Journal of Data-centric Machine Learning Research (2025) Submitted 01/25; Revised 08/25; Published 08/25

SEESAW: Do Graph Neural Networks Improve Node Representation Learning for All?

Yushun Dong YUSHUN.DONG@FSU.EDU
Florida State University
Tallahassee, FL, USA

William Shiao WSHIA002@UCR.EDU
University of California, Riverside
Riverside, CA, USA

Yozen Liu YLIU2@SNAP.COM
Snap Inc.
Santa Monica, CA, USA

Jundong Li JUNDONG@VIRGINIA.EDU
University of Virginia
Charlottesville, VA, USA

Neil Shah NSHAH@SNAP.COM
Snap Inc.
Bellevue, WA, United States

Tong Zhao TONG@SNAP.COM
Snap Inc.
Bellevue, WA, United States

Reviewed on OpenReview: *https://openreview.net/forum?id=zfPqOouLFn*

Abstract

Graph Neural Networks (GNNs) have garnered increasing attention in recent years, given their significant proficiency in various graph learning tasks. Consequently, there has been a notable transition away from the conventional and prevalent shallow graph embedding methods which pre-dated GNNs. However, in tandem with this transition which is presupposed in the literature, an imperative question arises: do GNNs always outperform shallow embedding methods in node representation learning? This question remains inadequately explored, as the field of graph machine learning still lacks a systematic understanding of their relative strengths and limitations. To address this gap, we propose a principled framework that unifies the ideologies of representative shallow graph embedding methods and GNNs. With comparative analysis, we show that GNNs actually bear drawbacks that are typically not shared by shallow embedding methods. These drawbacks are often masked by data characteristics in commonly used benchmarks and thus not well-discussed in the literature, leading to potential suboptimal performance when GNNs are indiscriminately adopted in applications. We further show that our analysis can be generalized to GNNs under various learning paradigms, which provides further insights to emphasize the research significance of shallow embedding methods. Finally, with these insights, we conclude with a guide to meet various needs of researchers and practitioners.

Keywords: Graph Learning, Comparative Benchmark, Graph Neural Networks

1 Introduction

Graph-structured data is ubiquitous across a plethora of applications, including recommender systems (Ying et al., 2018b; Fan et al., 2019; Tang et al., 2022), predictive user behavior models (Pal et al., 2020; Zhao et al., 2021), and chemistry analysis (You et al., 2018; Li et al., 2018). To gain deeper understanding on graph data and exploit the rich relational information, there has been a surge of interest in learning informative representations for graphs (Hamilton et al., 2017a; Xu et al., 2019). These methods typically learn representations via optimizing mappings that encode nodes or subgraphs as data points in a low-dimensional hidden space (Kipf and Welling, 2017). Their primary goal is to preserve as much task-relevant information of the graph (e.g., the proximity of nodes over the graph topology) as possible. Once the mapping is optimized, the learned representations can serve as the input features to perform a wide spectrum of downstream tasks on graphs, such as node classification (Kipf and Welling, 2017) and link prediction (Zhang and Chen, 2018).

In general, commonly used graph representation learning methods in practice can be divided into two categories, i.e., *shallow graph embedding methods* and *deep graph learning methods* (Hamilton et al., 2017b). Shallow graph embedding methods are mostly characterized by using an embedding lookup table as the mapping from nodes to their representations. For example, DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) directly consider node representations as free parameters. These representations are optimized with a skip-gram model (Mikolov et al., 2013) based on randomly generated walks. On the other hand, deep graph learning methods learn mappings from node attribute space to the latent space. For example, GNNs typically take node attributes and graph topology as input and exploit the topology and attribute information concurrently via neighborhood aggregation. In practice, GNNs are often found to show superior performance in node representation learning to power various tasks over attributed graphs (Dwivedi et al., 2023), such as node classification (Kipf and Welling, 2017; Xu et al., 2019; Dwivedi et al., 2023), link prediction (Zhang and Chen, 2018; Chamberlain et al., 2023; Ying et al., 2018b), and graph classification (Xu et al., 2019; Ying et al., 2019). Such a success has placed GNNs among the most popular graph representation learning methods, attracting increasing attention from researchers and practitioners (Zhou et al., 2020; Wu et al., 2020).

Nevertheless, close on the heels of the tremendous success of GNNs, several recent studies have revealed that GNNs may also exhibit weaknesses in downstream tasks compared with shallow embedding methods across different scenarios (Wang et al., 2022a; Chamberlain et al., 2023; Kipf and Welling, 2016). For example, DeepWalk can easily outperform Variational Graph Auto-Ecoders (Kipf and Welling, 2016), which is commonly believed to exhibit better performance, on multiple real-world graph datasets (Wang et al., 2022a). Additionally, multiple other shallow embedding methods (Bordes et al., 2013; Trouillon et al., 2016; Yang et al., 2015; Postavaru et al., 2020) also exhibited superior performances over GNNs in link prediction tasks (Chamberlain et al., 2023). Moreover, graph embedding methods have been widely deployed in various high-stake application scenarios to aid decision making in industry (Dong et al., 2023a; Chang et al., 2021). Correspondingly, if practitioners shift from shallow embedding methods to GNNs without careful proof-of-concept evaluations, they may end up with suboptimal results when GNNs are not suitable for their data and task (Altae-Tran et al., 2017; Chen et al., 2018; Li et al., 2017). There-

fore, given the rising interest in GNNs within the graph machine learning field, there is an urgent need to have a systematic understanding about when GNNs fall short in node representation learning (compared with shallow embedding methods), which helps foster the meticulous advancement of this area. To bridge this research gap, we ask:

When do GNNs exhibit drawbacks compared with shallow embedding methods?

To answer this question, we pioneer a comprehensive investigation SEESAW (**S**hallow **E**mbedding M**E**thods ve**S**us Gr**A**ph Neural Net**W**orks) to systematically compare the two branches of node representation learning methods. Specifically, we first perform a systematic analysis to compare the data flow of the two branches with a unified framework. Through such analysis, we attribute the primary differences between shallow embedding methods and GNNs to two factors: *(i)* whether the learning method uses a prior based on node attributes for representation learning; and *(ii)* whether the learning method explicitly performs neighborhood aggregation. Then we present a comprehensive study to compare the performance of methods from the two branches, and explore whether these differences bring drawbacks to GNNs or not. Despite the significant performance superiority of GNNs in most use-cases, we highlight two key drawbacks based on their differences from shallow embedding methods. First, in terms of the learning priors, we found that when only a limited number of attributes are available (i.e., in attribute-poor scenarios), the representations yielded by GNNs usually collapse into a lower-dimensional subspace (instead of spanning the entire available hidden space), a.k.a. dimensional collapse (Zhuo et al., 2023; Jing et al., 2022; He and Ozay, 2022). Second, in terms of neighborhood aggregation, we found that performing aggregation is prone to jeopardizing the performance over certain subgroups, e.g., nodes connected in a heterophilic manner. Armed with the above-mentioned observations, we emphasize the research significance of shallow embedding methods, and present a guide for practitioners to select an appropriate learning models given their settings. In particular, despite the overall performance superiority of GNNs, we suggest adopting shallow embedding methods instead of more commonly used GNNs in *(i)* attribute-poor scenarios, as shallow embedding methods excel at avoiding dimensional collapse by avoiding using node attributes; *(ii)* highly heterophilic networks, as shallow embedding methods do not perform neighborhood aggregation that jeopardizes the performances of heterophilic nodes.

2 Preliminaries

Notations. We denote an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of n nodes; $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Let $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and $\boldsymbol{X} \in \mathbb{R}^{n \times c}$ be the adjacency matrix and attribute matrix of $\mathcal{G}$, respectively. Here n represents the total number of nodes, while c is the number of dimensions[1] of the node attributes. In self-supervised node representation learning, an embedding model is denoted as $f_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ denotes the learnable parameters. Specifically, $f_{\boldsymbol{\theta}}$ takes a node v_i as input, and outputs its associated embedding. In node classification, a decoder takes the embedding of a node as input, and outputs the predicted label. In link prediction, a decoder takes the representations of a pair of nodes as input and output the probability of being connected.

1. For simplicity, we refer to the total number of dimensions of a space as its dimensionality.

Shallow Embedding Methods. Common shallow embedding methods include those based on matrix factorization and those based on random walks. Without loss of generality, in this paper, we focus on the walk-based ones, since they are observed to yield better performance and thus become the most popular options amongst shallow embedding methods (Hamilton et al., 2017b). Specifically, the mapping from nodes to representations in walk-based shallow embedding methods is usually an embedding lookup table. Such mapping is optimized to extract topological information into node representations. We formulate the mapping as $f_{\boldsymbol{\theta}}(v_i) = \boldsymbol{Z}\boldsymbol{v}_i$, where $\boldsymbol{Z} \in \mathbb{R}^{d\times n}$ is a matrix of representations, while $\boldsymbol{v}_i \in \mathbb{I}^n$ is a one-hot vector indicating the column in $\boldsymbol{Z}$ associated with node v_i. Here the learnable parameter set $\boldsymbol{\theta} = \{\boldsymbol{Z}\}$, which is usually optimized with a walk-based objective. We denote the embedding of node v_i as $\mathbf{z}_{v_i}$ (i.e., the i-th column of $\boldsymbol{Z}$), and a common walk-based objective (Perozzi et al., 2014) is

$$\mathscr{L}_{\text{walk}}\left(\mathbf{z}_{v_i}\right) = -\log\left(\sigma\left(\mathbf{z}_{v_i}^{\top}\mathbf{z}_{v_j}\right)\right) - Q\cdot\mathbb{E}_{v_k\sim P_n(v)}\log\left(\sigma\left(-\mathbf{z}_{v_i}^{\top}\mathbf{z}_{v_k}\right)\right). \tag{1}$$

Here $\mathbf{z}_{v_j}$ denotes the embedding of node v_j, which is a node that co-occurs near v_i within a fixed-length random walk; $\sigma(\cdot)$ denotes the activation function; Q is the number of negative samples; P_n is a negative sampling distribution. The topology proximity can be preserved in the embedding of each node via optimizing $\mathscr{L}_{\text{walk}}$ for each node.

Graph Neural Networks (GNNs). There have been a plethora of GNNs designed for different purposes over the years. Here we introduce the general paradigm of GNNs (Wu et al., 2020). Typically, a GNN model takes the input $\mathcal{G}$ and outputs $\boldsymbol{Z}$ as the learned embedding matrix for the nodes in $\mathcal{V}$. The basic operation of GNN between l-th and $(l+1)$-th layer can be summarized as $\boldsymbol{z}_{v_i}^{(l+1)} = \sigma(\text{COMBINE}(\boldsymbol{z}_{v_i}^{(l)}, \text{AGG}(\{\boldsymbol{z}_{v_j}^{(l)} : v_j \in \mathcal{N}(v_i)\})))$, where $\boldsymbol{z}_{v_i}^{(l)}$ and $\boldsymbol{z}_{v_i}^{(l+1)}$ is the embedding of node v_i at l-th and $(l+1)$-th layer, respectively. In the first layer, $\boldsymbol{z}_{v_i}^{(0)}$ can be initialized as the input node feature $\boldsymbol{x}_{v_i}$. $\mathcal{N}(v_i)$ is the set of one-hop neighbors of v_i according to $\boldsymbol{A}$. AGG$(\cdot)$ represents the aggregating function, e.g., weighted sum. COMBINE$(\cdot)$ is the combining function for output of AGG$(\cdot)$ and $\boldsymbol{z}_{v_i}^{(l)}$, which combines the representation from the centering node and the representations of its neighbors. Various objective functions can be adopted to optimize GNNs, including supervised (e.g., cross-entropy loss in classification) and self-supervised objectives (e.g., the objective in Equation 1).

3 Shallow Embedding Methods vs. GNNs: A Unified View

Here we design a systematic analysis named SEESAW to characterize the connections and differences between shallow graph embedding methods and GNNs. Based on such analysis, we aim to reveal the benefits and drawbacks of GNNs brought by the identified differences.

To perform rigorous analysis between the two branches of representation learning methods, it is critical to enforce a fair comparison. As discussed in section 2, most popular shallow embedding methods (e.g., DeepWalk and node2vec) are optimized in a self-supervised learning paradigm with an objective based on random walks On the other hand, GNNs can be optimized either in an end-to-end or self-supervised learning paradigm with various types of objectives. Considering the overlapping in the objectives and learning paradigms of two

branches, we utilize the widely studied self-supervised learning paradigm with a walk-based optimization objective (Equation 1) (Hamilton et al., 2017a) to establish a unified view.

We unify the data flow of the two branches of methods from the perspective of prior-posterior process, with Figure 1 presenting an overview of it. Specifically, we consider the distribution of node representations in the hidden space after model initialization as the prior distribution. For shallow graph embedding methods, as their embedding matrix is randomly initialized, the adopted distribution for node embedding initialization is the prior for representation learning, e.g., a uniform distribution. For GNNs e.g., GCN (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017a), the node attributes transformed by the randomly initialized learnable parameters are regarded as the prior distribution of node representations in the hidden space. Based on the prior formulations, we characterize the first difference below.

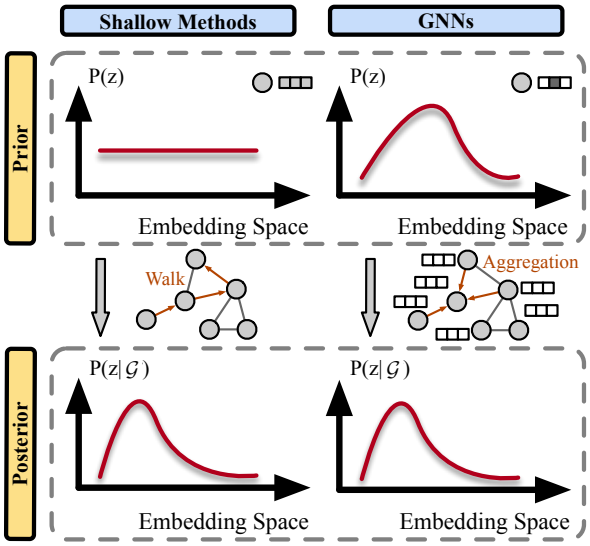


Figure 1: A unified view of data flow.

Difference 1 (Difference in Learning Priors). *Shallow embedding methods take any distribution as the prior of representations in the hidden space, while GNNs take the transformed node attributes as the prior of representations.*

Both branches of models perform optimization w.r.t. the graph topology based on the prior. As such, the proximity of nodes over the topology could be preserved in the learned representations. For shallow graph embedding methods, the prior distribution is directly optimized with a walk-based objective function. For GNNs, the output node representations can be optimized with the same objective, but only after the layer-wise neighborhood aggregation is performed. Therefore, we characterize the second difference from the perspective of neighborhood aggregation.

