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

Derek Lim[*]
Cornell University
dl772@cornell.edu

Felix Hohne[*]
Cornell University
fmh42@cornell.edu

Xiuyu Li[*]
Cornell University
xl289@cornell.edu

Sijia Linda Huang
Cornell University
sh837@cornell.edu

Vaishnavi Gupta
Cornell University
vg222@cornell.edu

Omkar Bhalerao
Cornell University
opb7@cornell.edu

Ser-Nam Lim
Facebook AI
sernam@gmail.com

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. Our codes and data are available at https://github.com/CUAI/Non-Homophily-Large-Scale.

1 Introduction

Graph learning methods generate predictions by leveraging complex inductive biases captured in the topology of the graph [7]. 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 [46, 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 [55, 13, 25, 11].

Further, while new GNNs that work better in these non-homophilous settings have been developed [82, 44, 81, 17, 15, 73, 36, 35, 9, 54], their evaluation is limited to a few graph datasets used by Pei et al. [58] (collected by [61, 66, 48]) that have certain undesirable properties such as small size, narrow range of application areas, and high variance between different train/test splits [82]. 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 [82, 1, 17], 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

[*]Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

be directly applied to the non-homophilous setting, as they oftentimes assume homophily in their construction [71, 32, 20, 10].

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 [71, 32, 20, 10]. Moreover, we give empirical evidence that existing minibatching techniques in graph learning [16, 77] significantly degrade 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 [77] in non-homophilous settings, showing that they substantially degrade performance in experiments. Also, we show empirically that scalable methods for graph learning like SGC and C&S [71, 32] 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 performance in large non-homophilous graphs, in large part 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 [79], 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.

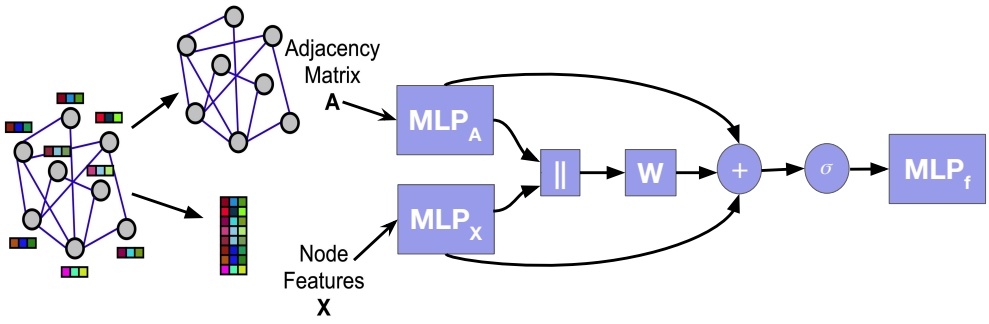


Figure 1: Our model LINKX separately embeds node features and adjacency information with MLPs, combines the embeddings together by concatenation, then uses a final MLP to generate predictions.

2 Prior Work

Graph Representation Learning. Graph neural networks [28, 38, 69] 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 [82, 6]. Indeed, a wide range of GNNs operate as low-pass graph filters [53, 71, 6] 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 [77, 16, 75, 28, 71, 32, 20, 10]. Many of these methods explicitly make use of an assumption of homophily in the data [71, 32, 20, 10]. By leveraging this assumption, several simple, inexpensive models are able to achieve state-of-the-art performance on homophilic datasets [71, 32]. 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 [28, 75, 14]. 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 [16, 76, 77]. 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 [82, 44, 81, 17, 15, 73, 36, 35]. Geom-GCN [58] introduces a geometric aggregation scheme, MixHop [1] proposes a graph convolutional layer that mixes powers of the adjacency matrix, GPR-GNN [17] features learnable weights that can be positive and negative in feature propagation, GCNII [15] allows deep graph convolutional networks with relieved oversmoothing, which empirically performs better in non-homophilous settings, and H_2GCN [82] 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 [82]. 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 [62, 74] are highly homophilous (see Appendix A) [82]. Recently, the Open Graph Benchmark [31] 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 [82] 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 used by Pei et al. [58] (and collected by [61, 66, 48]); see our Table 1 for statistics. 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 [63], evaluation on the datasets of Pei et al. [58] is plagued by high variance across different train/test splits (see results in [82]). The small size of these datasets may tend to create models that are more prone to overfitting [21], which prevents the scaling up of GNNs designed for non-homophilous settings.

Peel [57] also studies node classification on network datasets with various types of relationships between edges and labels. However, they only study methods that act on graph topology, 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 An Improved Homophily Measure

Various metrics have been proposed to measure the homophily of a graph. However, these metrics are sensitive to the number of classes and the number of nodes in each class. Let $G = (V, E)$ be a graph with n nodes, none of which are isolated. Further let each node $u \in V$ have a class label $k_u \in \{0, 1, \ldots, C-1\}$ for some number of classes C, and denote by C_k the set of nodes in class k. The edge homophily [82] is the proportion of edges that connect two nodes of the same class:

$$h = \frac{|\{(u, v) \in E : k_u = k_v\}|}{|E|}. \tag{1}$$

Another related measure is what we call the node homophily [58], defined as $\frac{1}{|V|} \sum_{u \in V} \frac{d_u^{(k_u)}}{d_u}$, in which d_u is the number of neighbors of node u, and $d_u^{(k_u)}$ is the number of neighbors of u that have the same class label. We focus on the edge homophily (1) in this work, but find that node homophily tends to have similar qualitative behavior in experiments.

The sensitivity of edge homophily to the number of classes and size of each class limits its utility. We consider a null model for graphs in which the graph topology is independent of the labels; suppose that nodes with corresponding labels are fixed, and include edges uniformly at random in the graph that are independent of node labels. Under this null model, a node $u \in V$ would be expected to have $d_u^{(k_u)}/d_u \approx |C_{k_u}|/n$ as the proportion of nodes of the same class that they connect to [3]. For a dataset with C balanced classes, we would thus expect the edge homophily to be around $\frac{1}{C}$, so the interpretation of the measure depends on the number of classes. Also, if classes are imbalanced, then the edge homophily may be misleadingly large. For instance, if 99% of nodes were of one class, then most edges would likely be within that same class, so the edge homophily would be high, even when the graph is generated from the null model where labels are independent of graph topology. Thus, the edge homophily does not capture deviation of the label distribution from the null model.

We introduce a metric that better captures the presence or absence of homophily. Unlike the edge homophily, our metric measures excess homophily that is not expected from the above null model where edges are randomly wired. Our metric does not distinguish between different non-homophilous settings (such as heterophily or independent edges); we believe that there are too many degrees of freedom in non-homophilous settings for a single scalar quantity to be able to distinguish them all. Our measure is given as:

$$\hat{h} = \frac{1}{C-1} \sum_{k=0}^{C-1} \left[h_k - \frac{|C_k|}{n} \right]_+, \tag{2}$$

where $[a]_+ = \max(a, 0)$, and h_k is the class-wise homophily metric

$$h_k = \frac{\sum_{u \in C_k} d_u^{(k_u)}}{\sum_{u \in C_k} d_u}. \tag{3}$$

Note that $\hat{h} \in [0, 1]$, with a fully homophilous graph (in which every node is only connected to nodes of the same class) having $\hat{h} = 1$. Since each class-wise homophily metric h_k only contributes positive deviations from the null expected proportion $|C_k|/n$, the class-imbalance problem is substantially mitigated. Also, graphs in which edges are independent of node labels are expected to have $\hat{h} \approx 0$, for any number of classes. Our measure $\hat{h}$ measures presence of homophily, but does not distinguish between the many types of possibly non-homophilous relationships. This is reasonable given the diversity of non-homophilous relationships. For example, non-homophily can imply independence of edges and classes, extreme heterophily, connections only among subsets of classes, or certain chemically / biologically determined relationships. Indeed, these relationships are very different, and are better captured by more than one scalar quantity, such as the compatibility matrices presented in the appendix. Further discussion is given in Appendix A.

