

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

Model	training AP \uparrow	test AP \uparrow	test time per epoch \downarrow	GPU usage (%) \uparrow
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) (Chen et al., 2017) 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) (Chen et al., 2022) 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 [Di Giovanni et al. \(2023\)](#), 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 [Fasino et al. \(2021\)](#) 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 $x_0, x_1, x_2 \dots$ 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 :

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

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

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

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 , $T_i := \min\{n \geq 0 | x_n = i\}$, and \bar{T}_i be the time when a random walk first arrived at node i after the first step, $\bar{T}_i := \min\{n > 0 | x_n = i\}$. With slight abuse of notation, we define access time $\mathfrak{t}(i, j)$ from i to j , access time $\mathfrak{t}(i \rightarrow j, k)$ from i to k where $x_1 = j$, and return time $\mathfrak{t}(i)$ in a graph \mathcal{G} as follows:

$$\begin{aligned} \mathfrak{t}(i, j) &:= \mathbb{E}[T_j | x_0 = i] \\ \mathfrak{t}(i \rightarrow j, k) &:= \mathbb{E}[T_k | x_1 = j, x_0 = i] \\ \mathfrak{t}(i; \mathcal{G}) &:= \mathbb{E}[\bar{T}_i | x_0 = i] \end{aligned}$$

We similarly denote access time and return time of BBRW as $(\tilde{\mathfrak{t}}(i, j), \tilde{\mathfrak{t}}(i \rightarrow j, k))$ and $\tilde{\mathfrak{t}}(i; \mathcal{G})$, 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., $\mathfrak{t}(v_n, v_{n+1})$ for $n \in 0, \dots, N - 1$. ([Lemma 2](#))

2. We evaluate the access time of neighboring nodes in the path, e.g., $\mathfrak{t}(v_n, v_{n+1})$, 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, \dots, v_N) ,*

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

Proof. The proof outline is as follows. Given a unique path (v_0, \dots, 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 $(v_0, v_1), (v_1, v_2), \dots, (v_{N-1}, v_N)$ 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.

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

From the Markov property of SRW:

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

Thus,

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

□

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 $\mathfrak{t}(i, j)$ is defined as follows:*

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

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,

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

where $\mathfrak{t}(i; \mathcal{G}_i)$ 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, as stated by Fasino et al. (2021), the following equation holds:

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

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

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 .*

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

where $\mathcal{E}(\mathcal{G})$ is edge set of graph \mathcal{G} and \mathcal{G}_n 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.

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

\square

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 $\tilde{\mathfrak{t}}$ 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 $\tilde{\mathfrak{t}}(i, j)$ 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, \dots, v_N) . Then the access time of BBRW between node v_0 and v_N can be derived as follows:*

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

Proof. The proof is similar to Lemma 2 such that we decompose the walk into a series of $N-1$ walks between $(v_0, v_1), (v_1, v_2), \dots, (v_{N-1}, v_N)$ 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 $\tilde{\mathfrak{t}}(v_{n-1} \rightarrow v_n, v_{n+1}) - 1$.

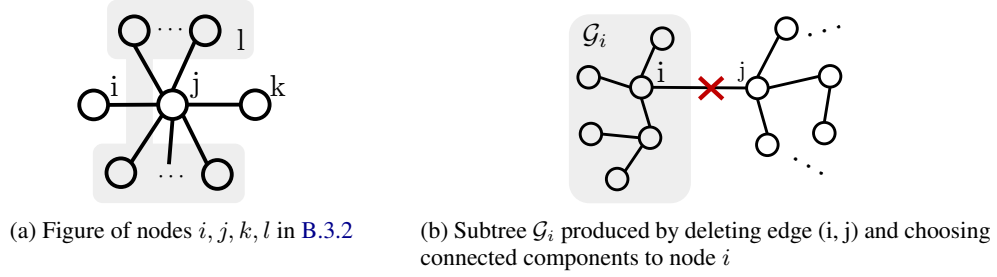


Figure 6: References for section B.3.2

Note that $\tilde{\tau}(v_{n-1} \rightarrow v_n, v_{n+1})$ counts the length of walk from v_{n-1} to v_{n+1} , hence should be subtracted 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.

$$\tilde{\tau}(v_0, v_N) = \sum_{n=0}^{N-1} \mathbb{E}[T_{v_{n+1}} - T_{v_n} | x_0 = v_0]$$

When the BBRW random walk reaches v_n at T_{v_n} , it previously passed the node v_{n-1} . (i.e., $x_{T_{v_n}-1} = v_{n-1}$). 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 $x_1 = v_n$. Thus,

$$\mathbb{E}[T_{v_{n+1}} - T_{v_n} | x_0 = v_0] = \tilde{\tau}(v_{n-1} \rightarrow v_n, v_{n+1}) - 1 \quad \text{for } l \geq 1.$$

□

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 $\tilde{\tau}(v_{n-1} \rightarrow v_n, v_{n+1})$.

