the literature, so we define the task of predicting certain marks on the accounts. About 20% of users in the dataset are marked "gone" on the site, which appears to often include spam users. Thus, we predict whether nodes are marked. The node features are user usage attributes like the Genius assigned expertise score, counts of contributions, and roles held by the user.

twitch-gamers [48] is a connected undirected graph of relationships between accounts on the streaming platform Twitch. Each node is a Twitch account, and edges exist between accounts that are mutual followers. The node features include number of views, creation and update dates, language, life time, and whether the account is dead. The binary classification task is to predict whether the channel has explicit content.

wiki is a dataset of Wikipedia articles, where nodes represent pages and edges represent links between them. We collect this new dataset, with a process that we describe further in Appendix D.1. Node features are constructed using averaged title and abstract GloVe embeddings [47]. Labels represent total page views over 60 days, which are partitioned into quintiles to make five classes.

Note that our proposed datasets are substantially larger than the previous ones, containing up to about 3M nodes (snap-patents) and 300M edges (wiki). As there is little study in large-scale non-homophilous graph learning, this strongly motivates the need for developing a new scalable approach that can accurately learn on non-homophilous graphs.

4 LINKX: A New Scalable Model

In this section, we introduce our novel model, LINKX, for scalable node classification in non-homophilous settings. LINKX is built out of multilayer perceptrons (MLPs) and linear transformations, thus making it simple and scalable. It also admits simple row-wise minibatching procedures

Table 1: Statistics for previously used datasets from Pei et al. [46]. #C is the number of node classes. The highest number of nodes or edges overall are bolded.

Dataset	# Nodes	# Edges	# Feat.	# C	Context	Edge hom.	$\hat{h}$ (ours)
Chameleon	2,277	36,101	2,325	5	Wiki pages	.23	.062
Cornell	183	295	1,703	5	Web pages	.30	.047
Actor	**7,600**	29,926	931	5	Actors in movies	.22	.011
Squirrel	5,201	**216,933**	2,089	5	Wiki pages	.22	.025
Texas	183	309	1,703	5	Web pages	.11	.001
Wisconsin	251	499	1,703	5	Web pages	.21	.094

Table 2: Statistics of our proposed non-homophilous graph datasets. # C is the number of distinct node classes. Note that our datasets come from more diverse applications areas and are much larger than those shown in Table 1, with up to 384x more nodes and 1398x more edges.

Dataset	# Nodes	# Edges	# Feat.	# C	Class types	Edge hom.	$\hat{h}$ (ours)
Penn94	41,554	1,362,229	5	2	gender	.470	.046
pokec	1,632,803	30,622,564	65	2	gender	.445	.000
arXiv-year	169,343	1,166,243	128	5	pub year	.222	.272
snap-patents	**2,923,922**	13,975,788	269	5	time granted	.073	.100
genius	421,961	984,979	12	2	marked act.	.618	.080
twitch-gamers	168,114	6,797,557	7	2	mature content	.545	.090
wiki	1,925,342	**303,434,860**	600	5	views	.389	.107

that allow it to perform well on large non-homophilous graphs. As a result, LINKX is able to circumvent aforementioned issues of graph minibatching and non-homophilous GNNs in large-scale non-homophilous settings.

4.1 Motivation from two simple baselines

Here, we detail two simple baselines for node classification that we build off of to develop LINKX.

MLP on node features. A naïve method for node classification is to ignore the graph topology and simply train an MLP on node features. For the same reason that the graph topology has more complicated relationships with label distributions in non-homophilous graphs, many GNNs are not able to effectively leverage the graph topology in these settings. Thus, MLPs can actually perform comparatively well on non-homophilous graphs — achieving higher or approximately equal performance to various GNNs [66].