Table 1: Statistics for previously used datasets from Pei et al. [58] (collected by [61, 66, 48]). #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

3.3 Proposed Datasets

Here, we detail the non-homophilous datasets that we propose for graph machine learning evaluation. Our datasets and tasks span diverse application areas. **Penn94** [67], **Pokec** [41], **genius** [43], and **twitch-gamers** [60] are online social networks, where the task is to predict reported gender, certain account labels, or use of explicit content on user accounts. For the citation networks **arXiv-year** [31] and **snap-patents** [42, 41] the goal is to predict year of paper publication or the year that a patent is granted. The dataset **wiki** consists of Wikipedia articles, where the goal is to predict total page views of each article. Detailed descriptions about the graph structure, node features, node labels, and licenses of each dataset are given in Appendix D.2.

Most of these datasets 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.3). This wiki dataset has 1,925,342 nodes and 303,434,860 edges, so training and inference require scalable algorithms.

Basic dataset statistics are given in Table 2. Note the substantial difference between the size of our datasets and those of Pei et al. [58] in Table 1; our datasets have up to 384x more nodes and 1398x more edges. The homophily 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. As there is little study in large-scale non-homophilous graph learning, our proposed large datasets strongly motivate the need for developing a new, scalable approach that can accurately learn on non-homophilous graphs.

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

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 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 on 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 [82].

LINK regression on graph topology. On the other extreme, there is LINK [79] — 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

$$\mathbf{Y} = \mathrm{softmax}(\mathbf{WA}). \tag{4}$$

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

$$(\mathbf{WA})_{ku} = \sum_{v \in \mathcal{N}(u)} \mathbf{W}_{kv}, \tag{5}$$

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:

$$\mathbf{h}_{\mathbf{A}} = \mathrm{MLP}_{\mathbf{A}}(\mathbf{A}) \in \mathbb{R}^{d \times n} \tag{6}$$

$$\mathbf{h}_{\mathbf{X}} = \mathrm{MLP}_{\mathbf{X}}(\mathbf{X}) \in \mathbb{R}^{d \times n} \tag{7}$$

$$\mathbf{Y} = \mathrm{MLP}_f\left(\sigma\left(\mathbf{W}[\mathbf{h}_{\mathbf{A}}; \mathbf{h}_{\mathbf{X}}] + \mathbf{h}_{\mathbf{A}} + \mathbf{h}_{\mathbf{X}}\right)\right), \tag{8}$$

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 $\mathbf{h_A}$ and $\mathbf{h_X}$ 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 $\mathbf{h_A}$ and the features $\mathbf{X}$ into $\mathbf{h_X}$ 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 [24, 78]. 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 $\mathbf{h_X}$ information is similar to just using LINK, and ignoring $\mathbf{h_A}$ 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 [77]. 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 [38] 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 [82]. 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 [15] 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 [82] 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, our LINKX model matches or outperforms other methods.

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 [57] and LINK (logistic regression on the adjacency matrix) [79] 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 [71] and C&S [32] 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 [26]. **Only graph topology:** label propagation (standard and two-hop) [80, 57], LINK [79]. **Simple methods:** SGC [71], C&S [32] and their two-hop variants. **General GNNs:** GCN [38], GAT [69], jumping knowledge networks (GCNJK, GATJK) [72], and APPNP [39]. **Non-homophilous methods:** H_2GCN [82], MixHop [1], GPR-GNN [17], GCNII [15], and LINKX (ours).

Minibatching methods. We also evaluate GNNs with various minibatching methods. We take GCNJK [72] 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 [16] and GraphSAINT [77] 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 [82, 58, 73]; 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. [58] (see [82] results). Secondly, as suggested by prior theory and experiments [82, 1, 17], 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 generally improve upon their one-hop counter-parts in these low-homophily settings.

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 [71, 32, 20, 10], 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

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

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.

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 new avenues of research in learning on non-homophilous graphs, along with better tools to test models and evaluate utility of new techniques.

While we do find utility in 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. [58] data also studies this setting exclusively [82, 17]. Using other Facebook 100 datasets besides Penn94 [67] 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 positive societal benefit. As a major application, it enables malicious node detection techniques in social networks and transaction networks that are not fooled by fraudsters' connections to legitimate users and customers. This is a widely studied task, and past works have noted that non-homophilous structures are present in many such networks [11, 25, 55]. We hope that this paper provides insight on the homophily limitations of existing scalable graph learning models and help researchers design scalable models that continue to work well in the non-homophilous regime, thus improving the quality of node classification on graphs more broadly. As our proposed datasets have diverse structures and our model performs well across all of these datasets, the potential for future application of our work to important non-homophilous tasks is high.

Nevertheless, our work could also have potential for different types of negative social consequences. Nefarious behavior by key actors could be one source of such consequences. Nonetheless, we expect that the actors that can make use of large-scale social networks for gender prediction as studied in our work are limited in number. Actors with both the capability and incentive to perform such operations probably mostly consist of entities with access to large social network data such as social media companies or government actors with auxiliary networks [50]. Smaller actors can perform certain attacks, but this may be made more difficult by resource requirements such as the need for certain external information [50] or the ability to add nodes and edges before an anonymized version of a social network is released [5]. Furthermore, additional actors could make use of deanonymization attacks [30, 49, 50] to reveal user identities in supposedly anonymized datasets.

Also, accidental consequences and implicit biases are a potential issue, even if the applications of the learning algorithms are benign and intended to benefit society [47]. Performance of algorithms may vary substantially between intersectional subgroups of subjects — as in the case of vision-based gender predictors [12] (and some have questioned the propriety of vision-based gender classifiers altogether). Thus, there may be disparate effects on different populations, so care should be taken to understand the impact of those differences across subgroups. Moreover, large datasets require computing resources, so projects can only be pursued by large entities at the possible expense of the individual and smaller research groups [8]. This is alleviated by the fact that our experiments are each run on one GPU, and hence have significantly less GPU computing requirements than much current deep learning research. Thus, smaller research groups and independent researchers should find our work beneficial, and should be able to build on it.

Finally, the nature of collection of online user information also comes with notable ethical concerns. Common notice-and-consent policies are often ineffective in actually protecting user privacy [52]. Indeed, users may not actually have much choice in using certain platforms or sharing data due to social or economic reasons. Also, users are generally unable to fully read and understand all of the different privacy policies that they come across, and may not understand the implications of having their data available for long periods of time to entities with powerful inference algorithms. Furthermore, people may rely on obscurity for privacy [29], but this assumption may be ignored in courts of law, and it may be directly broken when data leaks or is released in aggregated form without sufficient privacy protections. Overall, while we believe that our work will benefit machine learning research and enable positive applications, we must still be aware of possible negative consequences.

