Non-Backtracking Graph Neural Network

Seonghyun Park[1†], **Narae Ryu**[2†], **Gahee Kim**[2], **Dongyeop Woo**[1],
Se-Young Yun[2‡], **Sungsoo Ahn**[1‡]
[1]POSTECH [2]KAIST
{shpark26,ehdduq1503,sungsoo.ahn}@postech.ac.kr
{nrryu,gaheekim,yunseyoung}@kaist.ac.kr

Abstract

The celebrated message-passing updates for graph neural networks allow the representation of large-scale graphs with local and computationally tractable updates. However, the local updates suffer from backtracking, i.e., a message flows through the same edge twice and revisits the previously visited node. Since the number of message flows increases exponentially with the number of updates, the redundancy in local updates prevents the graph neural network from accurately recognizing a particular message flow for downstream tasks. In this work, we propose to resolve such a redundancy via the non-backtracking graph neural network (NBA-GNN) that updates a message without incorporating the message from the previously visited node. We further investigate how NBA-GNN alleviates the over-squashing of GNNs, and establish a connection between NBA-GNN and the impressive performance of non-backtracking updates for stochastic block model recovery. We empirically verify the effectiveness of our NBA-GNN on long-range graph benchmark and transductive node classification problems.

1 Introduction

Recently, graph neural networks (GNNs) [1–3] have shown great success in various applications, including but not limited to, molecular property prediction [4] and community detection [5]. Such success can be largely attributed to the message-passing structure of GNNs, which provides a computationally tractable way of incorporating the overall graph through iterative updates based on local neighborhoods. However, the message-passing structure also brings challenges due to the parallel updates and memory-less behavior of messages passed along the graph.

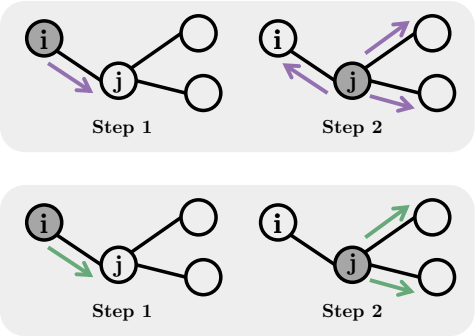


Figure 1: Message flows of simple (above) and non-backtracking (below) updates.

In particular, the message flow in a GNN is prone to backtracking, where the message from vertex i to vertex j is reincorporated in the subsequent message from j to i, e.g., Figure 1. Since the message-passing iteratively aggregates the information, the GNN inevitably encounters an exponential surge in the number of message flows, proportionate to the vertex degrees. This issue is compounded by backtracking, which accelerates the growth of message flows with redundant information.

† Equal Contribution.
‡ Co-corresponding author.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Interestingly, despite these challenges, non-backtracking updates—a potential solution—have been largely overlooked in the existing GNN research, while they have been thoroughly investigated for non-GNN message-passing algorithms or random walks [6, 7] (Figure 1). For example, given a pair of vertices i, j, the belief propagation algorithm [8] forbids an $i \to j$ message from incorporating the $j \to i$ message. Another example is the non-backtracking random walks [9] which are non-Markovian walks that do not traverse the same edge twice and revisit the previous node. Such classic algorithms have demonstrated great success in applications like probabilistic graphical model inference and stochastic block models [10–12]. In particular, the spectrum of the non-backtracking operator contains more useful information than that of the adjacency matrix in revealing the hidden structure of a graph model [11].

Contribution. In this work, we propose the non-backtracking graph neural network (NBA-GNN) which employs non-backtracking updates on the messages, i.e., forbids the message from vertex i to vertex j from being incorporated in the message from vertex j to i. To this end, we associate the hidden features with transitions between a pair of vertices, e.g., $h_{j\to i}$, and update them from features associated with non-backtracking transitions, e.g., $h_{k\to j}$ for $k \neq i$.

To motivate our work, we formulate "message flows" as the sensitivity of a GNN with respect to walks in the graph. Then we explain how the message flows are redundant; the GNN's sensitivity of a walk with backtracking transitions can be covered by that of other non-backtracking walks. We explain how the redundancy is harmful to the GNN since the number of walks increases exponentially as the number of layers grows and the GNN becomes insensitive to a particular walk information. Hence, reducing the redundancy by only considering non-backtracking walks would benefit the message-passing updates to better recognize each walk's information. We further make a connection from our sensitivity analysis to the over-squashing phenomenon for GNNs [13–15] in terms of access time.

Furthermore, we analyze our NBA-GNNs from the perspective of over-squashing and their expressive capability to recover sparse stochastic block models (SBMs). To this end, we prove that NBA-GNN improves the Jacobian-based measure of over-squashing [13] compared to its original GNN counterpart.

Next, we investigate NBA-GNN's proficiency in node classification within SBMs and its ability to distinguish between graphs originating from the Erdős–Rényi model or the SBM, from the results of [11, 16]. Unlike traditional GNNs that operate on adjacency matrices and necessitate an average degree of at least $\Omega(\log n)$, NBA-GNN demonstrates the ability to perform node classification with a substantially lower average degree bound of $\omega(1)$ and $n^{o(1)}$. Furthermore, the algorithm can accurately classify graphs even when the average degree remains constant.

Finally, we empirically evaluate our NBA-GNN on the long-range graph benchmark [17] and transductive node classification problems [18, 19]. We observe that our NBA-GNN demonstrates competitive performance and even achieves state-of-the-art performance on the long-rage graph benchmark. For the node classification tasks, we demonstrate that NBA-GNN consistently improves over its conventional GNN counterpart.

To summarize, our contributions are as follows:

- We propose NBA-GNN as a solution for the message flow redundancy problem in GNNs.
- We analyze how the NBA-GNN alleviates over-squashing and is expressive enough to recover sparse stochastic block models with an average degree of $o(\log n)$.
- We empirically verify our NBA-GNNs to show state-of-the-art performance on the long-range graph benchmark and consistently improve over the conventional GNNs across various tasks.

2 Related works

Non-backtracking algorithms. Non-backtracking updates have been considered by many classical algorithms [20, 21]. Belief propagation [8] infers the marginal distribution on probabilistic graphical models, and has demonstrated success for tree graphs [22] and graphs with large girth [23]. Moreover, non-backtracking walks have been utilized in graph kernels between labeled graphs [24, 25] , and further has been used for node classification [26]. Moreover, the non-backtracking has been shown to

yield better spectral separation properties, and its eigenspace contains information about the hidden structure of a graph model [11, 16].

Additionally, LGNN [27] has first used the non-backtracking operator in GNNs, though limited to community detection tasks. RFGNN [28] has removed redundancy by computing non-redundant computation graph for every nodes. We highlight the differences between the related works and our NBA-GNN in Appendix A, due to limited space.

Analyzing over-squashing of GNNs. When a node receives information from a k-hop neighbor node, an exponential number of messages passes through node representations with fixed-sized vectors. This leads to the loss of information, denoted as *over-squashing* [29], and has been formalized in terms of sensitivity [13, 15]. Hence, sensitivity is defined as the Jacobian of a node feature at a GNN layer with respect to the input node, and can be upper bounded via the graph topology. Stemming from this, graph rewiring methods mitigate over-squashing by adding or removing edges for an optimal graph topology to compute [13–15]. Another line of work uses global aspects, e.g., connecting a virtual global node [30] or using graph Transformers [31–34].

To resolve this issue, prior works have considered graph rewiring methods that add or remove edges to mitigate over-squashing by creating an optimal computation graph. One approach is using topological metrics, e.g., curvature [13] or effective resistance [14]. Another line of work uses global aspects, e.g., connecting a virtual global node [30] or using graph Transformers [31–35].

Expressivity of GNNs with the Weisfeiler-Lehman (WL) test. In the past few years, researchers have conducted significant studies on various aspects of the expressive power of GNNs. One line of research involves comparing GNNs with the WL test [36] to assess their expressiveness. For instance, graph isomorphism network (GIN) was introduced to demonstrate that MPNNs are at best as powerful as the WL test [3]. Besides, the expressivity of GNNs have been analyzed by linear algebraic tools and eigenvalue decomposition of graph operators [37].

Expres sivity of GNNs for the stochastic block model (SBM). Furthermore, certain studies have analyzed the expressive power of GNNs using variations of the SBM [38]. [39] established conditions for the existence of graph attention networks (GATs) that can precisely classify nodes in the contextual stochastic block model (CSBM) with high probability. Similarly, [40] investigated the effects of graph convolutions within a network on the XOR-CSBM. These works focused primarily on the probability distribution of node features, such as the distance between the means of feature vectors.

3 Non-backtracking graph neural network

3.1 Motivation from sensitivity analysis

We first explain how the conventional message-passing updates are prone to backtracking. To this end, consider a simple, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{N}(i)$ denotes the neighbor of the node i. Each node $i \in \mathcal{V}$ is associated with a feature x_i. Then the conventional graph neural networks (GNNs), i.e., message-passing neural networks (MPNNs) [4], iteratively updates the t-th layer node-wise hidden feature $h_i^{(t)}$ as follows:

$$h_i^{(t+1)} = \phi^{(t)}\left(h_i^{(t)}, \left\{\psi^{(t)}\left(h_i^{(t)}, h_j^{(t)}\right) : j \in \mathcal{N}(i)\right\}\right), \tag{1}$$

where $\phi^{(t)}$ and $\psi^{(t)}$ are architecture-specific non-linear update and permutation invariant aggregation functions, respectively. Our key observation is that the message from the node feature $h_i^{(t)}$ to the node feature $h_j^{(t+1)}$ is reincorporated in the node feature $h_i^{(t+2)}$, e.g., Figure 3a shows the computation graph of conventional GNNs where redundant messages are incorporated.

Sensitivity analysis. To concretely describe the backtracking nature of message-passing updates, we formulate the sensitivity of the final node feature $h_i^{(T)}$ with respect to the input as follows:

$$\sum_{j \in \mathcal{V}} \frac{\partial h_i^{(T)}}{\partial h_j^{(0)}} = \sum_{s \in \mathcal{W}(i)} \prod_{t=1}^{T} \frac{\partial h_{s(t)}^{(t)}}{\partial h_{s(t-1)}^{(t-1)}}, \tag{2}$$

where $h_i^{(0)} = x_i$, $\mathcal{W}(i)$ denotes the set of T-step walks ending at node i, and $s(t)$ denotes the t-th node in the walk $s \in \mathcal{W}(i)$. Intuitively, this equation shows that a GNN with T layer recognizes the graph via aggregation of random walks with length T. Our key observation from Equation (2) is on how the feature $h_i^{(T)}$ is insensitive to the information to an initial node feature $h_j^{(0)}$, due to the information being "squashed" by the aggregation over the exponential number of walks $\mathcal{W}(i)$. A similar analysis has been conducted on how a node feature $h_i^{(T)}$ is insensitive to the far-away initial node feature $h_j^{(0)} = x_j$, i.e., the over-squashing phenomenon of GNNs [13].

Redundancy of walks with backtracking. In particular, a walk s randomly sampled from $\mathcal{W}(i)$ is likely to contain a transition that backtracks, i.e., $s(t) = s(t+2)$ for some $t < T$. Then the walk s would be redundant since the information is contained in two other walks in $\mathcal{W}(i)$: $s(0), \ldots, s(t+1)$ and $s(0), \ldots, s(t+1), s(t) = s(t+2), s(t+3), ..., s(T)$. See Figure 2 for an illustration. This idea leads to the conclusion that non-backtracking walks, i.e., walks that do not contain backtracking transitions, are sufficient to express the information in the walks $\mathcal{W}(i)$. Since the exponential number of walks in $\mathcal{W}(i)$ causes the GNN to be insensitive to a particular walk information, it makes sense to design a non-backtracking GNN that is sensitive to the constrained set of non-backtracking walks. We note that similar concepts of redundancy for GNNs have been studied in prior works [28, 41]. See Appendix A for a detailed comparison.

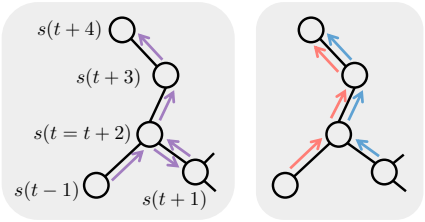


Figure 2: Two non-backtracking walks (right) are sufficient to express information in a walk with backtracking transition.

Relation to over-squashing. Finally, we point out an intriguing motivation for our work in terms of over-squashing. In particular, we note that the lower bound for the Jacobian obstruction has been analyzed [15], measuring the degree of over-squashing in terms of access time with respect to a simple random walk. They conclude that the degree of over-squashing, i.e., the size of Jacobian obstruction, is higher for a pair of nodes with longer access time.

Hence, to design a GNN architecture robust to the over-squashing phenomenon, one could (i) propose a random walk that has shorter access time in general for a pair of nodes in the graph and (ii) design a GNN that aligns with the optimized random walk. Non-backtracking random walks have been empirically shown and believed to generally yield faster access time than simple random walks [42, 43]. Hence, one could aim to design a GNN architecture that aligns with the non-backtracking random walks.

However, to the best of our knowledge, there is no formal proof of scenarios where non-backtracking random walks yield smaller access time. As a motivating example, we provide a theoretical result comparing the access time of non-backtracking and simple random walks for tree graphs.

Proposition 1. *Given a tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a pair of nodes $i, j \in \mathcal{V}$, the access time of begrudgingly backtracking random walk is equal or smaller than that of a simple random walk, where the equality holds if and only if the random walk length is 1.*

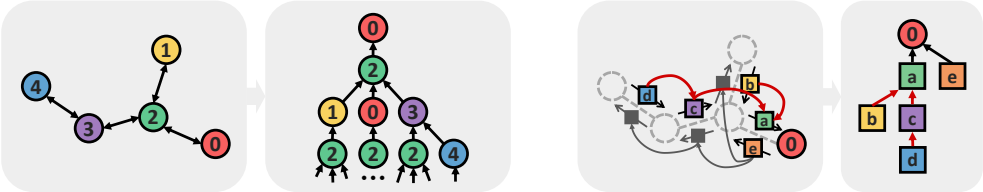
The begrudgingly backtracking random walk [7] modifies non-backtracking random walks to remove "dead ends" for tree graphs. Please refer to Appendix B for the proof.