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

$$\tilde{\tau}(i \rightarrow j, k) - 1 = \tilde{\tau}(j, k) - \frac{d_i - 1}{d_j} \tilde{\tau}(i; \mathcal{G}_i) - \frac{2}{d_j}, \quad (9)$$

where $\tilde{\tau}(i; \mathcal{H})$ 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 $i \rightarrow j$ to k can be expressed by its subset $j \rightarrow l, k$ where $l \neq i$ due to the non-backtracking property.

$$\tilde{\tau}(i \rightarrow j, k) - 1 = \frac{1}{d_j - 1} + \sum_{l \in \mathcal{N}(j) \setminus \{i, k\}} \frac{1}{d_j - 1} \tilde{\tau}(j \rightarrow l, k) \quad (10)$$

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 .

$$\tilde{\tau}(j, k) = \frac{1}{d_j} \sum_{l \in \mathcal{N}(j)} \tilde{\tau}(j \rightarrow l, k), \quad (11)$$

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

$$\tilde{\tau}(j \rightarrow i, k) = 1 + (d_i - 1) \tilde{\tau}(i; \mathcal{G}_i) + \tilde{\tau}(i \rightarrow j, k). \quad (12)$$

This is similar to Equation (7), where \mathcal{G}_i describes how the walk from $j \rightarrow i$ 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:

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

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

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

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

Thus, we can conclude for $\tilde{\tau}(i \rightarrow j, k) - 1$:

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

□

B.3.3 RETURN TIME WITH RESPECT TO A SUBGRAPH

Now we need the formula for the return time $\tilde{\tau}(i; \mathcal{G})$ 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,*

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

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 $x_0 = i$, we return to i at second transition (i.e., $x_2 = i$) since all the neighbors of i is leaf node. Since $d_i = |\mathcal{E}|$ for tree with height 1, $\tilde{\tau}(i; \mathcal{G}) = 2 = \frac{2|\mathcal{E}|}{d_i}$.

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 $x_1 = l$ 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,

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

where $\tilde{t}(l; \mathcal{G})$ 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 , $\tilde{t}(l; \mathcal{G}_l) = \frac{2|\mathcal{E}(\mathcal{G}_l)|}{d_l - 1}$. Therefore,

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

□

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 ,*

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

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 $\tilde{t}'(i)$, 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:

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

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

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 .*

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

where $\mathcal{E}(\mathcal{G})$ is the edge set of \mathcal{G} and \mathcal{G}_n 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),

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

□

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}_n be the subtree produced by deleting edge (v_n, v_{n+1}) and choosing connected component of v_n . Then,

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

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 $C \in \mathbb{R}^{2|\mathcal{E}| \times |\mathcal{V}|}$:

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

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 $(D_{out})_{(j \rightarrow i), (j \rightarrow i)} = \sum_{\ell \rightarrow k} B_{(j \rightarrow i), (\ell \rightarrow k)}$ and $(D_{in})_{(j \rightarrow i), (j \rightarrow i)} = \sum_{\ell \rightarrow k} B_{(\ell \rightarrow k), (j \rightarrow i)}$, for each index $j \rightarrow i$. Next, we introduce \hat{B} as the normalized non-backtracking matrix augmented with self-loops: $\hat{B} = (D_{out} + I)^{-\frac{1}{2}}(B + I)(D_{in} + I)^{-\frac{1}{2}}$. Then one obtains the following sensitivity bound of NBA-GNN. We also let \tilde{C} denote a matrix where $\tilde{C}_{(k \rightarrow j), i} = C_{(k \rightarrow j), i} + C_{(j \rightarrow k), i}$.