Acknowledgements

We thank Abhay Singh, Austin Benson, and Horace He for insightful discussions. We also thank the rest of Cornell University Artificial Intelligence for their support and discussion. We thank Facebook AI for funding equipment that made this work possible.

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] Lars Backstrom, Cynthia Dwork, and Jon M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW '07*, 2007.

[6] 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.

[7] 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.

[8] Abeba Birhane, Pratyusha Kalluri, Dallas Card, William Agnew, Ravit Dotan, and Michelle Bao. The values encoded in machine learning research, 2021.

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

[10] 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.

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

[12] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91. PMLR, 23–24 Feb 2018. URL `https://proceedings.mlr.press/v81/buolamwini18a.html`.

[13] 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.

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

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

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

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

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

[24] 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.

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

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

[27] Bronwyn H Hall, Adam B Jaffe, and Manuel Trajtenberg. The nber patent citation data file: Lessons, insights and methodological tools. Working Paper 8498, National Bureau of Economic Research, October 2001. URL `http://www.nber.org/papers/w8498`.

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

[29] Woodrow Hartzog and Frederic Stutzman. The case for online obscurity. *California Law Review*, 101(1):1–49, 2013. ISSN 00081221. URL `http://www.jstor.org/stable/23409387`.

[30] Michael Hay, Gerome Miklau, David Jensen, Donald Towsley, and Philipp Weis. Resisting structural identification in anonymized social networks. *PVLDB*, 1:102–114, 08 2008. doi: 10.14778/1453856.1453873.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

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

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

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

[40] M. Zabovsky L. Takac. Data analysis in public social networks. *International Scientific Conference & International Workshop Present Day Trends of Innovations*, 2012.

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

[42] 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.

[43] 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.

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

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

[46] 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.

[47] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019.

[48] Tom Mitchell et al. The 4 universities data set, 1997. URL `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`.

[49] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008. doi: 10.1109/SP.2008.33.

[50] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 173–187, 2009. doi: 10.1109/SP.2009.22.

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

[52] Helen Nissenbaum. A contextual approach to privacy online. *Daedalus*, 140:32–48, 2011.

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

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[62] 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.

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

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

[65] 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.

[66] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816, 2009.

[67] 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.

[68] Nate Veldt, Austin R Benson, and Jon Kleinberg. Higher-order homophily is combinatorially impossible. *arXiv preprint arXiv:2103.11818*, 2021.

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

[70] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft Academic Graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 02 2020. ISSN 2641-3337. doi: 10.1162/qss_a_00021. URL `https://doi.org/10.1162/qss_a_00021`.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] Michael Zimmer. Facebook data of 1.2 million users from 2005 released: Limited exposure, but very problematic, Feb 2011. URL `https://michaelzimmer.org/2011/02/15/facebook-data-of-1-2-million-users-from-2005-released/`.

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 wiki dataset that we collect, as it is encyclopedic in nature.

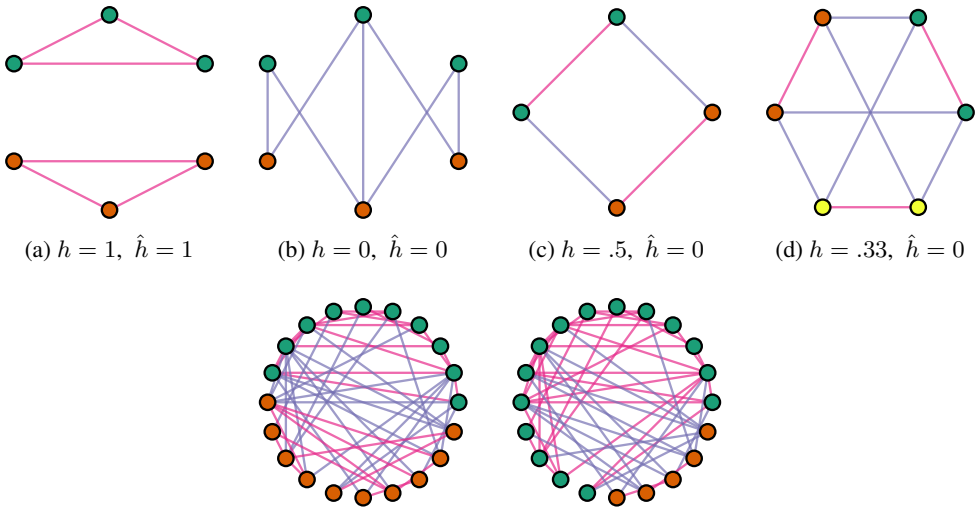
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] In Appendix D, we discuss this. Our datasets are primarily numerical, so they do not contain offensive content. To the best of our knowledge, our datasets do not contain 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]



(a) $h = 1,\ \hat{h} = 1$ (b) $h = 0,\ \hat{h} = 0$ (c) $h = .5,\ \hat{h} = 0$ (d) $h = .33,\ \hat{h} = 0$

(e) $h = .53,\ \hat{h} = .07$ (f) $h = .66,\ \hat{h} = .04$

Figure 2: Examples of graphs with different label-topology relationships and comparison of our measure $\hat{h}$ with the edge homophily ratio h. The node classes are labeled by color. Pink edges link nodes of the same class, while purple edges link nodes of different classes. (a,b) Pure homophily and pure heterophily. Both measures equal 1 in homophily and 0 in heterophily. (c,d) Graphs where each node is connected to one member of each class. Edge homophily depends on the number of classes, while our measure $\hat{h}$ does not. (e,f) Random Erdős-Rényi graphs in which edges are independent of labels. Edge homophily is sensitive to class imbalance, while our measure $\hat{h}$ is not.

A Compatibility Matrices and Statistics

A.1 Measuring Homophily

In Section 3.2, we consider metrics that attempt to capture the level of class-label homophily in a graph in a single scalar quantity. These metrics could be useful for practitioners who need to choose appropriate graph learning algorithms for some graph data that they have — performance of algorithms heavily depends on the homophily of the graph. Moreover, they are useful for choosing datasets to benchmark on, as we do in this paper.

A better representation of homophily may be given by the compatibility matrix, which consists of C^2 values instead of a single scalar value. Following previous work [82], for a graph G with C node classes we define the $C \times C$ compatibility matrix $\mathbf{H}$ by

$$\mathbf{H}_{kl} = \frac{|(u,v) \in E : k_u = k,\ k_v = l|}{|(u,v) \in E : k_u = k|}. \tag{9}$$

This captures finer details of label-topology relationships in graphs than single scalar metrics capture. For classes k and l, the entry $\mathbf{H}_{kl}$ measures the proportion of edges from nodes of class k that are connected to nodes of class l. A homophilous graph has high values of $\mathbf{H}_{kk}$ for each class k.

Compatibility matrices for our proposed datasets are shown in Figure 3. As evidenced by the different patterns, the proposed datasets show interesting types of label-topology relationships besides homophily. For instance, the citation datasets arXiv-year and snap-patents have primarily lower-triangular structure, since most citations reference past work. In wiki, low view articles tend to link to each other, while mid-rank articles frequently link to highly viewed articles and vice versa. In Pokec, there is some heterophily, in that one gender has some preference for friends of another gender.