3.2 Method description

In this section, we present our **N**on-**BA**cktracking GNNs (NBA-GNN) with the motivation described in Section 3.1. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, our NBA-GNN associates a pair of hidden features $h_{i\to j}^{(t)}, h_{j\to i}^{(t)}$ for each edge $\{i, j\}$. Then the non-backtracking message passing update for a hidden feature $h_{j\to i}^{(t)}$ is defined as follows:

$$h_{j\to i}^{(t+1)} = \phi^{(t)}\left(h_{j\to i}^{(t)}, \left\{\psi^{(t)}\left(h_{k\to j}^{(t)}, h_{j\to i}^{(t)}\right) : k \in \mathcal{N}(j) \setminus \{i\}\right\}\right), \tag{3}$$

where $\phi^{(t)}$ and $\psi^{(t)}$ are backbone-specific non-linear update and permutation-invariant aggregation functions at the t-th layer, respectively. For example, $\psi^{(t)}$ and $\phi^{(t)}$ are multi-layer perceptron and



(a) Computation graph of typical GNNs

(b) Computation graph of NBA-GNNs

Figure 3: Computation graph of typical GNN and NBA-GNN predicting node "0". (a) Redundant messages increase the size of the computation graph. (b) NBA-GNN assigns a pair of features for each edge and update them via non-backtracking message passing, reducing redundant messages.

summation over a set for the graph isomorphism network [3], respectively. Given the update in Equation (3), one can observe that the message $h^{(t)}_{i\to j}$ is never incorporated in the message $h^{(t+1)}_{j\to i}$, and hence the update is free from backtracking.

Initialization and node-wise aggregation of messages. The message at the 0-th layer $h^{(0)}_{i\to j}$ is initialized by encoding the node features x_i, x_j, and the edge feature e_{ij} using a non-linear function ϕ. After updating hidden features for each edge based on Equation (3), we apply permutation-invariant pooling over all the messages for graph-wise predictions. Since we use hidden features for each edge, we construct the node-wise predictions at the final layer as follows:

$$h_i = \sigma\left(\rho\left\{h^{(T)}_{j\to i} : j \in \mathcal{N}(i)\right\}, \rho\left\{h^{(T)}_{i\to j} : j \in \mathcal{N}(i)\right\}\right), \tag{4}$$

where σ is a non-linear aggregation function with different weights for incoming edges $j \to i$ and outgoing edges $i \to j$, ρ is a non-linear aggregation function invariant to the permutation of nodes in $\mathcal{N}(i)$. We provide a computation graph of NBA-GNNs in Figure 3b to summarize our algorithm.

Begrudgingly backtracking update. While the messages from our update are resistant to backtracking, a message $h^{(t)}_{j\to i}$ may get trapped in node i for the special case when $\mathcal{N}(i) = \{j\}$. To resolve this issue, we introduce a simple trick coined begrudgingly backtracking update [7] that updates $h^{(t+1)}_{i\to j}$ using $h^{(t)}_{j\to i}$ only when $\mathcal{N}(i) = \{j\}$. We empirically verify the effectiveness of begrudgingly backtracking updates in Section 5.3.

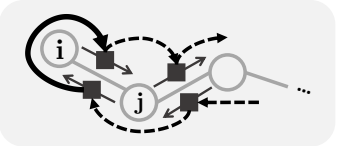


Figure 4: Begrudgingly backtracking update (solid).

Implementation. To better understand our NBA-GNN, we provide an example of non-backtracking message-passing updates with a GCN backbone [1], coined NBA-GCN. The message update at the t-th layer of NBA-GCN can be written as follows:

$$h^{(t+1)}_{j\to i} = h^{(t)}_{j\to i} + \sigma_{\text{GCN}}\left(\frac{1}{|\mathcal{N}(j)| - 1}\mathbf{W}^{(t)} \sum_{k\in\mathcal{N}(j)\setminus\{i\}} h^{(t)}_{k\to j}\right), \tag{5}$$

where σ_{GCN} is an element-wise nonlinear function, e.g., rectified linear unit [44], $\mathbf{W}^{(t)}$ is the weight matrix, and the messages are normalized by their population $|\mathcal{N}(j)| - 1$.

4 Theoretical analysis

In this section, we analyze the proposed NBA-GNN framework. To be specific, we show that (a) our NBA-GNN improves the upper bound for sensitivity-based measures of GNN over-squashing and (b) NBA-GNNs can detect the underlying structure of SBMs even for very sparse graphs.

4.1 Sensitivity analysis on over-squashing

We first analyze how NBA-GNNs alleviate the over-squashing issue. To this end, we use the Jacobian of node-wise output with respect to another node initial feature as a way to assess over-squashing

effect [13]. We first derive the Jacobian quantification of over-squashing for the NBA-GNNs, and then demonstrate that the upper bound for Jacobian is larger than that of the conventional GNNs [13].

To derive our analysis, we introduce the non-backtracking matrix $B \in \{0,1\}^{2|\mathcal{E}|\times 2|\mathcal{E}|}$ and the incidence matrix $C \in \mathbb{R}^{2|\mathcal{E}|\times|\mathcal{V}|}$ which describe the NBA-GNN message-passing and node-wise aggregation via linear operation, respectively. To be specific, the backtracking matrix B and the incidence matrix C are defined as follows:

$$B_{(\ell\to k),(j\to i)} = \begin{cases} 1 & \text{if } k=j, \ell\neq i \\ 0 & \text{otherwise} \end{cases}, \qquad C_{(k\to j),i} = \begin{cases} 1 & \text{if } j=i \text{ or } k=i \\ 0 & \text{otherwise} \end{cases}.$$

We also define D_{out} and D_{in} as the out-degree and in-degree matrices of NBA-GNN, respectively, counting the number of outgoing and incoming edges for each edge. These are diagonal matrices with $(D_{out})_{(j\to i),(j\to i)} = \sum_{\ell\to k} B_{(j\to i),(\ell\to k)}$ and $(D_{in})_{(j\to i),(j\to i)} = \sum_{\ell\to k} B_{(\ell\to k),(j\to i)}$. Next, we introduce $\widehat{B}$ as the normalized non-backtracking matrix augmented with self-loops: $\widehat{B} = (D_{out}+I)^{-\frac{1}{2}}(B+I)(D_{in}+I)^{-\frac{1}{2}}$. Finally, we let $\tilde{C}$ denote a matrix where $\tilde{C}_{(k\to j),i} = C_{(k\to j),i} + C_{(j\to k),i}$. Then, one obtains the following sensitivity bound of NBA-GNN.

Lemma 1. *Consider two nodes $i,j \in \mathcal{V}$ with distance T given a $(T-1)$-layer NBA-GNN as described in Equation (3) and Equation (4). Suppose $|\nabla\phi^{(t)}|, |\nabla\sigma| \leq \alpha$, $|\nabla\psi^{(t)}|, |\nabla\rho| \leq \beta$, and $|\nabla\phi| \leq \gamma$ for $0 \leq t < T$. Then the following holds:*

$$\left\| \frac{\partial h_j}{\partial x_i} \right\| \leq (\alpha\beta)^T \gamma (\tilde{C}^\top \widehat{B}^{(T-1)} \tilde{C})_{j,i}.$$

We provide the proof in Appendix C. Lemma 1 states how *the over-squashing effect is controlled by the power of $\widehat{B}$*. Consequently, one can infer that increasing the upper bound likely leads to mitigation of the GNN over-squashing effect [13].

From this motivation, we provide an analysis to support our claim on how the NBA-GNN suffers less from over-squashing effect due to its larger sensitivity bound.

Theorem 1. *Let $\widehat{A}$ denote the degree-normalized matrix. Then, for any pair of nodes $i,j \in \mathcal{V}$ with distance T, the sensitivity bound of NBA-GNN is larger than that of conventional GNNs [13], i.e., $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i} \geq (\widehat{A}^T)_{j,i}$. For d-regular graphs, $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i}$ decays slower by $O(d^{-T})$, while $(\widehat{A}^T)_{j,i}$ decays with $O((d+1)^{-T})$.*

Note that $(\alpha\beta)^T(\widehat{A}^T)_{j,i}$ provides an upper bound for the sensitivity term in conventional GNNs [13]. We provide the proof in Appendix C. Our proof is based on comparing the degree-normalized number of non-backtracking and simple walks from node i to node j.

4.2 Expressive power of NBA-GNN in the lens of SBMs

In the literature on the expressive capabilities of GNNs, comparisons with the well-known k-WL test are common. However, even when certain models surpass the k-WL test in performance, evaluating their relative merits remains a nontrivial task. Furthermore, due to the substantial performance gap between the 1-WL and 3-WL tests, many algorithms fall into a range between these two tests, making it more difficult to compare them with each other. It is also worth noting that comparing GNNs with the WL test does not always accurately reflect their performance on real-world datasets.

To address these issues, several studies have turned to spectral analysis of GNNs. From a spectral viewpoint, GNNs can be seen as functions of the eigenvectors and eigenvalues of the given graph. [45] showed that GNNs operate as low-pass filters on the graph spectrum, and [46] analyzed the use of various GNNs as filters to extract the relevant graph spectrum and measure their expressive power. Moreover, [47] argue that the expressive power of GNNs is influenced by the topological information contained in the graph spectrum.

The eigenvalues and the corresponding adjacency matrix eigenvectors play a pivotal role in establishing the fundamental limits of community detection in SBM [48–52]. The adjacency matrix exhibits a spectral separation property, and an eigenvector containing information about the assignments of the vertex community becomes apparent [53]. Furthermore, by analyzing the eigenvalues of the adjacency

Table 1: Comparison of conventional MPNNs and GNNs in the long-range graph benchmark, with and without Laplacian positional encoding. We also denote the relative improvement by Imp.

Model	`Peptides-func`		`Peptides-struct`		`PascalVOC-SP`	
	AP ↑	Imp.	MAE ↓	Imp.	F1 ↑	Imp.
GCN	0.5930 ± 0.0023		0.3496 ± 0.0013		0.1268 ± 0.0060	
+ NBA	0.6951 ± 0.0024	+17%	0.2656 ± 0.0009	+22%	0.2537 ± 0.0054	+100%
+ NBA+LapPE	**0.7206** ± **0.0028**	+22%	**0.2472** ± **0.0008**	+29%	**0.3005** ± **0.0010**	+137%
GIN	0.5498 ± 0.0079		0.3547 ± 0.0045		0.1265 ± 0.0076	
+ NBA	0.6961 ± 0.0045	+27%	0.2534 ± 0.0025	+29%	0.3040 ± 0.0119	+140%
+ NBA+LapPE	**0.7071** ± **0.0067**	+29%	**0.2424** ± **0.0010**	+32%	**0.3223** ± **0.0010**	+155%
GatedGCN	0.5864 ± 0.0077		0.3420 ± 0.0013		0.2873 ± 0.0219	
+ NBA	0.6429 ± 0.0062	+10%	0.2539 ± 0.0011	+26%	0.3910 ± 0.0010	+36%
+ NBA+LapPE	**0.6982** ± **0.0014**	+19%	**0.2466** ± **0.0012**	+28%	**0.3969** ± **0.0027**	+38%

matrix, it is feasible to determine whether a graph originates from the Erdős–Rényi (ER) model or the SBM [54, 55]. However, these spectral properties are particularly salient when the average degree of the graph satisfies $\Omega(\log n)$. For graphs with average degrees $o(\log n)$, vertices with higher degrees predominate, affecting eigenvalues and complicating the discovery of the underlying structure of the graph [56].

In contrast, the non-backtracking matrix exhibits several advantageous properties even for constant-degree cases. In [16], the non-backtracking matrix demonstrates a spectral separation property and establishes the presence of an eigenvector containing information about vertex community assignments, when the average degree only satisfies $\omega(1)$ and $n^{o(1)}$. Furthermore, [11] have demonstrated that by inspecting the eigenvalues of the non-backtracking matrix, it is possible to discern whether a graph originates from the ER model or the SBM, even when the graph's average degree remains constant. This capability sets NBA-GNNs apart and enhances their performance in both node and graph classification tasks, especially in sparse settings. These lines of reasoning lead to the formulation of the following theorem.

Theorem 2. ***(Informal)*** *Assume that the average degree in the stochastic block model satisfies the conditions of being at least $\omega(1)$ and $n^{o(1)}$. In such a scenario, the NBA-GNN can map from graph $\mathcal{G}$ to node labels.*

Theorem 3. ***(Informal)*** *Suppose we have a pair of graphs with a constant average degree, one generated from the stochastic block model and the other from the Erdős–Rényi model. In this scenario, the NBA-GNN is capable of distinguishing between them.*

For an in-depth exploration of this argument, please refer to Appendix D. The rationale behind these valuable properties of the non-backtracking matrix B in sparse scenarios lies in the fact that the matrix B^k exhibits similarity to the k-hop adjacency matrix, while A^k is mainly influenced by high-degree vertices. For these reasons, NBA-GNNs would outperform traditional GNNs in both node and graph classification tasks, particularly in sparse graph environments.

5 Experiment

In this section, we assess the effectiveness of NBA-GNNs across multiple benchmarks on graph classification, graph regression, and node classification tasks. Our method shows competitive performance compared to graph Transformers within long-range graph benchmarks, and robust improvements in handling transductive node classification tasks. We also conduct ablation studies to verify our method, and provide experimental details in Appendix E.

5.1 Long-range graph benchmark

The long-range graph benchmark (LRGB) [17] is a set of tasks that require learning long-range interactions. We validate our method using three datasets from the LRGB benchmark: graph classification (`Peptides-func`), graph regression (`Peptides-struct`), and node classification (`PascalVOC-SP`).

Table 2: Evaluation of NBA-GNN on the LRGB benchmark. We color the **first-**, **second-** and **third-**best results. Performance within a standard deviation of one another is considered equal. Non-reported values are denoted by -.