Difference 2 (Difference in Updating Operations). *Shallow graph embedding methods do not explicitly perform neighborhood aggregation, while GNNs do.*

According to the two differences characterized above, we formulate the research problem this paper aims to explore, with the goal of revealing the implicit drawbacks of GNNs compared with shallow methods and re-calibrate the meticulous advancement of this field.

Problem 1 *Given an attributed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, consider the above two classes of node representation learning methods, i.e., the shallow embedding methods and GNN methods, we seek to characterize (1) the implicit drawbacks of using an attribute-based prior; and (2) the implicit drawbacks of performing neighborhood aggregation.*

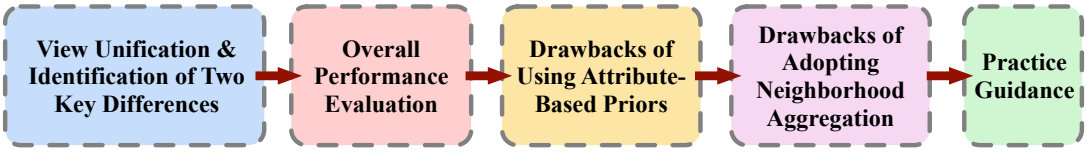


Figure 2: The pipeline of SEESAW: exploring the drawbacks of (1) using attribute-based priors and (2) adopting neighborhood aggregation of GNNs.

In the following section, we conduct comprehensive experiments to analyze what the two differences bring to GNNs. Through thorough analysis, we aim to present a systematic understanding of *(i)* the overall superiority of GNNs when node attributes are abundant and *(ii)* the scenarios where shallow embedding methods exhibit superiority while GNNs fall short. We show a brief pipeline of the proposed SEESAW exploratory study in Figure 2. Additionally, we note that GNNs can also be optimized with other learning paradigms such as contrastive learning (Zhu et al., 2020b; Ying et al., 2018a) and end-to-end training (Kipf and Welling, 2017). Nonetheless, they generally still adhere to the unified view delineated in Figure 1. Consequently, we also incorporate them to facilitate a comprehensive and generalizable analysis below.

4 Empirical Analysis: What Do the Differences Bring to GNNs?

4.1 Experimental Settings

Datasets and Tasks. We conduct experiments with 10 commonly used real-world graph benchmark datasets with different scales, including `Cora` (Yang et al., 2016), `CiteSeer` (Yang et al., 2016), `PubMed` (Yang et al., 2016), `CoraFull` (Bojchevski and Günnemann, 2018), `DBLPFull` (Bojchevski and Günnemann, 2018), `Amazon-Computers` (Shchur et al., 2018), `Amazon-Photo` (Shchur et al., 2018), `Coauthor-CS` (Shchur et al., 2018), `Coauthor-Physics` (Shchur et al., 2018), and `Flickr` (Zeng et al., 2020). More details including dataset statistics are in subsection B.1. We present the most representative results in this section, and we include more comprehensive evaluations in Appendix C.

Models. We adopt representative models from shallow embedding methods and GNNs for analysis. Specifically, we utilize DeepWalk (Perozzi et al., 2014) as the representative shallow embedding method, and select GCN with the walk-based loss in Equation 1 (Walk-GCN) as the default GNN for comparison, unless otherwise indicated. To take a step further, we will also present comprehensive empirical results of GNNs in different designs and learning paradigms to demonstrate the generality of our analysis. In terms of GNNs with different designs, there are multiple GNNs designed with contrastive and non-contrastive objectives under the same self-supervised learning paradigm (Shiao et al., 2023). We adopt four representative ones from both branches to study. For contrastive self-supervised GNNs, we adopt GRACE (Zhu et al., 2020b) and a GCN trained with max-margin loss (ML-GCN) (Ying et al., 2018a). For non-contrastive self-supervised GNNs, we adopt Graph Barlow Twins (GBT) (Bielak et al., 2022) and Bootstrapped Graph Latents (BGRL) (Thakoor et al., 2022). In terms of different learning paradigms, we also adopt the vanilla GCN trained in an end-to-end manner (E2E-GCN) for comparison. We report the average results across

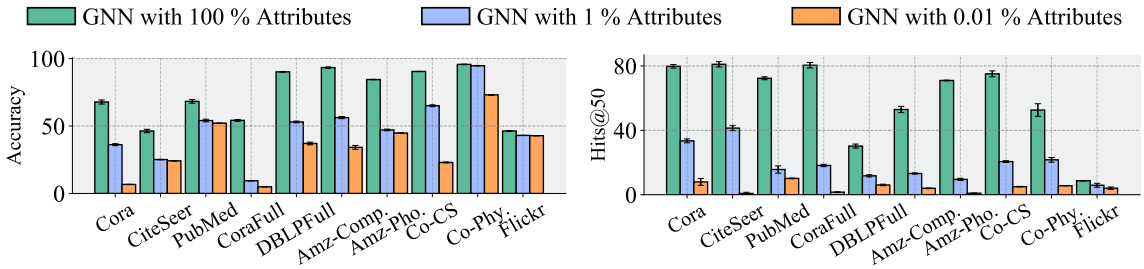
Table 1: Node classification accuracy comparison between shallow methods and GNNs trained under different learning paradigms on 10 real-world graph datasets, with the best performances highlighted in **Bold**. Best performances showing statistically significant superiority over the shallow method are marked with grey. All accuracy values are in percentages.

	Shallow	**Walk-GCN**	**GBT**	**BGRL**	**ML-GCN**	**GRACE**	**E2E-GCN**
Cora	69.33 ± 0.9	70.07 ± 1.4	75.63 ± 1.4	74.33 ± 1.0	71.87 ± 1.9	80.87 ± 0.4	**81.40 ± 0.4**
CiteSeer	46.37 ± 0.9	51.77 ± 1.4	56.90 ± 0.8	57.03 ± 2.8	53.07 ± 0.3	69.83 ± 0.7	**70.90 ± 0.5**
PubMed	60.77 ± 0.3	71.13 ± 2.9	78.63 ± 0.3	77.27 ± 3.5	72.03 ± 1.7	**80.50 ± 0.4**	79.00 ± 0.4
CoraFull	50.37 ± 1.1	53.77 ± 0.5	**57.69 ± 0.7**	54.10 ± 0.9	49.32 ± 0.3	50.23 ± 1.1	52.18 ± 8.3
DBLPFull	81.68 ± 0.7	84.54 ± 0.4	85.07 ± 0.0	84.74 ± 0.3	81.66 ± 0.4	83.25 ± 0.3	**85.19 ± 0.4**
Amz-C.	88.23 ± 0.8	88.74 ± 1.3	89.15 ± 0.3	88.46 ± 0.6	89.12 ± 0.5	86.30 ± 0.2	**91.03 ± 0.5**
Amz-P.	92.57 ± 0.5	93.64 ± 0.3	93.07 ± 0.4	**93.68 ± 0.5**	93.36 ± 0.5	92.07 ± 0.2	91.66 ± 0.6
Co-CS	87.69 ± 0.2	90.08 ± 0.3	**93.75 ± 0.2**	92.92 ± 0.1	92.95 ± 0.2	92.68 ± 0.6	93.23 ± 0.1
Co-Phy.	93.40 ± 0.4	95.83 ± 0.3	95.84 ± 0.2	95.74 ± 0.0	95.19 ± 0.0	OOM	**95.86 ± 0.2**
Flickr	**52.45 ± 0.1**	46.26 ± 0.2	51.87 ± 0.1	51.89 ± 0.2	51.16 ± 0.3	OOM	48.19 ± 0.2

three separate runs with standard deviation. We present more experimental details such as dataset split, evaluation protocol, and implementation details in Appendix B.2. We further show a detailed comparison of the complexity and the scalability between the adopted methods in in Appendix C.14.

4.2 Overall Performance Evaluation

We first present an overall performance comparison between shallow embedding methods and GNNs in Table 1. Without loss of generality, we take node classification as an exemplary downstream task. We observe that GNNs exhibit significant superiority over shallow embedding methods in most datasets. We assume that such superiority comes from two perspectives, where each perspective associates with one difference identified in section 3. First, in terms of the learning prior, GNNs are able to exploit the information encoded in the node attributes, which could lead to more task-relevant information in the learned node representations. As a comparison, shallow embedding methods typically are not capable of incorporating node attributes. Second, in terms of neighborhood aggregation, GNNs are able to exploit more abundant localized information by explicitly performing neighborhood aggregation between a node and its direct neighbors. On the other hand, shallow embedding methods preserve the topological proximity only through optimizing the walk-based loss, which may leave relatively more neighbors unexplored. We note that consistent observations have also been reported in recent studies on other graph learning tasks such as link prediction (Shiao et al., 2023; Zhu et al., 2020b; Thakoor et al., 2022). We present more discussions in Appendix C. Interestingly, the only dataset which shallow embedding method exhibits superiority on is Flickr. We attribute such phenomenon to the limited number of available node attributes and less homophily in this dataset, which will be further discussed in the following subsections.



(a) Accuracy in Node Classification.

(b) Hits@50 in Link Prediction.

Figure 3: A comparison between GNNs with different ratios of available node attribute dimensionality in node classification and link prediction. (a): Comparison of node classification accuracy; (b): Comparison of Hits@50 in link prediction.

Despite the significant superiority of GNNs over shallow embedding methods, we found GNNs could also exhibit clear drawbacks associated with the two differences. We discuss the drawbacks of GNNs brought by 1 and 2 in the following two subsections.

4.3 Are There Any Drawbacks of Using An Attribute-Based Prior?

We note that using an attribute-based prior could enable GNNs to exploit the information from both attributes and graph topology, which usually brings advantages. However, we found that their differences may also jeopardize the performance of GNNs in certain scenarios. Here we focus on the potential drawbacks brought by using a prior based on node attributes in GNNs, i.e., 1. Our rationale is that if the prior is directly obtained based on node attributes, then the performance of GNNs could also heavily rely on the quality of attributes. Nevertheless, we note that in practice, the rich high-dimensional node attributes are not always available. In those attribute-poor scenarios (i.e., when the attributes are only partially available), GNNs may struggle to learn high quality node representations, and thus could also end up with limited performance in downstream tasks. On the contrary, shallow graph embedding methods typically do not bear such a problem since they usually do not rely on node attributes. In light of this, here we study the drawbacks of GNNs in attribute-poor scenarios. We present our observations below, and we introduce the complementary results under the same settings in Appendix C.

Observation 1: GNN Performance Drops Under Limited Input Attributes. We first compare the performance of GNNs under different attribute dimensionalities by manually controlling the number of available attribute dimensions. Specifically, we refer to the ratio of available attribute dimensions for GNN as *attribute dimensionality ratio*, and we vary such a ratio in {100%, 1%, 0.01%} with a minimum number of node attribute dimensionality being one. Here, the subset of attributes are randomly selected from the original node attributes. We present the performance comparison between different ratios of attribute dimensions on node classification and link prediction in Figure 3(a) and Figure 3(b), respectively. We observe that, when using 100% node attributes, GNN yields satisfying performance on all datasets, demonstrating the superiority of GNNs when abundant attributes are available. However, when the attribute dimensionality becomes limited,

i.e., in cases with 1% and 0.01% node attributes, the performance of GNN drops significantly in both tasks. Such a phenomenon reveals that limited attribute dimensionality jeopardizes the performance of GNNs. As a comparison, shallow embedding methods typically do not bear such an issue, since they usually do not take node attributes as input. In fact, when the attribute dimensionality is limited, the representations yielded by GNNs collapse into a lower-dimensional subspace instead of spanning the entire hidden space, a.k.a. dimensional collapse (Zhuo et al., 2023; Jing et al., 2022; He and Ozay, 2022). We provide analysis between dimensional collapse and the common learning low-rank node representations in Appendix C.12 and show more detailed discussion in Observation 2 below.

Observation 2: Limited Input Attributes Typically Cause Dimensional Collapse. To explore to what extent dimensional collapse happens, here we characterize the *effective dimension* as the rank of the learned embedding matrix. Then, we measure the level of dimensional collapse with the ratio of the effective dimension number to the total number of hidden dimensions, i.e., r/d, where r is the rank of the node representation matrix and d is dimension of the hidden space. As a comparison, shallow embedding method consistently yields a maximal effective dimension ratio, As such, the lower the ratio, the more severe the dimension collapse becomes. Figure 4 presents the effective dimension ratio across different datasets and attribute dimensionalities in node classification task. We observe that more available node attribute dimensions play a critical role in learning representations that span a higher dimensional space, which prevents GNNs from dimensional collapse. which attributes to the difference in their learning priors. Specifically, shallow embedding methods randomly initialize the representations, and thus it consistently tends to learn full-rank node representation matrix. As such, in attribute-poor scenarios, it's difficult for GNNs to learn representations spanning a higher dimensional space. We present further discussions on the ranks of the learned representations in subsection C.4.

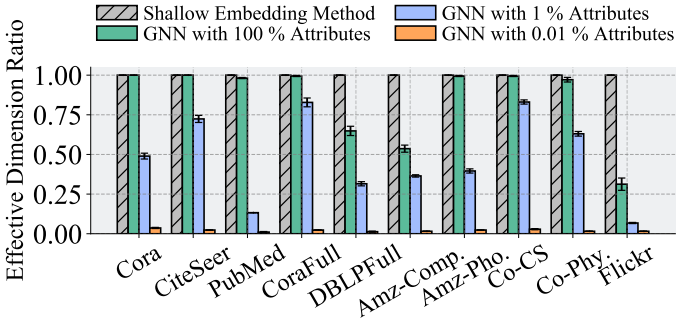
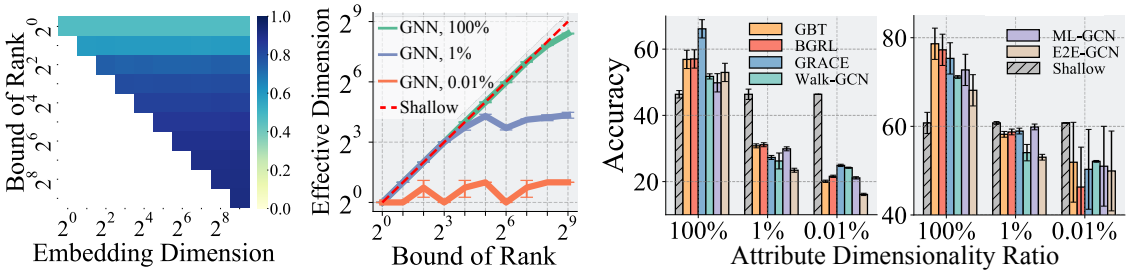


Figure 4: Representation effective dimension ratios.