LINK regression on graph topology. On the other extreme, there is LINK [63] — a simple baseline that only utilizes graph topology. In particular, we consider LINK regression, which trains a logistic regression model in which each node's features are taken from a column of the adjacency matrix. Letting $\mathbf{A} \in \{0,1\}^{n \times n}$ be the binary adjacency matrix of the graph, and $\mathbf{W} \in \mathbb{R}^{c \times n}$ be a learned weight matrix, LINK computes class probabilities as

$$\mathbf{Y} = \mathrm{softmax}(\mathbf{WA}). \tag{1}$$

Let $u \in \{1, \ldots, n\}$ be a specific node, and let $k \in \{1, \ldots, c\}$ be a specific class. Then, expanding the matrix multiplication, the log-odds of node u belonging to class k is given by

$$(\mathbf{WA})_{ku} = \sum_{v \in \mathcal{N}(u)} \mathbf{W}_{kv}, \tag{2}$$

where $\mathcal{N}(u)$ contains the 1-hop neighbors of u. In other words, the logit is given by the sum of weights $\mathbf{W}_{kv}$ across the 1-hop neighbors of u. If a specific node v has many neighbors of class k, then $\mathbf{W}_{kv}$ is probably large, as we would expect with a high probability that any neighbor of v is of class k. In this sense, LINK is like a 2-hop method: for a given node u, the probability of being in a given class is related to the class memberships of u's 2-hop neighbors in $\mathcal{N}(v)$ for each neighbor $v \in \mathcal{N}(u)$. Related interpretations of LINK as a method acting on 2-hop paths between nodes are given by Altenburger and Ugander [3].

Though it is simple and has been overlooked in the recent non-homophilous GNN literature, LINK has been found to perform well in certain node classification tasks like gender prediction in social networks [3, 4]. A major reason why LINK does well in many settings is exactly because it acts as a 2-hop method. For example, while 1-hop neighbors are often not so informative for gender prediction in social networks due to lack of homophily, 2-hop neighbors are very informative due to so-called "monophily," whereby many nodes have extreme preferences for connecting to a certain class [3]. Beyond just gender prediction, we show in Section 5 that LINK empirically outperforms many models across the various application areas of the non-homophilous datasets we propose.

4.2 LINKX

We combine these two simple baselines through simple linear transformations and component-wise nonlinearities. Let $\mathbf{X} \in \mathbb{R}^{D \times n}$ denote the matrix of node features with input dimension D, and let $[\mathbf{h}_1; \mathbf{h}_2]$ denote concatenation of vectors $\mathbf{h}_1$ and $\mathbf{h}_2$. Then our model outputs predictions $\mathbf{Y}$ through the following mapping:

$$\mathbf{h}_{\mathbf{A}} = \mathrm{MLP}_{\mathbf{A}}(\mathbf{A}) \in \mathbb{R}^{d \times n} \tag{3}$$

$$\mathbf{h}_{\mathbf{X}} = \mathrm{MLP}_{\mathbf{X}}(\mathbf{X}) \in \mathbb{R}^{d \times n} \tag{4}$$

$$\mathbf{Y} = \mathrm{MLP}_f\left(\sigma\left(\mathbf{W}[\mathbf{h}_{\mathbf{A}}; \mathbf{h}_{\mathbf{X}}] + \mathbf{h}_{\mathbf{A}} + \mathbf{h}_{\mathbf{X}}\right)\right), \tag{5}$$

in which d is the hidden dimension, $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ is a weight matrix, and σ is a component-wise nonlinearity (which we take to be ReLU). We call our model LINKX, as it extends LINK with node feature information from the matrix $\mathbf{X}$. A diagram of LINKX is given in Figure 1.

First, LINKX computes hidden representations $\mathbf{h}_{\mathbf{A}}$ of the adjacency (extending LINK) and $\mathbf{h}_{\mathbf{X}}$ of the feature matrix (as in node-feature MLPs). Then it combines these hidden representations through a linear transform $\mathbf{W}$ of their concatenation, with skip connections that add back in $\mathbf{h}_{\mathbf{A}}$ and $\mathbf{h}_{\mathbf{X}}$ to better preserve pure adjacency or node feature information. Finally, it puts this combined representation through a non-linearity and another MLP to make a prediction.

