Deep Graph Neural Networks via Posteriori-Sampling-based Node-Adaptive Residual Module

Jingbo Zhou[1,2]* **Yixuan Du**[2,3]* **Ruqiong Zhang**[2,3]* **Jun Xia**[1], **Zhizhi Yu**[3],
Zelin Zang[1], **Di Jin**[3], **Carl Yang**[4], **Rui Zhang**[2]† **Stan Z. Li**[1]†
[1]Westlake University, [2]Jilin University, [3]Tianjin University, [4]Emory University
{zhoujingbo, stan.zq.li}@westlake.edu.cn

Abstract

Graph Neural Networks (GNNs), a type of neural network that can learn from graph-structured data through neighborhood information aggregation, have shown superior performance in various downstream tasks. However, as the number of layers increases, node representations become indistinguishable, which is known as over-smoothing. To address this issue, many residual methods have emerged. In this paper, we focus on the over-smoothing issue and related residual methods. Firstly, we revisit over-smoothing from the perspective of overlapping neighborhood subgraphs, and based on this, we explain how residual methods can alleviate over-smoothing by integrating multiple orders neighborhood subgraphs to avoid the indistinguishability of the single high-order neighborhood subgraphs. Additionally, we reveal the drawbacks of previous residual methods, such as the lack of node adaptability and severe loss of high-order neighborhood subgraph information, and propose a **Posterior-Sampling-based, Node-Adaptive Residual module (PSNR)**. We theoretically demonstrate that PSNR can alleviate the drawbacks of previous residual methods. Furthermore, extensive experiments verify the superiority of the PSNR module in fully observed node classification and missing feature scenarios. Our code is available at https://github.com/jingbo02/PSNR-GNN.

1 Introduction

GNNs have emerged in recent years as the most powerful model for processing graph-structured data and have demonstrated exceptional performance across various fields, such as social networks [22], recommender systems [6], and drug discovery [4]. Through the message-passing mechanism that propagates and aggregates information of neighboring nodes, GNNs provide a general framework for learning information on graph structure. Despite the remarkable success, according to previous studies [17, 32], GNNs show significant performance degradation as the number of layers increase. One of the main reasons for this situation is over-smoothing [17, 21, 32, 15]. Over-smoothing refers to the phenomenon in which node representations become increasingly similar to each other as GNNs recursively aggregate more neighborhood information. This indistinguishability will inevitably degrade the performance of deep GNNs, restricting their ability to effectively model long-range dependencies among multi-hop neighbors.

Several methods have recently been proposed to alleviate over-smoothing in deep GNNs. According to [26], these methods fall into three categories: Normalization and Regularization [34, 33], Change of GNN dynamics [5], and Residual connections [32, 16]. Among all of them, the residual-based

---

*Equal contribution.

†Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

method is inspired by the success of residual neural networks (ResNets) [9] in computer vision. This type of method introduces a residual connection to the GNNs architecture. For example, JKNet [32] learns node representations by aggregating the outputs of all previous layers at the last layer. DenseGCN [16] concatenates the results of the current layer and all previous layers as the node representations of this layer. APPNP [15] uses the initial residual connection to retain the initial feature information with probability α, and utilizes information aggregated at the current layer with probability $1 - \alpha$. GCNII [3] shares a similar framework with APPNP, and it further introduces an identical mapping to avoid overfitting.

In this paper, we study the over-smoothing issue in GNNs, with a particular emphasis on residual methods. First, we revisit the over-smoothing phenomenon of GNNs from the new perspective of overlapping neighborhood subgraphs and explain the essential reason why the residual method can alleviate the over-smoothing. In essence, these methods mainly use multiple neighborhood subgraph aggregations to alleviate the indistinguishability of the single neighborhood subgraph aggregation, thereby improving the performance of the model. On this basis, we find that these residual methods often lack node adaptivity in utilizing multi-order neighborhood subgraph information, and at the same time, they still struggle to mitigate information loss when dealing with high-order neighborhood subgraphs, which hinders further improvement in the performance of deep GNNs. Although some residual methods, such as DenseGNN, can avoid these drawbacks, they tend to introduce more parameters at deeper layers. This can lead to significant memory consumption and is prone to gradient explosion, limiting the scalability of the methods.

Considering these limitations, we propose a **Posteriori-Sampling-based Node-Adaptative Residual Module (PSNR)**. More specifically, this module introduces a graph encoder to learn the posterior distribution of the required residual coefficients for each node in different layers with minor overhead. And then, we can obtain the specific fine-grained node-adaptive residual coefficients by sampling from the distribution. The contributions of this paper are as follows:

- *Perspective:* We revisit the over-smoothing issue from a novel perspective of high-order neighborhood subgraph coincidences and explain why the residual methods can alleviate it. Through this lens, we reveal several significant drawbacks of prior residual methods that limit the performance and scalability of GNNs.

- *Method:* We propose PSNR, a lightweight and model-agnostic module to mitigate the drawbacks of previous residual methods and provide theoretical justification for its advantages.

- *Experiments:* Extensive experiments verify that the PSNR module can effectively mitigate oversmoothing and further improve the performance of GNNs, especially in the case of missing feature that require deep GNNs.

2   Related Work

2.1   Notations

A connected undirected graph is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. The node features are represented in the matrix $\mathbf{H} \in \mathbb{R}^{N \times d}$, where d represents the length of the feature. Let $\mathbf{A} \in \{0, 1\}^{N \times N}$ denotes the adjacency matrix and $\mathbf{A}_{ij} = 1$ only if an edge exists between nodes v_i and v_j. $\mathbf{D} \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix, where each element d_i represents the number of edges connected to node v_i. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}, \tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ represent the adjacency matrix and degree matrix with self-loop, respectively.

2.2   Graph Neural Networks

A GNN layer updates the representation of each node via aggregating itself and its neighbors' representations. Specifically, a layer's output $\mathbf{H}'$ consists of new representations $\mathbf{h}'$ of each node computed as:
$$\mathbf{h}'_i = f_\theta \left(\mathbf{h}_i, \text{AGGREGATE} \left(\{\mathbf{h}_j \mid v_j \in \mathcal{V}, (v_i, v_j) \in \mathcal{E}\} \right) \right), \tag{1}$$

where $\mathbf{h}'_i$ indicates the new representation of node v_i and $f_\theta(\cdot)$ denotes the update function. The difference between different GNNs lies in the update function $\mathbf{f}_\theta(\cdot)$ and the $\text{AGGREGATE}(\cdot)$ function, which are also key to the performance of GNNs. Graph Convolutional Network (GCN) [14]

2

Table 1: Common residual connections for GNNs.

Residual Connection	Corresponding GCN	Formula
Res	ResGCN [16]	$\mathbf{H}_k = \mathbf{H}_{k-1} + \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_{k-1}\mathbf{W}_{k-1}\right)$
InitialRes	APPNP [15]	$\mathbf{H}_k = (1-\alpha)\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_{k-1} + \alpha\mathbf{H}$
Dense	DenseGCN [16]	$\mathbf{H}_k = \mathrm{AGG}(\mathbf{H}, \mathbf{H}_1, \ldots, \mathbf{H}_{k-1})$
JK	JKNet [32]	$\mathbf{H}_{output} = \mathrm{AGG}(\mathbf{H}_1, \ldots, \mathbf{H}_{k-1})$

is a classical massage-passing GNNs follows layer-wise propagation rule:

$$\mathbf{H}_{k+1} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_k\mathbf{W}_k\right), \tag{2}$$

where $\mathbf{H}_k$ is the feature matrix of the k-th layer, $\mathbf{W}_k$ is a layer-specific learnable weight matrix, $\sigma(\cdot)$ denotes an activation function.

2.3 Residual connection

Several works have used residual connection to alleviate the over-smoothing issue. Common residual connections for GNNs are summarized in Table 1, where $\mathbf{H}_k$ represents the output of the k-th layer, $\mathbf{W}_k$ is a learnable weight matrix for the k-th layer, α serves as a hyperparameter denoting the residual coefficient, and $\sigma(\cdot)$ denotes an activation function. Additionally, in DenseGNN, AGG represents a function with the concatenation of outputs from all previous layers as the input to the current layer, while in JKNet, AGG refers to the aggregation of all previous representations through concatenation, max-pooling, or LSTM-attention only at the final layer. Details can be found in Appendix B.

3 Why does the residual method alleviate over-smoothing?

In this section, we revisit over-smoothing from the perspective of overlapping high-order neighborhood subgraphs. Based on this, we elucidate the role of various residual methods in alleviating over-smoothing and identify their shortcomings.

3.1 Revisit over-smoothing from the perspective of neighborhood subgraphs overlapping

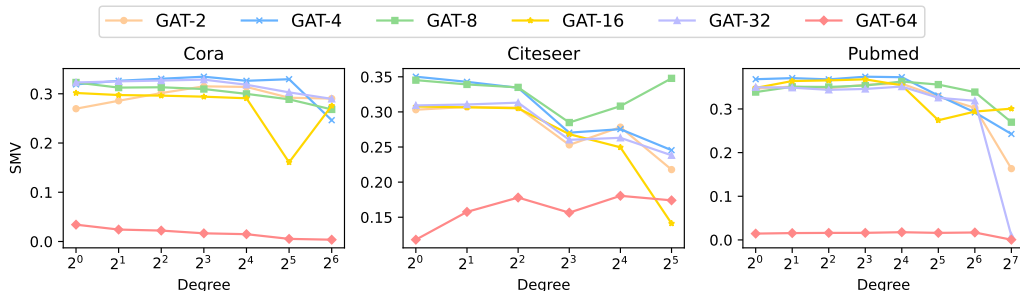



Figure 1: SMV for node groups of different degrees. More results are shown in Appendix C.

For message-passing GNNs without the residual connection, the information domain of each node after k-layer aggregation is a corresponding k-order neighborhood subgraph. Intuitively, the size of its k-order neighborhood subgraph grows exponentially as k increases, leading to a significant increase in the overlap between the k-order neighborhood subgraphs of different nodes. As a result, the aggregation result of different nodes on their respective k-order neighborhood subgraphs becomes indistinguishable. This explanation can be partially validated from the perspective of node degrees. Considering nodes with high degrees tend to have larger neighborhood subgraph overlap, the correlation between neighborhood subgraph overlap and oversmoothiong can be validated if nodes with higher degree exhibit more pronounced over-smoothing. To verify this point, we conduct experiments on three graph datasets: Cora, Citeseer, and Pubmed. Initially, nodes are grouped based on their degrees, with nodes having degrees falling within the range of $[2^i, 2^{i+1})$ assigned to the

3

i-th group. Subsequently, we perform aggregation with different layers of GCN and GAT and then calculate the degree of smoothing of the node representations within each group separately. The metric proposed in [19] is used to measure the smoothness of the node representations within each group, namely smoothness metric value (SMV), which calculates the average distances between the nodes within the group:

$$\text{SMV}(\mathbf{X}) = \frac{1}{N(N-1)} \sum_{i \neq j} \mathcal{D}\left(\mathbf{X}_{i,:}, \mathbf{X}_{j,:}\right), \tag{3}$$

where $\mathcal{D}(\cdot, \cdot)$ denotes the normalized Euclidean distance between two vectors:

$$\mathcal{D}(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \left\| \frac{\mathbf{x}}{\|\mathbf{x}\|} - \frac{\mathbf{y}}{\|\mathbf{y}\|} \right\|_2. \tag{4}$$

From the definition, a smaller value of SMV indicates a greater similarity in node representations. We show the result of GAT in Figure 1. More results can be found in Appendix C. It can be observed that the groups of nodes with higher degree tend to be more similar to each other within the group in different layers. This finding supports our claim.