However, the compatibility matrix can be unwieldy for datasets with many classes. For the cases where a single scalar representation of homophily is desired, our introduced measure better captures the presence or absence of homophily than existing metrics. Figure 2 compares our measure $\hat{h}$ to edge homophily on example graphs.

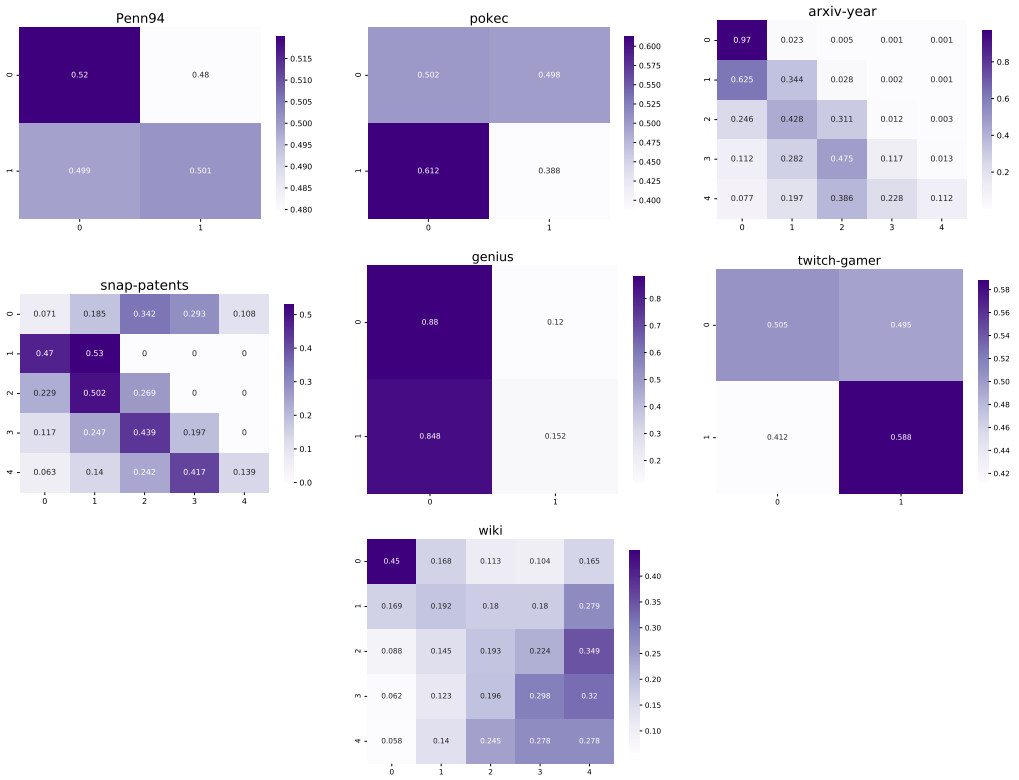


Figure 3: Compatibility matrices of our proposed datasets. These datasets from a variety of different contexts exhibit a wide range of non-homophilous structures.

A.2 Homophilous Data Statistics

Table 5: Statistics for homophilic graph datasets. # C is the number of node classes.

Dataset	# Nodes	# Edges	# C	Edge hom.	$\hat{h}$ (ours)
Cora	2,708	5,278	7	.81	.766
Citeseer	3,327	4,552	6	.74	.627
Pubmed	19,717	44,324	3	.80	.664
ogbn-arXiv	169,343	1,166,243	40	.66	.416
ogbn-products	2,449,029	61,859,140	47	.81	.459
oeis	226,282	761,687	5	.50	.532

In contrast to the different compatibility matrix structures of our proposed non-homophilous datasets, much other graph data have primarily homophilous relationships, as can be seen in Figure 4 and Table 5. The Cora, CiteSeer, PubMed, ogbn-arXiv, and ogbn-products datasets are widely used as benchmarks for node classification [74, 31], and are highly homophilous, as can be seen by the diagonally dominant structure of the compatibility matrices and by the high edge homophily and $\hat{h}$.

We collected the oeis dataset displayed in the bottom right of Figure 4. The nodes are entries in the Online Encyclopedia of Integer Sequences [64], and directed edges link an entry to any other entry that it cites. In analogy to arXiv-year and snap-patents, the node labels are the time of posting of the sequence. However, in this case the graph relationships are homophilous, even as we vary the number of distinct classes (time periods). This is in part due to differences between posting in this online encyclopedia and publication of academic papers or patents. For instance, there is less overhead to posting an entry in the OEIS, so users often post separate related entries and variants of these entries in rapid succession. Also, an entry in the encyclopedia often inspires other people to work on similar entries, which can be created in much less time than an academic follow-up work to a given paper.

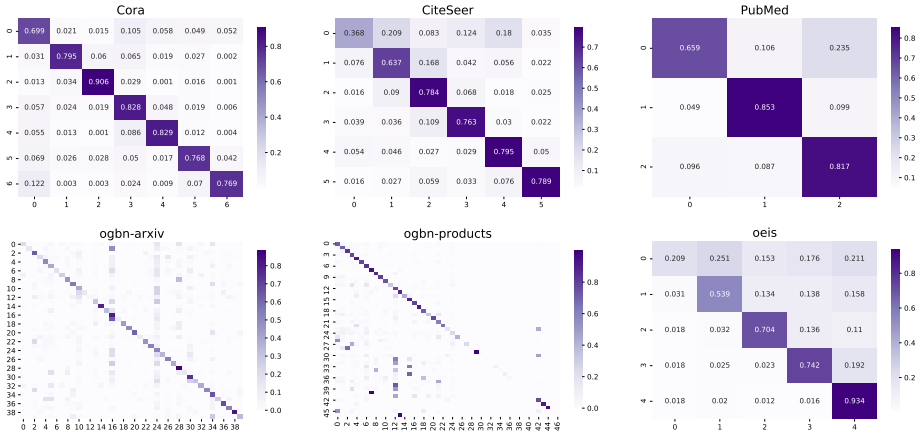


Figure 4: Compatibility matrices of homophilic datasets. The diagonal dominance indicates strong homophily.

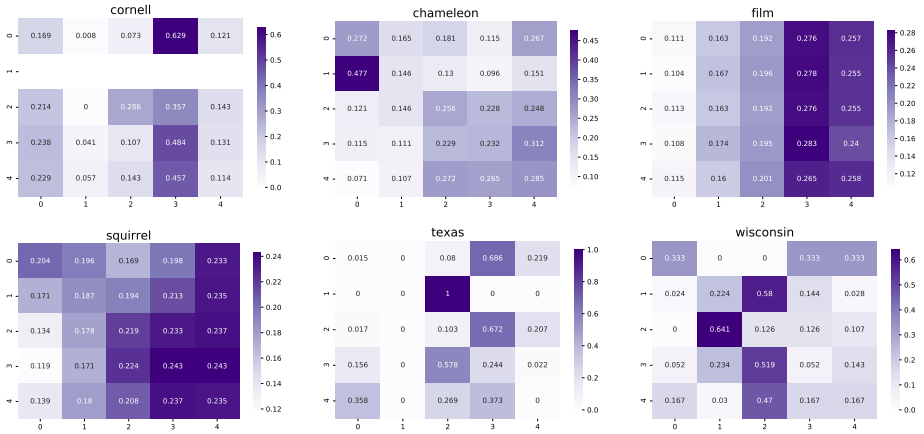


Figure 5: Compatibility matrices of datasets in Pei et al. [58] (collected by [61, 66, 48]). The "film" dataset is also referred to as "Actor". Note that there are no edges leading out of the nodes of class 1 in the Cornell dataset, so there is an empty row in its matrix.