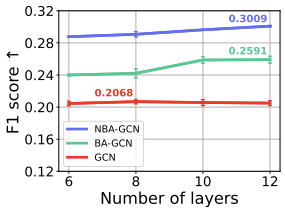
Method	**Model**	`Peptides-func` AP ↑	`Peptides-struct` MAE ↓	`VOC-SP` F1 ↑
GNNs	GCN	0.5930 ± 0.0023	0.3496 ± 0.0013	0.1268 ± 0.0060
	GIN	0.5498 ± 0.0079	0.3547 ± 0.0045	0.1265 ± 0.0076
	GatedGCN	0.5864 ± 0.0077	0.3420 ± 0.0013	0.2873 ± 0.0219
	GatedGCN+PE	0.6069 ± 0.0035	0.3357 ± 0.0006	0.2860 ± 0.0085
Subgraph GNNs	MixHop-GCN	0.6592 ± 0.0036	0.2921 ± 0.0023	0.2506 ± 0.0133
	MixHop-GCN+LapPE	0.6843 ± 0.0049	0.2614 ± 0.0023	0.2218 ± 0.0174
	PathNN	0.6816 ± 0.0026	0.2545 ± 0.0032	-
	CIN++	0.6569 ± 0.0117	0.2523 ± 0.0013	-
Transformers	Transformer+LapPE	0.6326 ± 0.0126	0.2529 ± 0.0016	0.2694 ± 0.0098
	GraphGPS+LapPE	0.6535 ± 0.0041	0.2500 ± 0.0005	**0.3748 ± 0.0109**
	SAN+LapPE	0.6384 ± 0.0121	0.2683 ± 0.0043	0.3230 ± 0.0039
	Exphormer	0.6527 ± 0.0043	0.2481 ± 0.0007	**0.3966 ± 0.0027**
	Graph MLP-Mixer/ViT	0.6970 ± 0.0080	**0.2449 ± 0.0016**	-
Rewiring methods	DIGL+MPNN	0.6469 ± 0.0019	0.3173 ± 0.0007	0.2824 ± 0.0039
	DIGL+MPNN+LapPE	0.6830 ± 0.0026	0.2616 ± 0.0018	0.2921 ± 0.0038
	DRew-GCN+LapPE	**0.7150 ± 0.0044**	0.2536 ± 0.0015	0.1851 ± 0.0092
	DRew-GIN+LapPE	**0.7126 ± 0.0045**	0.2606 ± 0.0014	0.2692 ± 0.0059
	DRew-GatedGCN+LapPE	0.6977 ± 0.0026	0.2539 ± 0.0007	0.3314 ± 0.0024
NBA-GNNs (Ours)	NBA-GCN	0.6951 ± 0.0024	0.2656 ± 0.0009	0.2537 ± 0.0054
	NBA-GCN+LapPE	**0.7207 ± 0.0028**	**0.2472 ± 0.0008**	0.3005 ± 0.0010
	NBA-GIN	0.6961 ± 0.0045	0.2775 ± 0.0057	0.3040 ± 0.0119
	NBA-GIN+LapPE	**0.7071 ± 0.0067**	**0.2424 ± 0.0010**	0.3223 ± 0.0063
	NBA-GatedGCN	0.6429 ± 0.0062	0.2539 ± 0.0011	**0.3910 ± 0.0010**
	NBA-GatedGCN+LapPE	0.6982 ± 0.0014	**0.2466 ± 0.0012**	**0.3969 ± 0.0027**

Table 3: Comparison of conventional GNNs and their NBA-GNN counterpart on transductive node classification tasks, with and without positional encoding. We mark the best numbers in bold.

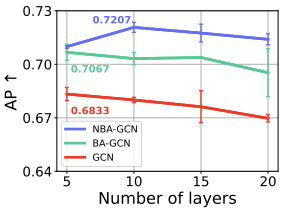
Model	`Cora`	`CiteSeer`	`PubMed`	`Texas`	`Wisconsin`	`Cornell`
GCN	0.8658±0.0060	0.7532±0.0134	0.8825±0.0042	0.6162±0.0634	0.6059±0.0438	0.5946±0.0662
+NBA	**0.8722±0.0095**	0.7585±0.0175	0.8826±0.0044	**0.7108±0.0796**	**0.7471±0.0386**	0.6108±0.0614
+NBA+LapPE	0.8720±0.0129	**0.7609±0.0186**	**0.8827±0.0048**	0.6811±0.0595	**0.7471±0.0466**	**0.6378±0.0317**
GraphSAGE	0.8632±0.0158	0.7559±0.0161	0.8864±0.0030	0.7108±0.0556	0.7706±0.0403	0.6027±0.0625
+NBA	**0.8702±0.0083**	0.7586±0.0213	**0.8871±0.0044**	0.7270±0.0905	**0.7765±0.0508**	**0.6459±0.0691**
+NBA+LapPE	0.8650±0.0120	**0.7621±0.0172**	0.8870±0.0037	**0.7486±0.0612**	0.7647±0.0531	0.6378±0.0544
GAT	0.8694±0.0119	0.7463±0.0159	0.8787±0.0046	0.6054±0.0386	0.6000±0.0491	0.4757±0.0614
+NBA	**0.8722±0.0120**	0.7549±0.0171	**0.8829±0.0043**	0.6622±0.0514	0.7059±0.0562	**0.5838±0.0558**
+NBA+LapPE	0.8692±0.0098	**0.7561±0.0175**	0.8822±0.0047	**0.6730±0.0348**	**0.7314±0.0531**	0.5784±0.0640
GatedGCN	0.8477±0.0156	0.7325±0.0192	**0.8671** ±0.0060	0.6108±0.0652	0.5824±0.0641	0.5216±0.0987
+NBA	**0.8523±0.0095**	**0.7405±0.0187**	0.8661±0.0035	0.6162±0.0490	0.6431±0.0356	**0.5649±0.0532**
+NBA+LapPE	0.8517±0.0130	0.7379±0.0193	0.8661±0.0047	**0.6243±0.0467**	**0.6569±0.0310**	0.5405±0.0785

We use performance scores from [17] and from each baseline papers: subgraph based GNNs [57–59], graph Transformers [32–35], and graph rewiring methods [60, 61]. For NBA-GNNS and NBA-GNNs with begrudgingly backtracking, we report the one with better performance. Furthermore, LapPE, i.e., Laplacian positional encoding [62], is applied, as it enhances the performance of NBA-GNNs in common cases.

As one can see in Table 1, NBA-GNNs show improvement regardless of the combined backbone GNNs, i.e., GCN [1], GIN [3], and GatedGCN [63]. Furthermore, even against a variety of baselines in Table 2, at least one NBA-GNNs shows competitive performance with the best baseline. Also, it is noteworthy that the improvement of NBA-GNNs is higher in dense graphs, `PascalVOC-SP` having an average degree of 8 while `Peptides-func`, `Peptides-struct` has an average degree of 2.



(a) F1 for changes in number of layers in `PascalVOC-SP`



(b) AP for changes in number of layers in `Peptides-func`

Model	BG.	AP ↑	MAE ↓
GCN	✗	0.7015	**0.2472**
	✓	**0.7207**	0.2547
GIN	✗	0.6825	**0.2424**
	✓	**0.7071**	0.2479
GatedGCN	✗	0.6710	**0.2466**
	✓	**0.6982**	0.2489

(c) Performance of begrudgingly, non-backtracking in sparse graphs

Figure 5: Ablation studies on the components of NBA-GNN.

5.2 Transductive node classification tasks

We conduct experiments on three citation networks (Cora, CiteSeer, and Pubmed) [18], and three heterophilic datasets (Texas, Wisconsin, and Cornell) [19] for transductive node classification. In our experiments, we employed various GNN architectures [1, 2, 64, 65], along with their corresponding NBA versions, as illustrated in Table 3. The results indicate that the NBA operation improves the performance of all GNN variants. Specifically, it demonstrates significant enhancements, particularly on heterophilic datasets. Given that related nodes in heterophilic graphs are often widely separated [66], the ability of NBA-GNN to alleviate over-squashing plays a vital role in classifying nodes in such scenarios.

5.3 Ablation studies

Here, we conduct ablation studies to empirically verify our framework. For simplicity, we use BA for backtracking GNNs and BG for begrudgingly backtracking GNNs. All experiments are averaged over 3 seeds. We use the suggested hyper-parameters for GCN [67].

Non-backtracking vs. backtracking GNNs. We first verify whether the performance improvements indeed stem from the non-backtracking updates. To this end, we compare our NBA-GNN with a backtracking variant, coined BA-GNN. To be specific, BA-GNN allows backtracking update prohibited in NBA-GNN , i.e., using $h^{(\ell)}_{i\to j}$ to update $h^{(\ell+1)}_{j\to i}$. From Figures 5a and 5b, one can observe how NBA-GNN outperforms the BA-GNN consistently irrelevant to the numbers of layers. Intriguingly, one can also observe that BA-GNN outperforms the naïve backbone, i.e., GCN, consistently.

Begrudgingly backtracking updates in sparse graphs. Additionally, we investigate the effect of begrudgingly backtracking in sparse graphs, i.e., `Peptides-func`, and `Peptides-struct`, in Figure 5c. One can see that begrudgingly backtracking is effective in `Peptides-func`, and shows similar performance in `Peptides-struct` (Note that the `PascalVOC-SP` does not have a vertex with degree one).

6 Conclusion

We have introduced message passing framework applicable to any GNN architectures, bringing non-backtracking into graph neural networks. As theoretically shown, NBA-GNNs mitigate over-squashing in terms of sensitivity, and enhance its expressive power for both node and graph classification tasks on SBMs. Additionally, we demonstrate that NBA-GNNs achieve competitive performance on the LRGB benchmark and outperform conventional GNNs across various tasks.

Limitations. The space complexity of our framework is larger than the conventional GNNs. It creates $2|\mathcal{E}|$ messages considering directions, and $\sum_{i\in\mathcal{V}}\binom{d_i}{2}$ connections between them where d_i is the degree of a node i. The time complexity is also $\sum_{i\in\mathcal{V}}\binom{d_i}{2}$, since we do not add additional message-passing to the backbone GNN. We provide more detail of the space and time complexity in Appendix F. Therefore, investigation of space complexity reduction for our work would be an interesting future work.

Acknowledgements

N. Ryu and S. Yun were supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT)(No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)). S. Park and S. Ahn were supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. IITP-2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)), the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1C1C1013366), and Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2022R1A6A1A03052954).

References

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 30, 2017.

[3] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[5] Joan Bruna and X Li. Community detection with graph neural networks. *stat*, 1050:27, 2017.

[6] Robert Fitzner and Remco van der Hofstad. Non-backtracking random walk. *Journal of Statistical Physics*, 150:264–284, 2013.

[7] Brian Rappaport, Anuththari Gamage, and Shuchin Aeron. Faster clustering via non-backtracking random walks. *stat*, 1050:26, 2017.

[8] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the AAAI conference on artificial intelligence*, pages 133–136, 1982.

[9] Noga Alon, Itai Benjamini, Eyal Lubetzky, and Sasha Sodin. Non-backtracking random walks mix faster. *Communications in Contemporary Mathematics*, 9(04):585–603, 2007.

[10] Laurent Massoulié. Community detection thresholds and the weak ramanujan property. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 694–703, 2014.

[11] Charles Bordenave, Marc Lelarge, and Laurent Massoulié. Non-backtracking spectrum of random graphs: community detection and non-regular ramanujan graphs. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1347–1357. IEEE, 2015.

[12] Emmanuel Abbe and Colin Sandon. Detection in the stochastic block model with multiple clusters: proof of the achievability conjectures, acyclic bp, and the information-computation gap. *arXiv preprint arXiv:1512.09080*, 2015.

[13] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

[14] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning*, pages 2528–2547. PMLR, 2023.

[15] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pages 7865–7885. PMLR, 2023.

[16] Ludovic Stephan and Laurent Massoulié. Non-backtracking spectra of weighted inhomogeneous random graphs. *Mathematical Statistics and Learning*, 5(3):201–271, 2022.

[17] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

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

[19] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2019.

[20] MEJ Newman. Spectral community detection in sparse networks. *arXiv preprint arXiv:1308.6494*, 2013.

[21] Mark Condie Kempton. Non-backtracking random walks and a weighted ihara's theorem. *Open Journal of Discrete Mathematics*, 2016.

[22] JinHyung Kim and Judea Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *International Joint Conference on Artificial Intelligence*, pages 0–0, 1983.

[23] Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*, 2013.

[24] Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Extensions of marginalized graph kernels. In *Proceedings of the twenty-first international conference on Machine learning*, page 70, 2004.

[25] Furqan Aziz, Richard C Wilson, and Edwin R Hancock. Backtrackless walks on a graph. *IEEE transactions on neural networks and learning systems*, 24(6):977–989, 2013.

[26] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52):20935–20940, 2013.

[27] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[28] Rongqin Chen, Shenghui Zhang, Ye Li, et al. Redundancy-free message passing for graph neural networks. *Advances in Neural Information Processing Systems*, 35:4316–4327, 2022.

[29] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[30] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between mpnn and graph transformer. *arXiv preprint arXiv:2301.11956*, 2023.

[31] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[32] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[33] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[34] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, 2023.

[35] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning*, pages 12724–12745. PMLR, 2023.

[36] AA Leman and Boris Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

[37] Charilaos I. Kanatsoulis and Alejandro Ribeiro. Representation power of graph neural networks: Improved expressivity via algebraic analysis, 2023.

[38] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[39] Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. Graph attention retrospective. *arXiv preprint arXiv:2202.13060*, 2022.

[40] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Effects of graph convolutions in deep networks. *arXiv preprint arXiv:2204.09297*, 2022.

[41] Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. Redundancy-free computation for graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 997–1005, 2020.

[42] Yuan Lin and Zhongzhi Zhang. Non-backtracking centrality based random walk on networks. *The Computer Journal*, 62(1):63–80, 2019.

[43] Dario Fasino, Arianna Tonetto, and Francesco Tudisco. Hitting times for second-order random walks. *European Journal of Applied Mathematics*, 34(4):642–666, 2023.

[44] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

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

[46] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2020.

[47] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

[48] Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on information theory*, 62(1):471–487, 2015.

[49] Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 670–688. IEEE, 2015.

[50] Bruce Hajek, Yihong Wu, and Jiaming Xu. Achieving exact cluster recovery threshold via semidefinite programming. *IEEE Transactions on Information Theory*, 62(5):2788–2797, 2016.

[51] Se-Young Yun and Alexandre Proutiere. Optimal cluster recovery in the labeled stochastic block model. *Advances in Neural Information Processing Systems*, 29, 2016.

[52] Se-Young Yun and Alexandre Proutière. Optimal sampling and clustering in the stochastic block model. *Advances in Neural Information Processing Systems*, 32, 2019.

[53] Jing Lei and Alessandro Rinaldo. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, pages 215–237, 2015.

[54] László Erdős, Antti Knowles, Horng-Tzer Yau, and Jun Yin. Spectral statistics of erdős—rényi graphs i: Local semicircle law. *The Annals of Probability*, pages 2279–2375, 2013.

[55] Konstantin Avrachenkov, Laura Cottatellucci, and Arun Kadavankandy. Spectral properties of random matrices for stochastic block model. In *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 537–544. IEEE, 2015.

[56] Florent Benaych-Georges, Charles Bordenave, and Antti Knowles. Largest eigenvalues of sparse inhomogeneous erdős–rényi graphs. *The Annals of Probability*, 47(3):1653–1676, 2019.

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

[58] Gaspard Michel, Giannis Nikolentzos, Johannes F Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *International Conference on Machine Learning*, pages 24737–24755. PMLR, 2023.

[59] Lorenzo Giusti, Teodora Reu, Francesco Ceccarelli, Cristian Bodnar, and Pietro Liò. Cin++: Enhancing topological message passing. *arXiv preprint arXiv:2306.03561*, 2023.

[60] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

[61] Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pages 12252–12267. PMLR, 2023.

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

[63] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

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

[65] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

[66] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[67] Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *arXiv preprint arXiv:2309.00367*, 2023.

[68] Dario Fasino, Arianna Tonetto, and Francesco Tudisco. Hitting times for non-backtracking random walks. *arXiv preprint arXiv:2105.14438*, 2021.

[69] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

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

[71] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[72] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[73] 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.

Table 4: Comparison of LGNN and NBA-GCN on `Peptides-func`.