3.2 The role of residual method in alleviating over-smoothing

After verifying the conclusion that neighborhood subgraph overlap leads to over-smoothing, a natural idea is to alleviate the overlap of the single neighborhood subgraph by utilizing multi-order neighborhood subgraph aggregations. In the following section, we show that the previous k-layer residual-based GNNs essentially represent different forms of utilizing neighborhood subgraph aggregations from 0 to k orders.

Table 2: Utilization of neighborhood subgraphs by various residual methods.

Model	Closed/Iterative form formulas
ResGCN	$\mathbf{H}_k = \sum_{j=0}^{k} \binom{j}{k} \mathbf{N}^j \mathbf{H}$
APPNP	$\mathbf{H}_k = (1-\alpha)^k \mathbf{N}^k \mathbf{H} + \alpha \sum_{j=0}^{k-1} \sum_{i=0}^{j} (-1)^{j-i} (1-\alpha)^i \mathbf{N}^i \mathbf{H}$
JKNet	$\mathbf{H}_k = \text{AGG}(\mathbf{N}\mathbf{H}, \dots, \mathbf{N}^{k-1}\mathbf{H})$
DenseGCN	$\mathbf{H}_k = \text{AGG}(\mathbf{H}_{k-1}, \dots, \mathbf{H}_1, \mathbf{H}_0)$

In the rest of this paper, we take GCN, a classical residual-free message-passing GNN, as an example. Assuming that $\mathbf{H}$ is non-negative, the ELU function and the weight matrix can be ignored for simplicity. Combined with the formula of GCN in Eq. 2, the k-order neighborhood subgraph aggregation can be formulated as $\mathbf{N}^k \mathbf{H}$, where $\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$. To show more intuitively how different residual models utilize multi-order neighborhood subgraph aggregation $\mathbf{N}^k \mathbf{H}$, we rewrite their formula in Table 2. Details of the derivation of the closed-form formula in this part are given in Appendix D. As can be observed, the output of GCN's residual-based variants contains multi-order matrix products that represent different order neighborhood subgraph aggregations from 0 to k. There are two main ways to exploit them: **(1)** Summation, such as ResGCN and APPNP. Such methods employ linear summation over the aggregation of different order neighborhood subgraphs; **(2)** Aggregation functions, such as DenseGNN and JKNet. Such methods make direct and explicit exploitation of different order neighborhood subgraph aggregations through operations such as concatenation.

However, the utilization of multi-order neighborhood subgraph aggregations in these methods presents the following issues: Firstly, the summation methods all use a fixed coefficient to sum the neighborhood subgraph aggregations. Consequently, these methods inherently presume that the information from the neighborhood subgraph of the same order is equally important for different nodes, which lacks node adaptivity. Secondly, for ResGNN, APPNP, and JKNet, when the number of layers increases, the output of these methods still involves many high-order matrix products that are over-smoothed. This leads to severe information loss when aggregating high-order neighborhood subgraphs, which in turn degrades model performance at deeper layers. Although DenseGNN seems to alleviate the above issues to some extent, the recursive use of all previous neighborhood subgraph aggregation would introduce more parameters as the model deepens. This increases memory consumption and raises the risk of gradient explosion at deeper layers.

4

4 The Proposed Method PSNR

4.1 Methodology

To solve the above issues, we propose a node-adaptive and lightweight residual module named PSNR. The motivation is to learn the adaptive residual coefficients for each node, thereby achieving fine-grained and node-level neighborhood subgraph aggregation to improve the performance of GNNs. However, directly learning these coefficients through backpropagation presents significant challenges. The primary challenge lies in the lack of transferability of learned coefficients. Specifically, in tasks such as semi-supervised node classification, nodes in the test and validation sets often cannot propagate information to the training nodes through multiple message-passing steps. Therefore, we cannot learn effective coefficients for these nodes during the training phase.

As a remedy, we regard node-level residual coefficients as hidden parameters. Our strategy involves estimating their posterior distribution $\mathbf{P}(\eta_k|\mathbf{A}, \mathbf{H}_k, k)$. In most cases, since the training, validation, and test sets originate from the same distribution, the posterior distribution learned from the training data possesses transferability to the validation and testing nodes. We can assume the posterior distribution to be Gaussian:

$$\eta_k \sim \mathcal{N}(\mu(\mathbf{A}, \mathbf{H}_k, k), \sigma^2(\mathbf{A}, \mathbf{H}_k, k)).$$

A graph encoder can be used to model this distribution. This encoder leverages graph topology and node information as inputs. Subsequently, to enable backpropagation, we employ the reparameterization trick. This technique enables us to represent the sampling process from the aforementioned distribution as follows:

Figure 2: The framework of PSNR Module.

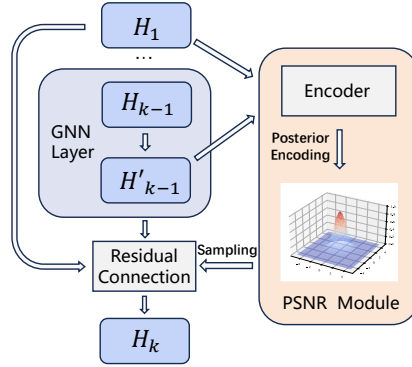
$$\eta_k = \mu(\mathbf{A}, \mathbf{H}_k, k) + \zeta \cdot \sigma(\mathbf{A}, \mathbf{H}_k, k), \ \zeta \sim \mathcal{N}(0, 1).$$

Furthermore, we parameterize $\mu(\mathbf{A}, \mathbf{H}_k, k)$ and $\sigma(\mathbf{A}, \mathbf{H}_k, k)$ as an arbitrary GNN layer. While the posterior distributions of residual coefficients vary across different layers, we do not parameterize a specific encoder for each layer. Instead, we employ the same graph encoder and use the positional embedding generated from the layer number k to differentiate the posterior distributions of residual coefficients for various layers. Consequently, the PSNR module can be formulated as follows:

$$\begin{aligned}
\mathbf{H}'_{k-1} &= \mathrm{GraphConv}\left(\mathbf{H}_{k-1}\right) \\
\mathbf{H}_k &= \mathbf{H}_1 + \phi(diag(\eta_{k-1}))\left(\mathbf{H}_1 - \mathbf{H}'_{k-1}\right), \ \eta_{k-1} \sim \mathcal{N}(\mu_{k-1}, {\sigma_{k-1}}^2) \\
\mu_k, \sigma_k &= \mathrm{GraphEncoder}_{\mu,\sigma}(\mathbf{H}_1 - \mathbf{H}'_k + \gamma\, \mathrm{LayerEmb}(k)),
\end{aligned} \tag{5}$$

where $\mathrm{GraphConv}(\cdot)$ is the k-th layer of any backbone GNN, $\mathbf{H}_k$ denotes the node representation matrix of the k-th layer, and $\mathbf{H}'_k$ represents the output matrix of the k-th layer. The first equation corresponds to the aggregation operation of the backbone GNN at the k-th layer. η_k represents the node-level residual coefficient at the k-th layer, where the element $\eta_k^{(i)}$ corresponds to the residual coefficient at the i-th node. In addition, η_k follows a high-dimensional Gaussian distribution: $\mathcal{N}(\mu_k, {\sigma_k}^2)$, and each time before each residual calculation, the distribution is first sampled to obtain the exact residual coefficients. μ_k and σ_k represent the mean and standard deviation of the distribution, respectively. $diag(\eta_k)$ represents a diagonal matrix transformed from η_k, where the i-th diagonal element is precisely the i-th element of η_k. $\phi(\cdot)$ represents the sigmoid function, which constrains the residual coefficient to $(0, 1)$. $\mathrm{GraphEncoder}_{\mu,\sigma}(\cdot)$ is a posterior encoder, which can be any graph convolution layer that is independent of the backbone GNN. $\mathrm{LayerEmb}(k)$ represents the positional embedding [29] with layer number k as input. γ is a learnable coefficient serving as the layer embedding factor. The analysis of the residual coefficients can be found in Appendix H.

Also, it is noteworthy that PSNR introduces randomness by sampling residual coefficients during both training and testing phases, thereby adding learnable random perturbations. This is different to other methods including DropEdge [24], GRAND [8] and DropMessage [7] that only incorporate randomness during the training phase and primarily introduce perturbations to node features and graph structure. We provide the theoretical analysis for this design in Section 4.2. Additionally, we clarify the difference between PSNR and other subgraph-based methods in Appendix F.

5

4.2 Theoretical justifications on the advantages of the PSNR module

In this section, we theoretically show that the PSNR module achieves finer-grained and node-adaptive neighborhood subgraph aggregations while avoiding the loss of high-order subgraph information. Firstly, combining with Equation 5, the matrix form of the iterative formula for PSNR-GCN is:

$$\mathbf{H}_k = \mathbf{H}_1 + \Lambda_{k-1} \left(\mathbf{H}_1 - \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{H}_{k-1} \right).$$

For simplicity, we use Λ_k to denote $diag(\eta_k)$. Subsequently, based on this recursive form of formula, we derive the closed-form expression of PSNR-GCN as:

$$\mathbf{H}_k = \sum_{i=2}^{k-1} \prod_{j=i}^{k-1} \tilde{\mathbf{N}}_j (\mathbf{M}_i - \mathbf{M}_{i-1}) + \prod_{i=1}^{k-1} \tilde{\mathbf{N}}_i (\mathbf{H}_1 + \mathbf{M}_1) - \mathbf{M}_{k-1},$$

where $\tilde{\mathbf{N}}_i = -\Lambda_{k-1}\mathbf{N}$, and $\mathbf{M}_k = -(\Lambda_k \mathbf{N} + \mathbf{I})^{-1}(\mathbf{I} + \Lambda_k)\mathbf{H}_1$. The detailed derivation of this formula can be found in Appendix E. The first two terms consist of cumulative product terms of different orders, similar to the form of $\mathbf{N}^k \mathbf{H}$, thus approximating a new version of neighborhood subgraph aggregation. Additionally, the formula involves the aggregation of all neighborhood subgraphs from 1 to k orders. This ensures that our method, like other residual methods, can comprehensively utilize multi-order neighborhood subgraph aggregations to enhance performance. Furthermore, since Λ_k is a diagonal matrix computed by a learnable posterior encoder with graph structure, node feature and layer number as input, the neighborhood subgraph aggregation of PSNR-GCN is fine-grained and node-adaptive. This sets it apart from methods like ResGCN and APPNP.