These related entries tend to cite each other, which contributes to homophilic relationships over time. Thus, the data here does not follow the special temporal citation structure of academic publications and patents.

A.3 Previous Non-Homophilous Data

For the six datasets in Pei et al. [58] often used in evaluation of graph representation learning methods in non-homophilous regimes [82], basic statistics are listed in Table 1 and compatibility matrices are displayed in Figure 5. We propose datasets that have up to orders of magnitude more nodes and edges and come from a wider range of contexts. There are several cases of class-imbalance in these previously used datasets, which may make the edge homophily misleading. As discussed in Appendix A.1, our measure may be able to alleviate issues with edge homophily in measuring homophily of these datasets, and offers a way to distinguish between the Chameleon, Actor, and Squirrel datasets that all have similar edge homophily.

A.4 General Non-homophilous Settings

Different settings in which non-homophilous / disassortative relationships are prevalent have been identified in the literature, and many of these non-homophilous settings are represented by our

proposed datasets. We list these general non-homophilous settings for reference, with our proposed datasets that belong to these settings in parentheses:

- Gender relations in social or interaction networks [3, 18, 34] (Penn94, Pokec).
- Technological and internet relationships, such as in web page connections [51, 58] (wiki).
- Malicious or fraudulent nodes, such as in auction networks [13, 55] (genius).
- Publication time in citation networks [57] (arXiv-year, snap-patents).
- Biological structures such as in food webs [25] and protein interactions [51].
- Specific online user attributes [60] (twitch gamers)

While not all example graph data from these contexts are non-homophilous, a diverse range are. In order to succeed in future applications in these contexts, it is of importance to develop methods that are able to handle non-homophilous structures.

A.5 Class-Imbalance and Metrics

In this section, we present experiments that demonstrate an instance in which our metric is not affected by imbalanced classes, while edge homophily is. We generate graphs in which node labels are independent of edges by randomly choosing node labels and generating graph edges by the Erdős-Rényi random graph model [22]. In particular, we fix the number of classes to two, the number of nodes to 100, and the probability of edge formation as .25 between every pair of nodes. Then we generate 100 samples of these random graphs, and compute the mean and standard deviation of both edge homophily h and our measure $\hat{h}$. As seen in Figure 6, our measure $\hat{h}$ is constantly near zero as we increase the size of the majority class, while the edge homophily h increases as the size of majority class increases.

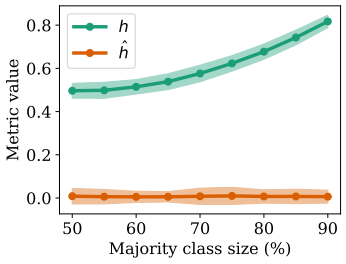


Figure 6: Comparison of edge homophily h and our measure $\hat{h}$ on random class-imbalanced graph data with edges independent of node labels. Three standard deviations are shaded. Our measure is mostly constant as the classes become more imbalanced, while edge homophily increases.

A.6 Degree Distributions of Proposed Non-homophilous datasets

Past work has found that the degree distribution can affect the performance of models on node classification [82]. As a result, we provide degree distributions of all of our proposed non-homophilous datasets in Figure 7. The degree distributions are all heavy-tailed, as is typically expected in real-world graph data. The wiki distribution also has an interesting additional property, where the mode appears to be at a degree between 10 and 100.

A.7 Two-hop homophily levels

While the homophily measures are based on one-hop information, we may also consider properties of two-hop neighborhoods. Even though there are limitations to measuring higher-order homophily [68], here we estimate an interesting quantity in two-hop neighborhoods. We use a node homophily measure, where the neighborhood of each node is defined to be the nodes of exactly two hops away:

$$\frac{1}{|V|}\sum_{v\in V}\frac{\text{Number of exact two-hop neighbors of } v \text{ with same class as } v}{\text{Number of } v\text{'s exact two-hop neighbors}}. \tag{10}$$

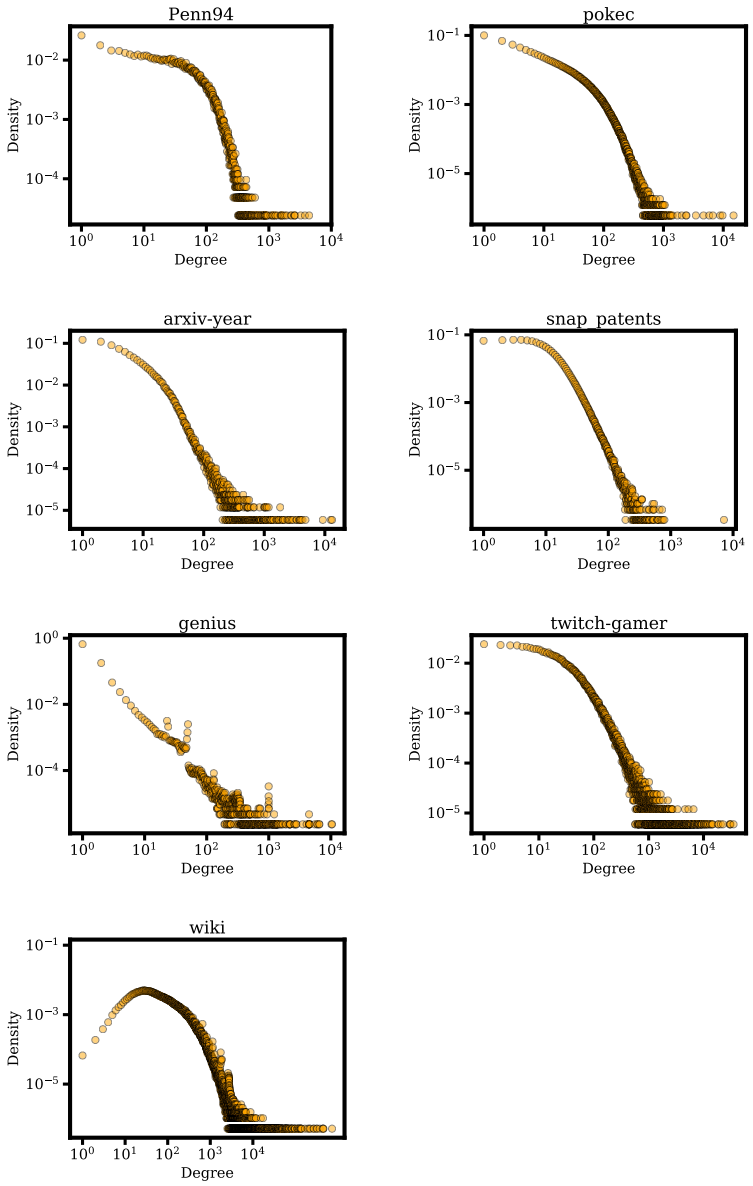


Figure 7: Degree distribution of our proposed non-homophilous datasets

This is an expensive operation to compute, so we approximate the sum over a random sample of k nodes, and then replace the normalization factor of $\frac{1}{|V|}$ by $\frac{1}{k}$. Note that for each node v in the sample, the inner sum is still using the two-hop neighborhoods of the original graph. Table 6 compares the computed one-hop and two-hop measures. The two-hop measure is mostly similar to or somewhat higher than the standard one-hop node homophily.