Separating then mixing adjacency and feature information. LINKX separately embeds the adjacency $\mathbf{A}$ to $\mathbf{h}_{\mathbf{A}}$ and the features $\mathbf{X}$ into $\mathbf{h}_{\mathbf{X}}$ before mixing them for a few reasons. First, we note that this design is reminiscent of fusion architectures in multimodal networks, where data from different modalities are processed and combined in a neural network [21, 62]. In our setting, we can view adjacency information and node feature information as separate modalities. Since node feature MLPs and LINK do well independently on different datasets, this allows us to preserve their individual performance if needed. Ignoring $\mathbf{h}_{\mathbf{X}}$ information is similar to just using LINK, and ignoring $\mathbf{h}_{\mathbf{A}}$ information is just using an node feature MLP. Still, to preserve the ability to just learn a similar mapping to LINK or to a node feature MLP, we find that having the additive skip connections helps to get performance at least as good as either baseline. Our initial empirical results showed that simply concatenating adjacency and node features as input to a network does worse overall empirically (see Appendix C.1).

There are also computational benefits to our design choices. Embedding $\mathbf{A}$ is beneficial for depth as adding more layers to the MLPs only gives an $\mathcal{O}(d^2)$ cost — depending only on the hidden dimension d — and thus does not scale in the number of edges $|E|$ as when adding layers to message-passing GNNs. This is because the graph information in $\mathbf{A}$ is already compressed to hidden feature vectors after the first linear mapping of $\mathrm{MLP}_{\mathbf{A}}$, and we do not need to propagate along the graph in later steps. Moreover, this enables a sparse-dense matrix product to compute the first linear mapping of $\mathrm{MLP}_{\mathbf{A}}$ on $\mathbf{A}$, which greatly increases efficiency as $\mathbf{A}$ is typically very sparse for real-world graphs. Separate embeddings are key here, as this would not be possible if we for instance concatenated $\mathbf{A}$ and $\mathbf{X}$ when $\mathbf{X}$ is large and dense.

Simple minibatching. Message-passing GNNs must take graph topology into account when minibatching with techniques such as neighbor sampling, subgraph sampling, or graph partitioning. However, LINKX does not require this, as it utilizes graph information solely through defining adjacency matrix columns as features. Thus, we can train LINKX with standard stochastic gradient descent variants by taking i.i.d. samples of nodes along with the corresponding columns of the adjacency and feature matrix as features. This is much simpler than the graph minibatching procedures for message-passing GNNs, which require specific hyperparameter choices, have to avoid exponential

blowup of number of neighbors per layer, and are generally more complex to implement [61]. In Section 5.3, we use the simple LINKX minibatching procedure for large-scale experiments that show that LINKX with this minibatching style outperforms GNNs with graph minibatching methods. This is especially important on the scale of the wiki dataset, where none of our tested methods — other than MLP — is capable of running on a Titan RTX GPU with 24 GB GPU RAM (see Section 5).

4.3 Complexity Analysis

Using the above notation, a forward pass of LINKX has a time complexity of $\mathcal{O}\left(d|E| + nd^2L\right)$, in which d is the hidden dimension (which we assume to be on the same order as the input feature dimension D), L is the number of layers, n is the number of nodes, and $|E|$ is the number of edges. We require a $\mathcal{O}(d|E|)$ cost for the first linear mapping of $\mathbf{A}$ and a $\mathcal{O}(d^2)$ cost per layer for MLP operations on hidden features, for L total layers and each of n nodes.

As mentioned above, message passing GNNs have to propagate using the adjacency in each layer, so they have an $L|E|$ term in the complexity. For instance, an L-layer GCN [32] with d hidden dimensions has $\mathcal{O}(dL|E| + nd^2L)$ complexity, as it costs $\mathcal{O}(d|E|)$ to propagate features in each layer, and $\mathcal{O}(nd^2)$ to multiply by the weight matrix in each layer.

Non-homophilous methods often make modifications to standard architectures that increase computational cost, such as using higher-order neighborhoods or using additional hidden embeddings [66]. For instance, the complexity of MixHop [1] is $\mathcal{O}(K(dL|E| + nd^2L))$, which has an extra factor K that is the number of adjacency powers to propagate with. The complexity of GCNII [12] is asymptotically the same as that of GCN, but in practice it requires more computations per layer due to residual connections and linear combinations, and it also often achieves best performance with a large number of layers L. H_2GCN [66] is significantly more expensive due to its usage of strict two-hop neighborhoods, which requires it to form the squared adjacency $\mathbf{A}^2$. This makes the memory requirements intractable even for medium sized graphs (see Section 5).