Observation 3: Dimensional Collapse Ties to Performance & Attribute Dimensionality. To further understand the influence of the dimensionality of the representation subspace, we conduct experiments by enforcing a bound over the effective dimensionality of the learned representations. In this way, we are able to adjust other factors while avoiding increasing rank (e.g., increasing the dimension of representations). This allows us to manually control the level of dimensional collapse in the learned embedding matrix and explore how rank reduction restricts the embedding space and hinders the model performance. For example, if the performance is only improved when the rank is allowed to improve while it does not change w.r.t. other factors, we are then confident to say that it is the rank being too low that is affecting the performance. Specifically, we propose to consider the learned embedding matrix from GNNs as $\boldsymbol{Z} = \boldsymbol{C}\boldsymbol{F}$, where $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$, $\boldsymbol{C} \in \mathbb{R}^{n\times r}$, and $\boldsymbol{F} \in \mathbb{R}^{r\times d}$ $(1 \leq r \leq d)$. As such, r naturally serves as an upper bound of the rank for $\boldsymbol{Z}$ without changing the dimensionality of the hidden space. In practice, we consider the output matrix of



(a) Performance. (b) Collapse Tendency. (c) Performances on `CiteSeer` and `PubMed`.

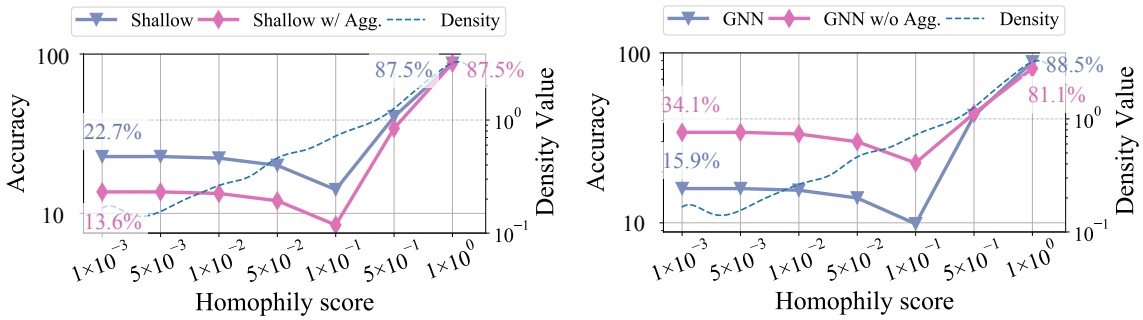
Figure 5: (a) and (b): Tendencies of performance and collapse under different rank bounds (enforced for embedding matrix). (c): Performance comparison of other popular GNNs across different available attribute dimensionality ratios.

GNN model and a matrix with learnable parameters as $\boldsymbol{C}$ and $\boldsymbol{F}$, respectively. First, we present the tendency of node classification accuracy on `Amazon-Computers` in Figure 5(a), where both r and d vary within a wide range between 2^0 and 2^9 to cover most commonly used values. We observe that the performance (indicated by the color) improves as long as the bound of the rank r improves under any embedding dimension number d. This further demonstrates that rank being too low can severely affect the performance no matter how the embedding dimension is improved or reduced. In other words, spanning a higher-dimensional subspace (within the available hidden space) is beneficial for the quality of the learned representations. Second, we present the tendency of effective dimension w.r.t. the bound of rank on `DBLPFull` under different input attribute dimensionalities in Figure 5(b). We observe that, in attribute-poor scenarios, it is difficult for GNNs to learn representations with large effective dimensions even if the bound raises to a larger value. We note that such observations are consistent across datasets and provide additional results and discussion in subsection C.3 and subsection C.4.

Finally, Figure 5(c) shows the performances of DeepWalk and GNNs under different learning paradigms under different node attribute availabilities. We can observe that the performance reduction due to dimensional collapse (as discussed above) can also be observed with other GNNs. By contrast, shallow embedding method learns representations spanning a maximum subspace (i.e., effective dimensionality equals to the bound of rank). These observations demonstrate that small attribute dimensionality indeed prevents a wide spectrum of GNNs from learning representations that span a larger subspace, while shallow embedding methods typically do not bear such an issue.

4.4 Are There Any Drawbacks of Performing Neighborhood Aggregation?

In this subsection, we explore the potential drawbacks brought by performing neighborhood aggregation in GNNs, i.e., 2. We note that explicitly performing neighborhood aggregation enables GNNs to extract information from its direct neighbors. In this way, GNNs allows each node to take advantage of abundant localized information around it. However, multiple existing works have pointed out that the neighborhood aggregation mechanism could jeopardize the performance of GNNs (Zhu et al., 2021; Luan et al., 2022). For example, when most



(a) Shallow Methods w/ & w/o Aggregation.

(b) GNNs w/ & w/o Aggregation.

Figure 6: Cumulative node classification accuracy comparison across nodes with different levels of homophily: (a) shallow embedding methods w/ vs. w/o neighborhood aggregation; (b) GNNs w/ vs. w/o neighborhood aggregation. Performance is on `Amazon-Computers` for both figures. The dashed curves represent density functions of node homophily score.

labels of a node's neighbor are different from its own (i.e., heterophilic cases), the learned embedding associated with this node could be misled by the information aggregated from its neighbors. To exclude the influence of differences in their priors, we propose to adopt *(i)* shallow embedding methods w/ neighborhood aggregation and *(ii)* GNNs w/o neighborhood aggregation for comparison with normal shallow embedding methods and GNNs, respectively. Specifically, for the former, we add a layer of the mean aggregator (Hamilton et al., 2017a) in DeepWalk during both training and inferencing. For the latter, we remove the neighborhood aggregation before the non-linear transformation.

Observation 4: Heterophilic Nodes Barely Benefit From Aggregation. We empirically validate whether neighborhood aggregation mechanism could jeopardize the performance of GNNs or not in the self-supervised learning paradigm. Figure 6(a) and Figure 6(b) present comparisons between (i) shallow embedding methods w/ and w/o neighborhood aggregation and (ii) GNNs w/ and w/o neighborhood aggregation, respectively. The homophily score of each node is measured with the ratio of its neighbors with the same labels (as this node) to the total number of its neighbors (Zhu et al., 2020a).

With the settings above, we can observe that the overall performance (i.e., the performance at homophily score equals to one) are similar between w/ and w/o aggregation for shallow embedding method, while the overall performance for GNN w/o aggregation reduces significantly compared with vanilla GNN. This demonstrates that neighborhood aggregation could be critical to effectively exploit the information encoded in node attributes.

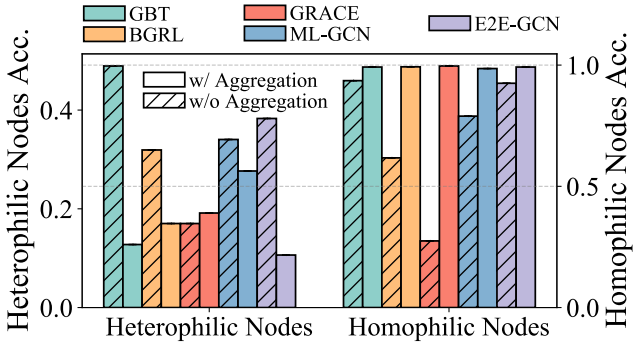


Figure 7: Performance comparison of other GNNs across heterophilic and homophilic nodes.

This validates the phenomenon of heterophily nodes suffering from neighborhood aggregation. Similar observations are found on other datasets, and more results are in subsection C.5. Furthermore, we observe that neighborhood aggregation reduces the performance on nodes with low levels of homophily (i.e., high levels of heterophily) for both models.

Finally, Figure 7 shows the performances of different GNNs w/ and w/o neighborhood aggregation for top 10% most heterophilic and homophilic nodes in `Amazon-Photo`. We observe that performance reduction from neighborhood aggregation widely exists in different GNNs across different learning paradigms.

5 Discussion: A Guide for Practitioners

Based on the discussion above, we conclude that it is necessary to meticulously select the branches of models to use, instead of adopting GNNs as a panacea. Armed with such insights, in this section, we provide a guide for practitioners to choose an appropriate type of models to learn high-quality node representations, such that the performance in various downstream tasks can be improved. It's worth noting that neither shallow embedding methods nor GNNs are flawless: it is difficult for shallow embedding methods to properly exploit information encoded in node attributes, while GNNs also bear drawbacks as discussed in subsection 4.3 and subsection 4.4. To properly handle their drawbacks, a straightforward way to leverage the advantages from both is by simply combining the representations learned from both types of models. The rationale is two-fold: *(i)* The number of effective dimensions of the learned representations (i.e., the rank of the node representation matrix) from GNNs could be promoted by representations from shallow methods, which helps to tackle the problem of dimensional collapse even in attribute-poor scenarios (i.e., when node attributes are only partially available). *(ii)* Information from both aggregated and unaggregated node attributes could be preserved at the same time, which improves the performance on heterophilic nodes.

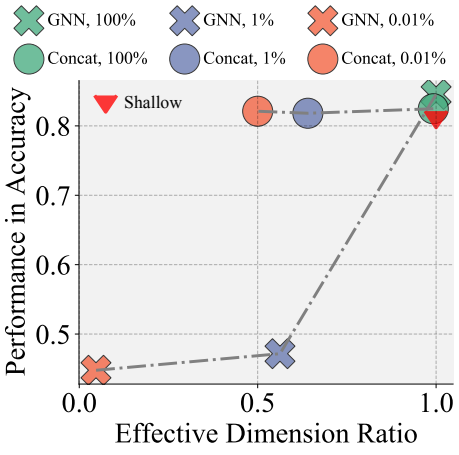


Figure 8: Comparison between GNNs, GNNs with concatenated representations from shallow embedding method (*Concat*), and shallow embedding methods (*Shallow*).

As an example, we found that simply concatenating the representations from both models helps achieve satisfying performance across different attribute dimensionality ratios. We present a performance comparison between shallow method, GNN, and the strategy of using the concatenation of node representations yielded by the two methods on `DBLPFull` dataset in Figure 8. We observe that the performance and effective dimension ratio of GNNs reduce significantly when attribute dimensionality ratio declines. However, by simply combining the representations yielded by the two methods, we are able to significantly reduce its sensitivity to attribute dimension-

ality ratio, achieving a more stable performance along both axes. We also have similar observations on other datasets. Nevertheless, such combination would result in significantly higher computational complexity as well as the need of maintaining two models instead of one, which is often much less preferred in industrial applications given cost and human resourcing concerns. Hence, it's of practical significance to provide guidance in choosing between the two branches of methods. Below, we formulate the guidance from two perspectives: *(i).* data perspective, as discussed in previous sections, and *(ii).* model perspective, which has been extensively discussed in literature. We further show a detailed empirical guidance about determining attribute-poor scenarios in Appendix C.13.

Data - Attribute-Rich vs. Attribute-Poor Networks. As discussed in subsection 4.3, GNNs often achieve superior performance in scenarios with rich attribute compared to shallow embedding methods. Correspondingly, adopting GNNs for representation learning on attribute-rich networks is an obvious choice. Nevertheless, in attribute-poor scenarios, e.g., when the attribute dimensionality is limited, GNNs are prone to exhibit dimensional collapse, while shallow embedding methods do not bear such a drawback. Therefore, we recommend adopting GNNs and shallow embedding methods on attribute-rich and attribute-poor networks, respectively.

Data - Homophilic vs. Heterophilic Networks. According to discussion in subsection 4.3, shallow embedding methods and GNNs exhibit different performance on nodes with different levels of heterophily. In particular, GNNs exhibit superior and inferior performance on homophilic and heterophilic nodes (compared with shallow embedding methods), respectively. A preliminary reason is that explicitly performing neighborhood aggregation is helpful for homophilic nodes while harmful to heterophilic nodes. Therefore, GNNs are recommended for representation learning if the network data is homophilic, otherwise shallow embedding methods could be more suitable.

Model - Transductive vs. Inductive Settings. As the shallow embedding methods rely on training an embedding vector for each of the node in the graph, they naturally do not support inductive learning. That is, given any newly appeared nodes, shallow embedding methods cannot produce it's representation without retraining or at least fine-tuning the model. On the other hand, as feature-based models, GNNs are naturally inductive and are able to inference node representations for the newly appeared nodes (Hamilton et al., 2017a). Hence, for use-cases such that the graphs are rapidly updating (e.g., social networks, e-commerce networks, etc), GNNs are recommended given their inductive bias, whereas shallow embedding methods require frequent costly retrains.

Model - Low-parameter vs. High-parameter Settings. Modern machine learning usually requires loading all learnable parameters into limited GPU memory to achieve higher training speed. The number of learnable parameters for shallow embedding methods grows linearly with the number of nodes. On the other hand, the parameters size of GNNs is only proportional to the dimension of node attributes and not the number of nodes. Therefore, for graphs with a large number of nodes, GPU training is not be feasible for shallow embedding methods without techniques such as parallelism (Li, 2023).

6 Conclusion

In this study, we aimed to emphasize the research significance of shallow embedding methods, challenging the prevailing emphasis on Graph Neural Networks (GNNs) in the field of graph representation learning. Specifically, we proposed a framework that unifies the data flow of representative shallow graph embedding methods and GNNs, allowing for systematic comparison. Through careful analysis, we characterized their primary differences from two perspectives: the use of attribute-based priors and explicit neighborhood aggregation. We then performed comprehensive experiments to analyze the benefits and drawbacks these differences bring to GNNs. Notably, we found that GNNs can suffer from dimensional collapse in attribute-poor scenarios and that neighborhood aggregation can harm performance for heterophilic nodes. These drawbacks were consistent across different GNN architectures and learning paradigms, highlighting their practical significance. Armed with these insights, we developed a structured guide for practitioners on selecting appropriate graph representation learning methods for different scenarios. We also demonstrated that in some cases, combining shallow methods with GNNs can leverage the strengths of both approaches. Our primary endeavor is to recalibrate the academic perspective, accentuating the benefits and drawbacks of GNNs compared with conventional shallow embedding methods. We hope our work enlighten researchers to foster the meticulous advancement.

Broader Impact Statement

This research has substantial positive societal impacts by advancing the effective deployment of graph learning technologies. By providing clear guidance on when to use simpler shallow embedding methods versus GNNs, it enables more efficient and equitable adoption of these technologies, particularly benefiting resource-constrained environments like developing regions and smaller organizations. The findings about handling heterophilic nodes and dimensional collapse can improve model performance across crucial applications. We do not see any significant negative societal impacts to discuss here.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their valuable feedback and Snap Inc. for supporting this work during the first author's internship.

References

Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. Watch your step: Learning node embeddings via graph attention. *Advances in neural information processing systems*, 31, 2018.

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.

Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

Amina Amara, Mohamed Ali Hadj Taieb, and Mohamed Ben Aouicha. Network representation learning systematic review: Ancestors and current development state. *Machine Learning with Applications*, 6:100130, 2021.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *Knowledge-Based Systems*, 256: 109631, 2022.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637, 2018.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9): 5103–5113, 2021.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. Mobility network models of covid-19 explain inequities and inform reopening. *Nature*, 589(7840):82–87, 2021.

Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Hejie Cui, Zijie Lu, Pan Li, and Carl Yang. On positional and structural node features for graph neural networks on non-attributed graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3898–3902, 2022.

Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE transactions on knowledge and data engineering*, 31(5):833–852, 2018.

Guimin Dong, Mingyue Tang, Zhiyuan Wang, Jiechao Gao, Sikun Guo, Lihua Cai, Robert Gutierrez, Bradford Campbel, Laura E Barnes, and Mehdi Boukhechba. Graph neural networks in iot: a survey. *ACM Transactions on Sensor Networks*, 19(2):1–50, 2023a.

Yushun Dong, Kaize Ding, Brian Jalaian, Shuiwang Ji, and Jundong Li. Adagnn: Graph neural networks with adaptive frequency response filter. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 392–401, 2021a.

Yushun Dong, Jian Kang, Hanghang Tong, and Jundong Li. Individual fairness for graph neural networks: A ranking based approach. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 300–310, 2021b.

Yushun Dong, Ninghao Liu, Brian Jalaian, and Jundong Li. Edits: Modeling and mitigating data bias for graph neural networks. In *Proceedings of the ACM web conference 2022*, pages 1259–1269, 2022.

Yushun Dong, Jing Ma, Song Wang, Chen Chen, and Jundong Li. Fairness in graph mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):10583–10602, 2023b.

Yushun Dong, Binchi Zhang, Zhenyu Lei, Na Zou, and Jundong Li. Idea: A flexible framework of certified unlearning for graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 621–630, 2024.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *J. Mach. Learn. Res.*, 24:43:1–43:48, 2023.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017a.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

Bobby He and Mete Ozay. Exploring the gap between collapsed & whitened features in self-supervised learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 8613–8634. PMLR, 2022.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Junying Li, Deng Cai, and Xiaofei He. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017.

Shen Li. Single-machine model parallel best practices, 2023. URL `https://pytorch.org/tutorials/intermediate/model_parallel_tutorial.html`.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022.

Seiji Maekawa, Koki Noda, Yuya Sasaki, et al. Beyond real-world benchmark datasets: An empirical study of node classification with gnns. *Advances in Neural Information Processing Systems*, 35:5562–5574, 2022.

Ilya Makarov, Dmitrii Kiselev, Nikita Nikitinsky, and Lovro Subelj. Survey on graph embeddings and their applications to machine learning problems on graphs. *PeerJ Computer Science*, 7:e357, 2021.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. Pinnersage: Multi-modal user embedding framework for recommendations at pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2311–2320, 2020.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

Stefan Postavaru, Anton Tsitsulin, Filipe Miguel Gonçalves de Almeida, Yingtao Tian, Silvio Lattanzi, and Bryan Perozzi. Instantembedding: Efficient local node representations. *CoRR*, abs/2010.06992, 2020.

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

William Shiao, Zhichun Guo, Tong Zhao, Evangelos E. Papalexakis, Yozen Liu, and Neil Shah. Link prediction with non-contrastive learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 787–795, 2023.

Xianfeng Tang, Yozen Liu, Xinran He, Suhang Wang, and Neil Shah. Friend story ranking with edge-contextual local graph convolutions. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1007–1015, 2022.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Velickovic, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022a.

Yu Wang, Yuying Zhao, Yushun Dong, Huiyuan Chen, Jundong Li, and Tyler Derr. Improving fairness in graph neural networks via mitigating sensitive attribute leakage. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 1938–1948, 2022b.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018a.

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, 2018b.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 6(1):3–28, 2018.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Tong Zhao, Tianwen Jiang, Neil Shah, and Meng Jiang. A synergistic approach for graph anomaly detection with pattern mining and feature learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2393–2405, 2021.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

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*, 33:7793–7804, 2020a.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11168–11176, 2021.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *CoRR*, abs/2006.04131, 2020b. URL `https://arxiv.org/abs/2006.04131`.

Zhijian Zhuo, Yifei Wang, Jinwen Ma, and Yisen Wang. Towards a unified theoretical understanding of non-contrastive learning via rank differential mechanism. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Appendix A. Related Work

Shallow Embedding Methods. Shallow embedding methods simply consider the mapping to representations in the hidden space as lookup tables, and the representations are optimized w.r.t. the objective function (Hamilton et al., 2017b; Perozzi et al., 2014; Grover and Leskovec, 2016). In general, shallow embedding methods can be divided into two types, i.e., those based on matrix factorization (Ahmed et al., 2013; Belkin and Niyogi, 2001; Cao et al., 2015; Ou et al., 2016) and those based on random walks. Among them, the approaches based on random walks, e.g., DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016), have shown superior performance in a plethora of settings.

Graph Neural Networks. Graph Neural Networks (GNNs) have emerged to be powerful frameworks to tackle learning problems on graphs (Wu et al., 2020; Zhou et al., 2020; Hamilton et al., 2017a; Kipf and Welling, 2017; Dong et al., 2021a). GNNs learn low-dimensional representations by extracting information from both attributes and graph topology (Velickovic et al., 2018; Dong et al., 2021b, 2022). Such success can be attributed to its carefully designed neighborhood aggregation Kipf and Welling (2017); Dong et al. (2023b), through which a node iteratively extracts information from its direct neighbors (Velickovic et al., 2018; Wu et al., 2020; Zhou et al., 2020; Wang et al., 2022b). Correspondingly, GNNs have been widely used in many real-world applications (Ying et al., 2018b; Fan et al., 2019; Pal et al., 2020; Zhao et al., 2021; You et al., 2018; Li et al., 2018; Dong et al., 2024).

Comparison of Shallow Embedding Methods vs. GNNs. Several existing works have compared traditional shallow embedding methods and GNNs from different perspectives, including optimization objectives (Cai et al., 2018; Zhang et al., 2018; Cui et al., 2018; Amara et al., 2021; Hamilton et al., 2017b), properties learned (Cai et al., 2018; Goyal and Ferrara, 2018; Zhang et al., 2018), computational complexity (Goyal and Ferrara, 2018; Cui et al., 2018), and applications (Cai et al., 2018; Amara et al., 2021). Nevertheless, the performance comparison over real-world datasets and practical settings of data availability and heterogeneity are ignored. A few recent studies have performed performance comparison between the two branches (Makarov et al., 2021), with the conclusion that GNNs consistently achieve superior performance. However, we argue they do not sufficiently explore the drawbacks of GNNs. Different from the works above, we present a systematic study to compare the two branches based on both theoretical and experimental discussion. This allows us to *(i)* elaborate on the drawbacks of GNNs in a finer granularity; and *(ii)* propose simple yet effective strategies to tackle the drawbacks.

Appendix B. Experimental Settings

Open-source code can be accessed at: https://github.com/snap-research/SEESAW.

B.1 Datasets

We perform the empirical evaluations on 10 real-world network datasets, which span different fields such as scientific publications (citation networks and co-authorship networks) and e-commerce (merchandise networks), including `Cora` (Yang et al., 2016), `CiteSeer` (Yang et al., 2016), `PubMed` (Yang et al., 2016), `CoraFull` (Bojchevski and Günnemann, 2018), `DBLPFull` (Bojchevski and Günnemann, 2018), `Amazon-Computers` (Shchur et al., 2018),

Amazon-Photo (Shchur et al., 2018), Coauthor-CS (Shchur et al., 2018), Coauthor-Physics (Shchur et al., 2018), and Flickr (Zeng et al., 2020). We present their statistics in Table 2. We directly utilize the APIs provided by PyTorch Geometric (Fey and Lenssen, 2019) to access and load all datasets.

Among these datasets, Cora, CiteSeer, PubMed, CoraFull, and DBLPFull are citation networks, where nodes represent documents and edges are citation links. Amazon-Computers and Amazon-Photo are networks of merchandise, where nodes denote goods and edges represent that two goods are frequently bought together. Coauthor-CS and Coauthor-Physics are co-author networks. Here nodes are authors, and two authors are connected if they have co-authored a paper. Flickr is a social network of images (as nodes), and a pair of images are connected if they share similar properties such as geographic locations.

B.2 Implementation Details

Dataset Split. We first introduce the dataset split for node classification tasks. For all datasets with available public splits (i.e., Cora, CiteSeer, and PubMed), we utilize the given public splits to train an MLP model based on the learned node representations and measure the utility such as node classification accuracy. For those datasets without available public splits (i.e., CoraFull, DBLPFull, Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics, and Flickr), we utilize a commonly used with 60%/20%/20% split for the train/validation/test split, following the same settings explored by other literature. (Maekawa et al., 2022; Chien et al., 2021) We first introduce the dataset split for link prediction tasks. For all datasets, we explore the random split of 85%/5%/10%, following the same standard as existing works (Shiao et al., 2023; Zhang and Chen, 2018; Cai et al., 2021). Only training edges are visible during the training phase.

Evaluation Protocol. We consider the representation learning models as the encoder, and we follow a standard evaluation protocol to train a decoder to perform downstream tasks on graphs based on the learned representations. Specifically, we train an MLP model as the decoder in both node classification and link prediction tasks, which is a commonly adopted approaches in a series of related works (Kipf and Welling, 2016; Tan et al., 2023). In node classification, we input each node embedding into an MLP model and predict the associated label. In link prediction, we take the Hadamard product for representations of each node pair. An MLP takes the resulted vector as input, and output the associated predicted probability of being connected. All results are presented as an average value across three different runs together with the associated standard deviation.

Details of GNNs for Comparison. In this paper, we adopt vanilla GCN and vanilla GraphSAGE as the most representative GNNs for comparison. In addition, we also adopted state-of-the-art contrastive and non-contrastive self-supervised learning GNNs. We present a more detailed discussion below. For contrastive self-supervised GNNs, we propose to adopt GRACE (Zhu et al., 2020b) and a GCN trained with max-margin loss, i.e., ML-GCN (Ying et al., 2018a). Specifically, GRACE first generates two correlated graph views by randomly performing corruption. Then, the embedding model is trained with a contrastive loss to maximize the agreement between node representations in these two views. On the other hand, ML-GCN is trained with a walk-based max-margin loss, which forces the agreement between nodes appear in same walks (measured with inner product) to exceed a

certain positive margin. For non-contrastive self-supervised GNNs, we adopt Graph Barlow Twins (GBT) (Bielak et al., 2022) and Bootstrapped Graph Latents (BGRL) (Thakoor et al., 2022). Specifically, GBT computes the representations cross-correlation matrix of two distorted views of a single graph. The objective function is formulated to force the cross-correlation matrix to be as close as possible to the identity matrix. In this way, no negative sample is needed for optimization, which improves the practical efficiency. On the other hand, BGRL maintains two distinct graph encoders, and learns the node representations by training an online encoder to predict the embedding of a target node. This also enables BGRL to avoid using negative samples during learning.

Downstream Tasks and Metrics. In this work, we use the two most commonly studied tasks for graph data: node classification and link prediction. Following the literature, we use node classification accuracy and F1 score for node classification (Dwivedi et al., 2023), and Hits@50 is adopted for link prediction (Shiao et al., 2023).

Machine Details. We ran our experiments on Google Cloud Platform. For all experimental results reported in this paper, we run the corresponding experiments on either NVIDIA P100 or V100 GPUs. Specifically, the machine is configured with 12 virtual CPU cores and 64 GB of RAM for most experiments.

Open-Sourced Code. Open-source code can be accessed at: https://github.com/snap-research/SEESAW.

Appendix C. Additional Experimental Results and Analysis

In this section, we present additional experimental results and corresponding analysis. Specifically, we first discuss the overall performance across different GNNs. Then, we present an additional analysis on the GNN performance on nodes with different levels of degrees between attribute-rich and attribute-poor scenarios. After that, we present comprehensive results from both shallow embedding methods and GNNs to demonstrate the relationship between dimensional collapse, embedding dimensionality, and performance. In the last two sections, we first present the relationship between the bound of the rank and the actual rank of the learned node embedding matrix between shallow embedding methods and GNNs. We then compare the performance of (1) shallow embedding method and that with the enforced neighborhood aggregation; (2) GNNs and that without the neighborhood aggregation.

C.1 Analysis: Performance of Different GNNs

We first present the performance of different state-of-the-art GNNs from a different perspective to reveal their superiority over shallow embedding methods in terms of overall performance. Here we take the widely studied node classification as an example, and we measure the performance with F1 score (macro). We present the corresponding performance in Table 3. We observe that GNNs exhibit significant performance superiority over shallow embedding method on almost all datasets, which remains consistent with the discussion in subsection 4.2. It is worth note that the only dataset where shallow embedding method exhibits superiority over all other GNNs is `Flickr`, where we have the smallest available node attribute dimensionality (see Table 2). This implies that such superiority could be undermined when only limited node attribute dimensionality is available, which is in align with the discussion in 1.

Table 2: Statistics of the 10 real-world datasets we adopted in this paper. We utilize `Amz-C.`, `Amz-P.`, `Co-CS`, and `Co-Phy.` to represent `Amazon-Computers`, `Amazon-Photo`, `Coauthor-CS`, and `Coauthor-Phy`, respectively.

	Cora	CiteSeer	PubMed	CoraFull	DBLPFull	Amz-C.	Amz-P.	Co-CS	Co-Phy.	Flickr
#nodes	2,708	3,327	19,717	19,793	17,716	13,752	7,650	18,333	34,493	89,250
#edges	10,556	9,104	88,648	126,842	105,734	491,722	238,162	163,788	495,924	899,756
#features	1,433	3,703	500	8,710	1,639	767	745	6,805	8,415	500
#classes	7	6	3	70	4	10	8	15	5	7
Density	0.29%	0.16%	0.05%	0.06%	0.07%	0.52%	0.81%	0.10%	0.08%	0.02%
Homophily	81.0%	74.0%	80.0%	56.7%	82.8%	77.7%	82.7%	80.8%	93.1%	31.9%
Attr. Sparsity	98.73%	99.15%	89.98%	99.35%	99.68%	65.16%	65.26%	99.12%	99.61%	53.61%
Avg. Degree	3.90	2.74	4.50	6.41	5.97	35.76	31.13	8.93	14.38	10.08
Std. Degree	5.23	3.38	7.43	8.79	9.35	70.31	47.28	9.11	15.57	31.75

C.2 Analysis: Performance on Nodes with Different Levels of Degrees

To gain a deeper understanding of the performance on a finer granularity, we propose to also explore the influence of limited attributes on nodes with different levels of degrees, which allows us to gain an understanding of performance w.r.t. available attribute dimensionality at a fine-grained level.

