This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

Another Perspective of Over-smoothing: Alleviating Semantic Over-smoothing in Deep GNNs

Jin Li, Qirong Zhang, Wenxi Liu, Antoni B. Chan, Yang-Geng Fu

***Abstract*—Graph neural networks (GNNs) are widely used for analyzing graph-structural data and solving graph-related tasks due to their powerful expressiveness. However, existing off-the-shelf GNN-based models usually consist of no more than three layers. Deeper GNNs usually suffer from severe performance degradation due to several issues including the infamous "over-smoothing" issue, which restricts the further development of GNNs. In this paper, we investigate the over-smoothing issue in deep GNNs. We discover that over-smoothing not only results in indistinguishable embeddings of graph nodes, but also alters and even corrupts their semantic structures, dubbed semantic over-smoothing. Existing techniques, e.g., graph normalization, aim at handling the former concern, but neglect the importance of preserving the semantic structures in the spatial domain, which hinders the further improvement of model performance. To alleviate the concern, we propose a cluster-keeping sparse aggregation strategy to preserve the semantic structure of embeddings in deep GNNs (especially for spatial GNNs). Particularly, our strategy heuristically redistributes the extent of aggregations for all the nodes from layers, instead of aggregating them equally, so that it enables aggregate concise yet meaningful information for deep layers. Without any bells and whistles, it can be easily implemented as a plug-and-play structure of GNNs via weighted residual connections. Last, we analyze the over-smoothing issue on the GNNs with weighted residual structures and conduct experiments to demonstrate the performance comparable to the state-of-the-arts.**

***Index Terms*—Deep graph neural networks, over-smoothing, node classification, clustering, sparse aggregation strategy.**

I. Introduction

As an emerging and promising technology, graph neural networks (GNNs) [1] pave a path to handle graph-like data structures thanks to their efficiency and powerful expressiveness, and thus draw increasing attention to various applications on social network [2], recommendation systems [3], point clouds [4], drug discovery [5], etc.

Neural techniques for graph analysis have been in existence since the previous century. Motivated by [6] applying neural networks to directed acyclic graphs, Gori et al. [7] first introduces the notion of Graph Neural Networks (*i.e.*, GNNs), which is further followed by [8] and [9]. These recurrent graph neural networks (RecGNNs) [1] attempt to learn node embeddings by iteratively propagating neighbor information until reaching a stable equilibrium. Some other GNNs aim to design some graph convolution in either the spectral or spatial domain. Those spectral-based methods try to learn a graph filter that adaptively preserves low- and high-frequency signals [1]. Yet, they suffer from high computational complexity when performing eigen-decomposition for graph Laplacian. On the other hand, graph convolutional neural network (GCN) [10] is proposed to linearly approximate universal filters of graph signals, which also has good interpretability in the spatial domain. Furthermore, as a typical kind of spatial-based model, message-passing neural networks (MPNNs) [11] generalize GCN and propose a framework to iteratively aggregate messages from the first-order neighbors of each node via some generalized operators. However, most existing GNN-based models consist of no more than three layers, whereas deeper GNNs usually suffer from severe performance degradation. There are many issues resulting in the performance degradation of deep GNNs, *e.g.*, over-fitting [12], over-smoothing [12, 13], memory limitation [14, 15], time consumption [14], difficulty in optimization [16] (*e.g.*, gradient vanishing or exploding, numerical instability [17]), and over-squashing [18].

Amongst these issues, over-smoothing is the most important one and has attracted more and more attention in recent years. On the deeper layers of GNNs, they tend to produce indistinguishable embeddings for different graph nodes, thus leading to an over-smoothing effect. Techniques are incorporated to alleviate over-smoothness by separating the embeddings, which can be mainly categorized as: 1) skip connections [19, 20]; 2) graph normalization [16, 21]; 3) random dropping [12, 22, 23]; 4) their combinations [17]. Yet, we observe that, in deep layers, GNNs not only produce too clustered embeddings to discriminate, but also tend to change or corrupt their underlying semantic structures, dubbed *semantic over-smoothing*. In particular, as the embeddings of graph nodes become too close in deep layers, although normalization tricks can be applied to expand their mutual distances, we show that the tricks actually cannot prevent the semantic structures from corruption based on our theoretical analysis and experiments. Thus, semantic over-smoothing is neglected by all existing anti-over-smoothing techniques, so that the performance of GNNs can hardly be further strengthened. More importantly, studying semantic over-smoothing provides an intuitive way to effectively handle the over-smoothing issue, which is interpretable in the spatial domain.