5 Experiments

We conduct two sets of experiments for node classification on our proposed non-homophilous datasets. One set of experiments does full batch gradient descent training for all applicable methods. This of course limits the size of each model, as the large datasets require substantial GPU memory to train on. Our other set of experiments uses minibatching methods. As all graph-based methods run out of memory on the wiki dataset, even on 24 GB GPUs, we only include wiki results in the minibatching section. In all settings, we find that our LINKX model matches or outperforms the representative methods that we compare against.

Table 3: Experimental results. Test accuracy is displayed for most datasets, while genius displays test ROC AUC. Standard deviations are over 5 train/val/test splits. The three best results per dataset are highlighted. (M) denotes some (or all) hyperparameter settings run out of memory.

	Penn94	pokec	arXiv-year	snap-patents	genius	twitch-gamers
MLP	73.61 ± 0.40	62.37 ± 0.02	36.70 ± 0.21	31.34 ± 0.05	86.68 ± 0.09	60.92 ± 0.07
L Prop 1-hop	63.21 ± 0.39	53.09 ± 0.05	43.42 ± 0.17	30.28 ± 0.09	66.02 ± 0.16	62.77 ± 0.24
L Prop 2-hop	74.13 ± 0.46	76.76 ± 0.03	46.07 ± 0.15	38.61 ± 0.07	67.04 ± 0.20	63.88 ± 0.24
LINK	80.79 ± 0.49	80.54 ± 0.03	53.97 ± 0.18	60.39 ± 0.07	73.56 ± 0.14	64.85 ± 0.21
SGC 1-hop	66.79 ± 0.27	53.61 ± 0.17	32.83 ± 0.13	30.31 ± 0.06	82.36 ± 0.37	58.97 ± 0.19
SGC 2-hop	76.09 ± 0.45	62.81 ± 1.42	32.27 ± 0.06	29.09 ± 0.09	82.10 ± 0.14	59.94 ± 0.21
C&S 1-hop	74.28 ± 1.19	62.35 ± 0.06	44.51 ± 0.16	35.55 ± 0.05	82.93 ± 0.15	64.86 ± 0.27
C&S 2-hop	78.40 ± 3.12	81.69 ± 0.09	49.78 ± 0.26	49.08 ± 0.04	84.94 ± 0.49	65.02 ± 0.16
GCN	82.47 ± 0.27	75.45 ± 0.17	46.02 ± 0.26	45.65 ± 0.04	87.42 ± 0.37	62.18 ± 0.26
GAT	81.53 ± 0.55	71.77 ± 6.18 (M)	46.05 ± 0.51	45.37 ± 0.44 (M)	55.80 ± 0.87	59.89 ± 4.12
GCNJK	81.63 ± 0.54	77.00 ± 0.14	46.28 ± 0.29	46.88 ± 0.13	89.30 ± 0.19	63.45 ± 0.22
GATJK	80.69 ± 0.36	71.19 ± 6.96 (M)	45.80 ± 0.72	44.78 ± 0.50	56.70 ± 2.07	59.98 ± 2.87
APPNP	74.33 ± 0.38	62.58 ± 0.08	38.15 ± 0.26	32.19 ± 0.07	85.36 ± 0.62	60.97 ± 0.10
H_2GCN	(M)	(M)	49.09 ± 0.10	(M)	(M)	(M)
MixHop	83.47 ± 0.71	81.07 ± 0.16	51.81 ± 0.17	52.16 ± 0.09 (M)	90.58 ± 0.16	65.64 ± 0.27
GPR-GNN	81.38 ± 0.16	78.83 ± 0.05	45.07 ± 0.21	40.19 ± 0.03	90.05 ± 0.31	61.89 ± 0.29
GCNII	82.92 ± 0.59	78.94 ± 0.11 (M)	47.21 ± 0.28	37.88 ± 0.69 (M)	90.24 ± 0.09	63.39 ± 0.61
LINKX	84.71 ± 0.52	82.04 ± 0.07	56.00 ± 1.34	61.95 ± 0.12	90.77 ± 0.27	66.06 ± 0.19