Specifically, we propose to first compute the ranking of all nodes based on their degree. Then we divide the nodes in the test set into high- and low-degree nodes with a percentile threshold. Without loss of generality, here we take the percentile threshold as 50%. We present the experimental results in Table 4. We observe that low-degree nodes bear more significant accuracy reduction in seven out of the 10 adopted datasets. A potential reason is that low-degree nodes rely more on the information contained in the attribute-based prior (than high-degree nodes). Specifically, compared with high-degree nodes, low-degree nodes tend to receive relatively less information from its neighbors through the neighborhood aggregation mechanism. Therefore, the information encoded in their attributes could dominate the performance. We have consistent observations in link prediction.

C.3 Analysis: Dimensional Collapse vs. Performance

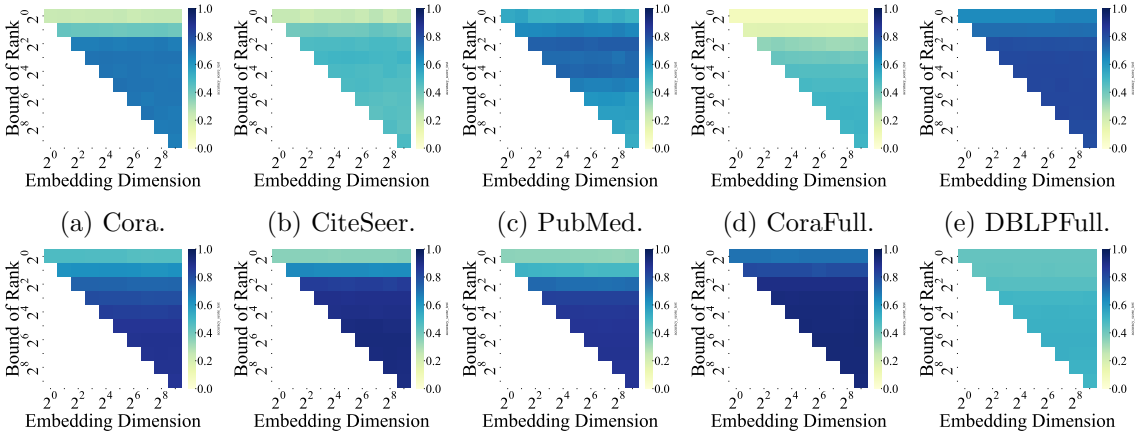
We then present the experimental results of performance w.r.t. the level of dimensional collapse. We utilize the same strategy introduced in *Observation 3* in subsection 4.3 to enforce different levels of dimensional collapse. Specifically, instead of directly obtaining the learned node embedding matrix from the GNN model, we consider the learned embedding matrix as $\boldsymbol{Z} = \boldsymbol{C}\boldsymbol{F}$, where $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$, $\boldsymbol{C} \in \mathbb{R}^{n\times r}$, and $\boldsymbol{F} \in \mathbb{R}^{r\times d}$ $(1 \leq r \leq d)$. In practice, $\boldsymbol{C}$ is the direct output of the GNN model, while $\boldsymbol{F}$ is a matrix with learnable parameters. We optimize both the learnable parameters in the GNN model and all elements in $\boldsymbol{F}$ during the end-to-end learning process. Without loss of generality, we present two sets of results as representative performances of shallow embedding methods and GNNs: (1) for shallow embedding methods, we present the results from DeepWalk in node classification task as an

Table 3: F1 score (macro) comparison between shallow embedding method, vanilla GCN, and other state-of-the-art self-supervised learning GNNs on 10 real-world graph datasets. We highlight the best performances in **Bold**. All numerical numbers are in percentages.

	Shallow	**Walk-GCN**	**GBT**	**BGRL**	**ML-GCN**	**GRACE**	**E2E-GCN**
Cora	69.45 ± 0.9	69.80 ± 0.7	74.85 ± 1.2	74.31 ± 1.5	67.93 ± 1.5	79.91 ± 0.2	**80.34 ± 0.7**
CiteSeer	44.82 ± 1.0	50.21 ± 1.2	54.65 ± 0.4	54.38 ± 2.7	48.39 ± 1.3	65.51 ± 1.2	**67.30 ± 0.8**
PubMed	58.92 ± 0.3	70.19 ± 3.0	77.79 ± 0.4	76.81 ± 3.2	71.98 ± 3.3	**81.45 ± 0.5**	78.49 ± 0.1
CoraFull	29.38 ± 0.5	32.37 ± 0.4	39.54 ± 1.0	37.37 ± 1.0	33.35 ± 1.3	32.15 ± 0.8	**40.55 ± 7.5**
DBLPFull	76.26 ± 0.9	80.32 ± 0.1	80.44 ± 0.1	**81.22 ± 0.3**	76.70 ± 0.6	80.89 ± 0.1	81.19 ± 0.3
Amz-C.	86.78 ± 0.9	86.54 ± 1.4	87.21 ± 0.7	85.73 ± 1.3	85.98 ± 1.2	79.92 ± 0.3	**89.45 ± 0.4**
Amz-P.	91.77 ± 1.0	**92.49 ± 0.5**	91.30 ± 0.7	92.28 ± 0.9	90.45 ± 0.6	90.31 ± 0.1	88.60 ± 0.8
Co-CS	84.25 ± 0.7	86.94 ± 0.7	91.68 ± 0.4	90.97 ± 0.4	89.13 ± 0.5	**91.94 ± 0.2**	91.40 ± 0.4
Co-Phy.	91.04 ± 0.5	94.42 ± 0.3	94.41 ± 0.3	94.25 ± 0.0	93.57 ± 0.3	OOM	**94.48 ± 0.3**
Flickr	**23.29 ± 0.0**	16.58 ± 0.0	21.00 ± 0.2	21.13 ± 0.4	20.23 ± 0.3	OOM	16.41 ± 0.1

Table 4: Node classification accuracy of GNNs under different numbers of attribute dimensions. We highlight the performances with the most significant reduction when transitioning from using 100% to 1% attributes in **Bold**. All numerical numbers are in percentages.

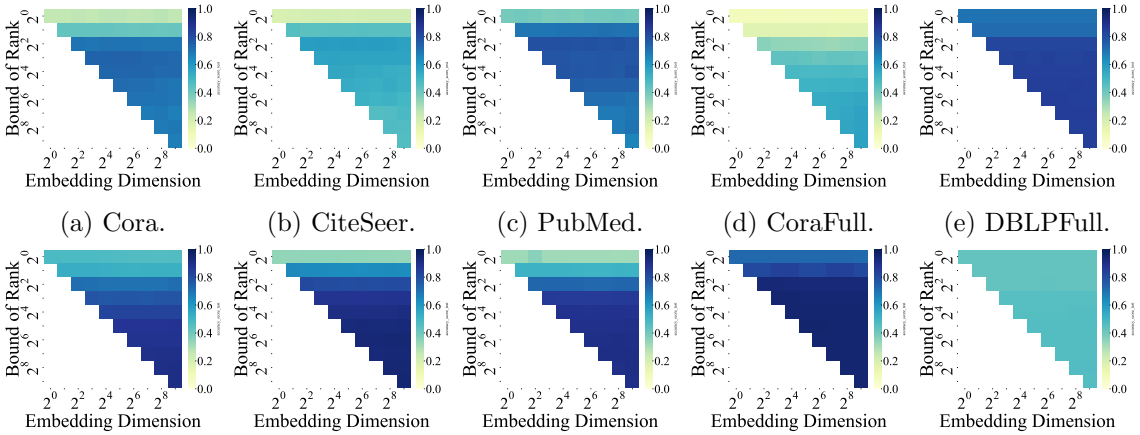
	GNN + 100% Attributes		**GNN + 1% Attributes**	
	High-Degree	**Low-Degree**	**High-Degree**	**Low-Degree**
Cora	74.7 ± 1.3	61.0 ± 1.7	**35.6 ± 0.8 (− 52.3 %)**	36.9 ± 0.7 (− 39.6 %)
CiteSeer	53.4 ± 0.9	39.3 ± 1.6	32.1 ± 0.4 (− 39.9 %)	**18.1 ± 0.1 (− 54.1 %)**
PubMed	73.1 ± 0.9	63.5 ± 2.1	**51.9 ± 1.3 (− 29.0 %)**	56.3 ± 0.6 (− 11.3 %)
CoraFull	59.3 ± 0.5	49.1 ± 1.1	11.8 ± 0.3 (− 80.1 %)	**7.09 ± 0.2 (− 85.6 %)**
DBLPFull	87.1 ± 0.3	81.8 ± 0.2	61.7 ± 0.8 (− 29.2 %)	**32.5 ± 1.4 (− 60.2 %)**
Amz-C.	92.4 ± 0.5	88.0 ± 0.4	63.9 ± 1.3 (− 30.9 %)	**42.2 ± 2.3 (− 52.0 %)**
Amz-P.	95.6 ± 0.2	91.0 ± 0.9	67.1 ± 1.7 (− 29.9 %)	**45.4 ± 0.9 (− 50.1 %)**
Co-CS	93.0 ± 0.4	87.9 ± 0.4	72.7 ± 1.5 (− 21.8 %)	**57.3 ± 0.5 (− 34.8 %)**
Co-Phy.	97.5 ± 0.3	93.8 ± 0.2	96.9 ± 0.1 (− 0.69 %)	**92.6 ± 0.3 (− 1.35 %)**
Flickr	43.0 ± 0.2	49.6 ± 0.1	**37.8 ± 0.1 (− 12.1 %)**	48.2 ± 0.0 (− 2.82 %)



(a) Cora. (b) CiteSeer. (c) PubMed. (d) CoraFull. (e) DBLPFull.

(f) Amz-Comp. (g) Amz-Photo. (h) Coauthor-CS. (i) Coauthor-Phy. (j) Flickr.

Figure 9: The node classification performance (measured with classification accuracy) of shallow embedding method on 10 different real-world graph datasets. A lower value of *Bound of Rank* implies a heavier level of dimensional collapse. We utilize Amz-Computers to refer to the dataset `Amazon-Computers`.



(a) Cora. (b) CiteSeer. (c) PubMed. (d) CoraFull. (e) DBLPFull.

(f) Amz-Comp. (g) Amz-Photo. (h) Coauthor-CS. (i) Coauthor-Phy. (j) Flickr.

Figure 10: The node classification performance (measured with classification accuracy) of GNN on 10 different real-world graph datasets. Here the available ratio of the node attribute dimensionality is 100%. A lower value of *Bound of Rank* implies a heavier level of dimensional collapse. We utilize Amz-Computers to refer to the dataset `Amazon-Computers`.

example in Figure 9.; (2) for GNNs, we present the results from GCN in node classification

task as an example in Figure 10. We also have similar observations in link prediction task, other shallow embedding methods, and GNNs. We have the following observations.

First, we observe that in all cases, improving the value of embedding dimension does not significantly change the performance, which implies that the value of embedding dimension does not play a key role in learning high-quality node representations.

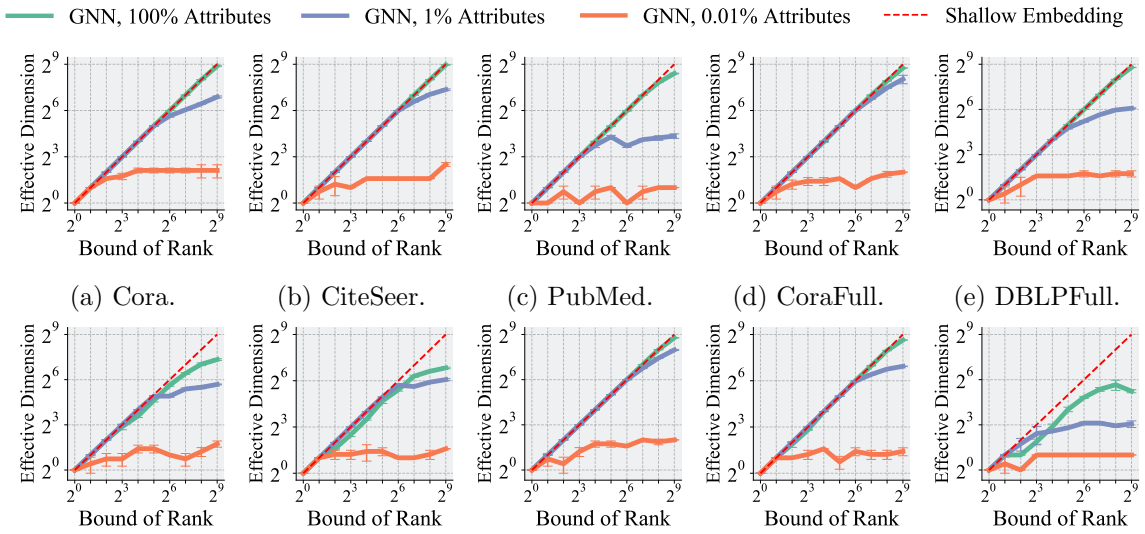
Second, we found that in almost all cases, improving the bound of the rank is able to significantly improve the performance (i.e., lead to deeper colors in the heatmap). Here, a smaller value of the bound of the rank generally implies a heavier level of dimensional collapse. This because no matter what value the embedding dimension takes, the learned representations will be guaranteed to collapse into a lower dimensional subspace as long as the the bound of the rank is small. The dimensionality of the lower dimensional subspace is upper-bounded by the bound of the rank. Such an observation demonstrates that as long as dimensional collapse happens, the performance then significantly drops no matter how large the embedding dimensionality is. On the contrary, if dimensional collapse is relieved, the performance is then also improved under a given embedding dimension. We note that improving the bound of the rank could also bring performance reduction in very few cases, e.g., on the PubMed dataset for both shallow embedding methods and GNNs. Such a phenomenon could be caused by overfitting, which goes beyond the scope of this paper.

To summarize, we conclude that no matter what value embedding dimension takes, dimensional collapse always leads to significantly performance drop, which implies a worse quality of the node representations. At the same time, mitigating the dimensional collapse will be beneficial to the performance in most cases. This remains consistent with our conclusion discussed in subsection 4.3.

C.4 Analysis: Ranks of Learned Representations

We now present an analysis on the rank of the learned node representations between shallow embedding methods and GNNs with different levels of attribute availability. Specifically, we utilize the strategy introduced in subsection 4.3 and subsection C.3 to control the level of dimensional collapse. At each value of the bound of the rank, we calculate the actual rank of the learned node embedding matrix. Note that the actual rank of the learned node embedding matrix cannot exceed the bound of the rank. Correspondingly, in an ideal case, an representation learning model should yield an embedding matrix with a rank equivalent to the bound of the rank, such that the node representations will span a hidden subspace as large as possible. In Figure 11, we present the curves of effective dimension vs. bound of rank for each of the 10 real-world graph datasets. In each subfigure, we present curves from four different scenarios, namely using shallow embedding method, using GNN with 100% attributes, using GNN with 1% attributes, and using GNN with 0.01% attributes. Here we adopt DeepWalk and GCN as the representative model for shallow embedding methods and GNNs, respectively. We have the following observations.