Beyond that, we can also demonstrate that the PSNR module can reduce the information loss of high-order subgraph aggregation, thereby further improving the performance of GNNs at deeper layers. Specifically, we aim to prove that as the order k increases, the smoothing rate of the cumulative product terms in PSNR is slower than that of the matrix power terms in other methods. Since we need to analyse the smoothing rate, which involves analyzing the asymptotic behavior of cumulative product terms or matrix power terms as the order increases, we can generalize the problem setup. Therefore, we only need to prove the following proposition to support our claim.

Proposition 1 *Let $\mathcal{S} = \{\mathbf{S}_j = \Lambda_j \mathbf{N} | j \in \mathbb{N}_0\}$, where Λ_j represents a random diagonal matrix with each diagonal element $\Lambda_{j,ii}$ satisfying $0 < \Lambda_{j,ii} < 1$, and let $\mathbf{X}$ be any feature matrix. Then, as the order k increases, the product of elements in the set $\mathcal{S}$ and the matrix $\mathbf{X}$, denoted as $\mathbf{X}^{(k)} = \prod_{i=1}^k \mathbf{S}_i \mathbf{X}$, converges to a rank-one matrix with identical rows slower than $\mathbf{X}_{GCN}^{(k)} = \mathbf{N}^k \mathbf{X}$.*

Since the diagonal elements of the diagonal matrix Λ are all results of the sigmoid function, we can assume that they have a lower bound $\epsilon > 0$. Therefore, in this setting, each element $\mathbf{S}_i$ in the set $\mathcal{S}$ is a row-stochastic matrix satisfying the following property.

Property 1 *For a row-stochastic matrix $\mathbf{S}_k$, there exists an $\epsilon > 0$ satisfying the following conditions:*
1. $\epsilon \leq \mathbf{S}_{k,ij} \leq 1$, if $(i,j) \in \mathcal{E}$,
2. $\mathbf{S}_{k,ij} = 0$, if $(i,j) \notin \mathcal{E}$.

We can refer to the conclusion in [31]. [31] describes the attention matrix of GAT as a row-stochastic matrix satisfying Property 1. Leveraging mathematical tools such as joint spectral radius from the perspective of dynamical systems, this work proves that the convergence rate of GAT is bounded below by GCN. Due to all the matrix $\mathbf{S}_i$ from $\mathcal{S}$ also satisfies Property 1, this conclusion can be borrowed to prove Proposition 1, thereby demonstrating that the PSNR module can alleviate the loss of high-order neighborhood subgraph information. Furthermore, this conclusion also explains the introduction of randomness during both the training and testing phases. This random perturbation further reduces the smoothing rate, thereby enhancing the model's performance.

4.3 Complexity analysis

We take PSNR-GCN as an example to provide a complexity analysis of the PSNR module. The time complexity of a vanilla GCN layer mainly comes from the matrix multiplication of $\mathbf{N}$ and $\mathbf{H}$, hence its complexity is $O(n^2 d)$. And the main computational parts of PSNR module are the computation of mean and standard deviation, sampling of $p_k^{(i)}$, scalar multiplication and matrix

6

addition, which correspond to a complexity of $O(n^2 d)$, $O(n)$, $O(nd)$, and $O(nd)$, respectively. Thus the time complexity of the PSNR module is $O(n^2 d)$ and the time complexity of a GCN layer equipped the PSNR module is $O(n^2 d)$. As for space complexity, PSNR module needs to store the computed mean and variance for each node, i.e., $O(2n)$, which can be approximately considered as $O(n)$. Section 5.6 compares memory consumption with other residual methods on large graphs.

5   Experiment

In this section, we assess the performance of the PSNR module in comparison to other methods and answer the following research questions (**RQ**): **RQ1.** How well does the PSNR alleviate oversmoothing? **RQ2.** How does PSNR perform compared to other baseline when used with different backbone? **RQ3.** Can PSNR enable deeper networks to perform better under the missing feature scenario? **RQ4.** How scalable is the PSNR on large graph datasets?

5.1   Datasets

We conducted experiments on ten real-world datasets, including three citation network datasets, i.e., Cora, Citeseer, Pubmed [27], two web network datasets, i.e., Chameleon and Squirrel [25], co-author/co-purchase network datasets, i.e., Coauthor-CS [28], Amazon-Photo [28] and three larger datasets, i.e., Flickr [20], Coauthor-Physics [28] and Ogbn-arxiv [10]. More details of these datasets and data-splitting procedures can be found in Appendix G.

5.2   Baselines and experimental settings

We consider two fundamental GNNs, GCN [14] and GAT [30]. For GCN, we test the performance of PSNR-GCN and its residual variant models, including ResGCN [16], DenseGCN [16], GCNII [3] and JKNet [32]. For GAT, we directly equip it with the following residual module: Res-GAT, InitialRes-GAT, Dense-GAT, JK-GAT and PSNR-GAT. And we adopt the GraphSAGE layer as the GraphEncoder of the PSNR module. The impact of different graph encoders on the experiments can be found in Appendix I. In addition, we compare three recent representative methods belonging to different categories aimed at alleviating oversmoothing issues: DropMessage [7] for the drop category, DeProp [18] for the norm category, and Half-Hop [1] for graph data processing. For the missing feature setting, we also conduct comparisons with several classical oversmoothing mitigation techniques, including BatchNorm [11], PairNorm [33], DGN [34], Decorr [12], DropEdge [24]. For all baselines, the linear layers in the models are initialized with a standard normal distribution, and the convolutional layers are initialized with Xavier initialization. The Adam optimizer [13] is used for training. Experimental results are obtained from the server with four core Intel(R) Xeon(R) Platinum 8358 CPUs @ 2.60GHZ, one NVIDIA A100 GPU (80G), and models and datasets used in this paper are implemented using the Deep Graph Library (DGL) and Pytorch Geometric (PyG). Further details on the specific parameter settings can be found in Appendix G.

5.3   Effectiveness in mitigating over-smoothing (RQ1)

In this section, we aim to assess whether the PSNR module can mitigate the oversmoothing phenomenon in deep GNNs. We select representative datasets Cora, Amazon-Photo, and Chameleon. Using GCN as the backbone network, we compare our method against several residual methods: ResGCN, GCNII, JKNet, and DenseGNN. We set the number of layers to 2, 4, 8, 16, 32, 64 and test on ten random splits, with the average accuracy serving as the final result. The experimental results are depicted in Figure 3. Consistent with the analysis in the main text, most methods can alleviate over-smoothing, but at deeper layers, such as 64 layer, over-smoothing still occurs. In contrast, compared to other residual methods, PSNR maintains stable performance even at deeper layers, demonstrating remarkable effectiveness in mitigating over-smoothing in deep GNNs. This is attributed to PSNR module can effectively reduce the loss of high-order neighborhood subgraph information. It is noteworthy that among the various residual methods, another initial residual method GCNII also alleviate over-smoothing well. However, the subsequent experiment will reveal that although the performance of GCNII remains relatively stable with varying layers, it leads to a decrease in overall performance.

7

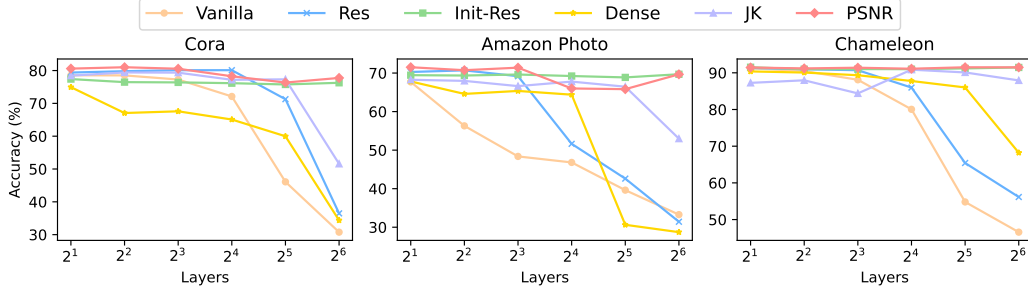



Figure 3: Different residual methods' effectiveness in mitigating over-smoothing.

Table 3: Summary of classification accuracy (%) results on real-world datasets. The best results are in bold, and the second best results are underlined.

Method	Cora	Citeseer	CS	Photo	Chameleon	Squirrel	*Avg. Rank*
GCN	78.56±1.57	66.00±2.37	90.19±0.83	91.22±0.73	67.70±0.17	47.43±1.31	6.67
ResGCN	80.11±0.98	66.40±2.29	90.86±0.99	91.31±0.85	70.70±1.54	52.43±1.72	3.67
GCNII	77.40±1.43	66.69±1.95	90.34±0.78	91.46±0.84	69.69±1.36	46.13±1.59	5.00
DenseGCN	74.94±2.46	62.54±2.56	90.28±1.06	90.38±1.02	67.82±1.27	49.57±1.58	7.17
JKNet	79.42±1.48	65.49±2.32	90.94±0.89	90.85±1.19	68.26±1.15	49.87±1.44	5.17
Half-Hop-GCN	80.74±1.80	67.76±2.19	90.30±1.83	89.32±1.17	60.07±2.34	42.78±1.01	6.33
DropMessage-GCN	80.00±2.22	67.66±1.84	90.64±2.34	91.40±0.90	65.91±2.43	45.71±1.76	4.83
DeProp-GCN	79.52±2.63	67.24±2.02	90.98±2.04	91.38±1.33	61.51±1.40	44.07±1.07	5.17
PSNR-GCN	**81.01±1.63**	**68.06±2.12**	**91.23±1.00**	**91.51±0.69**	**71.51±1.90**	**54.95±1.73**	**1.00**
GAT	79.21±1.31	67.12±1.59	89.43±1.62	89.64±0.84	68.04±1.36	47.93±1.99	6.00
Res-GAT	79.72±1.94	67.19±1.91	89.92±0.86	91.40±0.74	**72.66±0.94**	55.98±2.12	2.83
Init-Res-GAT	79.67±1.71	66.84±2.52	89.61±1.02	91.53±1.06	69.89±1.81	51.29±1.42	4.17
JK-GAT	80.04±1.61	65.83±2.21	90.10±0.94	90.82±1.24	67.98±1.71	50.43±1.45	4.50
Dense-GAT	73.39±1.52	61.23±2.53	89.72±0.86	88.92±1.66	67.36±1.95	50.25±0.88	7.00
DropMessage-GAT	80.36±1.92	67.82±2.04	**90.67±1.68**	90.98±0.90	63.23±2.23	45.23±1.35	4.33
DeProp-GAT	76.00±2.08	61.16±3.50	87.34±1.42	89.76±1.52	64.23±3.22	46.29±3.24	8.00
Half-Hop-GAT	77.24±1.69	66.74±1.98	89.66±1.45	89.92±0.77	62.86±2.04	47.84±4.36	6.83
PSNR-GAT	**80.47±1.62**	**68.01±2.14**	90.38±1.21	**91.64±0.61**	72.24±1.69	**60.85±1.61**	1.33

5.4 Fully observed feature setting (RQ2)

The PSNR module can effectively address the performance degradation of GNNs at deeper layers, but can it further enhance the overall performance of the model? In this section, we comprehensively evaluate the PSNR module across a wider range of datasets. Under the fully observed feature setting, we set the number of layers to 2, 4, 8, 16, 32, 64, and conduct experiments on ten random splits for each dataset, taking the average accuracy as the final result for each layer. The results of all layers of the model can be found in Appendix A. To evaluate the overall performance, we record the best results of each model in all layers for each dataset in Table 3.