Consequently, one can express our NBA-GNN updates:

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

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:

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

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. (Topping et al., 2022) 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

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

Now, we show that the sensitivity bound for non-backtracking GNNs can be defined as following. Following (Topping et al., 2022), 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:

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

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

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

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

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

$$(21)$$

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

Thus considering the message update from Equation (15),

$$\begin{aligned} \frac{\partial h_{k \rightarrow \ell}^{(T-1)}}{\partial x_i} &= \partial_1 \phi^{(T-2)} \frac{\partial h_{k \rightarrow \ell}^{(T-2)}}{\partial x_i} + \partial_2 \phi^{(T-2)} \left(\sum_{k' \rightarrow \ell' \in N_e^-(k)} \hat{B}_{k' \rightarrow \ell', k \rightarrow \ell} \frac{\partial h_{k \rightarrow \ell}^{(T-2)}}{\partial x_i} \right) \\ &= \partial_2 \phi^{(T-2)} \left(\sum_{k' \rightarrow \ell' \in N_e^-(k)} \hat{B}_{k' \rightarrow \ell', k \rightarrow \ell} \frac{\partial h_{k \rightarrow \ell}^{(T-2)}}{\partial x_i} \right), \end{aligned}$$

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

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.

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

$$\leq \alpha \beta \left(\sum_{k' \rightarrow \ell' \in N_e^-(k)} \hat{B}_{k' \rightarrow \ell', k \rightarrow \ell} \left\| \frac{\partial h_{k \rightarrow \ell}^{(T-2)}}{\partial x_i} \right\| \right) \quad (23)$$

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

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

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

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

$$\begin{aligned} \left\| \frac{\partial h_j}{\partial x_i} \right\| &\leq (\alpha \beta)^T \gamma \left(\sum_{(s_0, \dots, s_T) \in s(i, j)} \left(\prod_{t=1}^{T-1} \hat{B}_{s_{t-1} \rightarrow s_t, s_t \rightarrow s_{t+1}} \right) \right) \\ &= (\alpha \beta)^T \gamma (\tilde{C}^\top \hat{B}^{T-1} \tilde{C})_{j, i} \end{aligned} \quad (27)$$

□

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 \hat{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 (Topping et al., 2022), i.e., $(\tilde{C}^\top \hat{B}^{T-1} \tilde{C})_{j,i} \geq (\hat{A}^T)_{j,i}$. For d -regular graphs, $(\tilde{C}^\top \hat{B}^{T-1} \tilde{C})_{j,i}$ decays slower by $O(d^{-T})$, while $(\hat{A}^T)_{j,i}$ 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 $s(i, j) = (s_0 = i, s_1, \dots, s_{T-1}, s_T = j)$.

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

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

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

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

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

$$(33)$$

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

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

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

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

$$\begin{aligned} (\hat{A}^T)_{j,i} &= (d+1)^{-T} \\ (\tilde{C}^\top \hat{B}^{T-1} \tilde{C})_{j,i} &= d^{-T}. \end{aligned}$$

So $(\tilde{C}^\top \hat{B}^{T-1} \tilde{C})_{j,i}$ decays slower by $O(d^{-T})$, while $(\hat{A}^T)_{j,i}$ 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, $\alpha = (\alpha_1, \dots, \alpha_K)$ represents the probability of each vertex being assigned to communities $\mathcal{V}_1, \dots, \mathcal{V}_K$, 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 $\Omega(\log n)$ (Abbe, 2017). 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 (Yun & Proutiere, 2016). 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 (Benaych-Georges et al., 2019).

However, real-world benchmark datasets often fall within the category of very sparse graphs. For example, the citation networks dataset discussed in (Sen et al., 2008) 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 $(n, 2, (\frac{1}{2}, \frac{1}{2}), (\frac{a}{n}, \frac{b}{n}))$, 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 $n \rightarrow \infty$.