To intuitively illustrate our motivation, we plot the scattering of embeddings produced by different stages with different methods on a synthetic graph in Fig. 1, where: 1) Line 1 shows that GCN easily suffers from the numerical over-smoothing issue, since the embeddings tend to approach too close (at a point or a very short segment); 2) Line 2 shows that GCN with isotropic normalization can only delay this issue with only constant numerical scaling (a longer segment); 3) Line 3 utilizes an-isotropic normalization, preventing embeddings from converging to a line and keeping diversity to the fullest;

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

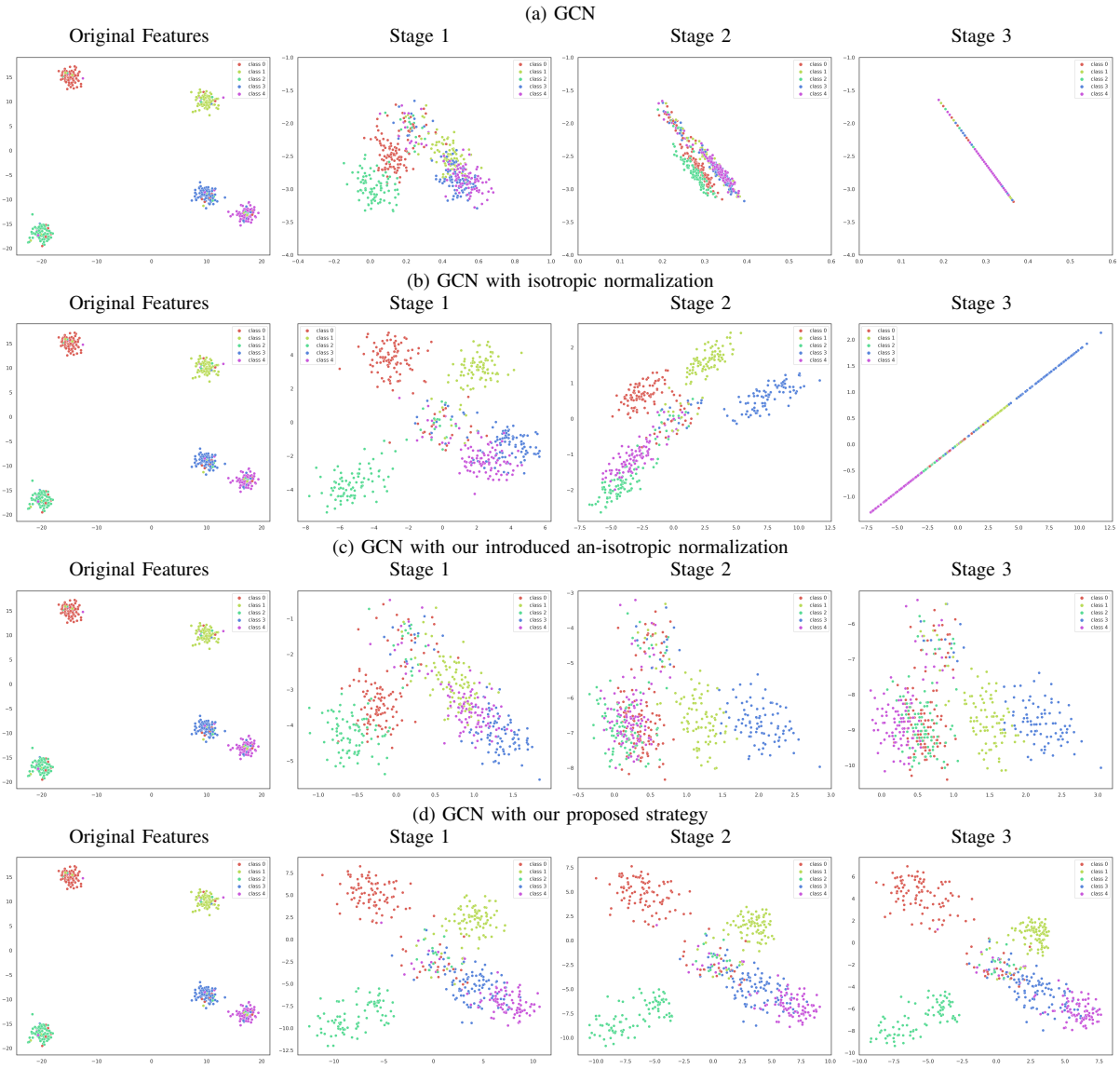

Fig. 1. Illustration of over-smoothing on synthetic graph data. From the top to the bottom row, we visualize the evolution of data as layers go deeper using GCN, GCN with isotropic normalization, GCN with our introduced an-isotropic normalization (see Sec. IV-B1 and supplementary material), and GCN with our proposed strategy, respectively. The specific data generation method and the specific numbers of layers of every model are provided in Sec. III of the supplementary material. **(a)-(b)** The former two models easily degrade to over-smoothing. **(c)** GCN with an-isotropic normalization alleviates numerical over-smoothing, yet it cannot preserve its structural information and thus suffers from semantic over-smoothing where different clusters shift and blend with each other. **(d)** In contrast, with the same number of layers as other models, our strategy manages to maintain the semantic structure, effectively alleviating semantic over-smoothing.

however, some semantic clusters in terms of ground-truth labels significantly blend and mix with each other (a more essential reason for performance degradation of downstream linear classifiers), which gives visual observation of semantic over-smoothing and can be hardly discovered by Line 1 and 2 in their limits; 4) With the aid of our proposed strategy, semantic clusters can naturally preserved with also relatively reduced intra-cluster variances compared to Line 3, which is definitely beneficial. Based on these visualizations, we can provide a comprehensive understanding for the semantic over-smoothing issue. Specifically, it is exhibited as the blending and collapsing of global semantic structure (*e.g.*, clustering, distribution, or geometrical relationships), even when the isotropic or more powerful an-isotropic normalization techniques are applied. The reason may be attributed to that the knowledge abstracted from the node embeddings of the input graph are encoded into semantic structure, far more important than the specific numeric values of those embeddings themselves. Thus, those normalization techniques that consider solving the over-smoothing issue only from a numerical perspective, *e.g.*, normalizing the average mutual numerical distances or keeping the comprehensive shape, are not enough to preserve the semantic structure (as shown on the 2-th and 3-th row of Fig. 1). Although shallow models perform worse than the state-of-the-art deep models, they can capture the basic semantic structure. Thus, in their deep layers, the slight and sparse structure-based refinement rather than remodeling can naturally and intuitively alleviates both numerical and semantical over-smoothing issues. This