5.1 Experimental Setup

Methods. We include both methods that are graph-agnostic and node-feature-agnostic as simple baselines. The node-feature-agnostic models of two-hop label propagation [45] and LINK (logistic regression on the adjacency matrix) [63] have been found to perform well in various non-homophilous settings, but they have often been overlooked by recent graph representation learning work. Also, we include SGC [55] and C&S [26] as simple, scalable methods that perform well on homophilic datasets. We include a two-hop propagation variant of C&S in analogy with two-step label propagation. In addition to representative general GNNs, we also include GNNs recently proposed for non-homophilous settings. The full list of methods is: **Only node features:** MLP [23]. **Only graph topology:** label propagation (standard and two-hop) [64, 45], LINK [63]. **Simple methods:** SGC [55], C&S [26] and their two-hop variants. **General GNNs:** GCN [32], GAT [54], jumping knowledge networks (GCNJK, GATJK) [56], and APPNP [33]. **Non-homophilous methods:** H_2GCN [66], MixHop [1], GPR-GNN [14], GCNII [12], and LINKX (ours).

Minibatching methods. We also evaluate GNNs with various minibatching methods. We take GCNJK [56] and MixHop [1] as our base models for evaluation, as they are representative of many GNN design choices and MixHop performs very well in full batch training. As other minibatching methods are trickier to make work with these models, we use the Cluster-GCN [13] and GraphSAINT [61] minibatching methods, which sample subgraphs. We include both the node based sampling and random walk based sampling variants of GraphSAINT. We compare these GNNs with MLP, LINK, and our LINKX, which use simple i.i.d. node minibatching.

Training and evaluation. Following other works in non-homophilous graph learning evaluation, we take a high proportion of training nodes [66, 46, 57]; we run each method on the same five random 50/25/25 train/val/test splits for each dataset. All methods requiring gradient-based optimization are run for 500 epochs, with test performance reported for the learned parameters of highest validation performance. We use ROC-AUC as the metric for the class-imbalanced genius dataset (about 80% of nodes are in the majority class), as it is less sensitive to class-imbalance than accuracy. For other datasets, we use classification accuracy as the metric. Further experimental details are in Appendix B.

5.2 Full-Batch Results

Table 3 lists the results of each method across the datasets that we propose. Our datasets reveal several important properties of non-homophilous node classification. Firstly, the stability of performance across runs is better for our datasets than those of Pei et al. [46] (see [66] results). Secondly, as suggested by prior theory and experiments [66, 1, 14], the non-homophilous GNNs usually do well — though not necessarily on every dataset.

The core assumption of homophily in SGC and C&S that enables them to be simple and efficient does not hold on these non-homophilous datasets, and thus the performance of these methods is typically relatively low. Still, as expected, two-hop variants improve upon their one-hop counter-parts. Indeed, our results indicate that simple 2-hop modifications of learning methods often improve performance in low-homophily.

One consequence of using larger datasets for benchmarks is that the tradeoff between scalability and learning performance of non-homophilous methods has become starker, with some methods facing memory issues. This tradeoff is especially important to consider in light of the fact that many scalable graph learning methods rely on implicit or explicit homophily assumptions [55, 26, 17, 8], and thus face issues when used in non-homophilous settings.

Finally, LINKX achieves superior performance on all datasets, taking advantage of LINK's power, while also being able to utilize node features where they provide additional information.

5.3 Minibatching Results

Our experimental results for minibatched methods on our proposed datasets are in Table 4. Since GraphSAINT does not partition the nodes of the graph into subgraphs that cover all nodes, we test on the full input graph for the smaller datasets and uniformly random partitions of the graph into 10 induced subgraphs for the larger datasets.

First, we note that both Cluster-GCN and GraphSAINT sampling lead to performance degradation for these methods on our proposed non-homophilous datasets. When compared to the full-batch training results of the previous section, classification accuracy is typically substantially lower. Further experiments in Appendix C.2 give evidence that the performance degradation is often more substantial in non-homophilous settings, and provides possible explanations for why this may be the case.

On the other hand, LINKX does not suffer much performance degradation with the simple i.i.d. node minibatching technique. In fact, it matches or outperforms all methods in this setting, often by a wide margin. Though LINK performs on par with LINKX in arXiv-year and pokec, our LINKX model significantly outperforms it on other datasets, again due to LINKX's ability to integrate node feature information. We again stress that the LINKX minibatching is very simple to implement, yet it still substantially outperforms other methods. Consequently, LINKX is generally well-suited for scalable node classification across a broad range of non-homophilous settings, surpassing even specially designed non-homophilous GNNs with current graph minibatching techniques.