First, the curve of effective dimension of shallow embedding method is a straight line of $y = x$ in all cases. This reveals the clear advantage of shallow embedding method in defending against dimensional collapse, since it can always learn node embedding matrices spanning the whole available hidden subspace. As discussed in subsection 4.3, such an advantage can be attributed to avoiding using a prior based on the node attributes.



(a) Cora. (b) CiteSeer. (c) PubMed. (d) CoraFull. (e) DBLPFull.

(f) Amz-Comp. (g) Amz-Photo. (h) Coauthor-CS. (i) Coauthor-Phy. (j) Flickr.

Figure 11: A comparison between shallow embedding method and GNNs under different ratios of available attributes on 10 real-world graph datasets. Here, *Effective Dimension* represents the rank of the learned node embedding matrix, which is upper bounded by the *Bound of Rank*. The dimensionality of hidden space is 2^9. We utilize Amz-Computers to refer to the dataset `Amazon-Computers`.

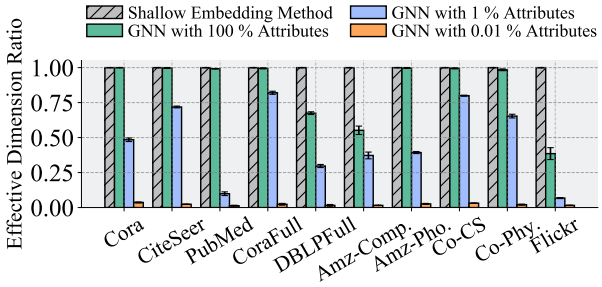


Figure 12: Effective dimension ratio of representations learned for link prediction task.

Second, we found that it is always difficult for GNNs to avoid dimensional collapse when the dimensionality of the input node attributes is limited. We take Cora dataset (given in Figure 11(a)) as an example. GNNs with 100% available node attributes can always achieve a value of effective dimension (i.e., the rank of the learned node embedding matrix) equal to the bound of the rank. Nevertheless, when only 1% node attributes are available, dimensional collapse begins to happen when the bound of rank is larger than 2^6. When only 0.01% node attributes are available, dimensional collapse even begins to happen when the bound of rank is larger than 2^2. As a comparison, shallow embedding method can always achieve an effective dimension that is equals to the bound of the rank. This demonstrates that available node attribute dimensionality directly influences the level of dimensional collapse. We also present the effective dimension ratio of the learned node representations in Figure 12, revealing that dimensional collapse also happens under representations learned for link prediction in GNNs, while shallow embedding methods do not encounter such a problem. As discussed in subsection 4.3, such a drawback should

be attributed to using a prior based on the node attributes. Such a conclusion remains consistent with the discussion in subsection 4.3.

C.5 Analysis: Performance w/ Aggregation vs. Performance w/o Aggregation

We finally present a comparison of the performance between representation learning models with and without neighborhood aggregation. Specifically, we adopt the same strategy as discussed in subsection 4.4, where shallow embedding method with neighborhood aggregation (with a mean aggregator) and GNNs without neighborhood aggregation are implemented for comparison. In this way, we are able to rigorously compare the effect of performing neighborhood aggregation within each branch of models, and whether using a prior based on node attributes or not will not influence the conclusion. Without loss of generality, we take DeepWalk and GraphSAGE as the representative example of shallow embedding methods and GNNs, respectively. For any node, we measure the homophily score with the ratio of the total number of its direct neighbors with the same label as itself to the total number of its direct neighbors. We have the following observations.

First, the shallow embedding methods without neighborhood aggregation can achieve better performance on those nodes with small homophily scores in almost all cases. As a comparison, the cumulative performance of shallow embedding methods with neighborhood aggregation improves faster than that of shallow embedding methods without neighborhood aggregation. This indicates a better performance of shallow embedding method with neighborhood aggregation on those nodes with large homophily scores.

Second, we found that a similar phenomenon also exists in GNNs. Specifically, the GNNs without neighborhood aggregation can achieve better performance on those nodes with small homophily scores in almost all cases. As a comparison, GNNs with neighborhood aggregation can always achieve a similar or even better performance compared with that without neighborhood aggregation. This indicates a better performance of GNNs with neighborhood aggregation on those nodes with large homophily scores.

To summarize, we conclude that for both types of models, performing neighborhood aggregation typically helps the performance on those nodes with larger homophily scores, while this could also do harm to the performance on those nodes with smaller homophily scores. This remains consistent with our conclusion discussed in subsection 4.4.

C.6 Analysis: Combining Shallow Embedding Methods and GNNs

We now present the analysis on the performance of combining shallow embedding methods and GNNs by directly concatenating their node representations. The rationale here is twofold: *(i)* The number of effective dimensions of the learned representations (i.e., the rank of the node embedding matrix) from GNNs could be promoted by representations from shallow methods, which helps to tackle the problem of dimensional collapse in attribute-poor scenarios (mentioned in subsection 4.3). *(ii)* Information from both aggregated and unaggregated node attributes could be preserved at the same time, which helps to alleviate the performance reduction on heterophilic nodes (mentioned in subsection 4.4). This remains consistent with the disucssion in section 5. Without loss of generality, we take DeepWalk and GCN as the shallow embedding model and the GNN model, respectively. We take the performance of node classification accuracy on the `DBLPFull` dataset as an example, and

present the results across different levels of available attribute dimensionalities in Figure 8. We present the observations below.

First, the performance of GNNs with 100% available node attributes is superior to that of shallow embedding method, which reveals that useful information could be encoded in the node attributes, which contribute to the performance of GNNs.

Second, when the available attribute dimensionality is decreased, the performance of GNNs reduces significantly together with the performance (measured with node classification accuracy). This generally reflects that less available node attributes will typically lead to dimensional collapse, which remains consistent with the discussion in subsection 4.3.

Third, when we use the concatenation of the node representations from the two methods, we observe that the performance does not significantly reduce when the available node attributes become limited. This validate the superiority of combining the representations from the two methods across scenarios with different available node attribute dimensionalities.

To summarize, we observe that although GNNs with 100% node attributes can achieve the best performance, its performance reduces significantly once the available node attributes are limited. However, by simply concatenating their representations, we can obtain much more stable performance across scenarios with different available node attribute dimensionalities. However, this approach will lead to a higher computational complexity, since both models need to be optimized. Hence such method can hardly be recommended in industrial settings due to the high computational cost.

C.7 Selection of Shallow Method

We select DeepWalk as a representative shallow graph embedding method to compare with in this paper. The reason why DeepWalk is adopted is that DeepWalk is a representative example of walk-based shallow methods in its design. Specifically, DeepWalk is among the most commonly used shallow graph embedding methods, and a large amount of following works under the umbrella of shallow methods are developed based on DeepWalk. Therefore, DeepWalk is among the best options we can choose to obtain generalizable analysis, and adopting more follow-up methods that share similar design with DeepWalk does not change the observation and conclusion.

C.8 Experimental Results with Heterophilic Graphs

We would like to note that our analysis does not depend on whether the adopted datasets are homophilic or not. Here we present the comparison between GNNs and shallow methods on the heterophilic dataset. Specifically, we select the Squirrel dataset and present the corresponding performances below as a representative example, since the Squirrel dataset has a comparable scale (5,201 nodes) with the datasets adopted in our paper and is also highly heterophilic (homophilic ratio 0.22). First, we perform experiments to evaluate dimensional collapse. Here utility is measured by node classification accuracy, while the effective dimension ratio (EDR) is measured by the ratio of the value of rank (of representation matrix) to the representation dimensionality. We present the experimental results in Table 5. We observe a similar tendency as presented in our paper: the GNN model bears severe dimensional collapse when available attributes become limited, while the shallow method is not influenced since it does not take any node attribute as its input.

Table 5: Performance comparison between shallow embedding method and GNNs on the Squirrel dataset under different levels of attribute availibility ratio.

	100%Att, Acc	**100%Att, EDR**	**1%Att, Acc**	**1%Att, EDR**	**0.01%Att, Acc**	**0.01%Att, EDR**
GNN	**38.8%**	74.2%	26.3%	21.9%	19.0%	1.56%
Shallow	31.5%	**99.6%**	**31.5%**	**99.6%**	**31.5%**	**99.6%**

Table 6: Node classification accuracy comparison between GNN (w/ Aggregation) and GNN w/o Aggregation on the Squirrel dataset under different of node heterophily scores.

	1e-3	**5e-3**	**1e-2**	**5e-2**	**1e-1**	**5e-1**	**1e0**
GNN (w/ Aggregation)	**26.88%**	26.88%	26.88%	26.73%	25.44%	**36.64%**	**38.75%**
GNN w/o Aggregation	**30.10%**	**30.10%**	**30.10%**	**30.69%**	**26.48%**	**32.20%**	33.17%

Table 7: Node classification accuracy comparison between shallow method (w/o Aggregation) and shallow method w/ Aggregation on Squirrel dataset under different levels of node heterophily score.

	1e-3	**5e-3**	**1e-2**	**5e-2**	**1e-1**	**5e-1**	**1e0**
Shallow w/ Aggregation	**21.86%**	21.86%	21.86%	23.23%	26.35%	**30.36%**	**31.54%**
Shallow (w/o Aggregation)	**25.14%**	**25.14%**	**25.14%**	**26.26%**	**26.71%**	**29.95%**	31.44%

Table 8: Node classification accuracy comparison between GNN and GNN under two types of attribute augmentation approaches (including concatenating a random matrix denoted as R and concatenating a matrix of structural features denoted as S onto the original node attribute matrix) on Cora dataset. Here we adopt the walk-based GCN (denoted as W-GCN) as the backbone GNN.

	100%Att-Acc	**100%Att-EDR**	**1%Att-Acc**	**1%Att-EDR**	**0.01%Att-Acc**	**0.01%Att-EDR**
W-GCN	67.8%	96.9%	36.9%	35.5%	32.3%	1.56%
W-GCN (R)	68.0%	97.3%	29.9%	67.2%	10.6%	5.08%
W-GCN (S)	69.0%	97.3%	37.3%	35.2%	32.3%	2.34%

Second, we also perform experiments to study how the performance changes from highly heterophilic nodes to highly homophilic nodes. We first present the study between GNN (w/ neighborhood aggregation) and GNN w/o neighborhood aggregation. We present their cumulative performances in node classification accuracy under different values of the homophilic score below (the same setting as in Figure 6 in our paper). We present the experimental results in Table 6. We observe a similar tendency as presented in our paper: neighborhood aggregation will jeopardize the performances over those highly heterophilic nodes while benefiting highly homophilic nodes. In addition, we also perform experiments to compare shallow method w/ neighborhood aggregation vs. shallow method w/o neighborhood aggregation. We present the experimental results in Table 7, and the observations remain consistent.

In conclusion, we also have similar observations on heterophilic datasets from both studied perspectives, and our analysis does not depend on whether the adopted datasets are homophilic or not.

C.9 Experimental Results with Attribute Augmentation

We performed experiments by (1) concatenating a random matrix with the same dimensionality as the original node attributes onto the node attribute matrix and (2) concatenating a matrix encoded with structural information following the state-of-the-art *degree* strategy (Cui et al., 2022) onto the node attribute matrix. We present the unsupervised learning performances on the Cora dataset below as an example. Here utility is measured by node classification accuracy, while the effective dimension ratio (EDR) is measured by the ratio of the value of rank (of representation matrix) to the representation dimensionality. We present the experimental results in Table 8. We observe that: (1) concatenating a matrix with structural information slightly improves the node classification accuracy, while such a strategy does not stop the significant drop in the rank of node representations; (2) concatenating random node attributes successfully improves the rank of the node representations, however, the classification accuracy is reduced. Therefore, both strategy does not really solve the problem of dimensional collapse, and we believe handling such a problem is non-trivial. Correspondingly, this paper is particularly interesting to researchers working in this area and such a problem is also worth to be explored in future works.

Table 9: Node classification accuracy comparison between GNN and APPNP on Cora dataset. Here we adopt the walk-based GCN as a representative GNN for comparison, and both models are optimized with the walk-based loss.

	100%Att-Acc	**100%Att-EDR**	**1%Att-Acc**	**1%Att-EDR**	**0.01%Att-Acc**	**0.01%Att-EDR**
GCN	67.8%	96.9%	36.9%	35.5%	32.3%	1.56%
APPNP	75.5%	56.3%	42.4%	13.7%	10.6%	0.78%

C.10 Observations over Other Types of GNNs

Performance of APPNP. We perform empirical experiments based on APPNP. We present the unsupervised learning performances on the Cora dataset below as an example. Here utility is measured by node classification accuracy, while the effective dimension ratio (EDR) is measured by the ratio of the value of rank (of representation matrix) to the representation dimensionality. We present the experimental results in Table 9. We observe that similar to GCN, APPNP also bears severe dimensional collapse (exhibited by the significant reduction in the value of EDR).

Performance of LINKX. We perform empirical experiments based on LINKX. We present the unsupervised learning performances on the Cora dataset below as an example. Here utility is measured by node classification accuracy, while the effective dimension ratio (EDR) is measured by the ratio of the value of rank (of representation matrix) to the representation dimensionality. We observe that compared with GCN, LINKX exhibits smaller values of EDR in attribute-rich scenarios (e.g., 100% available node attributes), while it also mitigates dimensional collapse in attribute-poor scenarios (e.g., 0.01% available node attributes). This demonstrates that (1) such an approach may jeopardize the effective dimension ratio in attribute-rich scenarios and (2) such an approach effectively helps to mitigate dimensional collapse in attribute-poor scenarios. However, we would also like to point out that even if LINKX successfully mitigates dimensional collapse for GNNs, it is not ideal, since it (1) sacrifices the capability of GNNs in inductive learning and (2) improves the computational complexity from $\mathcal{O}(n * k)$ to $\mathcal{O}(n^2)$ to perform inference (k is the number of node attributes and n is the number of nodes). Therefore, the problem of dimensional collapse is non-trivial to handle, and more analysis can be a great follow-up study of our work.

Table 10: Node classification accuracy comparison between GNN and LINKX on Cora dataset. Here we adopt the walk-based GCN as a representative GNN for comparison, and both models are optimized with the walk-based loss.

	100%Att-Acc	**100%Att-EDR**	**1%Att-Acc**	**1%Att-EDR**	**0.01%Att-Acc**	**0.01%Att-EDR**
GCN	67.8%	96.9%	36.9%	35.5%	32.3%	1.56%
LINKX	65.9%	50.4%	64.6%	49.6%	68.8%	24.2%

C.11 Discussion: Works Combining Shallow Methods and GNNs

Here we present two representative works that aim to combine the advantage of shallow graph embedding methods and GNNs (Abu-El-Haija et al., 2018; Chien et al., 2020). Specifically, (Abu-El-Haija et al., 2018) successfully achieves optimization for the context hyperparameters of shallow graph embedding methods. However, the proposed approach cannot take node attributes as the input and thus fails to effectively utilize the information encoded in node attributes; (Chien et al., 2020) explored to generalize the GNNs to adaptively learn high-quality node representations under both homophilic and heterophilic node label patterns. Nevertheless, it fails to avoid using the node attributes as the learning prior, and thus still follows a design that has been proved to bear dimensional collapse.