Table 6: Comparison of node homophily over one-hop neighborhoods and strict two-hop neighborhoods. The measure over two-hop neighborhoods is estimated with $k = 500$ sampled nodes.

Dataset	Two-hop	One-hop
Penn94	.474	.483
pokec	.611	.428
arXiv-year	.341	.289
snap-patents	.330	.221
genius	.749	.508
twitch-gamers	.513	.556

B Experimental Details

For gradient-based optimization, we use the AdamW optimizer [37, 45] with weight decay .001 and learning rate .01 by default, unless we tune the optimizer for a particular method (as noted below in B.1). Hyperparameter tuning is conducted using grid search for most methods. Tuning for C&S is done as in the original paper [32], which uses Optuna [2] for Bayesian hyperparameter optimization. All graphs are treated as undirected besides arXiv-year and snap-patents, in which the directed nature of the edges capture useful temporal information; however, we find that label propagation and C&S (which builds on label propagation) perform better with undirected graphs in these cases, so we keep the graphs as undirected for these methods.

We implement our methods and run experiments in PyTorch [56] (3-clause BSD license), and make heavy use of the PyTorch-Geometric library [23] (MIT license) for graph representation learning. For full-batch training, simple methods are run on a NVIDIA 2080 Ti with 11 GB GPU memory. In cases where the NVIDIA 2080Ti did not provide enough memory, we re-ran experiments on a NVIDIA Titan RTX with 24 GB GPU memory, reporting (M) in Table 3 if the GPU memory was still insufficient. For minibatch training on wiki, we also make use of NVIDIA RTX 3090 GPUs with 24 GB GPU memory. Each experiment was run on one GPU at a time.

B.1 Full-batch Hyperparameters

Experimental results for full-batch training are reported on the hyperparameter settings below, where we choose the settings that achieve the highest performance on the validation set. We choose hyperparameter grids that do not necessarily give optimal performance, but hopefully cover enough regimes so that each model is reasonably evaluated on each dataset. Unless otherwise stated, each GNN has dropout of .5 [65] and BatchNorm [33] in each layer. The hyperparameter grids for the different methods are:

- MLP: hidden dimension $\in \{16, 32, 64, 128, 256\}$, number of layers $\in \{2, 3\}$. We use ReLU activations.
- Label propagation: $\alpha \in \{.01, .1, .25, .5, .75, .9, .99\}$. We use 50 propagation iterations.
- LINK: weight decay $\in \{.001, .01, .1\}$.
- SGC: weight decay $\in \{.001, .01, .1\}$.
- C&S: Normalized adjacency matrix $A_1, A_2 \in \{D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, D^{-1}A, AD^{-1}\}$ for the residual propagation and label propagation, where A is the adjacency matrix of the graph and D is the diagonal degree matrix; $\alpha_1, \alpha_2 \in (0.0, 1.0)$ for the two propagations. Both Autoscale and FDiff-scale were used for all experiments, and scale $\in (0.1, 10.0)$ was searched in FDiff-scale settings. The base predictor is chosen as the best MLP model for each dataset.

- GCN: lr $\in \{.1, .01, .001\}$, hidden dimension $\in \{4, 8, 16, 32, 64\}$, except for snap-patents and pokec, where we omit hidden dimension = 64. Each activation is a ReLU. 2 layers were used for all experiments.
- GAT: lr $\in \{.1, .01, .001\}$. For snap-patents and pokec: hidden channels $\in \{4, 8, 12\}$ and gat heads $\in \{2, 4\}$. For all other datasets: hidden channels $\in \{4, 8, 12, 32\}$ and gat heads $\in \{2, 4, 8\}$. We use the ELU activation [19]. 2 layers were used for all experiments.
- GCNJK: Identical for GCN, also including JK Type $\in \{\text{cat}, \max\}$.
- GATJK: Identical for GAT, also including JK Type $\in \{\text{cat}, \max\}$.
- APPNP: MLP hidden dimension $\in \{16, 32, 64, 128, 256\}$, learning rate $\in \{.01, .05, .002\}$, $\alpha \in \{.1, .2, .5, .9\}$. We truncate the series at the $K = 10$th power of the adjacency.
- H_2GCN: hidden dimension $\in \{8, 16, 32, 64\}$, number of layers $\in \{1, 2\}$, dropout $\in \{0, .5\}$. The architecture follows Section 3.2 of [82].
- MixHop: hidden dimension $\in \{8, 16, 32\}$, number of layers $\in \{2, 3\}$. Each layer has uses the 0th, 1st, and 2nd powers of the adjacency and has ReLU activations. The last layer is a linear projection layer, instead of the attention output mechanism in [1].
- GPR-GNN: The basic setup and grid is the same as that of APPNP. We use their Personalized PageRank weight initialization.
- GCNII: number of layers $\in \{2, 8, 16, 32, 64\}$, strength of the initial residual connection $\alpha_\ell \in \{0.1, 0.2, 0.5\}$, hyperparameter used to compute the strength of the identity mapping $\lambda \in \{0.5, 1.0, 1.5\}$.
- LINKX: We use take $\text{MLP}_{\mathbf{A}}$ and $\text{MLP}_{\mathbf{X}}$ to be one layer networks, i.e. linear mappings of size $d \times n$ and $d \times D$, respectively. The hidden dimension is taken to be $d \in \{16, 32, 128, 256\}$, and the number of layers of MLP_f is in $\{1, 2, 3\}$.

B.2 Minibatching hyperparameters

The setup for minibatching is similar to the setup of full-batch training as above, with some differences that we note here. For all GNNs, we fix the hidden dimension to 128, which is a common hidden dimension used in Cluster-GCN [16] and GraphSAINT [77]. We use concatenation jumping knowledge connections [72] for GCNJK. For GCNJK and MixHop, our hyperparameter grid only chooses a number of layers $L \in \{2, 4\}$, along with the hyperparameters for the minibatching methods that we give below.

- MLP, LINK, and LINKX use the same hyperparameter grids as in the full-batch setting, and they are trained with standard minibatching with a batch size of $n/10$. We train with one such batch in each of 500 epochs.
- All Cluster-GCN based experiments partition the graph into 200 parts and process a number of parts in $\{1, 5\}$ at once. We train over all partitions in each of 500 epochs.
- All GraphSAINT-Node based experiments have a budget of nodes in $\{5{,}000, 10{,}000\}$. We train over five subgraphs for each of 500 epochs.
- All GraphSAINT-RandWalk based experiments use a random walk length of the same size as the number of layers L of the GNN, and use a number of roots in $\{5{,}000, 10{,}000\}$. We train over five subgraphs for each of 500 epochs.

These hyperparameter settings are in the same range as those used in the original papers [16, 77] for datasets that have similar node counts to our proposed datasets. For a total number of nodes n in the full graph, note that the MLP, LINK, and LINKX minibatching method processes about $50n$ total nodes across all batches in training. The Cluster-GCN method processes $500n$ nodes. When the number of nodes is 10,000, the GraphSAINT-Node method approximately processes between $601n$ nodes (Penn94) and $8.5n$ nodes (snap-patents). The GraphSAINT-RandWalk method with 10,000 root nodes processes a few times more nodes (usually between two to five times more) than the GraphSAINT-Node method, as more nodes are sampled from the random walks (and we take the number of layers L to be 2 or 4).