As can be observed from the Table 3, PSNR outperforms all baselines in most cases. For example, compared to the vanilla model, PSNR improves the test accuracy on the Squirrel dataset by 7.52% and 12.92%, respectively. Compared with the vanilla GCN and GAT, the proposed PSNR can significantly improve the performance under the fully observed feature setting.

5.5 Missing feature setting (RQ3)

When do we need the deeper GNN? Real-world data often contain missing features. In that scenario, previous research [33] has shown that deep GNNs can help improve performance. For the nodes with missing features, due to the lack of information, they need a deeper network to gather more neighborhood information, thereby achieving better node representations. However, deep GNNs face the issue of performance degradation. In this section, we examine if PSNR module can improve the performance of GNNs in the context of missing feature.

Consistent with [33, 18, 12], we evaluate the performance of GNNs on three datasets, Cora, Citeseer, and Pubmed, and remove their node features from validation and test sets. Under this setting, the test nodes need more propagation layers to reach the training nodes. We reuse the metrics that already reported in [12] for None, BatchNorm [11], PairNorm [33], DGN [34], DeCorr [12], and DropEdge [24]. For all residual-based models, the results are obtained by varying the number of layers in

8

Table 4: Test accuracy (%) on missing feature setting. The best results are in bold and the second best results are underlined.

	GCN						GAT					
	Cora		Citeseer		Pubmed		Cora		Citeseer		Pubmed	
Module	Acc	#K	Acc	#K	Acc	#K	Acc	#K	Acc	#K	Acc	#K
None	57.3	3	44.0	6	36.4	4	50.1	2	40.8	4	38.5	4
BatchNorm	71.8	20	45.1	25	70.4	30	72.7	5	48.7	5	60.7	4
PairNorm	65.6	20	43.6	25	63.1	30	68.8	8	50.3	6	63.2	20
DGN	76.3	20	50.2	30	72.0	30	75.8	8	54.5	5	72.3	20
DeCorr	73.8	20	49.1	30	73.3	15	72.8	15	46.5	6	72.4	15
DropEdge	67.0	6	44.2	8	69.3	6	67.2	6	48.2	6	67.2	6
Res	_76.8_	_8_	60.4	10	76.6	6	76.5	8	60.5	6	_76.9_	_8_
Init-Res	65.1	6	50.7	15	70.4	10	_77.1_	_8_	60.6	8	76.8	6
Dense	66.2	4	51.5	2	74.1	8	68.5	10	52.7	2	75.1	10
JK	75.5	30	60.4	8	_76.9_	_6_	77.0	10	60.3	4	76.8	6
DropMessage	75.5	10	_61.0_	_6_	74.6	6	76.5	6	_61.1_	_8_	76.6	6
DeProp	71.4	6	59.4	2	76.1	4	68.04	2	48.3	2	75.8	4
Half-Hop	73.7	8	59.48	6	76.5	6	76.0	20	59.6	4	_76.9_	_6_
PSNR	**77.3**	**20**	**61.1**	**15**	**77.0**	**30**	**77.9**	**8**	**61.9**	**15**	**77.3**	**10**

$\{2, 4, 6, 8, 10, 15, 20, 30\}$ and running five times for each number of layers. We select the layer #K that achieves the best performance and report its average accuracy. The results are reported in Table 4. By examining the results in Table 4, under the missing feature setting, the optimal number of layers to achieve the best performance is significantly higher than in the fully observed feature setting, demonstrating the importance of deep GNNs. And PSNR outperform other baselines in all cases through alleviating over-smoothing more effectively. Specially, on the Pubmed dataset, PSNR boost the accuracy of GCN and GAT by 40.6% and 38.8%, respectively.

5.6 Performance on large graphs (RQ4)

To validate the scalability of PSNR, we conducted additional experiments on three larger graph datasets i.e., Coauthor-Physics, Flickr and Ogbn-arxiv, to further validate the effectiveness and scalability of our method. Specifically, we selected the GCN backbone for our experiment. We report the performance of GCN and various residual methods on three datasets and the memory consumption on the largest dataset, Ogbn-arxiv. The experimental results are presented in Table 5, from which we observe that PSNR-GCN scales well and achieves the best results across all three large datasets. Meanwhile, in terms of memory consumption, PSNR is slightly higher than GCN and ResGCN, comparable to GCNII with the same initial residuals, and significantly lower than JKNet and DenseGCN. Regarding training time, PSNR is roughly at the same level as JKNet, and its time is shorter than that of DenseNet.

Table 5: Comparison of different methods across various datasets and memory consumption (MB) and training time (ms / epoch) on Ogbn-arxiv. The best performance for each dataset is in bold, while the second best is underlined.

Method	Phy	Flickr	Ogbn-arxiv	Memory	Time
GCN	95.32 ± 0.11	51.40 ± 0.33	64.37 ± 0.41	2421	30.10
ResGCN	95.61 ± 0.18	51.90 ± 0.16	$\underline{66.32 \pm 0.59}$	2463	36.06
GCNII	$\underline{95.90 \pm 0.14}$	46.18 ± 0.21	61.43 ± 1.63	2525	33.33
JKNet	95.88 ± 0.15	51.65 ± 0.31	60.46 ± 1.21	2921	40.05
DenseGCN	95.50 ± 0.12	$\underline{52.18 \pm 0.25}$	62.46 ± 1.58	3131	52.24
PSNR-GCN	$\mathbf{95.92 \pm 0.17}$	$\mathbf{52.47 \pm 0.16}$	$\mathbf{67.81 \pm 0.57}$	2539	42.93

6 Conclusion and Future Work

In this paper, we addressed the oversmoothing in Graph Neural Networks (GNNs) with a focus on residual methods. We revisit the oversmoothing from the perspective of overlapping neighborhood subgraphs, explaining why residual methods can alleviate it. Our analysis revealed that current residual methods often lack node adaptivity and struggle with information loss in high-order neighborhoods subgraphs. To overcome these limitations, we introduce the Posteriori-Sampling-based Node-Adaptive Residual Module (PSNR). This innovative module uses a graph encoder to learn the posterior distribution of residual coefficients for each node at different layers, enabling fine-grained,

9

node-adaptive neighborhood subgraph aggregation with minimal overhead. Extensive experiments confirmed that the PSNR module can effectively mitigate oversmoothing and improve performance, particularly in scenarios requiring deep networks. Despite the significant progress made by PSNR, training a deeper GNN remains challenging, prompting the need for further research in the future.

7 Acknowledgements

This work was supported by National Science and Technology Major Project (No. 2022ZD0115101), National Natural Science Foundation of China Project (No. 623B2086, No. U21A20427), Project (No. WU2022A009) from the Center of Synthetic Biology and Integrated Bioengineering of Westlake University and Integrated Bioengineering of Westlake University and Project (No. WU2023C019) from the Westlake University Industries of the Future Research Funding. Carl Yang was not supported by any funds from China.

References

[1] Mehdi Azabou, Venkataramana Ganesh, Shantanu Thakoor, Chi-Heng Lin, Lakshmi Sathidevi, Ran Liu, Michal Valko, Petar Velickovic, and Eva L. Dyer. Half-hop: A graph upsampling approach for slowing down message passing. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 1341–1360. PMLR, 2023.

[2] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[3] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of ICML*, volume 119, pages 1725–1735, 2020.

[4] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of NeurIPS*, pages 2224–2232, 2015.

[5] Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: novel architectures for graph neural networks motivated by partial differential equations. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 3836–3849, 2021.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *Proceedings of WWW*, pages 417–426, 2019.

[7] Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. Dropmessage: Unifying random dropping for graph neural networks. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 4267–4275. AAAI Press, 2023.

[8] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778, 2016.

10

[10] 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. *CoRR*, abs/2005.00687, 2020.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of ICML*, volume 37, pages 448–456, 2015.

[12] Wei Jin, Xiaorui Liu, Yao Ma, Charu C. Aggarwal, and Jiliang Tang. Feature overcorrelation in deep graph neural networks: A new perspective. In *Proceedings of SIGKDD*, pages 709–719, 2022.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*, 2017.

[15] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of ICLR*, 2019.

[16] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of ICCV*, pages 9266–9275, 2019.

[17] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of AAAI*, pages 3538–3545, 2018.

[18] Hua Liu, Haoyu Han, Wei Jin, Xiaorui Liu, and Hui Liu. Enhancing graph representations learning with decorrelated propagation. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1466–1476. ACM, 2023.

[19] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 338–348. ACM, 2020.

[20] Julian McAuley and Jure Leskovec. Image labeling on a network: using social-network metadata for image classification. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV 12*, pages 828–841. Springer, 2012.

[21] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *Proceedings of ICLR*, 2020.

[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*, pages 701–710, 2014.

[23] Chendi Qian, Gaurav Rattan, Floris Geerts, Mathias Niepert, and Christopher Morris. Ordered subgraph aggregation networks. In *NeurIPS*, 2022.

[24] Y. Rong, Wen-bing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *Proceedings of ICLR*, 2019.

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

[26] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *CoRR*, abs/2303.10993, 2023.

[27] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

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

11

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, P. Lio', and Yoshua Bengio. Graph attention networks. *Proceedings of ICLR*, 2017.

[31] Xinyi Wu, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[32] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of ICML*, pages 5449–5458, 2018.

[33] Lingxiao Zhao and L. Akoglu. Pairnorm: Tackling oversmoothing in gnns. *Proceedings of ICLR*, 2019.

[34] Kaixiong Zhou, Xiao Huang, Yuening Li, D. Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *Proceedings of NeurIPS*, 2020.

12

A   Fully Observed Node Classification

Table 6: Node classification accuracy (%) on GCN backbone. The best results across different layers are highlighted.