C.12 Discussion: Dimensional Collapse vs. Learning Low-Rank Representations

In this subsection, we distinguish between dimensional collapse in graph machine learning and the common objective of learning low-rank node representations to help the audience understanding their differences. While both involve reduced-rank representation matrices, they differ fundamentally in their causes, controllability, and desirability. Dimensional collapse, as demonstrated in our experiments, is an unintended consequence that occurs when GNNs are trained with limited node attributes, causing learned representations to involuntarily collapse into a lower-dimensional subspace regardless of the available hidden space dimensionality. Such collapse is also seen in other domains such as Zhuo et al. (2023); Jing et al. (2022); He and Ozay (2022) and is detrimental to performance and occurs due to the attribute-based prior inherent in GNN architectures. In contrast, learning low-rank representations is a deliberate design choice or regularization technique where practitioners intentionally constrain the rank to achieve benefits such as computational efficiency, noise reduction, or improved generalization. The key distinction lies in control and intent: dimensional collapse is an unwanted side effect that degrades model performance in attribute-poor scenarios, while low-rank learning is a purposeful strategy that can enhance model performance when appropriately applied. Our findings show that shallow embedding methods avoid this uncontrolled collapse by not relying on attribute-based priors, consistently maintaining full-rank representations that span the entire available hidden space.

C.13 Discussion: Empirical Guidance for Determining Attribute-Poor Graph Learning Scenarios

Based on the experimental results presented in this paper, practitioners can identify attribute-poor graph learning scenarios through several empirical indicators. Primary indicators include: (1) Limited attribute dimensionality - when the number of available node features is significantly reduced compared to the original dataset (e.g., less than 10% of original features), as demonstrated in our experiments where GNN performance dropped substantially when attribute dimensionality ratios fell to 1% or 0.01%; (2) Low feature-to-node ratio - graphs where the number of node attributes is much smaller than the number of nodes, as evidenced by datasets like Flickr (500 features for 89,250 nodes) where shallow methods outperformed GNNs; and (3) High dimensional collapse tendency - measurable through

the effective dimension ratio (rank of embedding matrix divided by embedding dimensionality), where ratios below 0.5 indicate severe dimensional collapse. Secondary indicators include: (1) Sparse or noisy attributes - when available features contain significant missing values or noise that may not provide meaningful signal; (2) Domain-specific constraints - scenarios where collecting rich node attributes is inherently difficult or expensive (e.g., privacy-sensitive domains, large-scale networks); and (3) Performance degradation patterns - observing substantial accuracy drops (>20%) when transitioning from full to limited attributes, particularly affecting low-degree nodes more severely as shown in our analysis. When these indicators are present, practitioners should consider shallow embedding methods over GNNs to avoid the dimensional collapse problem and maintain robust performance across varying attribute availability scenarios.

C.14 Discussion: Comparison of Complexity and Scalability

To provide a comprehensive understanding of the computational requirements for different graph representation learning methods, we present a theoretical time complexity analysis of all approaches evaluated in our study. Our analysis considers multiple dimensions of computational cost, including training complexity, inference complexity, and space complexity.

Notation and Complexity Factors. In our complexity analysis, we use the following notation: $|V|$ represents the number of nodes in the graph, $|E|$ denotes the number of edges, L indicates the number of GNN layers used in deep methods, d refers to the hidden dimension size of learned representations, W represents the number of random walks generated per node (for walk-based methods), L_{walk} denotes the length of each random walk, K indicates the number of negative samples used in contrastive learning objectives, and C represents the number of classes in classification tasks.

Complexity Comparison. Table 11 presents a detailed theoretical time complexity analysis for all methods compared in our study. The analysis reveals fundamental differences between shallow embedding methods and graph neural networks in terms of computational requirements. Shallow embedding methods, such as DeepWalk, exhibit complexity that is independent of the number of edges in the graph, with training complexity of $O(|V| \times W \times L_{walk} \times K \times d)$ and highly efficient inference through simple lookup operations with $O(|V| \times d)$ complexity. In contrast, all GNN-based methods show complexity that scales linearly with the number of edges, with the basic GNN component requiring $O(L \times (|E| \times d + |V| \times d^2))$ operations for both training and inference. The analysis further reveals that contrastive learning methods like GRACE incur additional computational overhead due to dual-view processing and contrastive loss calculation, resulting in training complexity of $O(2 \times L \times (|E| \times d + |V| \times d^2) + |V|^2 \times d)$. Non-contrastive methods such as BGRL and GBT demonstrate more favorable complexity profiles by avoiding expensive negative sampling procedures. Methods that combine GNN architectures with walk-based objectives (Walk-GCN and ML-GCN) exhibit hybrid complexity that includes both GNN operations and random walk generation costs.

Scalability Comparison. Table 12 provides a practical scalability assessment of these methods, ranking them from best (1) to worst (7) based on computational efficiency. The analysis shows that shallow embedding methods offer the best scalability for certain scenarios, particularly when dealing with small graphs or sparse node features, while BGRL and

GBT represent the most scalable options among GNN-based approaches. The table also indicates each method's dependency on edge count, memory efficiency during training, and recommended use cases, providing practical guidance for method selection based on specific application requirements and computational constraints.

Table 11: Theoretical Time Complexity Analysis of Methods in Table 1.

Method	Training Complexity	Inference Complexity	Space Complexity	Key Characteristics
Shallow Embedding Methods				
Shallow	$O(\|V\| \times W \times L \times K \times d)$	$O(\|V\| \times d)$	$O(\|V\| \times d)$	Edge-independent complexity Lookup-based inference
Graph Neural Networks				
Walk-GCN	$O(L \times (\|E\| \times d + \|V\| \times d^2)$ $+\|V\| \times W \times L \times K \times d)$	$O(L \times (\|E\| \times d + \|V\| \times d^2))$	$O(L \times \|V\| \times d + L \times d^2)$	GCN + Random walk loss Combines structure & context
GBT	$O(L \times (\|E\| \times d + \|V\| \times d^2)$ $+\|V\| \times d^2)$	$O(L \times (\|E\| \times d + \|V\| \times d^2))$	$O(L \times \|V\| \times d + d^2)$	Non-contrastive learning Cross-correlation matrix
BGRL	$O(L \times (\|E\| \times d + \|V\| \times d^2))$	$O(L \times (\|E\| \times d + \|V\| \times d^2))$	$O(2 \times L \times \|V\| \times d + L \times d^2)$	Bootstrap learning No negative sampling
ML-GCN	$O(L \times (\|E\| \times d + \|V\| \times d^2)$ $+\|V\| \times W \times L \times d)$	$O(L \times (\|E\| \times d + \|V\| \times d^2))$	$O(L \times \|V\| \times d + L \times d^2)$	Max-margin loss Walk-based training
GRACE	$O(2 \times L \times (\|E\| \times d + \|V\| \times d^2)$ $+\|V\|^2 \times d)$	$O(L \times (\|E\| \times d + \|V\| \times d^2))$	$O(2 \times L \times \|V\| \times d + L \times d^2)$	Dual-view contrastive Graph augmentation
E2E-GCN	$O(L \times (\|E\| \times d + \|V\| \times d^2)$ $+C \times \|V\| \times d)$	$O(L \times (\|E\| \times d + \|V\| \times d^2)$ $+C \times \|V\| \times d)$	$O(L \times \|V\| \times d + L \times d^2)$	End-to-end supervised Direct classification

Table 12: Complexity Factors and Scalability Analysis: *Edge* indicates whether the method's complexity depends on the number of edges; *Scalability* indicates the rank from 1 (best) to 7 (worst) based on computational efficiency; *Memory* refers to memory usage during training; Use Case indicates optimal application scenarios for each method.

Method	Edge	Scalability	Memory	Use Case
Shallow	×	1 (Best)	High	Small graphs, sparse features
BGRL	✓	2	Medium	Large sparse graphs
GBT	✓	3	Medium	Medium-scale graphs
E2E-GCN	✓	4	Medium	Supervised tasks
ML-GCN	✓	5	Low	Walk-based applications
Walk-GCN	✓	6	Low	Combined structure-context
GRACE	✓	7 (Worst)	Very Low	High-performance requirements

C.15 Discussion: Dataset Statistics and Model Behavior

Based on careful examination of the dataset statistics in Table 2 and the experimental results throughout the paper, there is no statistically significant correlation between key dataset characteristics and the fundamental model behaviors observed in our study. For instance, despite substantial variations in feature sparsity across datasets (ranging from 53.61% in Flickr to 99.68% in DBLPFull), node counts (from 2,708 in Cora to 89,250 in Flickr), and degree distributions (average degrees from 2.74 in CiteSeer to 35.76 in Amazon-Computers), the core phenomena of dimensional collapse in attribute-poor scenarios and performance degradation from neighborhood aggregation in heterophilic settings remain consistent across all datasets. Notably, even datasets with vastly different structural properties—such as the

dense Amazon-Computers network (0.52% density, 767 features) versus the sparse Flickr network (0.02% density, 500 features)—exhibit similar patterns of GNN performance drops when attribute dimensionality is reduced from 100% to 1% or 0.01%, as demonstrated in Figures 3 and 5. The dimensional collapse phenomenon occurs regardless of whether datasets have high feature dimensionality (8,710 features in CoraFull) or low feature dimensionality (500 features in PubMed and Flickr), and similarly affects both small-scale networks like Cora and large-scale networks like Flickr. While we acknowledge that homophily levels do influence the effectiveness of neighborhood aggregation—as extensively demonstrated in Section 4.4 through our analysis of heterophilic versus homophilic nodes—the fundamental architectural differences between shallow embedding methods and GNNs (attribute-based priors and neighborhood aggregation mechanisms) consistently manifest their respective advantages and drawbacks across diverse graph structures, scales, and feature distributions, suggesting that these behaviors are intrinsic to the methodological approaches rather than artifacts of specific dataset characteristics.

C.16 Discussion: Computational Cost of the Combined Method

Table 13 reveals that the combined method, which concatenates representations from both shallow embedding methods and GNNs, incurs substantially higher computational costs compared to using either method independently. The total runtime for the combined approach consistently exceeds the sum of individual method runtimes across all datasets, with particularly pronounced overhead on larger graphs like Flickr (1549.8s total vs. 334.1s for shallow and 1210.7s for GNN). This burden extends beyond simple addition due to the need to maintain and optimize two separate models simultaneously, manage their interactions, and handle the increased dimensionality of concatenated representations. The memory requirements also represent an increase that could pose challenges in resource-constrained environments. For instance, on CoraFull, the combined method requires 683.4 MB compared to much lower requirements for individual methods. This analysis underscores a critical limitation: while concatenation can leverage strengths from both approaches to handle attribute-poor scenarios and heterophilic nodes, the computational overhead makes it impractical for many real-world applications, especially in industrial settings where efficiency is paramount. These findings highlight the urgent need for novel architectures that can inherently capture the advantages of both shallow embedding methods (dimensional collapse resistance) and GNNs (attribute utilization and structured learning) without requiring the maintenance of two separate models or suffering from high computational costs.

C.17 Discussion: Further Explorations of Other Graph Learning Methods

Table 14 demonstrates that even advanced graph transformer architectures like Graphormer fail to overcome the fundamental limitations identified in this study. When evaluated on the DBLPFull dataset, Graphormer shows a dramatic performance degradation as attribute availability decreases, dropping from 82.2% accuracy with full attributes to 45.9% with 1% attributes and 44.9% with 0.01% attributes. This pattern mirrors the behavior observed with traditional GNNs in Table 4, confirming that the dimensional collapse phenomenon is not merely an artifact of specific GNN architectures but rather a fundamental issue stemming from attribute-based priors. The rank ratio results are particularly telling, plummeting

Table 13: Computational cost of shallow method, GNN, and combined method across different dataset. All numerical values of running time are in seconds (s) and all numerical values of memory are in Megabyte (MB).

Dataset	Total Time	Shallow Time	GNN Time	Memory
Cora	84.9 ± 0.3	21.0 ± 0.2	63.4 ± 0.2	32.4 ± 0.0
CiteSeer	91.4 ± 0.4	23.4 ± 0.1	67.5 ± 0.3	65.9 ± 0.0
PubMed	289.2 ± 0.5	77.0 ± 1.1	211.4 ± 0.6	59.3 ± 0.0
CoraFull	321.4 ± 0.4	82.3 ± 1.2	237.7 ± 1.6	683.4 ± 0.0
DBLPFull	271.5 ± 4.0	71.9 ± 1.5	198.5 ± 2.5	132.2 ± 0.0
Amz-C.	249.4 ± 0.8	60.4 ± 0.7	187.3 ± 0.1	60.7 ± 0.0
Amz-P.	150.9 ± 2.0	38.3 ± 0.1	111.6 ± 2.1	40.5 ± 0.0
Co-CS	296.2 ± 0.5	77.0 ± 0.8	216.4 ± 1.6	500.1 ± 0.0
Co-Phy.	710.9 ± 12.5	165.7 ± 5.7	477.8 ± 4.1	188.9 ± 0.0
Flickr	1549.8 ± 5.1	334.1 ± 2.2	1210.7 ± 2.9	208.5 ± 0.0

from 31.5% to just 2.0% in extreme attribute-poor scenarios, indicating severe dimensional collapse. Moreover, the performance disparity between low-degree and high-degree nodes persists, with low-degree nodes experiencing more severe accuracy reductions (79.1% to 30.4%) compared to high-degree nodes (85.3% to 59.3%). These findings suggest that simply adopting more sophisticated architectures or attention mechanisms does not address the core issues identified in this work. The consistency of these results across different GNN variants, including now graph transformers, reinforces the paper's central argument about the inherent limitations of attribute-dependent graph learning methods and further emphasizes the critical need for fundamentally new approaches that can combine the robustness of shallow methods with the expressiveness of deep architectures.

Table 14: Experimental Results using Graphormer as the GNN backbone with different feature dimensions on DBLPFull dataset.