In (Stephan & Massoulié, 2022), 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. (Stephan & Massoulié, 2022) Suppose we have a SBM with parameters $(n, 2, (\frac{1}{2}, \frac{1}{2}), (\frac{a}{n}, \frac{b}{n}))$. In this case, we have two eigenvalues $\mu_1 > \mu_2$ of $n \text{diag}(\alpha)P$, 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 v_i with corresponding eigenvalues $\lambda_1 > \dots > \lambda_K \geq \lambda_{K+1} \geq \dots \geq \lambda_{2m}$, 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 $(n, 2, (\frac{1}{2}, \frac{1}{2}), (\frac{a}{n}, \frac{b}{n}))$ 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 $n \rightarrow \infty$.*

To establish [Theorem 3](#), we rely on the following [Proposition 6](#) from (Bordenave et al., 2015):

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:*

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

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

[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 v_1 and v_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 $|(T^+\nu_2 - T^+\xi_2)_i|_2 = O(f(a, b))$ 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, \dots, x_f and a matrix $X = [x_1, \dots, x_f] \in \mathbb{R}^{2m \times f}$, which has x_1, \dots, 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 orthonormalization 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:

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

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:

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

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

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

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 $k \geq K_\ell$, $\|e_\ell^{(k)} - \text{sign}(c_\ell^{(\ell)}) \nu_\ell\| < \epsilon_\ell$. To simplify the notations, we define $K_0 = \max_{1 \leq \ell < L} K_\ell$ and $z_\ell^{(k)} = e_\ell^{(k)} - \text{sign}(c_\ell^{(\ell)}) \nu_\ell$ for all ℓ . Then, we must show there exists K_L such that for all $k \geq K_L$, $\|z_L^{(k)}\| < \epsilon_L$. With no loss of generality, let's assume that $\text{sign}(c_\ell^{(\ell)}) = 1$ 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

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

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 $\|y_L^{(k)}\| \leq 3L\epsilon_0 d_{\max} \sqrt{2m}$ by the following lemma.

Lemma 10. *Let $B \in \mathbb{R}^{2m \times 2m}$ be a 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

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

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

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

$$\hat{e}_L^{(k+p)} = \nu_L + \sum_{i>L} \frac{\langle \nu_i, A e_L^{(k)} \rangle}{\langle \nu_L, B e_L^{(k)} \rangle} \left(\frac{\lambda_i}{\lambda_L} \right)^{p-1} \nu_i - \frac{\sum_{i \geq L} \sum_{j=1}^{p-1} C_j \left(\langle \nu_i, B y_L^{(k+j-1)} \rangle \lambda_i^{p-j-1} \nu_i \right) - y_L^{(k+p)}}{C_0 \lambda_L^{p-1} \langle \nu_L, B e_L^{(k)} \rangle}. \quad (37)$$

Take p_0 such that $\left(\frac{\lambda_i}{\lambda_L} \right)^{p_0-1} \leq \frac{\epsilon \langle \nu_L, B e_L^{(k)} \rangle}{4(L-N) \langle \nu_i, B e_L^{(k)} \rangle}$, then, the ℓ^2 -norm of the second term of (37) is less than $\epsilon_L/4$.

Meanwhile, the ℓ^2 -norm of the third term in (37) is upper-bounded by

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

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

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

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

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

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:

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

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:

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

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

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

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:

$$H^{(t+1)} = \phi^{(t)} \left(H^{(t)}, \psi^{(t)} \left(B^\top H^{(t)}, H^{(t)} \right) \right), \quad (38)$$

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

Now, the NBA-GNN implementation example in Section 3 using GCN (Kipf & Welling, 2016) as a backbone, can be written as the following:

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

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

$$H^{(t+1)} = \hat{B} H^{(t)} \mathbf{W}^{(t)}. \quad (39)$$

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 (Dwivedi et al., 2022), 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 $\sim 500k$ 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 lr gb benchmak (Dwivedi et al., 2022) and table1 of DRew(Gutteridge et al., 2023).
- We use an AdamW optimizer(Loshchilov & Hutter, 2018) 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 (Hu et al., 2020; 2021), 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 (Tönshoff et al., 2023).

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 (Sen et al., 2008)), and three heterophilic datasets (Texas, Wisconsin, and Cornell in (Pei et al., 2019)), 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 (Yang et al., 2016). 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 (Loshchilov & Hutter, 2018) 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- $\{\text{GraphSAGE}, \text{GraphSAGE+NBA+PE}\}$, Texas-GatedGCN+NBA, Wisconsin- $\{\text{GraphSAGE+NBA}, \text{GraphSAGE+NBA+PE}\}$ and Cornell- $\{\text{GraphSAGE}, \text{GraphSAGE+NBA}, \text{GraphSAGE+NBA+PE}, \text{GatedGCN+NBA}\}$, where the optimal number of layers is found to be four. Furthermore, for Cora- $\{\text{GraphSAGE+NBA+PE}, \text{GAT+NBA}\}$, 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.