C Further Experiments

C.1 Benefits of Separate Embeddings

Table 7: Comparing separately embedding $\mathbf{A}$ and $\mathbf{X}$ (as in LINKX) with direct concatenation. LINKX outperforms the concatenation based model.

	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
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
MLP($[\mathbf{A}; \mathbf{X}]$)	84.64 ± 0.33	81.74 ± 0.15	54.15 ± 0.20	59.12 ± 0.29	91.61 ± 0.05	64.89 ± 0.18
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

Here, we give more evidence to justify separately embedding $\mathbf{A}$ and $\mathbf{X}$ in LINKX. Recall from Section 4.2 that there are computational benefits to separately embedding them. Table 7 give results for concatenating $\mathbf{A}$ and $\mathbf{X}$ so that they are jointly embedded as $\mathrm{MLP}([\mathbf{A}; \mathbf{X}])$. For this concatenation model, we search over the same hyperparameter grid as that of MLP_f in LINKX. We see that LINKX generally outperforms the concatenation model (besides on the genius dataset). Moreover, LINKX always outperforms both MLP and LINK, while the concatenation model does not do as well as LINK on snap-patents and is within a standard deviation on twitch-gamers.

C.2 Minibatching experiments

Table 8: Comparison of different methods on the arXiv graphs with (homophilous) ogbn-arXiv subject labels and (non-homophilous) arXiv-year publication year labels. Percent relative error of minibatch methods against corresponding full batch methods are in parentheses.

	ogbn-arXiv Coarse	arXiv-year Coarse
GCNJK-Full	70.33	44.18
GCNJK-Cluster	68.28 (2.9%)	42.88 (3.0%)
GCNJK-SAINT-Node	67.69 (3.7%)	41.48 (6.1%)
GCNJK-SAINT-RW	69.57 (1.1%)	43.58 (1.4%)
MixHop-Full	72.26	48.14
MixHop-Cluster	70.16 (3.0%)	46.90 (2.8%)
MixHop-SAINT-Node	67.26 (7.1%)	41.68 (14.7%)
MixHop-SAINT-RW	71.54 (1.0%)	45.94 (5.0%)

	ogbn-arXiv Fine	arXiv-year Fine
GCNJK-Full	70.33	44.18
GCNJK-Cluster	68.01 (3.3%)	42.49 (3.8%)
GCNJK-SAINT-Node	66.18 (5.9%)	39.88 (9.7%)
GCNJK-SAINT-RW	68.96 (1.9%)	42.47 (3.9%)
MixHop-Full	72.26	48.14
MixHop-Cluster	69.37 (3.5%)	46.60 (3.2%)
MixHop-SAINT-Node	64.71 (10.1%)	39.17 (18.7%)
MixHop-SAINT-RW	70.48 (2.0%)	43.29 (10.1%)

Homophilous vs Non-Homophilous. To see differences in graph minibatching between homophilous and non-homophilous settings, we setup experiments using the same graph with homophilous labels and then non-homophilous labels; we take the ogbn-arXiv graph and train GNNs with minibatching with on the original ogbn-arXiv subject area labels (that are homophilous, see Table 5), and also train GNNs with minibatching on the arXiv-year labels that we defined in this work (that are non-homophilous, see Table 2). For the minibatching methods, we use similar hyperparameters as in Appendix C.2 (Cluster-GCN uses 200 partitions and processes 5 parts at once, GraphSAINT-Node has a 10,000 node budget and GraphSAINT-RandWalk has 10,000 roots), which we call the "coarse" setting. In the so-called "fine" setting, we use smaller subgraph minibatches (Cluster-GCN uses 750 partitions and processes 5 parts at once, GraphSAINT-Node has a 2,500 node budget and GraphSAINT-RandWalk has 2,500 roots). Both the GCNJK and MixHop architectures are fixed to have two layers and 128 hidden dimensions.

Table 8 shows the results of these arXiv experiments on two different label sets. We see that performance degradation as measured by percent relative error in test accuracy is more substantial in the non-homophilous arXiv-year experiments for the GraphSAINT methods, while it is mostly comparable for the Cluster-GCN methods.

Graph minibatching could perform poorly in non-homophilous settings for a variety of reasons. It has been shown both theoretically and empirically that higher-order information from more than one-hop neighbors are important for classification in certain non-homophilous settings [82, 3]. One reason why graph minibatching may have issues here is that it is difficult to preserve higher-order neighborhood structure when minibatching on graphs. For instance, suppose we minibatch a graph with n nodes by taking an induced subgraph on $n/2$ randomly selected nodes. Then for a node u in the subgraph, each one hop path $u \to v$ has probability $1/2$ of being in the subgraph, whereas each two-hop path $u \to w \to v$ has probability $1/4$ of being in the subgraph, since both w and v must be sampled.

Finally, even if the percent relative error of minibatching methods is similar in homophilous vs. non-homophilous graphs, the performance degradation would generally be more detrimental in non-homophilous graphs. This is because the gap between GNNs and methods that do not use the graph topology like MLPs are lower in many non-homophilous settings. Since MLPs do not face much performance degradation when trained with simple i.i.d. node minibatching, the gap between GNNs and MLPs is even lower in minibatched settings. Indeed, we see that MLPs perform on par with or outperform many GNNs in the minibatched setting in Table 4.

Table 9: Minibatching results when using GCN as a base model. The setup is the same as in Section 5.3. LINKX results are provided as reference.

	Penn94	pokec †	arXiv-year	snap-patents †	genius	twitch-gamers †	wiki †
GCN-Cluster	70.24±0.24	67.33±0.21	45.80±0.22	38.53±0.08	82.12±0.40	60.71±0.15	(T)
GCN-SAINT-Node	69.41±0.55	59.91±0.08	44.92±0.23	27.16±0.16	80.95±0.09	59.03±0.19	42.59±0.09
GCN-SAINT-RW	69.79±0.31	62.50±0.18	48.07±0.21	32.75±0.08	80.98±0.25	59.52±0.08	44.22±0.18
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

GCN results. In Section 5.3, we use GCNJK and MixHop as base models for minibatching evaluation. For completeness, we also include results for using GCN as a base model in Table 5.3. The results are qualitatively the same, though GCN generally performs worse than the other two GNNs.

C.3 Experiments on Prior Datasets

Although the Pei et al. [58] datasets suffer issues with evaluation of graph learning methods as discussed in the main paper, we test LINKX on these datasets for comparison. We use the 10 fixed splits of [58], directly reporting results from papers where they also use the official splits, and re-running methods when this was not the case. In particular, GPR-GNN used larger 60-20-20 random splits, while GCNII did not evaluate on Actor and Squirrel. Thus, we re-run GPR-GNN and GCNII using the hyperparameters of Section B.1.

Table 10: Comparison of non-homophilous methods on datasets of Pei et al. [58] (collected by [61, 66, 48]). † represents re-run result, if unofficial dataset splits were used in their method paper, or the paper did not evaluate their method on the specific dataset. Best results up to a standard deviation are highlighted. Geom-GCN and GCNII did not report standard deviation information.