Rank	Ratio	Accuracy	Accuracy Low Deg.	Accuracy High Deg.
31.5 ± 0.7	100%	82.2 ± 0.3	79.1 ± 0.8	85.3 ± 0.2
18.0 ± 0.0	1%	45.9 ± 0.3	32.1 ± 1.2	59.7 ± 0.5
2.0 ± 0.0	0.01%	44.9 ± 0.5	30.4 ± 1.1	59.3 ± 0.2

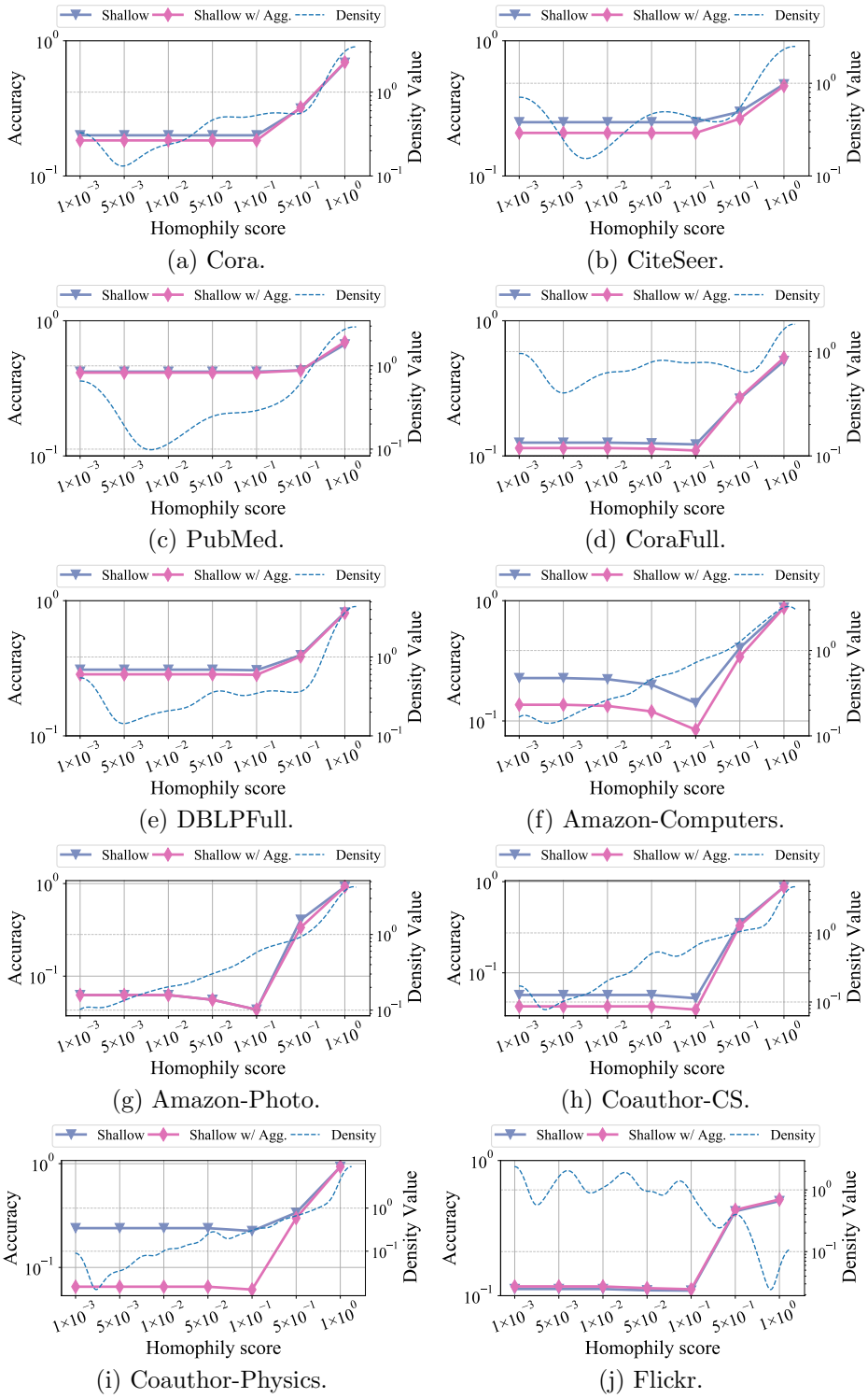
C.18 Discussion: Further Explorations of Other Combination Method

Table 15 presents results from a more sophisticated combination approach that goes beyond simple concatenation by incorporating a contrastive learning objective to reduce mutual information between shallow and GNN representations. Such an enhanced strategy helps to enforce the two methods to extract different (likely complementary) information during the training process, which demonstrates notable improvements over the naive concatenation strategy across multiple datasets. For instance, on DBLPFull, the method maintains strong

performance even in extreme attribute-poor scenarios (81.4% at 0.01% attributes vs. 84.3% at 100% attributes), showing remarkable robustness compared to standard GNNs that typically experience severe dimensional collapse. Similarly impressive results are observed on datasets like Amazon-Computers and Coauthor-CS, where the accuracy degradation is minimal despite drastic reductions in available attributes. The method successfully prevents dimensional collapse across most datasets, with many maintaining high rank ratios even under extreme attribute scarcity. However, these performance gains come at the cost of increased computational complexity from the joint training scheme and additional contrastive objectives, creating a challenging trade-off between utility and efficiency. The disparity between low-degree and high-degree node performance also persists, suggesting that the contrastive objective alone cannot fully reconcile the fundamental differences in how shallow methods and GNNs process graph information. While this enhanced combination method represents a significant step forward and achieves impressive robustness in attribute-poor scenarios, the difficulty in balancing computational efficiency with performance gains underscores the paper's core message: the field urgently requires novel architectures that can inherently capture the advantages of both approaches without the computational overhead of maintaining and coordinating multiple models.

Table 15: Experimental Results of the contrastive learning method across different datasets. Ratio represents the proportion of attribute dimensions over the original number of dimensions adopted for GNN training.

Dataset	Rank_B	Dim	Rank	Ratio	Accuracy	Accuracy Low Deg.	Accuracy High Deg.
Cora	32.0	32.0	32.0 ± 0.0	100%	69.8 ± 1.4	63.9 ± 1.5	79.7 ± 0.7
Cora	32.0	32.0	31.5 ± 0.7	1%	69.7 ± 1.9	61.5 ± 3.8	76.9 ± 0.0
Cora	32.0	32.0	4.0 ± 0.0	0.01%	70.6 ± 0.4	64.0 ± 0.8	77.1 ± 0.0
CiteSeer	32.0	32.0	32.0 ± 0.0	100%	50.2 ± 0.6	44.6 ± 3.1	58.9 ± 0.6
CiteSeer	32.0	32.0	32.0 ± 0.0	1%	45.2 ± 0.8	32.8 ± 0.0	57.7 ± 1.7
CiteSeer	32.0	32.0	3.0 ± 0.0	0.01%	47.6 ± 0.2	36.8 ± 0.6	58.6 ± 0.1
PubMed	32.0	32.0	30.5 ± 0.7	100%	74.5 ± 1.1	71.1 ± 3.0	77.8 ± 0.1
PubMed	32.0	32.0	20.5 ± 0.7	1%	71.1 ± 0.1	69.6 ± 0.0	72.5 ± 0.3
PubMed	32.0	32.0	1.5 ± 0.7	0.01%	72.5 ± 1.2	68.7 ± 1.5	76.3 ± 0.8
CoraFull	32.0	32.0	32.0 ± 0.0	100%	52.6 ± 0.7	47.5 ± 0.5	57.8 ± 0.9
CoraFull	32.0	32.0	32.0 ± 0.0	1%	47.8 ± 1.5	42.6 ± 1.1	53.0 ± 1.9
CoraFull	32.0	32.0	2.0 ± 0.0	0.01%	47.8 ± 0.9	42.8 ± 0.7	52.9 ± 1.1
DBLPFull	32.0	32.0	32.0 ± 0.0	100%	84.3 ± 0.2	81.5 ± 0.5	87.1 ± 0.1
DBLPFull	32.0	32.0	26.5 ± 2.1	1%	81.5 ± 0.5	76.7 ± 0.7	86.2 ± 0.4
DBLPFull	32.0	32.0	3.0 ± 0.0	0.01%	81.4 ± 0.2	76.4 ± 0.6	86.4 ± 0.1
Amz-C.	32.0	32.0	26.5 ± 4.9	100%	89.6 ± 0.6	87.6 ± 0.8	91.5 ± 0.5
Amz-C.	32.0	32.0	28.0 ± 1.4	1%	87.9 ± 0.2	85.2 ± 0.4	90.6 ± 0.8
Amz-C.	32.0	32.0	100%	0.01%	87.4 ± 0.7	84.9 ± 0.3	90.0 ± 1.1
Amz-P.	32.0	32.0	22.0 ± 0.0	100%	93.9 ± 0.0	91.7 ± 0.1	96.1 ± 0.1
Amz-P.	32.0	32.0	29.5 ± 0.7	1%	92.1 ± 0.1	88.6 ± 0.2	95.5 ± 0.5
Amz-P.	32.0	32.0	2.5 ± 0.7	0.01%	91.5 ± 0.5	87.8 ± 0.6	95.2 ± 0.4
Co-CS	32.0	32.0	32.0 ± 0.0	100%	89.5 ± 0.2	86.6 ± 0.5	92.4 ± 0.1
Co-CS	32.0	32.0	32.0 ± 0.0	1%	88.5 ± 0.4	85.2 ± 0.4	91.9 ± 0.3
Co-CS	32.0	32.0	3.5 ± 0.7	0.01%	87.3 ± 0.2	83.7 ± 0.0	91.0 ± 0.4
Co-Phy.	32.0	32.0	32.0 ± 0.0	100%	95.4 ± 0.2	93.2 ± 0.3	97.5 ± 0.0
Co-Phy.	32.0	32.0	30.5 ± 0.7	1%	95.2 ± 0.0	93.2 ± 0.2	97.2 ± 0.1
Co-Phy.	32.0	32.0	100%	0.01%	93.4 ± 0.2	90.0 ± 0.5	96.8 ± 0.0
Flickr	32.0	32.0	16.0 ± 2.8	100%	52.4 ± 0.1	54.3 ± 0.2	50.5 ± 0.4
Flickr	32.0	32.0	7.5 ± 0.7	1%	52.3 ± 0.1	54.3 ± 0.3	50.3 ± 0.5
Flickr	32.0	32.0	2.0 ± 0.0	0.01%	52.1 ± 0.1	54.1 ± 0.5	50.0 ± 0.3



(a) Cora. (b) CiteSeer.

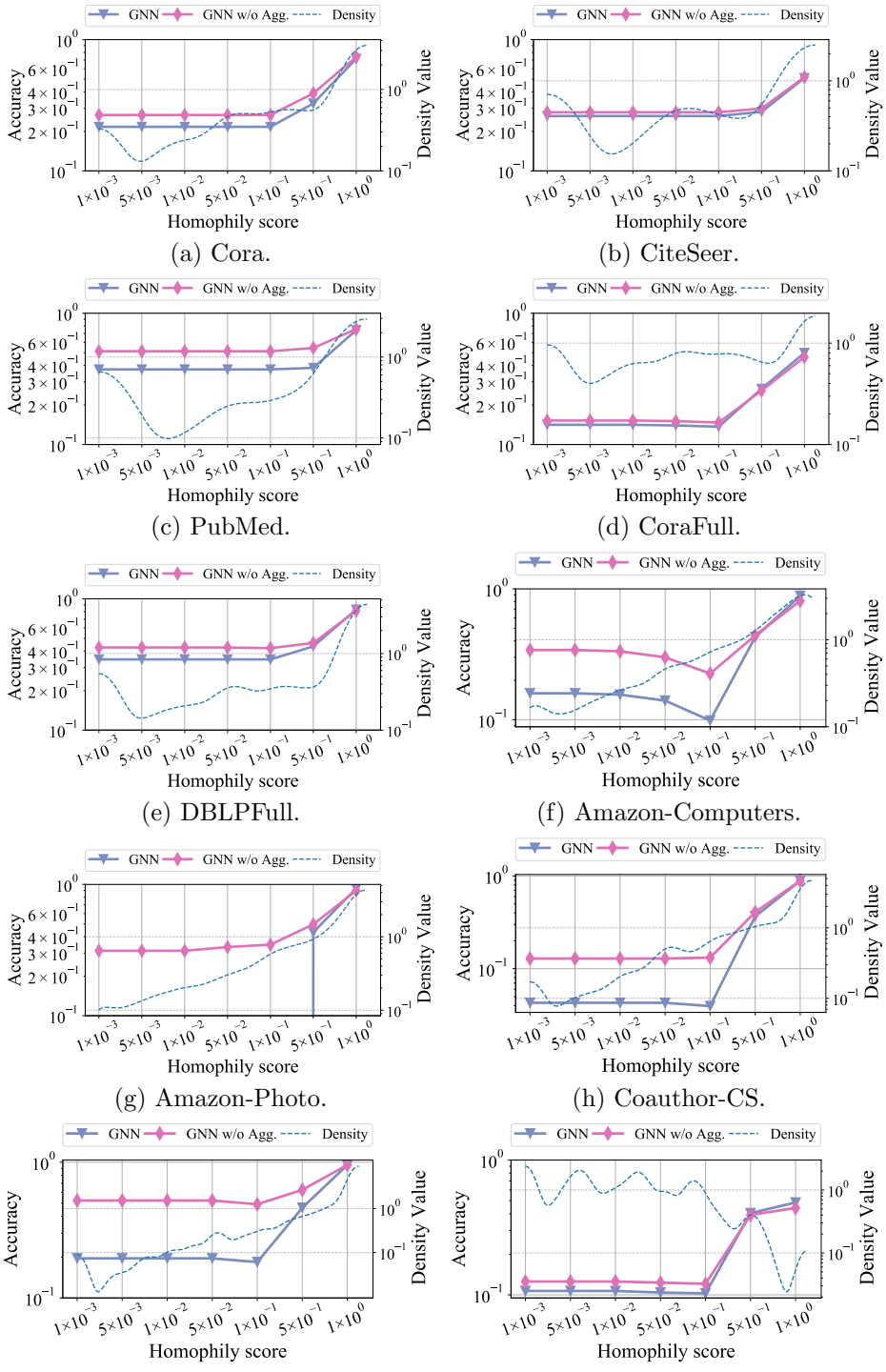
(c) PubMed. (d) CoraFull.

(e) DBLPFull. (f) Amazon-Computers.

(g) Amazon-Photo. (h) Coauthor-CS.

(i) Coauthor-Physics. (j) Flickr.

Figure 13: A comparison between shallow embedding methods and shallow embedding methods with an enforced neighborhood aggregation on 10 real-world graph datasets Here performance is measured with the cumulative node classification accuracy, and the density curve (marked with dashed line) represents the density of nodes with a certain homophily score in the test set.



(a) Cora. (b) CiteSeer.

(c) PubMed. (d) CoraFull.

(e) DBLPFull. (f) Amazon-Computers.

(g) Amazon-Photo. (h) Coauthor-CS.

(i) Coauthor-Physics. (j) Flickr.

Figure 14: A comparison between GNNs and GNNs without the neighborhood aggregation on 10 real-world graph datasets Here performance is measured with the cumulative node classification accuracy, and the density curve (marked with dashed line) denotes the density of nodes with a certain homophily score in the test set.