Datasets	Model	Layer 2	Layer 4	Layer 8	Layer 16	Layer 32	Layer 64
Cora	GCN	78.56±1.57	78.52±2.02	77.27±2.29	72.11±6.30	46.09±12.12	30.76±1.30
	ResGCN	79.39±1.12	79.80±1.53	80.10±1.46	80.11±0.98	71.27±3.65	36.48±7.53
	GCNII	77.40±1.43	76.47±1.96	76.39±1.88	76.15±1.70	75.76±1.83	76.27±1.46
	DenseGCN	74.94±2.46	67.06±2.02	67.58±2.66	65.08±3.71	59.98±2.30	34.37±5.69
	JKNet	78.44±2.10	79.42±1.48	79.36±2.23	77.18±2.17	77.33±2.88	51.58±7.29
	DropMessage-GCN	80.00±2.22	79.88±1.81	78.14±1.65	76.40±1.22	52.18±1.06	30.34±0.63
	DeProp-GCN	79.52±2.63	78.44±2.39	33.02±5.64	31.28±3.24	30.20±0.01	30.20±0.01
	Half-Hop-GCN	78.92±2.08	80.74±1.80	80.24±1.61	46.66±13.37	43.68±7.14	31.03±7.39
	PSNR-GCN	80.59±1.57	81.01±1.63	80.55±1.57	78.26±1.36	76.34±2.18	77.75±2.27
Citeseer	GCN	66.00±2.37	64.13±2.31	64.13±2.10	58.52±3.31	27.21±3.98	27.52±5.04
	Res-GCN	66.11±1.65	66.40±2.29	65.97±1.93	65.46±2.02	48.70±3.77	33.74±4.79
	GCNII	66.69±1.95	66.18±1.74	66.50±1.77	66.31±2.08	66.27±1.92	66.50±2.59
	Dense-GCN	62.54±2.56	58.99±3.74	54.35±5.42	50.10±5.24	49.09±4.03	31.43±5.54
	JKNet	65.49±2.32	64.40±2.28	64.18±3.07	63.77±1.87	60.88±3.53	29.66±7.51
	DropMessage-GCN	67.66±1.84	67.04±2.20	63.84±2.66	59.82±2.75	22.68±3.77	21.38±1.72
	Half-Hop-GCN	67.76±2.19	67.46±1.99	67.02±2.53	54.48±3.93	38.40±8.33	24.14±3.48
	DeProp-GCN	67.24±2.02	64.86±2.93	29.08±10.03	21.72±0.91	21.94±1.18	20.98±1.13
	PSNR-GCN	68.06±2.12	66.03±1.93	65.46±1.59	65.81±2.39	65.52±1.76	65.85±2.30
CS	GCN	90.19±0.83	88.81±0.62	86.93±0.83	84.47±0.95	71.40±4.93	36.74±4.57
	Res-GCN	90.86±0.99	90.63±0.96	89.97±0.90	88.40±0.90	85.01±1.53	64.39±3.86
	GCNII	90.34±0.78	90.02±0.81	90.07±0.77	90.09±0.94	90.17±0.52	89.93±0.70
	Dense-GCN	89.01±1.09	89.99±1.27	90.28±1.06	89.33±1.73	88.92±1.08	88.40±1.37
	JKNet	90.81±1.35	90.94±0.89	90.53±1.41	89.52±1.11	88.81±1.14	88.44±1.44
	DropMessage-GCN	90.64±2.34	89.18±1.76	89.38±1.52	85.62±7.01	87.94±6.18	84.44±1.13
	Half-Hop-GCN	90.30±1.83	89.30±2.23	88.98±1.81	82.98±1.88	54.60±4.55	25.36±10.27
	DeProp-GCN	90.98±2.04	89.10±2.23	71.24±4.18	72.34±2.26	52.60±0.03	22.60±0.01
	PSNR-GCN	91.23±1.00	90.70±1.49	90.26±1.17	90.26±0.98	90.52±1.02	90.30±0.88
Photo	GCN	91.22±0.73	90.49±0.76	88.10±1.02	80.05±4.25	54.80±8.00	46.57±9.64
	Res-GCN	91.31±0.85	90.97±0.78	90.71±0.78	85.98±2.36	65.42±6.75	56.15±10.43
	GCNII	91.02±0.93	90.98±0.92	91.02±0.70	90.99±0.76	91.20±0.81	91.46±0.84
	Dense-GCN	90.38±1.02	90.07±1.76	89.34±1.40	87.77±2.00	86.01±1.91	68.22±17.58
	JKNet	87.26±1.77	87.96±1.91	84.39±2.76	90.85±1.19	90.10±1.20	87.93±2.66
	DropMessage-GCN	91.40±0.90	90.22±1.19	87.57±2.74	87.82±1.21	86.12±1.05	80.40±1.07
	Half-Hop-GCN	52.00±17.28	67.48±24.16	89.32±1.17	83.66±3.02	64.40±4.30	39.00±9.42
	DeProp-GCN	91.38±1.33	89.50±5.31	78.12±3.66	81.86±9.36	87.74±4.19	84.43±2.31
	PSNR-GCN	91.44±0.82	91.20±1.03	91.39±0.76	91.11±0.68	91.51±0.69	91.49±0.88
Chameleon	GCN	67.70±0.17	56.30±2.28	48.37±1.56	46.81±2.06	39.62±2.13	33.27±1.86
	Res-GCN	70.28±1.14	70.70±1.54	69.18±0.78	51.64±2.11	42.61±3.60	31.42±2.73
	GCNII	69.45±1.63	69.36±1.13	69.62±1.61	69.23±1.21	68.87±2.16	69.69±1.36
	Dense-GCN	67.82±1.27	64.59±1.29	65.36±1.50	64.39±1.37	30.63±3.98	28.72±4.66
	JKNet	68.26±1.15	67.97±1.59	66.61±1.76	67.78±1.60	66.46±2.25	52.98±2.79
	DropMessage-GCN	65.91±2.43	60.50±2.84	47.29±2.00	42.66±1.60	34.56±1.61	31.74±1.56
	Half-Hop-GCN	60.07±2.34	55.72±3.12	52.64±1.99	43.19±2.55	36.26±2.12	29.34±2.20
	DeProp-GCN	61.51±1.40	58.25±2.22	46.46±2.18	31.88±2.52	30.81±2.07	33.36±0.96
	PSNR-GCN	71.51±1.90	70.74±2.24	71.40±1.83	66.00±1.97	65.82±2.02	69.67±4.59
Squirrel	GCN	47.43±1.31	41.68±1.80	37.43±1.35	33.17±1.22	31.99±1.14	29.83±1.98
	Res-GCN	51.72±1.63	52.43±1.72	50.38±1.90	38.35±1.56	26.30±2.06	22.66±1.01
	GCNII	46.13±1.59	45.28±1.59	45.71±2.08	45.37±1.72	45.06±1.50	45.32±1.85
	Dense-GCN	49.57±1.58	49.01±0.98	49.51±1.30	49.08±1.36	49.29±1.25	49.04±1.30
	JKNet	49.87±1.44	49.11±2.27	46.62±1.44	24.79±3.88	45.73±1.80	41.75±1.96
	DropMessage-GCN	45.71±1.76	38.68±2.02	26.67±3.06	25.71±1.15	23.77±1.18	20.06±0.23
	Half-Hop-GCN	42.78±1.01	41.87±1.36	37.84±1.68	31.60±2.12	26.36±1.66	21.07±1.53
	DeProp-GCN	44.07±1.07	40.98±1.29	33.38±1.70	28.33±0.96	28.42±0.54	29.98±2.78
	PSNR-GCN	54.95±1.73	54.13±1.41	50.68±1.45	50.22±6.32	50.06±0.97	50.24±1.58

13

Table 7: Node classification accuracy (%) on GAT backbone. The best results across different layers are highlighted.