Model	training AP ↑	test AP ↑	test time per epoch ↓	GPU usage (%) ↓
LGNN	0.4202	0.3778	87.761	97.10
NBA-GCN+LapPE	0.9724	0.7207	0.541	51.18

A Comparison with related works

This section delves into the detailed exploration of key related works, highlighting essential distinctions from our NBA-GNN. In addition, we present simple experiments to prove the superiority of our model.

A.1 Line graph neural networks

Line Graph Neural Networks (LGNN) [27] were the pioneers in applying the non-backtracking operator to GNNs. However, it is important to note that LGNN only utilizes non-backtracking within a layer, whereas NBA-GNN applies non-backtracking across all layers. In other words, our main contribution over LGNN is in consideration of the non-backtracking operator specifically for the message redundancy problem. Notably, the computational complexity of LGNN is acknowledged to be close to $|V| \log(|V|)$ by its authors.

In our experiments on `Peptides-func`, we compare the performance and complexity of LGNN and NBA-GCN+LapPE, as detailed in Table 4. Given that LGNN was initially proposed using only node degrees for node features, we incorporated an additional node feature encoder, similar to our NBA-GNN setup. Specifically, we used a batch size of 64, hidden dimension of 24, and 4 layers for LGNN. Refer to Appendix E.2 for the hyperparameters of NBA-GCN. The drawbacks of LGNN in terms of time and space complexity become evident when compared to NBA-GNN.

A.2 Redundant-free graph neural networks

Redundant-Free Graph Neural Networks (RFGNN) [28] shares the common motivation with NBA-GNN, aiming to reduce redundant messages in the computation graph. RFGNN achieves this by constructing a tree coined Truncated ePath Tree (TPT) for every node. TPT ensures that there are no repeated nodes along the simple path from the root to the leaf, except for the root node which can appear twice in the path. This approach differs significantly from NBA-GNN, which eliminates redundancy by identifying non-backtracking edge adjacency.

In terms of complexity, RFGNN has a space and time complexity of $\mathcal{O}\left(\frac{|V|!}{|V-t-1|!}\right)$, where $|V|$ is the number of nodes and t is the number of GNN layers. In contrast, NBA-GNN achieves superior complexity with space complexity $\mathcal{O}(2|\mathcal{E}|)$ and time complexity $\mathcal{O}(d_{avg}|\mathcal{E}|)$, which remains **irrelevant to the number of layers**. The preprocessing process time complexity, involved in finding non-backtracking edge adjacency, is $\mathcal{O}(|\mathcal{E}|^2)$. This scalability allows NBA-GNN to handle larger graph dataset compared to RFGNN. Furthermore, in theoretical aspects, our work distinguishes itself by providing the sensitivity upper bound of non-backtracking updates, rather than comparing the relative inference of paths.

B Proofs for Section 3.1

In this section, we present the findings discussed in Section 3.1. Similar to [15], we concentrate on the relationship between over-squashing and access time of non-backtracking random walks. Our study establishes that the access time using a begrudgingly backtracking random walk (BBRW) is smaller than that of a simple random walk (SRW) between two nodes, in tree graphs. Also, it is noteworthy that the gap between these two access times increases as the length of the walk grows.

We have derived formulas to compare the access times between BBRW and SRW. First, we show that on the tree graph, the access time equals the sum of access times between neighboring nodes. We note that the access time between neighboring nodes, which is the cut-point, can be represented in terms of return time. The formulas for return time in [68] and Lemma 6 allow us to derive the formulas for access time between neighboring nodes. Finally, we derive and compare the access time of BBRW and SRW.

B.1 Preliminaries

Simple and begrudgingly random walks. A random walk on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of $\mathcal{V}$-valued random variable $\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2 \dots$ where x_{n+1} is chosen randomly from neighborhood of x_n. Different types of random walks have different probabilities for selecting neighboring nodes.

Simple random walk (SRW) choose a next node j uniformly from the neighbors of current node i:

$$P(\mathrm{x}_{n+1} = j | \mathrm{x}_n = i) = \begin{cases} \frac{1}{d_i} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}.$$

On the other hand, begrudgingly backtracking random walk (BBRW) tries to avoid the previous node when there is another option:

$$P(\mathrm{x}_{n+2} = k | \mathrm{x}_{n+1} = j, \mathrm{x}_n = i) = \begin{cases} \frac{1}{d_j - 1} & \text{when } (j,k) \in \mathcal{E},\ k \neq i, \text{ and } |\mathcal{N}(j) \setminus \{i\}| \geq 1 \\ 1 & \text{when } (j,k) \in \mathcal{E} \text{ and } |\mathcal{N}(j) \setminus \{i\}| = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Access time. Consider a SRW starting at a node $i \in \mathcal{V}$. Let T_i denote the time when a SRW first arrived at node i, $T_i := \min\{n \geq 0 | \mathrm{x}_n = i\}$, and $\overline{T}_i$ be the time when a random walk first arrived at node i after the first step, $\overline{T}_i := \min\{n > 0 | \mathrm{x}_n = i\}$. With slight abuse of notation, we define access time $\mathtt{t}(i,j)$ from i to j, access time $\mathtt{t}(i \to j, k)$ from i to k where $\mathrm{x}_1 = j$, and return time $\mathtt{t}(i)$ in a graph $\mathcal{G}$ as follows:

$$\begin{aligned} \mathtt{t}(i,j) &:= \mathbb{E}[T_j | \mathrm{x}_0 = i] \\ \mathtt{t}(i \to j, k) &:= \mathbb{E}[T_k | \mathrm{x}_1 = j, \mathrm{x}_0 = i] \\ \mathtt{t}(i; \mathcal{G}) &:= \mathbb{E}[\overline{T}_i | \mathrm{x}_0 = i] \end{aligned}$$

We similarly denote access time and return time of BBRW as $(\tilde{\mathtt{t}}(i,j), \tilde{\mathtt{t}}(i \to j, k))$ and $\tilde{\mathtt{t}}(i; \mathcal{G})$, respectively. Note that the first step of BBRW shows the same behavior as the SRW since there is no previous node on the first step.

Finally, for a tree graph $\mathcal{G}$ and pair of nodes i, j in Proposition 1, we denote the unique paths between i and j as $(v_0, \dots, v_N)$ with $i = v_0$, $j = v_N$. We also let N denote the distance between the nodes i and j.

B.2 Access time of simple random walks

In this Section, we derive formulas for the access time of simple random walks on tree graphs, Proposition 2. We show it by the following process:

1. We decompose the access time for two nodes i and j into a summation of access time of neighboring nodes in the path, i.e., $\mathtt{t}(v_n, v_{n+1})$ for $l \in 0, \cdots, N-1$. (Lemma 2)
2. We evaluate the access time of neighboring nodes in the path, e.g., $\mathtt{t}(v_n, v_{n+1})$, in using the number of edges in a graph. (Lemma 3)

B.2.1 Decomposition of access time

First, we show that the access time equals the sum of access times between neighboring nodes. When a random walker travels from v_0 to v_N, it must pass through all nodes v_n on the paths. We can consider the entire time taken as the summation of the time intervals between when the walker arrived at v_n and when it arrived at v_{n+1}. It is an intuitive way of understanding Lemma 2.

Lemma 2. *Given a tree $\mathcal{G}$ and path $(v_0, \ldots, v_N)$,*

$$\mathtt{t}(v_0, v_N) = \sum_{n=0}^{N-1} \mathtt{t}(v_n, v_{n+1}).$$

Proof. The proof outline is as follows. Given a unique path $(v_0, \ldots, v_N)$ between v_0 and v_N, any walk between (v_0, v_N) can be decomposed into a series of $N-1$ walks between $(v_0, v_1), (v_1, v_2), \ldots (v_{N-1}, v_N)$ such that the walk between (v_n, v_{n+1}) does not contain a node v_{n+1} except at the endpoint. The expected length of each walk is $t(v_n, v_{n+1})$.

To be specific, consider the following decomposition.

$$\mathtt{t}(v_0, v_N) = \mathbb{E}[T_{v_N} | \mathrm{x}_0 = v_0] = \sum_{n=0}^{N-1} \mathbb{E}[T_{v_{n+1}} - T_{v_n} | \mathrm{x}_0 = v_0] \tag{6}$$

From the Markov property of SRW:

$$\mathbb{E}[T_{v_{n+1}} - T_{v_n} | \mathrm{x}_0 = v_0] = \mathtt{t}(v_n, v_{n+1})$$

Thus,

$$\mathtt{t}(v_0, v_N) = \sum_{n=0}^{N-1} \mathbb{E}[T_{v_{n+1}} - T_{v_n} | \mathrm{x}_0 = v_0] = \sum_{n=0}^{N-1} \mathtt{t}(v_n, v_{n+1})$$

□

B.2.2 Access time between neighbors

Now, we derive the formula for the access time between neighboring nodes.

Lemma 3. *Given a tree graph $\mathcal{G}$ and adjacent nodes i, j, the associated access time $\mathtt{t}(i, j)$ is defined as follows:*

$$\mathtt{t}(i, j) = 1 + 2|\mathcal{E}(\mathcal{G}_i)|,$$

where $\mathcal{E}(\mathcal{G})$ is edge set of graph $\mathcal{G}$ and $\mathcal{G}_i$ is the subtree produced by deleting edge (i, j) and choosing connected component of i.

Proof. Given a random walk from i to j that only contains j once, every time the walk lands at the node i, its future events can be categorized into two scenarios:

1. The walk transitions from node i to j based on transitioning with respect to the edge $(i, j) \in \mathcal{E}$ with probability $\frac{1}{d_i}$. The walk terminates.
2. The walk fails to reach j and continues the walk in the subtree $\mathcal{G}_i$ until arriving at the node i again . Note that the walk cannot arrive at node j without arriving at the node i in prior. In other words, the walk continues for the return time of i with respect to the graph $\mathcal{G}_i$.

The two scenarios implies that, every time the walk arrives at node i, the walk terminates with probability $\frac{1}{d_i}$. Then the number of trials follow the geometric distribution and consequently, the average number of trial is d_i. In other words, the walk falls into the scenario of type 2, for $d_i - 1$ times in average. We have to traverse at least one edge (i, j). The expected total penalty is the product of average failure penalty and the average number of failure. Thus,

$$\mathtt{t}(i, j) = 1 + (d_i - 1)\mathtt{t}(i; \mathcal{G}_i), \tag{7}$$

where $\mathtt{t}(i; \mathcal{G}_i)$ is the return time of i with respect to the subgraph $\mathcal{G}_i$. Since the return time is the ratio of the sum of the degree to the degree of the node [68], the following equation holds:

$$\mathtt{t}(i; \mathcal{G}_i) = \frac{2|\mathcal{E}(\mathcal{G}_i)|}{d_i - 1},$$

which directly implies our conclusion of $\mathtt{t}(i, j) = 1 + 2|\mathcal{E}(\mathcal{G}_i)|$. □

B.2.3 Main result

Using Lemma 2 and Lemma 3, we arrive at our main result for SRW on trees.

Proposition 2. *Given a tree $\mathcal{G}$ and pair of nodes $i, j \in \mathcal{V}$, the following equations holds for the access time of SRW between u and v.*

$$\mathtt{t}(i,j) = \sum_{n=0}^{N-1} (1 + 2|\mathcal{E}(\mathcal{G}_l)|), \tag{8}$$

where $\mathcal{E}(\mathcal{G})$ is edge set of graph $\mathcal{G}$ and $\mathcal{G}_l$ is the subtree produced by deleting edge (v_n, v_{n+1}) and choosing connected component of v_n.

Proof. The proof is a straightforward combination of Lemma 2 and Lemma 3.

$$\mathtt{t}(i,j) = \sum_{n=0}^{N-1} \mathtt{t}(v_n, v_{n+1}) = \sum_{n=0}^{N-1} (1 + 2|\mathcal{E}(\mathcal{G}_l)|).$$

□

B.3 Access time of begrudgingly backtracking random walks

In this section, we derive formulas for the access time of begrudgingly backtracking random walks on tree graphs. As mentioned in the preliminaries, we use $\tilde{\mathtt{t}}$ to denote the access time of begrudgingly backtracking. At a high level, this section consists of the following proofs.

1. We decompose the access time for two nodes v_0 and v_N into summations of access time of neighboring nodes in the path. (Lemma 4)
2. The decomposed term in 1. can be formulated using (i) the return time and (ii) the access time between neighboring nodes. (Lemma 5)
3. The return time of a node can be formulated in terms of the number of edges and the degree of a node. (Lemma 6)
4. The access time between neighboring nodes can be formulated using the number of edges and the degree of the node. (Lemma 7)

B.3.1 Decomposition of access time

We start by decomposing the access time $\tilde{\mathtt{t}}(i,j)$ similar to the one in Lemma 2. However, a key difference exists in the BBRW random walk. When the walk first arrives at node v_n, it previously passed the node v_{n-1}. Therefore, we cannot return to v_{n-1} upon the first failure to reach v_{n+1}.

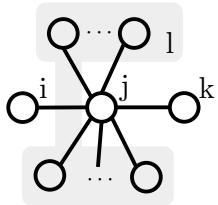
Lemma 4. *Consider a tree $\mathcal{G}$ and path $(v_0, \ldots, v_N)$. Then the access time of BBRW between node v_0 and v_N can be derived as follows:*

$$\tilde{\mathtt{t}}(v_0, v_N) = \tilde{\mathtt{t}}(v_0, v_0 \to v_1) + \sum_{l=1}^{N-1} \left(\tilde{\mathtt{t}}(v_{n-1} \to v_n, v_{n+1}) - 1 \right)$$

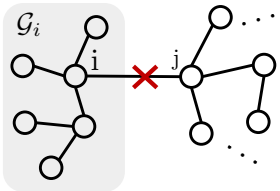
Proof. The proof is similar to Lemma 2 such that we decompose the walk into a series of $N-1$ walks between $(v_0, v_1), (v_1, v_2), \ldots, (v_{N-1}, v_N)$ such that the walk between v_n, v_{n+1} does note contain a node v_{n+1} except at the endpoint. The expected length of each walk is $\tilde{\mathtt{t}}(v_{n-1} \to v_n, v_{n+1}) - 1$. Note that $\tilde{\mathtt{t}}(v_{n-1} \to v_n, v_{n+1})$ counts the length of walk from v_{n-1} to v_{n+1}, hence should be substracted by represent the length of walk from v_n to v_{n+1}.