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

evidently indicates the roles that deeper layers play, leading to the simple yet effective design of the proposed method presented in Sec. V. In Sec. IV-B, we show more observation and analysis on semantic over-smoothing, including a more detailed description, comparison with the numerical issue, how it was discovered, the potential cause, and why studying it is significant.

To alleviate the semantic over-smoothing issue, we propose a simple and effective cluster-keeping sparse aggregation strategy to adaptively choose the nodes to aggregate and how much information they will devote to aggregation. This strategy enables deep GNNs to aggregate concise yet meaningful information while relieving the semantic over-smoothing issue, and it can be easily implemented as a plug-and-play structure of GNNs via weighted residual connections. Last, we conduct a theoretical analysis on a GCN with weighted residual structures, which reflects the motivation of sparse aggregation, as well as the convergence property and expressiveness of our model against semantic over-smoothing. We also perform experiments in which our proposed strategy is incorporated with the vanilla GCN [10] and Simple Graph Convolution (SGC) [24], respectively, and they achieve comparable or even better performance against the existing state-of-the-arts in public benchmarks.

Overall, the contributions of this work are summarized as follows:

- We discover that over-smoothing not only results in indistinguishable embeddings for different graph nodes, but also alters and even corrupts their semantic structures in deep layers, called semantic over-smoothing. This phenomenon is neglected by previous methods, and we demonstrate that studying it provides a novel and intuitive way to handle the over-smoothing effect in deep GNNs.
- We propose a cluster-keeping sparse aggregation strategy to effectively alleviate semantic over-smoothing and enable deep GNNs to aggregate sparse yet crucial information on deep layers.
- A theoretical analysis on a GCN with weighted residual structures is conducted, which reflects the motivation of sparse aggregation, as well as the convergence property and expressiveness of our model against semantic over-smoothing.
- In experiments, we show that our proposed strategy can be easily incorporated with the vanilla GCN and SGC, and achieves comparable or even better performance against the existing state-of-the-art methods in public benchmarks.