Datasets	Model	Layer 2	Layer 4	Layer 8	Layer 16	Layer 32	Layer 64
Cora	GAT	79.21±1.31	78.83±1.78	41.48±2.84	30.20±0.00	30.20±0.00	30.20±0.00
	Res-GAT	79.42±1.46	79.72±1.94	78.88±1.69	78.71±2.13	77.80±2.25	30.22±0.12
	InitRes-GAT	79.32±1.65	79.39±1.45	79.00±1.97	79.16±1.68	79.67±1.71	79.42±1.97
	Dense-GAT	73.39±1.52	69.07±2.76	58.85±3.02	60.34±2.12	59.44±3.03	55.80±3.12
	JK-GAT	79.32±1.06	80.04±1.61	77.28±1.62	77.58±1.51	77.53±1.29	77.58±2.91
	DropMessage-GAT	80.06±2.50	80.36±1.92	79.20±0.22	76.54±2.49	52.54±1.02	30.17±0.06
	DeProp-GAT	76.00±2.08	69.06±3.94	30.22±0.06	31.20±0.02	30.20±0.01	30.20±0.01
	Half-Hop-GAT	77.20±1.86	76.98±1.42	77.24±1.69	75.84±1.68	70.50±4.69	30.20±0.10
	PSNR-GAT	80.47±1.62	80.22±0.98	79.96±1.69	80.02±1.64	79.43±1.78	79.69±1.29
Citeseer	GAT	67.12±1.59	65.14±3.02	21.90±2.40	21.10±0.00	21.10±0.00	21.10±0.00
	Res-GAT	67.19±1.91	65.65±2.21	63.26±2.31	63.47±1.18	62.82±2.92	22.66±1.31
	InitRes-GAT	66.84±2.52	64.15±2.44	65.80±1.89	64.89±2.65	64.69±2.49	65.11±2.65
	Dense-GAT	61.63±2.53	54.84±2.34	50.69±3.79	49.21±3.48	47.24±3.08	46.34±3.62
	JK-GAT	65.83±2.21	64.79±2.46	63.44±2.51	65.00±1.43	63.57±2.27	62.08±2.26
	DropMessage-GAT	67.82±2.04	66.64±2.51	65.10±3.32	58.20±2.55	21.90±1.64	20.90±2.04
	DeProp-GAT	61.16±3.50	51.52±11.25	23.20±2.77	21.12±1.76	21.00±1.67	20.50±0.92
	Half-Hop-GAT	66.74±1.98	66.70±2.58	63.12±3.49	60.00±3.30	45.98±4.63	24.32±4.46
	PSNR-GAT	68.01±2.14	65.61±2.05	66.46±2.14	65.48±2.59	66.36±2.25	65.50±2.35
CS	GAT	89.43±1.62	77.23±10.14	72.63±5.30	38.81±6.52	22.60±0.00	22.60±0.00
	Res-GAT	89.48±1.25	83.48±3.39	89.92±0.86	64.30±19.93	23.10±1.45	22.46±2.22
	InitRes-GAT	89.61±1.02	77.82±5.90	89.43±1.23	88.50±1.17	88.20±1.22	86.90±2.30
	Dense-GAT	89.72±0.86	88.81±0.89	88.28±0.72	87.01±1.52	86.47±1.13	87.23±1.78
	JK-GAT	89.77±1.02	90.10±0.94	90.03±1.26	89.68±1.41	89.66±1.04	89.86±1.12
	DropMessage-GAT	90.67±1.68	89.20±1.58	87.58±1.46	85.68±1.29	84.58±1.40	81.43±1.13
	DeProp-GAT	87.34±1.42	84.18±1.99	72.26±2.77	72.36±2.34	52.60±0.02	22.60±0.01
	Half-Hop-GAT	89.66±1.45	89.30±1.84	84.12±2.27	84.75±2.96	56.20±5.96	29.61±9.70
	PSNR-GAT	90.38±1.12	86.31±2.68	84.48±1.24	83.84±1.38	81.85±3.55	79.85±5.24
Photo	GAT	89.64±0.84	42.4±18.86	51.13±10.00	27.97±7.71	25.40±0.00	25.40±0.00
	Res-GAT	89.03±1.33	89.30±4.68	91.40±0.74	88.60±1.79	25.61±4.23	26.16±1.88
	InitRes-GAT	88.92±2.22	31.74±3.12	91.23±1.22	90.72±1.40	91.53±1.06	91.11±1.21
	Dense-GAT	88.92±1.66	88.10±1.50	87.03±2.22	86.79±1.97	86.24±2.65	85.63±2.15
	JK-GAT	87.55±1.39	90.82±1.24	90.35±1.61	90.30±1.37	90.29±1.49	90.07±1.48
	DropMessage-GAT	90.98±0.90	90.29±0.97	82.15±2.82	84.96±1.56	85.10±0.90	82.15±1.32
	DeProp-GAT	89.76±1.52	87.98±1.05	82.54±1.01	81.25±2.88	82.26±2.17	80.00±2.11
	Half-Hop-GAT	89.92±0.77	85.48±1.89	85.16±2.77	72.12±1.54	67.84±1.08	45.51±0.65
	PSNR-GAT	90.93±1.42	91.48±0.94	91.33±1.05	91.05±0.95	91.64±0.61	91.37±0.96
Chameleon	GAT	68.04±1.36	48.37±3.71	26.86±6.17	22.86±0.00	22.86±0.00	22.86±0.00
	Res-GAT	70.90±1.34	67.87±3.43	72.66±0.94	66.68±1.13	27.54±3.63	27.27±3.44
	InitRes-GAT	69.87±1.65	61.63±5.02	69.89±1.81	69.34±1.34	68.62±1.32	68.53±1.23
	Dense-GAT	67.36±1.95	66.87±2.07	66.21±2.03	61.95±2.14	62.32±2.10	61.71±1.69
	JK-GAT	67.98±1.71	65.71±1.40	66.75±1.88	66.99±2.23	66.24±1.42	66.31±1.56
	DropMessage-GAT	63.23±2.23	56.98±2.54	48.53±2.95	47.00±1.93	40.19±1.82	33.75±2.23
	DeProp-GAT	64.23±3.22	53.31±4.19	42.29±2.77	41.19±2.17	32.29±2.77	32.26±0.85
	Half-Hop-GAT	62.86±2.04	60.34±2.01	50.58±7.19	43.05±1.74	33.75±2.37	32.27±1.42
	PSNR-GAT	71.29±1.64	72.04±1.82	71.58±1.99	71.78±1.51	71.47±2.54	72.24±1.69
Squirrel	GAT	47.93±1.99	32.96±1.85	20.57±1.02	20.00±0.00	20.00±0.00	20.00±0.00
	Res-GAT	52.87±1.68	49.87±4.74	55.98±2.12	50.31±1.68	23.07±1.98	22.01±1.26
	InitRes-GAT	51.29±1.42	43.92±5.77	49.83±1.55	50.34±1.17	50.30±1.81	50.46±1.75
	Dense-GAT	50.25±0.88	49.57±1.80	46.71±1.69	47.83±1.70	47.54±1.58	46.87±1.65
	JK-GAT	50.43±1.45	44.41±2.33	49.08±0.79	49.31±1.99	48.87±1.74	49.56±1.31
	DropMessage-GAT	45.23±1.35	40.31±1.25	30.69±1.84	29.44±1.34	28.35±2.50	25.47±1.33
	DeProp-GAT	45.76±1.32	46.29±3.24	29.99±0.28	28.97±0.56	28.91±0.23	20.00±0.04
	Half-Hop-GAT	43.46±1.57	47.84±4.36	42.53±1.53	29.19±1.50	24.37±1.54	23.79±0.65
	PSNR-GAT	57.81±2.08	60.85±1.61	59.58±2.09	60.43±2.20	60.00±2.20	60.20±1.53

14

B   Residual Connection Defination

Common residual connection for GNNs and their corresponding GNNs are described below.

Res. Res is composed of multiple residual blocks containing few stacked layers. Taking the initial input of the n-th residual block as $\mathbf{X}_n$, and the stacked nonlinear layers within the residual block as $\mathbf{F}(\mathbf{X})$:

$$\mathbf{X}_{n+1} = \mathbf{F}(\mathbf{X}_n) + \mathbf{X}_n,$$

where residual mapping and identity mapping refer to $\mathbf{F}(\mathbf{X})$ and $\mathbf{X}$ on the right side of the above equation, respectively. Inspired by Res, Guohao Li & Matthias Müller(2019) proposed a residual connection learning framework for GCN and called this model ResGCN which can be simply described as follows:

$$\mathbf{H}_k = \sigma \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}_{k-1} \mathbf{W}_{k-1} \right) + \mathbf{H}_{k-1}.$$

InitialRes. InitialRes is proposed for the first time in APPNP, unlike Res that carries information from the previous layers, it constructs a connection to the initial representation $\mathbf{X}_0$ at each layer:

$$\mathbf{X}_{n+1} = (1 - \alpha)\mathbf{H}(\mathbf{X}_n) + \alpha\mathbf{X}_0,$$

where $\mathbf{H}(\mathbf{X})$ denotes the aggregation operation within one layer. InitialRes ensures that each node's representation retains at least an α-sized portion of the initial feature information. Correspondingly, APPNP can be formulated as:

$$\mathbf{H}_k = (1 - \alpha) \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}_{k-1} + \alpha\mathbf{H}.$$

Based on APPNP, GCNII introduces identity mapping from Res to make up for the deficiency in APPNP.

Dense. Dense proposes a more efficient way to reuse features between layers. The input is the outputs of all previous layers of the network and at each layer Dense concats them together:

$$\mathbf{X}_{n+1} = \mathbf{H}([\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_n]),$$

where $[\cdot]$ denotes the concatenation of the feature map for the output of layers 0 to n. Inspired by Dense, DenseGCN applies a similar idea to GCN, i.e., let the output of the k-th layer contains transformations from all previous GCN layers to exploit the information from different GCN layers:

$$\mathbf{H}_k = \mathbf{AGG}_{dense}(\mathbf{H}, \mathbf{H}_1, \dots, \mathbf{H}_{k-1}).$$

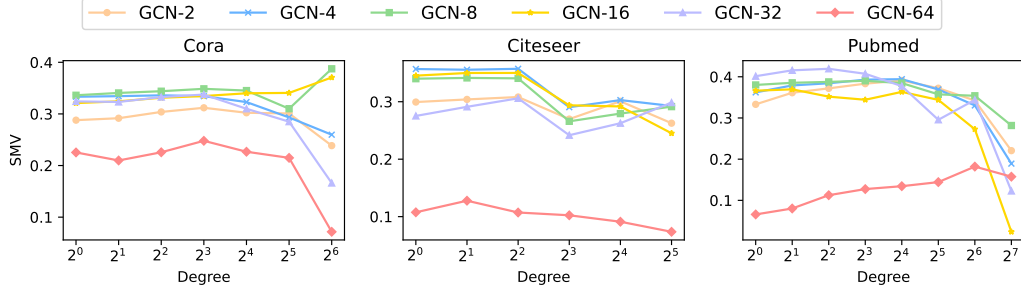
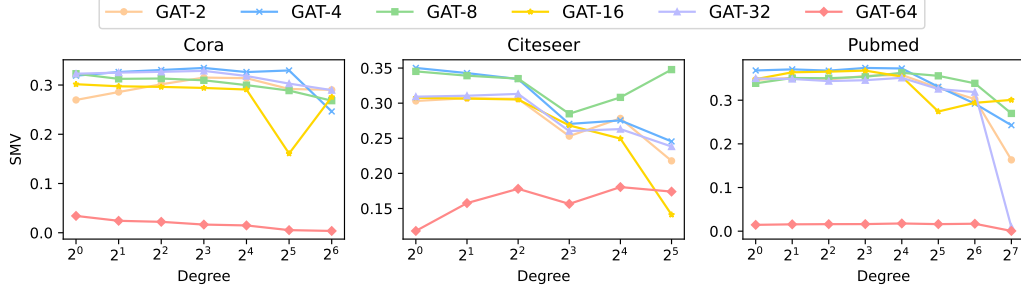
JK. At the last layer, JK sifts from all previous representations $[\mathbf{X}_1, \dots, \mathbf{X}_N]$ and combines them:

$$\mathbf{X}_{output} = \mathbf{AGG}(\mathbf{X}_1, \dots, \mathbf{X}_N).$$

The $\mathcal{AGG}$ operation includes concatenation, Maxpooling and LSTM-attention. When it is introduced to GNN, i.e., JKNet, can be formulated as:

$$\mathbf{H}_{output} = \mathbf{AGG}_{jk}(\mathbf{H}_1, \dots, \mathbf{H}_{k-1}).$$

15

C   SMV for Node Groups of Different Degrees

Result of GCN.

Result of GAT.