To be specific, the proof of Lemma 2, we mentioned that Equation (6) holds for general random walks.

$$\tilde{\mathtt{t}}(v_0, v_N) = \sum_{n=0}^{N-1} \mathbb{E}[T_{v_{n+1}} - T_{v_n} | \mathrm{x}_0 = v_0]$$



(a) Figure of nodes i, j, k, l in B.3.2



(b) Subtree $\mathcal{G}_i$ produced by deleting edge (i, j) and choosing connected components to node i

Figure 6: References for section B.3.2

When the BBRW random walk reaches v_n at T_{v_n}, it previously passed the node v_{n-1}. (i.e., $\mathrm{x}_{T_{v_n}-1} = v_{n-1}$). Therefore, we can think the random walk after the time $t = T_{v_n} - 1$ as random walk starting at v_{n-1} with the second node $\mathrm{x}_1 = v_n$. Thus,

$$\mathbb{E}[T_{v_{n+1}} - T_{v_n} | \mathrm{x}_0 = v_0] = \tilde{\mathtt{t}}(v_{n-1} \to v_n, v_{n+1}) - 1 \quad \text{for} \quad l \geq 1.$$

□

B.3.2 Expressing transition-conditioned access time using subgraph-return time.

We further prove the following formula for computing the transition-conditioned access time $\tilde{\mathtt{t}}(v_{n-1} \to v_n, v_{n+1})$.

Lemma 5. *Given a tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and three nodes i, j, and k such that $(i, j), (j, k) \in \mathcal{E}$,*

$$\tilde{\mathtt{t}}(i \to j, k) - 1 = \tilde{\mathtt{t}}(j, k) - \frac{d_i - 1}{d_j} \tilde{\mathtt{t}}(i; \mathcal{G}_i) - \frac{2}{d_j}, \tag{9}$$

where $\tilde{\mathtt{t}}(i; \mathcal{H})$ is return time of node i in a subgraph $\mathcal{H}$, and $\mathcal{G}_i$ is the subtree produced by deleting edge (i, j) and choosing connected components of i.

Proof. We start with establishing basic equalities for access time of BBRW.

The first equality describes how the access time of a random walk from $i \to j$ to k can be expressed by its subset $j \to l, k$ where $l \neq i$ due to the non-backtracking property.

$$\tilde{\mathtt{t}}(i \to j, k) - 1 = \frac{1}{d_j - 1} + \sum_{l \in \mathcal{N}(j) \setminus \{i,k\}} \frac{1}{d_j - 1} \tilde{\mathtt{t}}(j \to l, k) \tag{10}$$

The next equality describes how the access time from j to k can be decomposed by considering subsequent transitions to some l in the neighborhood of j.

$$\tilde{\mathtt{t}}(j, k) = \frac{1}{d_j} \sum_{l \in \mathcal{N}(j)} \tilde{\mathtt{t}}(j \to l, k), \tag{11}$$

When plugging in Equation (11) into Equation (10), one can see that we need to consider $l = i, k$ for $\tilde{\mathtt{t}}(j \to l, k)$. In case of k, it is trivially 1. For the case of i, i.e., $\tilde{\mathtt{t}}(j \to i, k)$, it can be defined as follow:

$$\tilde{\mathtt{t}}(j \to i, k) = 1 + (d_i - 1)\tilde{\mathtt{t}}(i; \mathcal{G}_i) + \tilde{\mathtt{t}}(i \to j, k). \tag{12}$$

This is similar to Equation (7), where $\mathcal{G}_i$ describes how the walk from $j \to i$ to k can be divided into two scenarios: (i) continues the walk in $\mathcal{G}_i$ or (ii) i transitions into j with probability $1/(d_i - 1)$.

Starting from Equation (10), one can derive the following relationship:

$$\begin{aligned}
\tilde{\mathtt{t}}(i \to j, k) - 1 &= \frac{1}{d_j - 1} \cdot 1 + \sum_{l \in \mathcal{N}(j) \setminus \{i,k\}} \frac{1}{d_j - 1} \tilde{\mathtt{t}}(j \to l, k) \\
&\overset{(a)}{=} \frac{1}{d_j - 1} + \frac{1}{d_j - 1} \left[\sum_{l \in \mathcal{N}(j)} \tilde{\mathtt{t}}(j \to l, k) - \tilde{\mathtt{t}}(j \to i, k) - \tilde{\mathtt{t}}(j \to k, k) \right] \\
&\overset{(b)}{=} \frac{1}{d_j - 1} + \frac{1}{d_j - 1} \left[d_j \tilde{\mathtt{t}}(j, k) - \tilde{\mathtt{t}}(j \to i, k) - 1 \right] \\
&\overset{(c)}{=} \frac{1}{d_j - 1} \left[d_j \tilde{\mathtt{t}}(j, k) - (d_i - 1) \tilde{\mathtt{t}}(i; \mathcal{G}_i) - \tilde{\mathtt{t}}(i \to j, k) - 1 \right],
\end{aligned}$$

where (a) is from Equation (10), (b) is from Equation (11) and $\tilde{\mathtt{t}}(j \to k, k) = 1$, (c) is from Equation (12).

Now, by multiplying $d_j - 1$ on both sides, we get

$$\begin{aligned}
(d_j - 1)\left(\tilde{\mathtt{t}}(i \to j, k) - 1 \right) &= d_j \tilde{\mathtt{t}}(j, k) - (d_i - 1) \tilde{\mathtt{t}}(i; \mathcal{G}_i) - \tilde{\mathtt{t}}(i \to j, k) - 1 \\
&= d_j \tilde{\mathtt{t}}(j, k) - (d_i - 1) \tilde{\mathtt{t}}(i; \mathcal{G}_i) - \left(\tilde{\mathtt{t}}(i \to j, k) - 1 \right) - 2.
\end{aligned}$$

Thus, we can conclude for $\tilde{\mathtt{t}}(i \to j, k) - 1$:

$$\tilde{\mathtt{t}}(i \to j, k) - 1 = \tilde{\mathtt{t}}(j, k) - \frac{d_i - 1}{d_j} \tilde{\mathtt{t}}(i; \mathcal{G}_i) - \frac{2}{d_j}.$$

□

B.3.3 Return time with respect to a subgraph

Now we need the formula for the return time $\tilde{\mathtt{t}}(i; \mathcal{G})$ with respect to a tree-graph $\mathcal{G}$. For the return time, we prove that the return time of BBRW is less than or equal to that of SRW.

Lemma 6. *Given a tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a node i, the return time of i is,*

$$\tilde{\mathtt{t}}(i; \mathcal{G}) = \frac{2|\mathcal{E}|}{d_i} \tag{13}$$

Proof. We can think the tree as rooted tree where root is i. We use mathematical induction with tree height of i.

Consider the base case where the height of tree is 1. Then, whatever we choose as the next node x_1 from $\mathrm{x}_0 = i$, we return to i at second transition (i.e., $\mathrm{x}_2 = i$) since all the neighbors of i is leaf node. Since $d_i = |\mathcal{E}|$ for tree with height 1, $\tilde{\mathtt{t}}(i; \mathcal{G}) = 2 = \frac{2|\mathcal{E}|}{d_i}$.

Now, assume that the lemma holds for the tree with a height less than $k \geq 1$. It suffices to show that the lemma also holds for a tree with height $k + 1$ and its root i.

From the same perspective in the proofs of the Lemma 3, we can view the random walk returning to i as follows:

1. We choose a node $\mathrm{x}_1 = l$ from $\mathcal{N}(i)$ uniformly. Then, from node l, we return to l to reach i. (i.e., we have to pay penalty amounts to return time of l)
2. In node l, we try to reach i by selecting edge (l, i) with probability $\frac{1}{d_l - 1}$. Thus, the average number of failure of failure is $d_l - 2$.
3. If we fail to reach i from l, we should return to l for a next chance to reach i. i.e., a average failure penalty amounts to a return time of l in subtree with root l.

Hence,

$$\tilde{\mathtt{t}}(i;\mathcal{G}) = 1 + \sum_{l\in\mathcal{N}(i)} \frac{1}{d_i}\left\{1 + \tilde{\mathtt{t}}(l;\mathcal{G}_l)\cdot\left((d_l-2)+1\right)\right\} = 2 + \sum_{l\in\mathcal{N}(i)} \frac{1}{d_i}\cdot\tilde{\mathtt{t}}(l;\mathcal{G}_l)\cdot(d_l-1),$$

where $\tilde{\mathtt{t}}(l;\mathcal{G})$ is return time of l with respect to $\mathcal{G}$ and $\mathcal{G}_l$ is the subtree with root l.

Since the subtree with root l has height less than k, $\tilde{\mathtt{t}}(l;\mathcal{G}_l) = \frac{2|\mathcal{E}(G_l)|}{d_l-1}$. Therefore,

$$\begin{aligned}
\tilde{\mathtt{t}}(i;\mathcal{G}) &= 2 + \frac{1}{d_i}\sum_{l\in\mathcal{N}(i)} \tilde{\mathtt{t}}(l;\mathcal{G}_l)\cdot(d_l-1) \\
&= 2 + \frac{1}{d_i}\sum_{l\in\mathcal{N}(i)} \frac{2|\mathcal{E}(\mathcal{G}_l)|}{d_l-1}\cdot(d_l-1) \\
&= 2 + \frac{1}{d_i}\sum_{l\in\mathcal{N}(i)} 2|\mathcal{E}(\mathcal{G}_l)| \\
&= \frac{2\left(d_i + \sum_{l\in\mathcal{N}(i)}|\mathcal{E}(\mathcal{G}_l)|\right)}{d_i} \\
&= \frac{2|\mathcal{E}|}{d_i}
\end{aligned}$$

□

B.3.4 Access time between neighbors

For access time between neighbors, we get the similar result to Lemma 3.

Lemma 7. *Given a tree $\mathcal{G}$ and adjacent nodes i, j,*

$$\tilde{\mathtt{t}}(i,j) = 1 + 2|\mathcal{E}(\mathcal{G}_i)|\cdot\frac{d_i-1}{d_i}, \tag{14}$$

where $\mathcal{E}(\mathcal{G})$ is edge set of graph $\mathcal{G}$, and $\mathcal{G}_i$ is the subtree produced by deleting edge (i,j) and choosing connected component of i.

Proof. From the same perspective in the proofs of the Lemma 3, we analyse success probability for each trial.

For the first trial, success probability is $\frac{1}{d_i}$ since there is no previous node. After the first trial, success probability is fixed to $\frac{1}{d_i-1}$. On the each trial, the average failure penalty amounts to $\tilde{\mathtt{t}}'(i)$, which is the return time of i on subtree $\mathcal{G}_i$.

Thus, average number of trial until first success is,

$$\text{(Average number of trial)} = \frac{1}{d_i}\cdot 1 + \frac{d_i-1}{d_i}\cdot\left(1+(d_i-1)\right)$$

The average number of failure is as follows:

$$\begin{aligned}
\text{(Average number of failure)} &= \text{(Average number of trial)} - 1 \\
&= \frac{1}{d_i}\cdot 1 + \frac{d_i-1}{d_i}\cdot(1+d_i-1) - 1 \\
&= \frac{(d_i-1)^2}{d_i}
\end{aligned}$$

Thus, expected total penalty is as follows:

$$\text{(Expected total penalty)} = \tilde{\mathtt{t}}(i;\mathcal{G}_i)\cdot\frac{(d_i-1)^2}{d_i}$$

Since $\tilde{\mathtt{t}}(i;\mathcal{G}_i) = \frac{2|\mathcal{E}(\mathcal{G}_i)|}{d_i-1}$ by Lemma 6, $\tilde{\mathtt{t}}(u,v) = 1 + 2|\mathcal{E}(\mathcal{G}_i)|\frac{d_i-1}{d_i}$. □

B.3.5 Main result

Combining the results provides the following theorem.

Proposition 3. *Given a tree $\mathcal{G}$ and pair of nodes i, j, the following equations holds for the access time of BBRW between i and j.*

$$\tilde{\mathtt{t}}(i,j) = \sum_{n=0}^{N-1}\left(1 + 2|\mathcal{E}(\mathcal{G}_l)| \cdot \frac{d_{v_n}-1}{d_{v_n}}\right) - \sum_{l=1}^{N-1}\frac{2|\mathcal{E}(\mathcal{G}_{l-1})|}{d_{v_n}} - \sum_{l=1}^{N-1}\frac{2}{d_{v_n}},$$

where $\mathcal{E}(\mathcal{G})$ is the edge set of $\mathcal{G}$ and $\mathcal{G}_l$ is the subtree produced by deleting edge (v_n, v_{n+1}) and choosing connected component of v_n.

Proof. By Lemma 4 and Equation (9),

$$\begin{aligned}
\tilde{\mathtt{t}}(v_0, v_N) &= \tilde{\mathtt{t}}(v_0, v_1) + \sum_{l=1}^{N-1}\left\{\tilde{\mathtt{t}}(v_{n-1} \to v_n, v_{n+1}) - 1\right\} \\
&= \tilde{\mathtt{t}}(v_0, v_1) + \sum_{l=1}^{N-1}\left\{\tilde{\mathtt{t}}(v_n, v_{n+1}) - \frac{d_{v_{n-1}}-1}{d_{v_n}}\tilde{\mathtt{t}}'(v_{n-1}) - \frac{2}{d_{v_n}}\right\} \\
&= \sum_{n=0}^{N-1}\tilde{\mathtt{t}}(v_n, v_{n+1}) - \sum_{l=1}^{N-1}\frac{d_{v_{n-1}}-1}{d_{v_n}}\tilde{\mathtt{t}}'(v_{n-1}) - \sum_{l=1}^{N-1}\frac{2}{d_{v_n}} \\
&= \sum_{n=0}^{N-1}\left(1 + 2|\mathcal{E}(\mathcal{G}_l)| \cdot \frac{d_{v_n}-1}{d_{v_n}}\right) - \sum_{l=1}^{N-1}\frac{2|\mathcal{E}(\mathcal{G}_{l-1})|}{d_{v_n}} - \sum_{l=1}^{N-1}\frac{2}{d_{v_n}}
\end{aligned}$$

□

B.4 Proof of Proposition 1

Finally, we compare the access time of two random walks in a tree. Recall Proposition 1.

Proposition 1. *Given a tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a pair of nodes $i, j \in \mathcal{V}$, the access time of begrudgingly backtracking random walk is equal or smaller than that of a simple random walk, where the equality holds if and only if the random walk length is 1.*

Proof. Let $\mathcal{E}(\mathcal{G})$ be the edge set of $\mathcal{G}$ and $\mathcal{G}_l$ be the subtree produced by deleting edge (v_n, v_{n+1}) and choosing connected component of v_n. Then,

$$\begin{aligned}
\tilde{\mathtt{t}}(v_0, v_N) - \mathtt{t}(v_0, v_N) &= \sum_{n=0}^{N-1}\left(1 + 2|\mathcal{E}(\mathcal{G}_l)| \cdot \frac{d_{v_n}-1}{d_{v_n}}\right) - \sum_{l=1}^{N-1}\frac{2|\mathcal{E}(\mathcal{G}_{l-1})|}{d_{v_n}} \\
&\quad - \sum_{l=1}^{N-1}\frac{2}{d_{v_n}} - \sum_{n=0}^{N-1}(1 + 2|\mathcal{E}(\mathcal{G}_l)|) \\
&= -\sum_{n=0}^{N-1}\frac{2|\mathcal{E}(\mathcal{G}_l)|}{d_{v_n}} - \sum_{l=1}^{N-1}\frac{2|\mathcal{E}(\mathcal{G}_{l-1})|}{d_{v_n}} - \sum_{l=1}^{N-1}\frac{2}{d_{v_n}} \leq 0
\end{aligned}$$

Therefore the access time of two nodes are always less or equal in begrudgingly backtracking random walk than simple random walks, where equality holds for random walks with length 1.