The remainder of this paper is organized as follows: In Sec. II-A and Sec. III, we introduce some related works and preliminaries, respectively. Then the intuition and motivation of semantic over-smoothing issue are presented in Sec. IV. After that, we will detailedly introduce our proposed clustering-keeping sparse aggregation strategy in Sec. V. Some theoretical discussion on semantic over-smoothness and the structure of GNN with residual connections are given in Sec. IV-B1 and V-D. Consequently, in Sec. VI extensive experiments are conducted to show the potential and the effectiveness of our method. Finally, we summarize this paper and give the limitation of our method in Sec. VII. Please refer to the supplementary material for more theoretical analysis, interpretations, experimental details, and visualizations.

II. Related Works

A. Graph Neural Networks

GNNs are widely used for various graph-related tasks including texts [25], images [26], traffic [27], generation [28], molecule [29], few-shot learning [30], electroencephalogram (*i.e.*, EEG) [31], and Internet of Things (*i.e.*, IoT) [32]. They also achieve state-of-the-art results [1, 33, 34]. In addition to some earlier recurrent graph neural networks (RecGNNs) [8, 9], there are two kinds of convolution-based GNNs: spectral-based and spatial-based, which have different motivations and interpretability in their designs for graph convolutions. From the perspective of graph signal processing, spectral-based approaches introduce some graph filters to remove noises in graph signals. Spatial GNNs directly define graph convolutions via information propagation in local environments (*e.g.*, neighborhoods). As a pioneering spectral-based model inspired by CNN, SpectralCNN [35] obtains Laplacian eigen-space (*i.e.*, eigen-vectors and eigen-values) via direct eigen-decomposition, which is followed by many models [10, 24, 36, 37, 38]. Graph Convolutional Network (GCN) [10], a significant model interpreted as both low-pass filters and neighborhood aggregations respectively in the spectral and spatial domain, has inspired many recent spatial-based models such as GAT [39], GIN [40], GeoGNN [41], etc. Hamilton et al. [42], Gao et al. [43], and Zeng et al. [15] develop GCN to solve large-scale problems. Some GNNs can be applied to handle heterogeneous graphs (*e.g.*, [44, 45, 46]) and make it possible to perform recommendation tasks [47, 48] and deal with knowledge graph [49, 50]. Other aspects of GNNs are also explored, *e.g.*, heterophily [51], expressiveness [52, 53, 54, 55, 56], interpretability [57, 58], pretraining [59], self-supervised learning [60]. The proposed method pertains to spatial GNNs and can be applied to nearly all of them, including those aforementioned.

B. Deep GNNs and the over-smoothing issue

Although GNNs have lots of variants, most of them are shallow networks that restrict their expressiveness and limit distant message passing. To make GNNs deep, prior works introduce modifications or tricks to overcome or alleviate the over-smoothing issue. To solve the over-smoothing issue, known methods can be categorized into three classes: skip connections, graph normalization (*e.g.*, Pairnorm [21], Nodenorm [61], Meannorm [16]) and random dropping (*e.g.*, DropEdge [12], DropNode [22], DropMessage [23]). Skip connection-based models originate from ResNet [62] and contain common residual connection (from last layer) [63, 64], initial connection (from the first layer) [19, 20, 63], dense connection (from every layer) [13, 65], and jump connection (from every layer to the last layer only) [13, 66]. Chen et al. [17] thoroughly surveys the relevant methods.

Also, we notice that there exist some works analysing or designing deep GNNs from other perspectives or domains.

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

Inspired by Scarselli et al. [67], Tiezzi et al. [68] proposes to learn deep GNNs via multi-layer constrained optimization in the Lagrangian framework, simultaneously learning the node states and their transition function, whose convergence can be implicitly expressed by some constraints. Keriven [69] theoretically analyses the over-smoothing issue for simplified linear GNNs with Mean aggregations from a perspective of probabilistic latent space modeling. Deep GNNs can also be applied to other domains. For example, to solve general visual tasks (*e.g.*, image recognition and object detection), ViG [70] makes a prediction based on the graph representation obtained from a graph constructed by splitting an image into patches and connecting the nearest ones. Besides, ViG can easily go deeper via normalization similar to Meannorm [16] and some customized feature transformations, which can effectively capture irregular and complex objects compared to traditional widely-used CNNs and transformers.