Table 4: Minibatching results on our proposed datasets. † denotes that 10 random partitions of the graphs are used for testing GraphSAINT sampling. (T) denotes that five runs takes ≥ 48 hours for a single hyperparameter setting. Best results up to a standard deviation are highlighted.

	Penn94	pokec †	arXiv-year	snap-patents †	genius	twitch-gamers †	wiki †
MLP Minibatch	74.24±0.55	62.14±0.05	36.89±0.11	22.96±0.81	82.35±0.38	61.01±0.06	37.38±0.21
LINK Minibatch	81.61±0.34	81.15±0.25	53.76±0.28	45.65±8.25	80.95±0.07	64.38±0.26	57.11±0.26
GCNJK-Cluster	69.99±0.85	72.67±0.05	44.05±0.11	37.62±0.31	83.04±0.56	61.15±0.16	(T)
GCNJK-SAINT-Node	72.80±0.43	63.68±0.06	44.30±0.22	26.97±0.10	80.96±0.09	59.50±0.35	44.86±0.19
GCNJK-SAINT-RW	72.29±0.49	65.00±0.11	47.40±0.17	33.05±0.06	81.04±0.14	59.82±0.27	47.39±0.19
MixHop-Cluster	75.79±0.44	76.67±0.07	48.41±0.31	46.82±0.11	81.12±0.10	62.95±0.08	(T)
MixHop-SAINT-Node	75.61±0.55	66.42±0.06	44.84±0.18	27.45±0.11	81.06±0.08	59.58±0.27	47.39±0.18
MixHop-SAINT-RW	76.38±0.50	67.92±0.06	50.55±0.20	34.21±0.07	82.25±0.78	60.39±0.16	49.15±0.26
LINKX Minibatch	84.50±0.65	81.27±0.38	53.74±0.27	60.27±0.29	85.81±0.10	65.84±0.19	59.80±0.41

6 Discussion and Conclusion

In this paper, we propose new, high-quality non-homophilous graph learning datasets, and we benchmark simple baselines and representative graph representation learning methods across these datasets. Further, we develop LINKX: a strong, simple, and scalable method for non-homophilous classification. Our experiments show that LINKX significantly outperforms other methods on our proposed datasets, thus providing one powerful method in the underexplored area of scalable learning on non-homophilous graphs. We hope that our contributions will provide researchers with many new avenues of research in learning on non-homophilous graphs, with better tools to test models and evaluate utility of new techniques.

While we do find utility of our proposed datasets and LINKX model, this work is somewhat limited by only focusing on transductive node classification. This setting is the most natural for studying performance in the absence of homophily, since here we define homophily in terms of the node labels, and previous non-homophilous GNN work using the Pei et al. [46] data also studies this setting exclusively [66, 14]. Using other Facebook 100 datasets besides Penn94 [53] would allow for inductive node classification, but LINKX does not directly generalize to this setting. Our proposed datasets and model LINKX could be used for link prediction, but this is left for future work.

Broader Impact. Fundamental research in graph learning on non-homophilous graphs has the potential for real societal benefit. In particular, it enables more advanced malicious node detection techniques that are not fooled by fraudsters' connections to legitimate users and customers.

Nevertheless, as with all technology, its impact on society depends on what purpose it is used for. Negative social consequences may occur if these methods are deployed for nefarious purposes; for example, node classification may be used to infer users' gender, sexual orientation, and political beliefs, and thus cause users to face discrimination and unfair treatment. We hope that this paper will raise more awareness and discussions regarding the homophily limitations of existing scalable graph learning models and help researchers design scalable models that continue to work well in the non-homophily regime, thus improving the quality of node classification on graphs more broadly.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29, 2019.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[3] Kristen M Altenburger and Johan Ugander. Monophily in social networks introduces similarity among friends-of-friends. *Nature human behaviour*, 2(4):284–290, 2018.