□

C Proofs for Section 4.1

C.1 Preliminaries

Let $\mathcal{G}$ be a graph with a set of n vertices $\mathcal{V}$, a set of m edges $\mathcal{E} \in \mathcal{V}^2$. We use x_i to denote the node-wise feature for $i \in \mathcal{V}$, and d_i for the degree of node $i \in \mathcal{V}$. The adjacency matrix $A \in \mathbb{R}^{n \times n}$ encodes the connectivity of graph $\mathcal{G}$. For node $i \in \mathcal{V}$, we define the set of incident outgoing edges of i as $N_e^+(i)$, the set of incident incoming edges of i as $N_e^-(i)$, and let $N_e(i) = N_e^+(i) \cup N_e^-(i)$ be the set of all incident edges of node i. Also, recall the non-backtracking matrix $B \in \{0,1\}^{2|\mathcal{E}| \times 2|\mathcal{E}|}$ and the incidence matrix $C \in \mathbb{R}^{2|\mathcal{E}| \times |\mathcal{V}|}$:

$$B_{(\ell \to k),(j \to i)} = \begin{cases} 1 & \text{if } k = j, \ell \neq i \\ 0 & \text{otherwise} \end{cases}, \qquad C_{(k \to j),i} = \begin{cases} 1 & \text{if } j = i \text{ or } k = i \\ 0 & \text{otherwise} \end{cases}.$$

We also define D_{out} and D_{in} as the out-degree and in-degree matrices of NBA-GNN, respectively, counting the number of outgoing and incoming edges for each edge. These are diagonal matrices with $(D_{out})_{(j \to i),(j \to i)} = \sum_{\ell \to k} B_{(j \to i),(\ell \to k)}$ and $(D_{in})_{(j \to i),(j \to i)} = \sum_{\ell \to k} B_{(\ell \to k),(j \to i)}$, for each index $j \to i$. Next, we introduce $\widehat{B}$ as the normalized non-backtracking matrix augmented with self-loops: $\widehat{B} = (D_{out} + I)^{-\frac{1}{2}}(B + I)(D_{in} + I)^{-\frac{1}{2}}$. Then one obtains the following sensitivity bound of NBA-GNN. We also let $\tilde{C}$ denote a matrix where $\tilde{C}_{(k \to j),i} = C_{(k \to j),i} + C_{(j \to k),i}$.

Consequently, one can express our NBA-GNN updates:

$$h_{j \to i}^{(t+1)} = \phi^{(t)} \left(h_{j \to i}^{(t)}, \sum_{(k,\ell) \in \mathcal{E}} \widehat{B}_{(\ell \to k),(j \to i)} \psi^{(t)} \left(h_{\ell \to k}^{(t)}, h_{j \to i}^{(t)} \right) \right), \tag{15}$$

where $\phi^{(t)}$ and $\psi^{(t)}$ corresponds to the update and the aggregation as described in Equation (3). Next, the node-wise aggregation step can be described as follows:

$$h_i = \sigma \left(\sum_{(j,k) \in \mathcal{E}} C_{(k \to j),i} \rho \left(h_{k \to j}^{(T)} \right), \sum_{(j,k) \in \mathcal{E}} C_{(j \to k),i} \rho \left(h_{j \to k}^{(T)} \right) \right). \tag{16}$$

C.2 Proof of Lemma 1

The original sensitivity bound for a hidden representation for node e and initial feature of node s is defined as following.

Proposition 4. ***Sensitivity bound.*** *[13] Assume an MPNN defined in Equation (15). Let two nodes $i, j \in \mathcal{V}$ with distance T. If $|\nabla \phi^{(t)}| \leq \alpha$ and $|\nabla \psi^{(t)}| \leq \beta$ for $0 \leq t < T$, then*

$$\left\| \frac{\partial h_j^{(T)}}{\partial x_i} \right\| \leq (\alpha\beta)^T (\widehat{A}^T)_{j,i}. \tag{17}$$

Now, we show that the sensitivity bound for non-backtracking GNNs can be defined as following. Following [13], we assume the node features and hidden representations are scalar for better understanding.

Lemma 1. *Consider two nodes $i, j \in \mathcal{V}$ with distance T given a $(T-1)$-layer NBA-GNN as described in Equation (3) and Equation (4). Suppose $|\nabla \phi^{(t)}|, |\nabla \sigma| \leq \alpha$, $|\nabla \psi^{(t)}|, |\nabla \rho| \leq \beta$, and $|\nabla \phi| \leq \gamma$ for $0 \leq t < T$. Then the following holds:*

$$\left\| \frac{\partial h_j}{\partial x_i} \right\| \leq (\alpha\beta)^T \gamma (\tilde{C}^\top \widehat{B}^{(T-1)} \tilde{C})_{j,i}.$$

Proof. Just a straight calculation using the chain rule is enough to derive the above upper bound.

$$\left\|\frac{\partial h_j}{\partial x_i}\right\| = \left\|\partial_1\sigma\partial_2\rho\left(\sum_{k\to\ell\in N_e^-(j)}\frac{\partial h_{k\to\ell}^{(T-1)}}{\partial x_i}\right) + \partial_1\sigma\partial_3\rho\left(\sum_{k\to\ell\in N_e^+(j)}\frac{\partial h_{k\to\ell}^{(T-1)}}{\partial x_i}\right)\right\| \tag{18}$$

$$= \left\|\partial_1\sigma\partial_2\rho\left(\sum_{k\to\ell\in N_e^-(j)}\frac{\partial h_{k\to\ell}^{(T-1)}}{\partial x_i}\right)\right\| \tag{19}$$

$$\leq \alpha\beta\left(\sum_{k\to\ell\in N_e^-(j)}\left\|\frac{\partial h_{k\to\ell}^{(T-1)}}{\partial x_i}\right\|\right), \tag{20}$$

$$\tag{21}$$

where the bound for the derivatives were $|\nabla\sigma| \leq \alpha, |\nabla\rho| \leq \beta$.

Thus considering the message update from Equation (15),

$$\frac{\partial h_{k\to\ell}^{(T-1)}}{\partial x_i} = \partial_1\phi^{(T-2)}\frac{\partial h_{k\to\ell}^{(T-2)}}{\partial x_i} + \partial_2\phi^{(T-2)}\left(\sum_{k'\to\ell'\in N_e^-(k)}\widehat{B}_{k'\to\ell',k\to\ell}\frac{\partial h_{k\to\ell}^{(T-2)}}{\partial x_i}\right)$$

$$= \partial_2\phi^{(T-2)}\left(\sum_{k'\to\ell'\in N_e^-(k)}\widehat{B}_{k'\to\ell',k\to\ell}\frac{\partial h_{k\to\ell}^{(T-2)}}{\partial x_i}\right),$$

since the distance between node i and node j is at least T, therefore $\frac{\partial h_{k\to\ell}^{(T-2)}}{\partial x_i} = 0$.

Now, let the path from node i to node j as $s(i,j)$, where s_t denotes the t-th node in the walk s, i.e., $s_0 = i, s_T = j$. Using the bound of the derivative of functions, we can iterate like the following.

$$\left\|\frac{\partial h_{k\to\ell}^{(T-1)}}{\partial x_i}\right\| \tag{22}$$

$$\leq \alpha\beta\left(\sum_{k'\to\ell'\in\mathcal{N}_e^-(k)}\widehat{B}_{k'\to\ell',k\to l}\left\|\frac{\partial h_{k\to l}^{(T-2)}}{\partial x_i}\right\|\right) \tag{23}$$

$$\leq (\alpha\beta)^{T-1}\left(\sum_{(s_0,\ldots,s_T)\in s(i,j)}\widehat{B}_{s_0\to s_1,s_1\to s_2}\cdot\cdots\cdot\widehat{B}_{s_{T-2}\to s_{T-1},s_{T-1}\to s_T}\cdot\left\|\frac{\partial h_{s_0\to s_1}^{(0)}}{\partial x_i}\right\|\right) \tag{24}$$

$$= (\alpha\beta)^{T-1}\left(\sum_{(s_0,\ldots,s_T)\in s(i,j)}\prod_{t=1}^{T-1}\widehat{B}_{s_{t-1}\to s_t,s_t\to s_{t+1}}\left\|\frac{\partial h_{s_0\to s_1}^{(0)}}{\partial x_i}\right\|\right) \tag{25}$$

$$\leq (\alpha\beta)^{T-1}\gamma\left(\sum_{(s_0,\ldots,s_T)\in s(i,j)}\left(\prod_{t=1}^{T-1}\widehat{B}_{s_{t-1}\to s_t,s_t\to s_{t+1}}\right)\right) \tag{26}$$

Substitute the inequality Equation (26) into Equation (20) to get the final result. Then, we can get

$$\left\|\frac{\partial h_j}{\partial x_i}\right\| \leq (\alpha\beta)^T\gamma\left(\sum_{(s_0,\ldots,s_T)\in s(i,j)}\left(\prod_{t=1}^{T-1}\widehat{B}_{s_{t-1}\to s_t,s_t\to s_{t+1}}\right)\right) \tag{27}$$

$$= (\alpha\beta)^T\gamma(\tilde{C}^\top\widehat{B}^{T-1}\tilde{C})_{j,i}$$

□

since paths can be expressed using the power of adjacency-type matrix.

C.3 Proof of Proposition 1

We have defined the sensitivity bound for NBA-GNNs. Now, we show that the sensitivity bound of NBA-GNNs are bigger than the sensitivity bound of GNNs.

Theorem 1. *Let $\widehat{A}$ denote the degree-normalized matrix. Then, for any pair of nodes $i, j \in \mathcal{V}$ with distance T, the sensitivity bound of NBA-GNN is larger than that of conventional GNNs [13], i.e., $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i} \geq (\widehat{A}^T)_{j,i}$. For d-regular graphs, $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i}$ decays slower by $O(d^{-T})$, while $(\widehat{A}^T)_{j,i}$ decays with $O((d+1)^{-T})$.*

Proof. Identical to the notations above, we denote the list of nodes from node i to node j as path $s(i,j)$ where $s(i,j) = (s_0 = i, s_1, \cdots, s_{T-1}, s_T = j)$.

$$(\widehat{A}^T)_{j,i} = \sum_{(s_0,\ldots,s_T)\in s(i,j)} \widehat{A}_{s_0,s_1} \cdots \widehat{A}_{s_{T-1},s_T} \tag{28}$$

$$= \sum_{(s_0,\ldots,s_T)\in s(i,j)} (d_i+1)^{-\frac{1}{2}} \cdot (d_j+1)^{-\frac{1}{2}} \cdot \prod_{t=1}^{T-1} (d_{s_t}+1)^{-1} \tag{29}$$

$$(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i} = \sum_{(s_0,\ldots,s_T)\in s(i,j)} d_{s_0}^{-\frac{1}{2}} \cdot \widehat{B}_{s_0\to s_1, s_1\to s_2} \cdots \widehat{B}_{s_{T-2}\to s_{T-1}, s_{T-1}\to s_T} \cdot d_{s_T}^{-\frac{1}{2}} \tag{30}$$

$$= \sum_{(s_0,\ldots,s_T)\in s(i,j)} d_{s_0}^{-\frac{1}{2}} \cdot d_{s_1}^{-1} \cdots d_{s_{T-1}}^{-1} \cdot d_{s_T}^{-\frac{1}{2}} \tag{31}$$

$$= \sum_{(s_0,\ldots,s_T)\in s(i,j)} d_{s_0}^{-\frac{1}{2}} \cdot d_{s_T}^{-\frac{1}{2}} \cdot \prod_{t=1}^{T-1} d_{s_t}^{-1} \tag{32}$$

$$\tag{33}$$

Now, for a path $(s_0 = i, \ldots, s_T = j)$,

$$(d_i+1)^{-\frac{1}{2}}(d_j+1)^{-\frac{1}{2}} \prod_{t=1}^{T-1} (d_{s_t}+1)^{-1} \leq d_i^{-\frac{1}{2}} d_j^{-\frac{1}{2}} \prod_{t=1}^{T-1} d_{s_t}^{-1} \tag{34}$$

Each term in $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i}$ is larger than $(\widehat{A}^T)_{j,i}$, thus $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i} \geq (\widehat{A}^T)_{j,i}$ is trivial.

For d-regular graphs, $d_i = d, \forall i \in \mathcal{V}$. Therefore the sensitivity bound can can be written as following:

$$(\widehat{A}^T)_{j,i} = (d+1)^{-T}$$
$$(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i} = d^{-T}.$$

So $(\tilde{C}^\top \widehat{B}^{T-1}\tilde{C})_{j,i}$ decays slower by $O(d^{-T})$, while $(\widehat{A}^T)_{j,i}$ decays with $O((d+1)^{-T})$.

□

D Proofs for Section 4.2

To assess the expressive capabilities, we initially make an assumption about the graphs, considering that it is generated from the Stochastic Block Model (SBM), which is defined as follows:

Definition 1. ***Stochastic Block Model (SBM)*** *is generated using parameters (n, K, α, P), where n denotes the number of vertices, K is the number of communities, $\alpha = (\alpha_1, ..., \alpha_K)$ represents the probability of each vertex being assigned to communities $\mathcal{V}_1, ..., \mathcal{V}_K$, and P_{ij} denotes the probability of an edge $(v, w) \in \mathcal{E}$ between $v \in \mathcal{V}_i$ and $w \in \mathcal{V}_j$.*

Numerous studies have focused on the problem of achieving exact recovery of communities within the SBM. However, these investigations typically address scenarios in which the average degree is at least on the order of $\Omega(\log n)$ [69]. It is well-established that the information-theoretic limit in such cases can be reached through the utilization of the spectral method, as demonstrated by [51]. In contrast, when dealing with a graph characterized by the average degree much smaller, specifically $o(\log n)$, recovery using the graph spectrum becomes a more challenging endeavor. This difficulty arises due to the fact that the $n^{1-o(1)}$ largest eigenvalues and their corresponding eigenvectors are influenced by the high-degree vertices, as discussed in [56].

However, real-world benchmark datasets often fall within the category of very sparse graphs. For example, the citation networks dataset discussed in [18] has an average degree of less than three. In such scenarios, relying solely on an adjacency matrix may not be an efficient approach for uncovering the hidden graph structure. Fortunately, an alternative strategy is available by utilizing a non-backtracking matrix as follows.