In contrast with previous works, we discover that over-smoothing issue not only leads to indistinguishable embeddings, but also corrupts the semantic structure. Thus, we propose a novel and intuitive perspective by handling semantic over-smoothing as a more interpretable way to tackle the over-smoothing issue in the spatial domain. More specifically, our method falls into the first class (*i.e.*, skip connections-based methods). However, we do not focus on exploring new GNN structures (*i.e.*, looking for new kinds of connections) as prior works (*e.g.*, GCNII [19] and JKNet [66]). Our goal is to alleviate the over-smoothing issue via some flexible sparse aggregation strategies considering only the simplest structure (*i.e.*, GNN with commonly used residual connections).

III. Preliminaries

a) Notation: Let $G = (V, E)$ be an undirected graph with node set V and edge set E, where $n = |V|, m = |E|$ represent the number of its nodes and edges respectively. We denote by $A \in \{0,1\}^{n\times n}$ and $X \in \mathbb{R}^{n\times d}$ its adjacency matrix and feature matrix where $x_i = X_{i,:} \in \mathbb{R}^d$ represents the feature vector of node i. For semi-supervised node classification task, every node in G has a ground-truth label $y_i \in \mathbb{N}$. We denote $I_n \in \mathbb{R}^{n\times n}$ as identity matrix of size n, and $\mathbf{1} \in \mathbb{R}^{n\times 1}$ is an all-one vector.

b) Graph convolutional network: GCN in spatial domain can be described as several stacked aggregation operations, linear transformations, and non-linear activation functions.

$$\begin{aligned} H^{(l+1)} &= \sigma\left(\hat{A}H^{(l)}W^{(l)}\right) \in \mathbb{R}^{n\times d}, \quad \forall l \in [0, L), \\ H^{(0)} &= X, \end{aligned} \tag{1}$$

where L is the number of layers and $\sigma(\cdot)$ denotes non-linear activation function (*e.g.*, ReLU) except the last layer where Softmax is deployed. $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the symmetrical normalized matrix of $\tilde{A}$ where $\tilde{A} = A + I$ is adjacency matrix with self-loops and diagonal matrix $\tilde{D}_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j}$. $H^{(l)} \in \mathbb{R}^{n\times d}$ and $W^{(l)} \in \mathbb{R}^{d\times d}$ represent the embedding matrix of nodes and the trainable parameters in the l-th layer. Sometimes, we use unsymmetrical normalized matrix of $\tilde{A}$, *i.e.*, $\hat{A}_{rw} = \tilde{D}^{-1}\tilde{A}$, which is a probability transition matrix for random walks.

c) Simple graph convolution: SGC simplifies GCN by dropping its non-linear activation functions in the aggregation process. The forward pass of SGC can be described as follows:

$$H^{(L)} = \hat{A}^L XW \in \mathbb{R}^{n\times d} \tag{2}$$

Our method can be applied to any MPNN-based GNN architecture, including GCN, SGC, GAT, GIN, etc. For the sake of simplicity, we apply SGC-style (*i.e.*, dropping activation functions) backbone to show our motivations and theoretical derivation.

IV. Intuition and Motivation

A. Over-Smoothing Issues

Assuming that we stack infinite numbers of GCN layers, the produced embeddings of graph nodes in different classes will become too close to distinguish, which is called the over-smoothing issue. Formally, prior works analyze it from both theory and experiments [13, 21]. For SGC on a connected graph, $H^{(L)}$ will converge to a line (see Fig 1) as L increases to infinity according to spectral analysis:

$$\begin{aligned} H^{(L)} &= \hat{A}^L XW = \left(U\Lambda U^T\right)^L XW \\ &= U\Lambda^L U^T XW = \left(\sum_{i=1}^{n} \lambda_i^L u_i u_i^T\right) XW, \\ \lim_{L\to\infty} H^{(L)} &= \left(u_1 u_1^T\right) XW = \tilde{D}^{\frac{1}{2}}\mathbf{1}\left(\mathbf{1}^T\tilde{D}^{\frac{1}{2}}XW\right), \end{aligned} \tag{3}$$

where we apply the eigen decomposition of $\hat{A} = U\Lambda U^T$, and u_i is the eigen vector with respect to the eigenvalue λ_i (assuming that $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_n$). Because the line where the embeddings converge only contains the information on node degrees [1], SGC drops structural information. A similar conclusion can generalize to GCN with non-trivial proof (see [71]).