[4] Kristen M Altenburger and Johan Ugander. Node attribute prediction: An evaluation of within- versus across-network tasks. *NeurIPS Workshop on Relational Representation Learning*, 2018.

[5] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2021.

[6] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[7] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. *arXiv preprint arXiv:2101.00797*, 2021.

[8] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2464–2473, 2020.

[9] Adam Breuer, Roee Eilat, and Udi Weinsberg. Friend or faux: Graph-based early detection of fake accounts on social networks, 2020.

[10] Duen Horng Chau, Shashank Pandit, and Christos Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. In *European conference on principles of data mining and knowledge discovery*, pages 103–114. Springer, 2006.

[11] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

[12] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

[13] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 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 & Data Mining*, pages 257–266, 2019.

[14] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[15] Alex Chin, Yatong Chen, Kristen M. Altenburger, and Johan Ugander. Decoupled smoothing on graphs. In *The World Wide Web Conference*, pages 263–272, 2019.

[16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[17] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *International Conference on Learning Representations*, 2020.

[18] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[19] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[20] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[21] Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. Early vs late fusion in multimodal convolutional neural networks. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–6. IEEE, 2020.

[22] Wolfgang Gatterbauer. Semi-supervised learning with heterophily. *arXiv preprint arXiv:1412.3100*, 2014.

[23] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[25] 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. *arXiv preprint arXiv:2005.00687*, 2020.

[26] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2021.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[28] Junteng Jia and Austion R Benson. Residual correlation in graph neural network regression. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 588–598, 2020.

[29] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 148–156, 2021.

[30] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[33] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

[34] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[35] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

[36] Derek Lim and Austin R Benson. Expertise and dynamics within crowdsourced musical knowledge curation: A case study of the genius platform. In *Proceedings of the International AAAI Conference on Web and Social Media*, 2021.

[37] Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks, 2020.

[38] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[39] 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.

[40] Mark EJ Newman. Mixing patterns in networks. *Physical review E*, 67(2):026126, 2003.

[41] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[42] Hoang NT, Takanori Maehara, and Tsuyoshi Murata. Stacked graph filter. *arXiv preprint arXiv:2011.10988*, 2020.

[43] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210, 2007.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[45] Leto Peel. Graph-based semi-supervised learning for relational networks. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 435–443. SIAM, 2017.

[46] 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*, 2019.

[47] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162.

[48] Benedek Rozemberczki and Rik Sarkar. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *arXiv preprint arXiv:2101.03091*, 2021.

[49] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[50] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[51] Neil JA Sloane. The on-line encyclopedia of integer sequences. In *Towards mechanized mathematical assistants*, pages 130–130. Springer, 2007.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[53] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[55] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[56] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[57] 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. *arXiv preprint arXiv:2102.06462*, 2021.

[58] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[59] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[60] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Accurate, efficient and scalable graph embedding. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 462–471. IEEE, 2019.

[61] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2019.

[62] Jin Zeng, Yanfeng Tong, Yunmu Huang, Qiong Yan, Wenxiu Sun, Jing Chen, and Yongtian Wang. Deep surface normal estimation with hierarchical rgb-d fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6153–6162, 2019.

[63] Elena Zheleva and Lise Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540, 2009.

[64] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[65] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. *arXiv preprint arXiv:2009.13566*, 2020.

[66] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have supported the claims in the main paper and appendix.
   (b) Did you describe the limitations of your work? [Yes] See discussion / conclusion Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See discussion / conclusion Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include our proposed datasets and our codes for reproducing the experimental results in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.1 and Appendix Section B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We include standard deviation of performance metrics across multiple runs in Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We provide GPU information in Appendix Section B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the creators of existing datasets and methods in the main paper. Also, in Appendix B, we cite codes that we used.
   (b) Did you mention the license of the assets? [Yes] In Appendix D, we note the licenses of datasets, if provided in published work. Also, we include the license of open-source codes we build off of in Appendix B.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our proposed datasets along with codes for reproducing our results in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We discuss this in Appendix D. Most of the datasets were constructed and presented in previously published academic work. This mostly does not apply to the data that we collect, as it is encyclopedic in nature.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] In Appendix D, we discuss this. Our datasets are primarily numerical, so they do not contain offensive content. To the best of our knowledge, our datasets do not contain personally identifiable content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]