D.1 Proof of Theorem 2

Let's rephrase the formal version of Theorem 2 as follows:

Theorem 2. *Consider a Stochastic Block Model (SBM) with parameters $\left(n, 2, \left(\frac{1}{2}, \frac{1}{2}\right), \left(\frac{a}{n}, \frac{b}{n}\right)\right)$, where $(a + b)$ satisfies the conditions of being at least $\omega(1)$ and $n^{o(1)}$. In such a scenario, the non-backtracking graph neural network can accurately map from graph $\mathcal{G}$ to node labels, with the probability of error decreasing to 0 as $n \to \infty$.*

In [16], the authors demonstrate that the non-backtracking matrix exhibits valuable properties when the average degree of vertices in the graph satisfies $\omega(1)$ and $n^{o(1)}$. When $K = 2$, we define a function $\sigma(v)$ for $v \in \mathcal{V}$, such that $\sigma(v) = 1$ if v is in the first class, and $\sigma(v) = -1$ in the second class. Then, they establish the following proposition:

Proposition 5. *[16] Suppose we have a SBM with parameters $\left(n, 2, \left(\frac{1}{2}, \frac{1}{2}\right), \left(\frac{a}{n}, \frac{b}{n}\right)\right)$. In this case, we have two eigenvalues $\mu_1 > \mu_2$ of $ndiag(\alpha)P$, and the eigenvector ϕ_2 corresponding to μ_2, where v-th component is set to $\sigma(v)$. Then, for any n larger than an absolute constant, the eigenvalues λ_1 and λ_2 of the non-backtracking matrix B satisfies $|\lambda_i - \mu_i| = o(1)$, and all other eigenvalues of B are confined in a circle with radius $(1 + o(1))\sqrt{\frac{a+b}{2}}$. Also, there exists an eigenvector ν_2 of the non-backtracking matrix B associated with λ_2 such that*

$$\langle \nu_2, \xi_2 \rangle \geq \sqrt{1 - \frac{8}{(a+b)(a-b)^2}} + o(1) := 1 - f(a, b)$$

where $\xi_2 = \frac{T\Theta\phi_2}{\|T\Theta\phi_2\|}$, $T \in \{0, 1\}^{2m \times n}, T_{ei} = 1\{e_2 = i\}$ and $\Theta \in \{0, 1\}^{n \times 2}, \Theta_{ij} = 1$ if the vertex i is in the j-th community, and 0 otherwise.

The proposition above highlights that the non-backtracking matrix possesses a spectral separation property, even in the case of very sparse graphs. Moreover, the existence of an eigenvector ν_2 such that $\langle \nu_2, \xi_2 \rangle = 1 - o(1)$ suggests that this eigenvector contains information about the community index of vertices. The foundation for these advantageous properties of the non-backtracking matrix B in sparse scenarios can be attributed to the observation that the matrix B^k shares similarities with the k-hop adjacency matrix, while A^k is predominantly influenced by high-degree vertices. Consequently, we can establish the following theorem:

Lemma 8. *Suppose we have a SBM with parameters defined in Proposition 5. Then, there exists a function of the eigenvectors of the non-backtracking matrix that can accurately recover the original community index of vertices.*

The proof for Lemma 8 can be found in Appendix D.3 In the following, we will demonstrate that the non-backtracking GNN can compute an approximation to the top K eigenvectors mentioned in Lemma 8. To achieve this, we first require the following lemma:

Lemma 9. *Assuming that the non-backtracking matrix B has a set of orthonormal eigenvectors ν_i with corresponding eigenvalues $\lambda_1 > ... > \lambda_K \geq \lambda_{K+1} \geq ... \geq \lambda_{2m}$, then there exists a sequence of convolutional layers in the non-backtracking GNNs capable of computing the eigenvectors of the non-backtracking matrix.*

For the proof of Lemma 9, please refer to Appendix D.4. With this lemma in mind, we can observe that a sequence of convolutional layers, followed by a multilayer perceptron, can approximate the function outlined in Lemma 8, leveraging the universal approximation theorem. This argument leads to Theorem 2.

D.2 Proof of Theorem 3

Let's rephrase the formal version of Theorem 3 as follows:

Theorem 3. *Consider two graphs, one generated from a SBM ($\mathcal{G}$) with parameters $\left(n, 2, \left(\frac{1}{2}, \frac{1}{2}\right), \left(\frac{a}{n}, \frac{b}{n}\right)\right)$ and the other from an Erdős–Rényi model ($\mathcal{H}$) with parameters $(n, \frac{c}{n})$, for some constants $a, b, c > 1$. When $(a-b)^2 > 2(a+b)$, the non-backtracking graph neural network is capable of distinguishing between $\mathcal{G}$ and $\mathcal{H}$ with probability tending to 1 as $n \to \infty$.*

To establish Theorem 3, we rely on the following Proposition 6 from [11]:

Proposition 6. *Suppose we have two graphs $\mathcal{G}$ and $\mathcal{H}$ as defined in the formal statement of Theorem 3. Then, the eigenvalues $\lambda_i(B)$ of the non-backtracking matrix satisfy the following:*

$$\lambda_1(B_{\mathcal{G}}) = \frac{a+b}{2} + o(1), \lambda_2(B_{\mathcal{G}}) = \frac{a-b}{2} + o(1), \quad \textit{and} \quad |\lambda_k(B_{\mathcal{G}})| \leq \sqrt{\frac{a+b}{2}} + o(1) \quad \textit{for } k > 2$$

$$\lambda_1(B_{\mathcal{H}}) = c + o(1) \quad \textit{and} \quad |\lambda_2(B_{\mathcal{H}})| \leq \sqrt{c} + o(1)$$

Proposition 6 informs us that by examining the distribution of eigenvalues, we can discern whether a graph originates from the Erdős–Rényi model or the SBM. Leveraging Lemma 9, we can obtain the top two normalized eigenvectors of the non-backtracking matrix using convolutional layers, denoted as ν_1 and ν_2. Applying the non-backtracking convolutional layer to these vectors yields resulting vectors with ℓ_2-norms corresponding to $\lambda_i(B)$. Consequently, we can distinguish between the two graphs, $\mathcal{G}$ and $\mathcal{H}$, by examining the output of the convolutional layer in the non-backtracking GNN.

D.3 Proof of Lemma 8

Let us revisit the matrix T, defined as $T_{ei} = 1\{e_2 = i\}$, and its pseudo-inverse denoted as $T^+ = D^{-1}T^\top$, where D is a diagonal matrix containing the degrees of vertices on the diagonal. Considering the definition of ξ_2 as provided in Proposition 5, we can deduce the label of vertex v. Specifically, if $(T^+\xi_2)_v > 0$, it implies that the vertex belongs to the first class; otherwise, it belongs to the other class.

Additionally, we are aware that $|\left(T^+\nu_2 - T^+\xi_2\right)_i|_2 = O(f(a,b))$ as indicated in Proposition 5, considering the property that the sum of each row of T^+ is equal to 1. Consequently, by examining the signs of elements in the vector $T^+\nu_2$, we can classify nodes without encountering any misclassified ones.

D.4 Proof of Lemma 9

Proof. Suppose we have $f(\leq 2m)$ arbitrary vectors $x_1, ..., x_f$ and a matrix $X = [x_1, ..., x_f] \in \mathbb{R}^{2m \times f}$, which has $x_1, ..., x_f$ as columns. Without loss of generality, we assume that $\|x_v\|_2 = 1$, and let $x_j = \sum_{i=1}^{2m} c_i^{(j)} \nu_i$ for $1 \leq j \leq f$. We will prove the lemma by showing that if we multiply X by B and repeatedly apply the Gram–Schmidt orthnormalization to the columns of the resulting matrix, the j-th column of the resulting matrix converges towards the direction of ν_j. Further, we will show that there exists a series of convolutional layers equivalent to this process.

First, we need to show $B^k x_1$ converges to the direction of ν_1. $B^k x_1$ can be expressed as:

$$B^k x_1 = \lambda_1^k \left(c_1^{(1)} \nu_1 + c_2^{(1)} \left(\frac{\lambda_2}{\lambda_1} \right)^k \nu_2 + ... + + c_{2m}^{(1)} \left(\frac{\lambda_{2m}}{\lambda_1} \right)^k \nu_{2m} \right).$$

Let's fix $\epsilon > 0$ and take K_0 such that for all $k \geq K_0$, the inequality $\left(\frac{\lambda_2}{\lambda_1}\right)^k < \frac{\epsilon |c_1^{(1)}|}{4m}$ holds. Furthermore, we define the following for brevity:

$$e_\ell^{(k)} := \begin{cases} \frac{A^k x_1}{\|A^k x_1\|} & \text{for } \ell = 1, \\ \frac{u_\ell^{(k)}}{\left\|u_\ell^{(k)}\right\|} & \text{where } u_\ell^{(k)} = GS(Be_\ell^{(k-1)}) \text{ for } \ell \geq 2. \end{cases} \tag{35}$$

Then, the distance between $e_1^{(k)}$ and sign $\left(c_1^{(1)}\right) \nu_1$ is

$$\begin{aligned}
\left\| e_1^{(k)} - \text{sign}\left(c_1^{(1)}\right) \nu_1 \right\|_2 &= \left\| \frac{c_1^{(1)} \nu_1 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i^{(1)} \nu_i}{\sqrt{(c_1^{(1)})^2 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2}} - \text{sign}\left(c_1^{(1)}\right) \nu_1 \right\|_2 \\
&\overset{(a)}{<} \left\| \frac{c_1^{(1)} \nu_1 - \text{sign}\left(c_1^{(1)}\right) \sqrt{(c_1^{(1)})^2 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2} \nu_1}{\sqrt{(c_1^{(1)})^2 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2}} \right\|_2 + \frac{\epsilon}{2} \\
&= \frac{-|c_1^{(1)}| + \sqrt{(c_1^{(1)})^2 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2}}{\sqrt{(c_1^{(1)})^2 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2}} + \frac{\epsilon}{2} \\
&\overset{(b)}{\leq} \frac{\sqrt{\sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2}}{\sqrt{(c_1^{(1)})^2 + \sum_{i=2}^{2m} \left(\frac{\lambda_i}{\lambda_1}\right)^{2k} (c_i^{(1)})^2}} + \frac{\epsilon}{2} \\
&\overset{(c)}{<} \epsilon,
\end{aligned}$$

where (a) and (c) are obtained from the assumption $\left(\frac{\lambda_2}{\lambda_1}\right)^k < \frac{\epsilon |c_1^{(1)}|}{4m}$, and for (b) we use the inequality $\sqrt{x^2 + y^2} \leq |x| + |y|$.

Moreover, we assume that for all $1 \leq \ell < L$ and any $\epsilon_\ell > 0$, there exists K_ℓ such that for all $k \geq K_\ell$, $\left\| e_\ell^{(k)} - \text{sign}\left(c_\ell^{(\ell)}\right) \nu_\ell \right\| < \epsilon_\ell$. To simplify the notations, we define $K_0 = \max_{1 \leq \ell < L} K_\ell$ and $z_\ell^{(k)} = e_\ell^{(k)} - \text{sign}\left(c_\ell^{(\ell)}\right) \nu_\ell$ for all ℓ. Then, we must show there exists K_L such that for all $k \geq K_L$, $\left\| z_L^{(k)} \right\| < \epsilon_L$. With no loss of generality, let's assume that sign $\left(c_\ell^{(\ell)}\right) = 1$ for all ℓ. Furthermore, let us fix $\epsilon_L > 0$ and $\epsilon_\ell = \min(1, \epsilon_0)$ for $1 \leq \ell < L$. Then, $u_L^{(k)}$ for $k \geq K_0$ can be written as

$$\begin{aligned}
u_L^{(k+1)} &= Be_L^{(k)} - \sum_{i<L} \langle e_i^{(k+1)}, Be_L^{(k)} \rangle e_i^{(k+1)} \\
&= Be_L^{(k)} - \sum_{i<L} \left(\langle \nu_i, Be_L^{(k)} \rangle \nu_i + \langle z_i^{(k+1)}, Be_L^{(k)} \rangle \nu_i + \langle \nu_i + z_i^{(k+1)}, Be_L^{(k)} \rangle z_i^{(k+1)} \right) \\
&\overset{(a)}{=} \sum_{i \geq L} \langle \nu_i, Be_L^{(k)} \rangle \nu_i - y_L^{(k)},
\end{aligned} \tag{36}$$

where (a) comes from the definition $y_L^{(k)} := \sum_{i<L} \langle z_i^{(k+1)}, Be_L^{(k)} \rangle \nu_i + \langle \nu_i + z_i^{(k+1)}, Be_L^{(k)} \rangle z_i^{(k+1)}$. Then, we can deduce $\left\| y_L^{(k)} \right\| \leq 3L\epsilon_0 d_{\max} \sqrt{2m}$ by the following lemma.

Lemma 10. *Let $B \in \mathbb{R}^{2m \times 2m}$ be an non-backtracking matrix of the graph $\mathcal{G}$, and d_{max} be the maximum degree of vertices in $\mathcal{G}$. Then, $\|Ax\|_2 < \epsilon_x d_{max} \sqrt{2m}$ if the ℓ^2-norm of the vector $x \in \mathbb{R}^{2m}$ is less than ϵ_x.*

Proof.

$$\|Bx\|_2 = \sqrt{\sum_i \left(\sum_j b_{ij} x_j \right)^2} < \sqrt{\sum_i \epsilon_x^2 \left(\sum_j b_{ij} \right)^2} \leq \epsilon_x d_{\max} \sqrt{2m}$$

□

In contrast, for any integer $p > 0$, $u_L^{(k+p)}$ is

$$\begin{aligned}
u_L^{(k+p)} &= \sum_{i \geq L} \langle \nu_i, Be_L^{(k+p-1)} \rangle \nu_i - y_L^{(k+p)} \\
&= \frac{1}{\left\| u_L^{(k+p-1)} \right\|} \sum_{i \geq L} \left\langle \nu_i, \sum_{j \geq L} \langle \nu_j, Be_L^{(k+p-2)} \rangle \lambda_j \nu_j - By_L^{(k+p-2)} \right\rangle \nu_i - y_L^{(k+p)} \\
&= \frac{1}{\left\| u_L^{(k+p-1)} \right\|} \sum_{i \geq L} \left(\lambda_i \langle \nu_i, Be_L^{(k+p-2)} \rangle - \langle \nu_i, By_L^{(k+p-2)} \rangle \right) \nu_i - y_L^{(k+p)} \\
&= \frac{1}{\left\| u_L^{(k+p-1)} \right\|} \sum_{i \geq L} \langle \nu_i, Be_L^{(k+p-2)} \rangle \lambda_i \nu_i - \frac{1}{\left\| u_L^{(k+p-1)} \right\|} \sum_{i \geq L} \langle \nu_i, By_L^{(k+p-2)} \rangle \nu_i - y_L^{(k+p)} \\
&\vdots \\
&= \frac{1}{\prod_{j=1}^{p-1} \left\| u_L^{(k+j)} \right\|} \sum_{i \geq L} \langle \nu_i, Be_L^{(k)} \rangle \lambda_i^{p-1} \nu_i \\
&\qquad - \sum_{j=1}^{p-1} \frac{1}{\left\| u_L^{(k+j)} \right\|} \left(\sum_{i \geq L} \langle \nu_i, By_L^{(k+j-1)} \rangle \lambda_i^{p-j-1} \nu_i \right) - y_L^{(k+p)} \\
&= \frac{1}{\prod_{j=1}^{p-1} \left\| u_L^{(k+j)} \right\|} \sum_{i \geq L} \langle \nu_i, Be_L^{(k)} \rangle \lambda_i^{p-1} \nu_i \\
&\qquad - \sum_{j=1}^{p-1} \frac{1}{\left\| u_L^{(k+j)} \right\|} \left(\sum_{i \geq L} \langle \nu_i, By_L^{(k+j-1)} \rangle \lambda_i^{p-j-1} \nu_i \right) - y_L^{(k+p)} \\
&\overset{(a)}{=} C_0 \lambda_L^{p-1} \left(\langle \nu_L, Be_L^{(k)} \rangle \nu_L + \sum_{i > L} \langle \nu_i, Be_L^{(k)} \rangle \left(\frac{\lambda_i}{\lambda_L} \right)^{p-1} \nu_i \right) \\
&\qquad - \sum_{i \geq L} \sum_{j=1}^{p-1} C_j \left(\langle \nu_i, By_L^{(k+j-1)} \rangle \lambda_i^{p-j-1} \nu_i \right) - y_L^{(k+p)},
\end{aligned}$$

where (a) stems from the definitions $C_0 = \frac{1}{\prod_{j=1}^{p-1} \left\| u_L^{(k+j)} \right\|}$ and $C_j = \frac{1}{\left\| u_L^{(k+j)} \right\|}$.