However, the analysis of prior methods focuses on the spectral domain, which cannot help us understand the behaviors of the embeddings, namely, how their structures vary and the respective influences of nodes in the spatial domain, except their convergence on indistinguishable embeddings. They treat nodes equally so that they cannot adaptively manipulate the aggregation in the spatial domain.

B. Semantic Over-Smoothing

A common practice to solve the over-smoothing issue is numerical scaling or normalization, *e.g.*, magnifying the embedding matrix by a large constant or stretching them to varied extents along different directions. However, the performance of deep GNNs using these numerical normalization tricks is still limited, since these tricks cannot thoroughly solve the over-smoothing issue. According to the observation and analysis on these tricks (see Fig. 1 and the theoretical justification in Sec. IV-B1), we conjecture that there may be *scale-invariant semantic information loss* as layers go deep in GNNs, which makes the embeddings indistinguishable and is often ignored by

[1] Besides, if we utilize the asymmetrically normalized adjacency matrix, *e.g.*, some forms of Mean aggregation, node embeddings would converge to a point (*i.e.*, $\mathbf{1}\left(\mathbf{1}^T XW\right)$).

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

prior works. Hence, we call the phenomenon of the semantic information loss as *semantic over-smoothing*, in which the semantic information may refer to the meaningful information on distribution, structure, or signals of labels. Empirically, it may be attributed to spatial over-aggregation and fast shifting of embeddings even after performing numerical normalization, and thus leading to semantic structure corruption. More theoretical analysis is presented in the supplementary material.

The existing methods mainly attempt to approach a better numerical limit on deep layers of GNNs or obtain a universal filter of features, so as to preserve a few more high-frequency information in the spectral domain [16]. To differentiate it from our semantic over-smoothing, we call this issue as *numerical over-smoothing*. The key difference of numerical and semantic over-smoothing can be observed from Fig. 1 where we plot embeddings produced by different methods in different stages. After utilizing the introduced graph an-isotropic normalization trick, we can clearly see in Line 3 of Fig. 1 the severe cluster blending and overlap, discovering semantic corruption. It is completely ignored and can be hardly solved by prior methods of tackling only numerical over-smoothing (Line 2). Because they have major limitations on the interpretability of the implications and significance of the preserved high-frequency information for spatial evolution or semantic varying of node embeddings, which usually hinders them to make full of flexible spatial operations to mine and keep various semantic knowledge (including comprehensive distribution, community or clustering structure, features and their relationship, label alignment, etc). Hence, studying semantic over-smoothing provides an intuitive perspective to deal with over-smoothing (Line 1 in Fig. 1) *directly* and *naturally* from the spatial domain, *e.g.*, by directly controlling, manipulating, or guiding the spatial aggregation and propagation operations to keep simple yet interpretable semantic knowledge (see above) in case of performance degradation due to semantic corruption in deeper layers (instead of considering only preserving high-frequency information or a universal filter that is not intuitive or understandable enough).

1) Discussion on Semantic Over-smoothing: In this section, we discuss about semantic over-smoothing and provide some intuition and interpretation. We describe how we discover it and why it is important from both theoretical and experimental aspects. The analysis provides justification on our intuition and motivation, and motivates us to look for a useful strategy to combat the semantic over-smoothing issue. The numerical over-smoothing issue tells us that embeddings of nodes will converge to a limit where all nodes will be too close to each other, which makes them indistinguishable. In order to solve numerical over-smoothing first, we first propose a method called *graph an-isotropy normalization, GAINorm*, as expressed below.