# Nodes	Texas 183	Wisconsin 251	Actor 7,600	Squirrel 5,201	Chameleon 2,277	Cornell 183
H_2GCN-1	84.86 ± 6.77	86.67 ± 4.69	35.86 ± 1.03	36.42 ± 1.89	57.11 ± 1.58	82.16 ± 4.80
H_2GCN-2	82.16 ± 5.28	85.88 ± 4.22	35.62 ± 1.30	37.90 ± 2.02	59.39 ± 1.98	82.16 ± 6.00
MixHop	77.84 ± 7.73	75.88 ± 4.90	32.22 ± 2.34	43.80 ± 1.48	60.50 ± 2.53	73.51 ± 6.34
GPR-GNN	76.22 ± 10.19†	75.69 ± 6.59†	33.12 ± 0.57†	54.35 ± 0.87†	62.85 ± 2.90†	68.65 ± 9.86†
GCNII	77.84	81.57	34.36 ± 0.77†	56.63 ± 1.17†	62.481	76.46
Geom-GCN-I	57.58	58.24	29.09	33.32	60.31	56.76
Geom-GCN-P	67.57	64.12	31.63	38.14	60.90	60.81
Geom-GCN-S	59.73	56.67	30.30	36.24	59.96	55.68
LINKX	74.60 ± 8.37	75.49 ± 5.72	36.10 ± 1.55	61.81 ± 1.80	68.42 ± 1.38	77.84 ± 5.81

For these experiments, we used the following hyper-parameter grid for LINKX: $\text{MLP}_{\mathbf{A}} \in \{1, 2\}$, $\text{MLP}_{\mathbf{X}} \in \{1, 2\}$, hidden channels $\in \{64, 128, 256, 512\}$, number of MLP_f layers $\in \{1, 2, 3, 4\}$ learning rate in $\{0.05, 0.01, 0.002\}$ and dropout $\in \{0.0, 0.5\}$.

LINKX performs well on datasets with thousands of nodes, despite being primarily a scalable method, only falling short on the tiniest of datasets. Overall, evaluation on these datasets is very noisy, highly dependent on hyperparameter grid selection, and few conclusions can be drawn on the relative performance of different methods in non-homophily more broadly. In particular, note that methods that do well on these datasets do not necessarily do well on our large, non-homophilous datasets. For example, while MixHop is the best non-homophilous GNN on the datasets we present in this paper, it does poorly on the datasets of Pei et al. [58]; in contrast, H_2GCN achieves excellent performance here, but its design choices make it run out of memory on even medium-sized datasets.

C.4 Experiments on Homophilous datasets

Though LINKX was not designed to do well on small homophilous datasets, we also report LINKX on the homophilous datasets of Cora, Citeseer and Pubmed [74], with the standard 48/32/20 training, validation, and test proportions. We use the same hyper-parameter grid for LINKX as in Section C.3.

Table 11: Comparison of on homophilous datasets. Results other than LINKX reported from [82]. Best three results per dataset are highlighted.

	CiteSeer	PubMed	Cora
# Nodes	3327	19717	2708
H_2GCN-1	77.07 ± 1.64	89.40 ± 0.34	86.92 ± 1.37
H_2GCN-2	76.88 ± 1.77	89.59 ± 0.33	87.81 ± 1.35
MixHop	76.26 ± 1.33	85.31 ± 0.61	87.61 ± 0.85
GCN	76.68 ± 1.64	87.38 ± 0.66	87.28 ± 1.26
GAT	75.46 ± 1.72	84.68 ± 0.44	82.68 ± 1.80
LINKX	73.19 ± 0.99	87.86 ± 0.77	84.64 ± 1.13

D Further dataset details

D.1 Licenses

In this section, we note the licenses of the datasets we collect:

- wiki: Wikipedia is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported License and the GNU Free Documentation License, unversioned, with no invariant sections, front-cover texts, or back-cover texts.
- Penn94: The Facebook 100 datasets are available online (https://archive.org/details/oxford-2005-facebook-matrix), and to the best of our knowledge were not released with a license, though the corresponding paper has an arXiv non-exclusive license to distribute. Upon release, there were privacy concerns [83], as the data release may not have respected certain privacy settings, and the initial release of the data included unique identifiers. We use a version that does not include the unique identifiers, but of course may still be subject to deanonymization attacks. While we do not add any sensitive information to the dataset, we acknowledge that deanonymization is possible, though our work does not directly contribute to deanonymization risks.
- Pokec: We retrieved the data from SNAP; the original source of the data is [40]. To the best of our knowledge, the data was not released with a license. This is a social network, so there may be privacy concerns with the user data. Still, we include it as a suitable benchmark that has been previously used, as the dataset is large and has interesting feature information. We only provide numerical values and do not provide any of the raw text in the dataset.
- arxiv-year: The dataset is licensed under ODC-BY. We originally downloaded the data from the Open Graph Benchmark [31], and the data is a subset of the Microsoft Academic Graph [70]
- snap-patents: The data was originally publically released by NBER in 2001 in a working paper by [27]. To the best of our knowledge, the dataset was not released with a license.
- genius: The dataset is open-sourced by the authors with no attached license. It was originally introduced in a conference paper [43]. While this is a social network and thus may

face privacy concerns, we believe that the task of predicting undesired nodes can be very beneficial to society, so we benchmark methods on it. We only provide numerical values, and omit raw text information that has been previously released in the dataset.

- twitch-gamer: The dataset is open-sourced by the authors in a repository with the MIT license. It was originally introduced in an academic paper [46]. This is also a social network that may face privacy concerns, but the prediction task of detecting situationally undesired nodes may be socially beneficial, so we benchmark methods on it. There is no raw text in the dataset.

D.2 Dataset Properties

Penn94 [67] 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 [41] 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 [31] 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 [42, 41] 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 [43] 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 [60] 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.3. Node features are constructed using averaged title and abstract GloVe embeddings [59]. Labels represent total page views over 60 days, which are partitioned into quintiles to make five classes.

D.3 Wiki collection details

Here, we detail the process of crawling and cleaning the wiki dataset. We generated the graph using a breadth-first search, where we started from the Wikipedia page on Hilbert Spaces, then proceeded to visit all its neighbors, and so forth. For each Wikipedia page visited, we used the MediaWiki web service API to get a list of pages linked to by the given page — this forms the directed edges of the graph. In each API query, we also received the number of page views per day in the past 60 days. We crawled these articles throughout April and May of 2021. To convert this into discrete labels, we used an even quantile function, which set view boundaries to make the number of nodes in each class as even as possible. The output of this function formed the labels of the graph.

For the node features, we formed 300 dimensional Wikipedia Glove vectors [59] for each word in the title and abstract, then averaged the word vectors in the title and abstract, thus resulting in 600 dimensional feature vectors for each node. For words not found in the Glove dictionary, we used the

zero vector. This procedure was modeled on the construction of the ogbn-arxiv dataset by [31]. In particular, we avoided a one-hot vector because of the vast dimensionality that would be required. Finally, to clean the dataset, we pruned edges if either of the nodes that it spanned where not in our subset of wikipedia articles. Ultimately, we decided to stop collection at approximately one third of English wikipedia due to limitations of computational resources and time. Already, it is not possible to run full batch experiments on the wiki dataset, while requiring over 80GB of CPU RAM for some minibatching techniques. As such, we believe that the full wikipedia dataset would have been too large and unwieldy to use as an evaluation dataset for many research labs, and our dataset is at a good size that may hopefully provide utility to many researchers.