Now, define $\hat{e}_L^{(k+p)} := \frac{u_L^{(k+p)}}{C_0 \lambda_L^{p-1} \langle \nu_L, Be_L^{(k)} \rangle}$. Then,

$$\hat{e}_L^{(k+p)} = \nu_L + \sum_{i>L} \frac{\langle \nu_i, Ae_L^{(k)} \rangle}{\langle \nu_L, Be_L^{(k)} \rangle} \left(\frac{\lambda_i}{\lambda_L} \right)^{p-1} \nu_i \tag{37}$$

$$- \frac{\sum_{i \geq L} \sum_{j=1}^{p-1} C_j \left(\langle \nu_i, By_L^{(k+j-1)} \rangle \lambda_i^{p-j-1} \nu_i \right) - y_L^{(k+p)}}{C_0 \lambda_L^{p-1} \langle \nu_L, Be_L^{(k)} \rangle}. \tag{38}$$

Take p_0 such that $\left(\frac{\lambda_i}{\lambda_L} \right)^{p_0 - 1} \leq \frac{\epsilon \langle \nu_L, Be_L^{(k)} \rangle}{4(L-N) \langle \nu_i, Be_L^{(k)} \rangle}$, then, the ℓ^2-norm of the second term of (38) is less than $\epsilon_L/4$.

Meanwhile, the ℓ^2-norm of the third term in (38) is upper-bounded by

$$\frac{3L\epsilon_0 d_{\max} \sqrt{2m}}{C_0 \lambda_L^{p-1} \langle \nu_L, Be_L^{(k)} \rangle} \left(C d_{\max} \sqrt{2m} \sum_{i \geq L} \frac{\lambda_i^{p-2} - 1}{\lambda_i - 1} + 1 \right),$$

where $C := \max_{j=1}^{p-1} C_j$.

Therefore, if we take $\epsilon_0 < \frac{\epsilon C_0 \lambda_L^{p-1} \langle \nu_L, Be_L^{(k)} \rangle}{12 L d_{\max} \sqrt{2m}} \left(C d_{\max} \sqrt{2m} \sum_{i \geq L} \frac{\lambda_i^{p-2} - 1}{\lambda_i - 1} + 1 \right)^{-1}$, we can get $\left\| \hat{e}_L^{(k+p)} - \nu_L \right\| < \epsilon_L/2$ for $p > p_0$. Finally, the upper bound of $\left\| z_L^{(k)} \right\|$ is

$$\begin{aligned} \left\| z_L^{(k)} \right\| &\leq \left\| e_L^{(k+p)} - \hat{e}_L^{(k+p)} \right\| + \left\| \hat{e}_L^{(k+p)} - \nu_L \right\| \\ &\overset{(a)}{<} \left| \left\| \hat{e}_L^{(k+p)} \right\| - 1 \right| + \frac{\epsilon_L}{2} \\ &\overset{(b)}{\leq} \epsilon_L, \end{aligned}$$

where (a) follows from $e_L^{(k+p)} = \frac{\hat{e}_L^{(k+p)}}{\left\| \hat{e}_L^{(k+p)} \right\|}$, and (b) is obtained from the inequality $1 - \epsilon_L \leq \left\| \hat{e}_L^{(k+p)} \right\| - 1 \leq 1 + \epsilon_L$. □

E Experiment Details

In this section, we provide details of NBA-GNN implementation and experiments.

E.1 Implementation

Message initialization and aggregation. For message initialization and final aggregation of messages, we have proposed functions ϕ, σ, ρ. To be specific, the message initialization can be written as following:

$$h_{i\to j}^{(0)} = \begin{cases} \phi(e_{ij}, x_i, x_j) & \text{(if } e_{ij} \text{ exists)} \\ \phi(x_i, x_j) & \text{(otherwise)} \end{cases}.$$

For our experiments, we use concatenation for ϕ, weighted sums for σ, and average for ρ.

Non-backtracking updates. Here, we provide the details of using the non-backtracking operator in our NBA-GNNS. To begin, let's revisit the non-backtracking matrix $B \in \{0,1\}^{2|\mathcal{E}|\times 2|\mathcal{E}|}$ from Section 4.1 defined as follows:

$$B_{(\ell\to k),(j\to i)} = \begin{cases} 1 & \text{if } k = j, \ell \neq i \\ 0 & \text{otherwise} \end{cases}.$$

Returning to our NBA-GNN, the non-backtracking message passing update for a hidden feature $h_{j\to i}^{(t)}$ at the $(t+1)$-th layer, introduced in Section 3.2, is expressed as follows:

$$h_{j\to i}^{(t+1)} = \phi^{(t)}\left(h_{j\to i}^{(t)}, \left\{\psi^{(t)}\left(h_{k\to j}^{(t)}, h_{j\to i}^{(t)}\right) : k \in \mathcal{N}(j) \setminus \{i\}\right\}\right), \tag{3}$$

where $\phi^{(t)}$ and $\psi^{(t)}$ are backbone-specific non-linear update and permutation-invariant aggregation functions at the t-th layer, respectively. Using the non-backtracking matrix B, we can rewrite Equation (3) as following:

$$H^{(t+1)} = \phi^{(t)}\left(H^{(t)}, \psi^{(t)}\left(B^\top H^{(t)}, H^{(t)}\right)\right), \tag{39}$$

where $H^{(t)} \in \mathbb{R}^{2|\mathcal{E}|\times d}$ are edge-wise features, i.e., each row representing $h_{j\to i}^{(t)}$.

Now, the NBA-GNN implementation example in Section 3 using GCN [1] as a backbone, can be written as the following:

$$h_{j\to i}^{(t+1)} = h_{j\to i}^{(t)} + \frac{1}{|\mathcal{N}(j)| - 1}\mathbf{W}^{(t)} \sum_{k\in\mathcal{N}(j)\setminus\{i\}} h_{k\to j}^{(t)}.$$

Recall the out-degree and in-degree matrices of NBA-GNN, denoted as D_{out} and D_{in}, where $(D_{out})_{(j\to i),(j\to i)} = \sum_{\ell\to k} B_{(j\to i),(\ell\to k)}$ and $(D_{in})_{(j\to i),(j\to i)} = \sum_{\ell\to k} B_{(\ell\to k),(j\to i)}$, for each index $j \to i$. We also introduced the normalized non-backtracking matrix augmented with self-loops as $\widehat{B} = (D_{out} + I)^{-\frac{1}{2}}(B + I)(D_{in} + I)^{-\frac{1}{2}}$. In summary, Equation (5) can be expressed as follows:

$$H^{(t+1)} = \widehat{B}H^{(t)}\mathbf{W}^{(t)}. \tag{40}$$

Hence, the message passing update in NBA-GCN is equivalent to the multiplication of non-backtracking operator and edge-wise features.

E.2 Long range graph benchmark

Dataset statistics. From LRGB [17], we experiment for 3 tasks: graph classification (`Peptides-func`), graph regression (`Peptides-struct`), and node classification (`PascalVOC-SP`). We provide the dataset statistics in Table 5. Note that for `PascalVOC-SP`, we use SLIC compactness of 30 and edge weights are based only on super-pixels coordinates.

Experiments Details All experiment results are averaged over three runs and trained for 300 epochs, with a ∼500k parameter budget. Baseline scores were taken from each paper.

Table 5: Statistics of LRGB datasets

Dataset	Total Graphs	Total Nodes	Avg Nodes	Mean Deg.	Total Edges	Avg Edges	Avg Short. Path.	Avg Diameter
PascalVOC-SP	11,355	5,443,545	479.40	8.00	43,548,360	3,835.17	8.05 ± 0.18	19.40 ± 0.65
Peptides-func	15,535	2,344,859	150.94	2.04	4,773,974	307.30	20.89 ± 9.79	56.99 ± 28.72
Peptides-struct	15,535	2,344,859	150.94	2.04	4,773,974	307.30	20.89 ± 9.79	56.99 ± 28.72

Table 6: Best hyperparameters for each GNN, dataset in LRGB benchmark

Model	Dataset	# Param.	# Layers	hidden dim.	dropout	Batch size	# epochs
GCN	PascalVOC-SP	472k	12	180	0.7	30	200
	Peptides-func	510k	10	186	0.1	200	300
	Peptides-struct	505k	20	144	0.1	200	300
GINE	PascalVOC-SP	472k	12	180	0.7	30	200
	Peptides-func	502k	10	144	0.1	200	300
	Peptides-struct	503k	10	144	0.1	200	300
GatedGCN	PascalVOC-SP	486k	10	96	0.25	30	200
	Peptides-func	511k	10	96	0.1	200	300
	Peptides-struct	511k	8	108	0.1	200	300

- All experiments are averaged over three seeds, 0∼2.
- Baseline numbers were took from the lrgb benchmak [17] and table1 of DRew[61].
- We use an AdamW optimizer[70] with lr decay=0.1 , min lr=1e-5, momentum=0.9, and base learning rate lr=0.001 (0.0005 for PascalVOC-SP).
- We use cosine scheduler with reduce factor=0.5 , schedule patience=10 with 50 warm-up.
- Laplacian positional encoding were used with hidden dimension 16 and 2 layers.
- We use the "Atom Encoder", "Bond Encoder" for Peptides-func, Peptides-struct from based on OGB molecular feature [71, 72], and the "VOCNode Encoder", "VOCEdge Encoder" for PascalVOC-SP.
- PascalVOC-SP results in Figure 5a all use the same setup described above.
- Peptides-func results in Figures 5b and 5c all use the same setup described above.
- GCN results in Figures 5a to 5c use the hyperparameters from [67].

We searched the following range of hyperparameters, and report the best in Table 6.

- We searched layers 6 to 12 for PascalVOC-SP 2, layers 5 to 20 for Peptides-func and Peptides-struct.
- The hidden dimension was chosen by the maximum number in parameter budget.
- Dropout was searched from 0.0∼0.8 for PascalVOC-SP in steps of 0.1, and 0.1∼0.4 in steps of 0.1 for Peptides-func and Peptides-struct.
- We used the batch size of 30 for PascalVOC-SP on GPU memory, and 200 for Peptides-func and Peptides-struct.

E.3 Transductive node classification

We conducted experiments involving three citation networks (Cora, CiteSeer, and Pubmed in [18]), and three heterophilic datasets (Texas, Wisconsin, and Cornell in [19]), focusing on transductive node classification. Our reported results in Table 3 are the averages obtained from 10 different seed runs to ensure robustness and reliability.

For the citation networks, we employed the dataset splitting procedure outlined in [73]. In contrast, for the heterophilic datasets, we randomly divided the nodes of each class into training (60%), validation (20%), and testing (20%) sets.

During the model training process, we utilized the AdamW optimizer [70] with a learning rate of 3e-5. The training duration spanned 1,000 epochs for citation networks and 100 epochs for heterophilic datasets. Following training, we selected the best epoch based on validation accuracy for evaluation on the test dataset. The model's hidden dimension and dropout ratio were set to 512 and 0.2, respectively, consistent across all datasets, after fine-tuning these hyperparameters on the Cora dataset. Additionally, we conducted optimization for the number of convolutional layers within the set $\{1, 2, 3, 4, 5\}$. The results revealed that the optimal number of layers is typically three for most of the models and datasets. However, there are exceptions, such as CiteSeer-GatedGCN, PubMed-{GraphSAGE, GraphSAGE+NBA+PE}, Texas-GatedGCN+NBA, Wisconsin-{GraphSAGE+NBA, GraphSAGE+NBA+PE} and Cornell-{GraphSAGE, GraphSAGE+NBA, GraphSAGE+NBA+PE, GatedGCN+NBA}, where the optimal number of layers is found to be four. Furthermore, for Cora-{GraphSAGE+NBA+PE, GAT+NBA},CiteSeer-GraphSAGE+NBA, and Cornell-GatedGCN+NBA+PE.

For reference, we have provided a summary of dataset statistics in Table 7

Table 7: Statistics of the datasets for the node classification task

Dataset	Total Graphs	Num Nodes	Num Edges	Dim Features	Num Classes
`Cora`	1	2,708	10,556	1,433	7
`CiteSeer`	1	3,327	9,104	3,703	6
`PubMed`	1	19,717	88,648	500	3
`Texas`	1	183	309	1,703	5
`Wisconsin`	1	251	499	1,703	5
`Cornell`	1	183	295	1,703	5

F Time and Space Complexity

Space complexity. NBA-GNNs construct message for every edge considering directions, and connects them in a non-backtracking matter. This requires $2|\mathcal{E}|$ number of messages, and $(d_{avg}-1)|\mathcal{E}|$ connections among messages, where d_{avg} is the average degree of the graph. This has not been a bottleneck at practice, and can be reduced by adjusting the batch size.

Time complexity. As we connect the messages in a non-backtracking matter, we must calculated the relation between edges in the pre-processing process. This non-backtracking edge adjacency can be computed in $\mathcal{O}(|\mathcal{E}|^2)$, and $\mathcal{O}(d_{avg}|\mathcal{E}|)$ if the data is provided in the form of adjacency list. However, this is only required once per data, can be pre-computed, independent with the number of layers, and re-used for runs.