Definition 1 (Graph an-isotropy normalization)**.** *We define this an-isotropy normalization trick by the following iterative process:*

$$\begin{aligned} H_t &= B_{t-1}\left(B_{t-1}^T B_{t-1}\right)^{-\frac{1}{2}}, \quad \forall\, t \geq 1 \\ B_{t-1} &= \left(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\hat{A}H_{t-1}, \quad H_0 = X, \end{aligned} \tag{4}$$

where $\hat{A}$ and X can be found in Sec. III-0b, and given matrices P and Q, here $Q = P^{\frac{1}{2}}$ means the square root of a matrix whose eigen-values are all non-negative, i.e., $Q^2 = P$. When $B_t^T B_t$ is not invertible for t, we define $0^{-1} = 0$ to omit all of its zero-valued eigenvalues and ensure its invertibility.

From the definition, we can easily identify its characteristics: 1) In the spatial domain, an-isotropy normalization essentially attempts to retain the variance as 1 along every direction; 2) In the meantime, for the spectral domain, it also ensures $H^T H = I_d$, keeping the channels of H orthogonal and thus carrying diverse information. Therefore, an-isotropy normalization is a powerful and capable normalization technique to effectively overcome numerical over-smoothing. As mentioned above, the semantic over-smoothing can be naturally observed after first alleviating the numerical one with the help of the an-isotropy normalization. Thus, to essentially understand the semantic evolution during propagation, in the following, we will show its convergence property and discuss its limitations, from which we can figure out what challenge still remains and why it can hardly handle the challenge.

Theorem 1. *For an unweighted undirected graph G (connected but without self-loops), A is its adjacency matrix and $\tilde{A} = A + I$, $\tilde{D}$ is the degree matrix of $\tilde{A}$ (i.e., $\tilde{D}_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j}$). Assuming the symmetrically normalized matrix of $\tilde{A}$ is $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, and let $W = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, using the graph an-isotropy normalization trick according to Definition 1, we have:*

1) $\mathbf{1}^T B_t = \mathbf{1}^T H_t = \mathbf{0}, \quad \forall\, t \geq 1$;

2) $H_t^T H_t = I_d, \quad \forall\, t \geq 1$;

3) $WH_t = H_t, \quad \forall\, t \geq 1$;

4) Let $\bar{A} = W\hat{A}W$, then: $B_t = \bar{A}H_t, \quad \forall\, t \geq 1$;

5) Assume that H_0 is sampled from standard normal distribution $\mathcal{N}(0,1)$ or uniform distribution $\mathcal{U}(0,1)$, H_t will converge in large probability when t approaches infinity. Moreover, the columns of H_t will converge to a basis of the linear space spanned by the eigen-vectors $U_d \in \mathbb{R}^{n\times d}$ corresponding to eigenvalues of $\bar{A}$ with the top-d largest absolute values. In other words, there exists some orthogonal matrix $Q_t \in \mathbb{R}^{d\times d}$, such that: in large probability, $H_t \to U_d Q_t$, as $t \to +\infty$.

According to the theorem above, using an-isotropy normalization, there exists an orthogonal matrix $Q_t \in \mathbb{R}^{d\times d}$ such that, with a high probability, H_t will converge to $U_d Q_t$, as $t \to +\infty$. As a powerful normalization technique, the an-isotropy normalization is able to derive the principal space of $\bar{A}$ and easily capture rich low-frequency structural information, as the theorem implies. However, when it converges, some potentially beneficial super-high-frequency structural information and the whole original feature knowledge can hardly be preserved, leading to semantic structure corruption.

To give a visual intuition, please observe and compare

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

the plots of embeddings obtained by GCN with isotropic and an-isotropic normalization (Line 2 and 3 in Fig. 1). The embeddings would never converge to a line or a point (see the note in Sec. IV-A), which will make nodes definitely indistinguishable by a simple downstream classifier, if an-isotropic normalization are utilized. However, by carefully observing Stage 1 and Stage 3 of Line 3 in Fig. 1, we notice that: 1) The purple, red, and green clusters completely blend with each other; 1) The purple cluster tends to run towards and blend with the green and the red clusters although they are quite distant from it considering the original features, while the purple cluster tends to run away from the blue one, which has quite similar features. The observations intuitively tell us that embeddings might tend to forget some important feature knowledge, which conforms to the aforementioned analysis and reveals the potential cause of the semantic over-smooth concerning cluster mixture (Stage 3 of Line 3) instead of embedding approaching phenomena (Stage 3 of Line 1 and 2).