D   Derivation of the closed-form formulas in the table

ResGCN: We can write the recursive formula for ResGCN in the following form:

$$\mathbf{H}_k = (\mathbf{I} + \mathbf{N})\mathbf{H}_{k-1}. \tag{6}$$

In turn, the following form can be obtained by recursion:

$$\mathbf{H}_k = (\mathbf{I} + \mathbf{N})^k \mathbf{H}. \tag{7}$$

Using the binomial theorem, we can obtain the closed-form formula for ResGCN as follows:

$$\mathbf{H}_k = \sum_{j=0}^{k} \mathbf{C}_k^j \mathbf{N}^j \mathbf{H}. \tag{8}$$

APPNP: According to the recurrence formula of APPNP:

$$\mathbf{H}_k = \alpha \mathbf{H} + (1 - \alpha) \mathbf{N} \mathbf{H}_{k-1}. \tag{9}$$

To obtain the closed-form formula, we can add a term $\mathbf{T}$ to both sides of the equation:

$$\mathbf{H}_k + \mathbf{T} = (1 - \alpha)\, \mathbf{N} \mathbf{H}_{k-1} + \alpha \mathbf{H} + \mathbf{T}. \tag{10}$$

We aim to transform the equation into the following form:

$$\mathbf{H}_k + \mathbf{T} = (1 - \alpha)\, \mathbf{N} \left(\mathbf{H}_{k-1} + \mathbf{T}\right). \tag{11}$$

Then we need to make sure that there exists a very $\mathbf{T}$ that satisfies the following equation:

$$(1 - \alpha)\, \mathbf{N}\mathbf{T} = \alpha \mathbf{H} + \mathbf{T}, \tag{12}$$

which can be transformed into the following form:

$$((1 - \alpha)\, \mathbf{N} - \mathbf{I})\, \mathbf{T} = \alpha \mathbf{H}. \tag{13}$$

We can proof the following lemma:

16

Lemma 1 *Given that $\alpha \in (0,1)$, $(1-\alpha)\mathbf{N} - \mathbf{I}$ is invertible.*

Proof 1 *Proving that $(1-\alpha)\mathbf{N} - \mathbf{I}$ is invertible is equivalent to demonstrating that it does not possess an eigenvalue of 0. Consider the Rayleigh quotient of $(1-\alpha)\mathbf{N} - \mathbf{I}$:*

$$\frac{\mathbf{X}^T\left((1-\alpha)\mathbf{N} - \mathbf{I}\right)\mathbf{X}}{\mathbf{X}^T\mathbf{X}} = (1-\alpha)\frac{\mathbf{X}^T\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)\mathbf{X}}{\mathbf{X}^T\mathbf{X}} - 1. \tag{14}$$

From spectral graph theory, we can know the following equation holds:

$$\mathbf{X}^T\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{L}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)\mathbf{X} = \sum_{(v_i,v_j)\in\mathcal{E}}\left(\frac{\mathbf{X}_i}{\sqrt{d_i+1}} - \frac{\mathbf{X}_j}{\sqrt{d_j+1}}\right)^2 > 0. \tag{15}$$

We can decompose $\mathbf{L}$ into $\tilde{\mathbf{D}} - \tilde{\mathbf{A}}$, then we have:

$$\frac{\mathbf{X}^T\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{D}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)\mathbf{X}}{\mathbf{X}^T\mathbf{X}} - \frac{\mathbf{X}^T\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)\mathbf{X}}{\mathbf{X}^T\mathbf{X}} > 0, \tag{16}$$

which is equivalent to:

$$\frac{\mathbf{X}^T\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)\mathbf{X}}{\mathbf{X}^T\mathbf{X}} < \frac{\mathbf{X}^T\mathbf{I}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} = 1. \tag{17}$$

Combining Eq. 14 and Ineq. 17, we can obtain:

$$\frac{\mathbf{X}^T\left((1-\alpha)\mathbf{N} - \mathbf{I}\right)\mathbf{X}}{\mathbf{X}^T\mathbf{X}} < (1-\alpha) - 1 = -\alpha < 0. \tag{18}$$

Hence, 0 can't be the eigenvalue of $(1-\alpha)\mathbf{N} - \mathbf{I}$. Therefore, $(1-\alpha)\mathbf{N} - \mathbf{I}$ is invertible.

Since **Lemma 1** holds, we can derive the concrete form of $\mathbf{T}$:

$$\mathbf{T} = \alpha\left((1-\alpha)\mathbf{N} - \mathbf{I}\right)^{-1}\mathbf{H}. \tag{19}$$

Thus we can keep recurring from Eq. 11 and obtain the following equation:

$$\mathbf{H}_k + \mathbf{T} = \left((1-\alpha)\mathbf{N}\right)^k\left(\mathbf{H} + \mathbf{T}\right), \tag{20}$$

which also can be written as:

$$\mathbf{H}_k = \left((1-\alpha)\mathbf{N}\right)^k\mathbf{H} + \left((1-\alpha)\mathbf{N}\right)^k\mathbf{T} - \mathbf{T}. \tag{21}$$

For the second and third terms in Eq. 21, we write $(1-\alpha)\mathbf{N}$ as $(1-\alpha)\mathbf{N} - \mathbf{I} + \mathbf{I}$. We can use the binomial theorem to write $\left((1-\alpha)\mathbf{N}\right)^k$ as $\sum_{j=0}^{k}\left((1-\alpha)\mathbf{N} - \mathbf{I}\right)^j$, then Eq. 21 can be written as :

$$\mathbf{H}_k = \left((1-\alpha)\mathbf{N}\right)^k\mathbf{H} + \sum_{j=1}^{k}\left((1-\alpha)\mathbf{N} - \mathbf{I}\right)^j\mathbf{T}. \tag{22}$$

Bring in the specific form of $\mathbf{T}$ and further derive the closed-form formula of APPNP:

$$\mathbf{H}_k = \left((1-\alpha)\mathbf{N}\right)^k\mathbf{H} + \alpha\sum_{j=0}^{k-1}\left((1-\alpha)\mathbf{N} - \mathbf{I}\right)^j\mathbf{H} \tag{23}$$

$$= (1-\alpha)^k\mathbf{N}^k\mathbf{H} + \alpha\sum_{j=0}^{k-1}\sum_{i=0}^{j}(-1)^{j-i}(1-\alpha)^i\mathbf{N}^i\mathbf{H}. \tag{24}$$

17

E Derivation of the closed-form formula of PSNR-GCN

For each diagonal element $\Lambda_{k,ii}$ of Λ_k, it is trivial to obtain:

$$0 < \Lambda_{k,ii} < 1.$$

To derive the closed-form formula of PSNR-GCN, we need proof the following lemma first.

Lemma 2 *Set all the diagonal elements of Λ to satisfy $0 < \Lambda_{ii} < 1$, then $(\Lambda\mathbf{N} + \mathbf{I})$ is invertible.*

Proof 2 *Proving that $\Lambda\mathbf{N} + \mathbf{I}$ is invertible is equivalent to demonstrating that its determinant is not equal to 0. Because all the diagonal elements of Λ satisfy $0 < \Lambda_{ii} < 1$, then Λ is invertible. And due to*

$$|\Lambda\mathbf{N} + \mathbf{I}| = |\Lambda||\mathbf{N} + \Lambda^{-1}|. \tag{25}$$

Therefore, proving that its determinant is not equal to 0 is equivalent to demonstrating that $|\mathbf{N} + \Lambda^{-1}|$ is not equal to 0, and further equivalent to demonstrating that $\mathbf{N} + \Lambda^{-1}$ does not have an eigenvalue of 0.

Consider the Rayleigh quotient of $\mathbf{N} + \Lambda^{-1}$:

$$\mathbf{R}_1 = \frac{\mathbf{X}^T \left(\mathbf{N} + \Lambda^{-1}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}}. \tag{26}$$

Split Eq. 26, and we can derive:

$$\mathbf{R}_1 = \frac{\mathbf{X}^T\mathbf{N}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} + \frac{\mathbf{X}^T\Lambda^{-1}\mathbf{X}}{\mathbf{X}^T\mathbf{X}}. \tag{27}$$

The second term of Eq. 27 can be easily written as follows:

$$\frac{\mathbf{X}^T\Lambda^{-1}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} = \frac{\sum_{i=1}^N \Lambda_{ii}^{-1} x_i^2}{\sum_{i=1}^N x_i^2}.$$

Since $0 < \Lambda_{ii} < 1$, therefore $\Lambda_{ii}^{-1} > 1$, then

$$\frac{\mathbf{X}^T\Lambda^{-1}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} > 1. \tag{28}$$

For the first item, we write its specific form as follows:

$$\frac{\mathbf{X}^T\mathbf{N}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} = \frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}}. \tag{29}$$

From spectral graph theory, we know that the following formula holds:

$$\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \left(\mathbf{A} + \mathbf{D}\right) \tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X} = \sum_{(v_i,v_j)\in\mathcal{E}} \left(\frac{\mathbf{X}_i}{\sqrt{d_i+1}} + \frac{\mathbf{X}_j}{\sqrt{d_j+1}}\right)^2 > 0. \tag{30}$$

Further mathematically transforming this formula, we can get the following form:

$$\frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \left(\mathbf{A} + \mathbf{D}\right) \tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}}$$

$$= \frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \left(\tilde{\mathbf{A}} + \tilde{\mathbf{D}} - 2\mathbf{I}\right) \tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}} \tag{31}$$

$$= \frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}} + \frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{D}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}} - \frac{2\mathbf{X}^T\tilde{\mathbf{D}}^{-1}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} \tag{32}$$

$$= \frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right) \mathbf{X}}{\mathbf{X}^T\mathbf{X}} + 1 - \frac{2\mathbf{X}^T\tilde{\mathbf{D}}^{-1}\mathbf{X}}{\mathbf{X}^T\mathbf{X}} > 0. \tag{33}$$

18

Further, we get the following result:

$$\frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \right) \mathbf{X}}{\mathbf{X}^T \mathbf{X}} > \frac{2 \mathbf{X}^T \tilde{\mathbf{D}}^{-1} \mathbf{X}}{\mathbf{X}^T \mathbf{X}} - 1. \tag{34}$$

It is trivial to obtain:

$$\frac{2 \mathbf{X}^T \tilde{\mathbf{D}}^{-1} \mathbf{X}}{\mathbf{X}^T \mathbf{X}} = \frac{2 \sum_{i=1}^{N} (\mathbf{d}_i + 1)^{-1} \mathbf{x}_i^2}{\sum_{i=1}^{N} x_i^2} > 0. \tag{35}$$

Combining Eq. 27, Ineq. 28, Ineq. 34 and Ineq. 35, we can get the following inequality:

$$\frac{\mathbf{X}^T \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} + \Lambda^{-1} \right) \mathbf{X}}{\mathbf{X}^T \mathbf{X}} > 0. \tag{36}$$

It can be obtained that the eigenvalue of $\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} + \Lambda^{-1}$ is greater than 0, so 0 is not an eigenvalue of it. Further, $\Lambda \mathbf{N} + \mathbf{I}$ is invertible.

Now, we derive the closed form of the formula. Given the following recursive formula:

$$\mathbf{H}_k = \mathbf{H}_1 + \Lambda_{k-1} \left(\mathbf{H}_1 - \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{H}_{k-1} \right), \tag{37}$$

where $\mathbf{H}_1 = \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{H}$, $\mathbf{\Lambda}_k = diag\{\lambda_k^{(1)}, ... \lambda_k^{(n)}\}$, $\lambda_k^{(i)} \sim Sigmoid(\mathcal{N}(\alpha_k^{(i)}, \beta_k^{(i)^2}))$. After mathematical transformation, Eq. 37 can be written as:

$$\mathbf{H}_k = (\mathbf{I} + \Lambda_{k-1}) \mathbf{H}_1 - \Lambda_{k-1} \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{H}_{k-1}. \tag{38}$$

Set $\mathbf{N} = \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2}$, then Eq. 38 can be abbreviated as:

$$\mathbf{H}_k = (\mathbf{I} + \Lambda_{k-1}) \mathbf{H}_1 - \Lambda_{k-1} \mathbf{N} \mathbf{H}_{k-1}. \tag{39}$$

We try to modify Eq. 39 to the form that is more suitable for obtaining the closed form of the formula:

$$\mathbf{H}_k + \mathbf{M}_{k-1} = -\Lambda_{k-1} \mathbf{N} \left(\mathbf{H}_{k-1} + \mathbf{M}_{k-1} \right). \tag{40}$$

To verify whether there exists such $\mathbf{M}$ that satisfies the equation, we need to solve the following equation:

$$-\Lambda_{k-1} \mathbf{N} \mathbf{M}_{k-1} = (\mathbf{I} + \Lambda_{k-1}) \mathbf{H}_1 + \mathbf{M}_{k-1}, \tag{41}$$

which is equivalent to the following form:

$$-\left(\Lambda_{k-1} \mathbf{N} + \mathbf{I}\right) \mathbf{M}_{k-1} = (\mathbf{I} + \Lambda_{k-1}) \mathbf{H}_1. \tag{42}$$

Based on the definition, all the diagonal elements of Λ_k satisfy $0 < \lambda_k^{(i)} < 1$, so according to **Lemma 2**, $(\Lambda_{k-1} \mathbf{N} + \mathbf{I})$ is invertible. Then $\mathbf{M}_{k-1} = -\left(\Lambda_{k-1} \mathbf{N} + \mathbf{I}\right)^{-1} (\mathbf{I} + \Lambda_{k-1}) \mathbf{H}_1$, which means such $\mathbf{M}_{k-1}$ that we require exists.

First, we perform the following mathematical transformation on Eq. 40:

$$\mathbf{H}_k + \mathbf{M}_{k-1} = -\Lambda_{k-1} \mathbf{N} \left(\mathbf{H}_{k-1} + \mathbf{M}_{k-2} + \mathbf{M}_{k-1} - \mathbf{M}_{k-2} \right), \tag{43}$$

which can be split into the following form:

$$\mathbf{H}_k + \mathbf{M}_{k-1} = -\Lambda_{k-1} \mathbf{N} \left(\mathbf{H}_{k-1} + \mathbf{M}_{k-2} \right) + \left(-\Lambda_{k-1} \mathbf{N} \right) \left(\mathbf{M}_{k-1} - \mathbf{M}_{k-2} \right). \tag{44}$$

Let $\tilde{\mathbf{N}}_{k-1}$ denotes $-\Lambda_{k-1} \mathbf{N}$, so the formula can be simply written as:

$$\mathbf{H}_k + \mathbf{M}_{k-1} = \tilde{\mathbf{N}}_{k-1} \left(\mathbf{H}_{k-1} + \mathbf{M}_{k-2} \right) + \tilde{\mathbf{N}}_{k-1} \left(\mathbf{M}_{k-1} - \mathbf{M}_{k-2} \right). \tag{45}$$

We first use Eq. 40 to recurse once, then derive the following formula:

$$\mathbf{H}_k + \mathbf{M}_{k-1} = \tilde{\mathbf{N}}_{k-1} \tilde{\mathbf{N}}_{k-2} \left(\mathbf{H}_{k-2} + \mathbf{M}_{k-3} \right) + \tilde{\mathbf{N}}_{k-1} \left(\mathbf{M}_{k-1} - \mathbf{M}_{k-2} \right). \tag{46}$$

By analogy, continuing to split and iterate, we can get the closed form of the output of the k-th layer:

$$\mathbf{H}_k = \sum_{i=2}^{k-1} \prod_{j=i}^{k-1} \tilde{\mathbf{N}}_j \left(\mathbf{M}_i - \mathbf{M}_{i-1} \right) + \prod_{i=1}^{k-1} \tilde{\mathbf{N}}_i \left(\mathbf{H}_1 + \mathbf{M}_1 \right) - \mathbf{M}_{k-1}. \tag{47}$$

19

F Comparing with other subgraph-based methods

It is worth noting that a lot of previous work mentions "subgraph". However, our approach is fundamentally different from these approaches.

Relation with other subgraph-based methods. While there are existing works [2, 23] related to subgraphs, they primarily focus on graph classification tasks, aiming to learn representations of entire graphs. Given the limited capacity of GNNs to effectively represent the entire graphs, these subgraph-based approaches employ various strategies to leverage information from multiple subgraph structures within the overall graph to improve the representation.

In contrast, our focus is on node-level tasks, specifically, enhancing the representations of individual nodes. In our context, we naturally refer to neighborhood subgraphs as different orders of ego-networks centered on a node (which can be regarded as subgraphs of the entire graph). Our work uncovers the relationship between these subgraphs and over-smoothing, as well as how to utilize them to enhance node representations. This distinction fundamentally sets our approach apart from other subgraph methods.

G Experiment Setup

G.1 Details of Datasets

The dataset statistics are shown in Table 8, and details on dataset splits are summarized as follows:

Experiment 5.3 & 5.4. For Cora, Citeseer, Coauthor-CS, Amazon-Photo, we randomly select 20 nodes per class for training set, 500 nodes for validation and 1000 nodes for testing. For Chameleon and Squirrel, we randomly divide each class's nodes into 60%, 20%, and 20% as the train, validation, and test sets, respectively.

Experiment 5.5. We follow the widely used semi-supervised setting in [14].

Experiment 5.6. For larger datasets, We randomly divide each class's nodes into 20%, 20%, and 60% as the train, validation, and test sets, respectively.

Table 8: Dataset statistics of real-world datasets.

	Cora	Citeseer	Pubmed	Amazon-Photo	Ogbn-arxiv	Chameleon	Squirrel	Coauthor-CS	Coauthor-Phy	Flickr
#Nodes	2708	3327	19717	7650	169343	2277	5201	18333	34493	89250
#Edges	5429	4732	119081	126842	1166243	36101	217073	81894	247962	899756
#Features	1433	3703	500	745	128	2325	2089	6805	8415	500
#Classes	7	6	3	8	39	5	5	5	10	7

G.2 Parameter Settings

We summarized the hyperparameters used in different experiments in Table 9.

Table 9: Hyperparameters for experiments

Experiment	Backbone	Learning Rate	Dropout	Weight Decay	Hidden State	Attention Head	Max Epoch	Early Stopping Epoch
Experiment 5.3 & 5.4	GCN	{0.01, 0.001}	0.5	0.0005	128	-	500	100
	GAT	{0.01, 0.001}	0.5	0.0005	64	3	500	100
Experiment 5.5	GCN	{0.01, 0.001}	0.5	0.0005	128	-	1000	1000
	GAT	{0.01, 0.001}	0.5	0.0005	32	1	1000	1000
Experiment 5.6	GCN	{0.01, 0.001}	0.5	0.0005	128	-	500	100

H. The Analysis of PSNR Residual Coefficients

We conducted an empirical study using an 8-layer PSNR-GCN trained on the Cora dataset to obtain the best-performing model. We saved the mean and standard deviation of the learned residual coefficient distribution for each layer. Nodes were evenly divided into four groups based on their degree, with each group containing a similar number of nodes, and the average mean and standard

20

deviation for each group across different layers are reported in Table 10. The following observations can be obtained from the table: the mean residual coefficient increases with the number of layers, suggesting that PSNR effectively retains high-order subgraph information. In certain layers, the variance of the residual coefficients rises, indicating that added randomness helps mitigate information loss in higher-order subgraphs. In the shallow layers, the mean values show no significant differences across node degrees; however, in deeper layers, nodes with higher degrees tend to exhibit lower mean values, indicating that these nodes retain more initial information due to significant subgraph overlap. All of these observations align with our expectations, illustrating how residual coefficients adapt based on node degree and layer depth.

Table 10: Summary of means and standard deviations for different layers

Layer	Mean Group 1	Mean Group 2	Mean Group 3	Mean Group 4	Std Group 1	Std Group 2	Std Group 3	Std Group 4
Layer 0	0.0043	0.0049	0.0063	0.0099	0.0002	0.0001	0.0002	0.0001
Layer 1	0.0009	0.0017	0.0013	0.0016	0.3102	0.3120	0.3130	0.2410
Layer 2	0.0000	0.0000	0.0003	0.0008	0.0480	0.0431	0.0446	0.0170
Layer 3	0.5582	0.6017	0.5972	0.4979	0.0084	0.0118	0.0105	0.0080
Layer 4	1.4898	1.5238	1.5212	1.3478	8.3354e-05	5.4541e-05	0.0001	0.0002
Layer 5	2.4080	2.3974	2.3981	2.1414	0.0062	0.0081	0.0078	0.0059
Layer 6	5.2347	5.3148	5.2070	3.8177	0.0046	0.0089	0.0075	0.0069
Layer 7	11.0787	11.1253	10.9247	8.7238	0.1642	0.2089	0.2128	0.0677

I. Comparison of Experimental Results of Different GraphEncoders.

In practice, in addition to SAGE, encoders such as GAT and GCN can also be utilized. To provide further insight, we have included results for different encoders on the classical semi-supervised node classification task. The results are summarized in the table blow.

Table 11: Different GraphEncoder performance for SSNC task (layer 2)

Graph Encoder	Cora	Citeseer	CS	Photo	Chameleon	Squirrel
GCN	**80.98±1.60**	68.46±2.28	90.52±0.82	**91.56±0.74**	**72.02±1.60**	56.14±1.51
GAT	80.89±1.63	**68.77±1.89**	90.61±0.89	91.18±0.92	71.97±1.28	**56.24±1.11**
SAGE	80.59±1.57	68.06±2.12	**91.23±1.00**	91.44±0.82	71.51±1.90	54.95±1.73

Table 12: Different GraphEncoder performance for SSNC task (layer 4)

Graph Encoder	Cora	Citeseer	CS	Photo	Chameleon	Squirrel
GCN	81.65±1.70	**68.11±1.24**	90.66±0.70	91.14±0.90	**71.58±2.07**	56.34±1.48
GAT	**82.21±1.41**	67.96±1.20	90.57±0.89	91.17±0.81	71.29±1.75	**56.50±1.45**
SAGE	81.01±1.63	66.03±1.93	**90.70±1.49**	**91.20±1.03**	70.74±2.24	54.13±1.41

The results indicate that each encoder has its strengths and performs differently across various datasets, demonstrating superior performance compared to the baseline and emphasizing the potential of PSNR.

21

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist",**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our paper's contributions and scope are presented in the abstract and introduction accurately.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

22

Justification: Limitations of our work are discussed in Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide the proofs of all theorems and the derivations of all formulas in Section 4.2, Appendix D and Appendix E, respectively.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the details of experiments in Appendix G, which can ensure the reproducibility of our experimental result.

   Guidelines:

23

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We only used public datasets and the anonymous link to the code is provided in the Abstract.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

24

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details of training and test in Appendix G.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The standard deviation of the experimental results is reported in most of the tables.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the details about the compute resources in Section 5.2

   Guidelines:

   - The answer NA means that the paper does not include experiments.

25

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Our research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We have discussed the potential positive societal impacts in section 6. It has no negative societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

26

Answer: [NA]

Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All the original owners of assets (e.g., code, data, models) used in this paper are properly credited, and the license and terms of use are explicitly mentioned and properly respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Yes, we provide the anonymous link to the code.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

27

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our work does not involve any crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

28