More specifically, one can get an intuition about how the semantic structure of nodes' embeddings varies from the above theorem. From the very beginning, the embeddings carry the only information of node features on the structure of clusters (Fig. 1 Line 3 Original Features). As the nodes aggregate more information through an appropriate number of layers, their embeddings additionally carry structural information of this graph, which obtains a good trade-off and thus forms well-organized semantic clusters. However, with further aggregations, more and more relatively low-frequency structural information will be absorbed into their embeddings and thus become more dominant than the high-frequency structural and all of the feature information. The reason may rest in that, d channels cannot hold all the information they receive, so some outdated information must be discarded. As more and more low-frequency information degrades the distinction between many essentially different nodes, the model would definitely forget some important semantic knowledge to maintain the well-formed structure, and thus the over-smoothing inevitably takes place even with Graph an-isotropy normalization (Fig. 1 Line 3 Stage 2 $\rightarrow$ 3). This motivates us to design some adaptive strategies to directly avoid semantic corruption and combat semantic over-smoothing, keeping intuitive and interpretable knowledge not restricted from frequency filtering. For more information on the proof, the experimental validation, and in-depth discussion, please refer to the supplementary material.

V. Cluster-keeping Sparse Aggregation Strategy

Over-smoothing in the spatial domain may be caused by over-aggregation, so an intuitive solution to relieve it is to redistribute weights to different nodes with an appropriate amount of total weights [2], termed as *sparse aggregation*. Here we propose to adopt GNNs with weighted residual connections to realize sparse aggregation, which allows keeping of clusters and thus preventing the semantic structures of embeddings from severe corruption. The formulation can be formally expressed as below:

$$H^{(l+1)} = GNN^{(l)}(\hat{A}, H^{(l)}) \tag{5}$$

$$= \Xi^{(l)}\sigma\left(f\left(\hat{A}, H^{(l)}\right)\right) + \left(I_n - \Xi^{(l)}\right) H^{(l)} \in \mathbb{R}^{n\times d}, \tag{6}$$

$$\Xi^{(l)}_{i,i} = \Theta^{(l)} \cdot \Phi^{(l)}_i \cdot \Psi^{(l)}_i \in [0, 1], \tag{7}$$

where $\Xi^{(l)} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose diagonal elements are the aggregation weights of every node in the layer l and $f(\cdot)$ implies the aggregation function of a GNN model. Besides, according to Eq. 7, the strategy concerns three factors [3], which attempts to control the comprehensive distribution of the embeddings uniformly and then considers additional customized local adjustment node-wisely. In specific, $\Theta^{(l)}$ refers to the global control coefficient on the layer l. $\Phi^{(l)}_i$ and $\Psi^{(l)}_i$ denote the silhouette-based and homophily-based local control coefficients of the node i on the layer l, respectively. Although the strategy is simple, it turns out to be effective as a strong baseline model to confront the over-smoothing issue. We provide a theoretical analysis on the proposed model in Sec. V-D. Details of our strategy are elaborated as below.

A. Initialization

The initial embeddings on the first few layers of GNNs may not establish complete structural information, since only very local information of nodes can be captured on these layers. So, we let GNNs aggregate for a few layers (*e.g.*, two or three layers in practice) until they gather sufficient information and establish solid semantic structures. Then, we perform the following steps of the strategy from the layer L_0.

B. Global control

After aggregating till L_0 layers, we first cluster the embeddings using the classic K-means algorithm. Embedding clustering might be viewed as a good estimation on label-based clusters supposing a L_0-layer GNN encoder can capture meaningful semantic structures. Assume that the node i is clustered into the z_i-th cluster, while the j-th cluster C_j has a centroid $c_j (1 \leq j \leq K)$ where K is the total number of clusters. In order to approximately estimate how much the layer l should aggregate, we design a global control coefficient (denoted by $\Theta^{(l)}$). It is designed according to the ratio between the compactness of embeddings output by the layer L_0 and the layer l. When the magnitude of the ratio does not change much, $\Theta^{(l)}$ can evaluate how well the embeddings can be clustered. With the embeddings well clustered, further aggregation will not be needed. However, when the magnitude of the ratio varies a lot, $\Theta^{(l)}$ will correspondingly control the distribution of embeddings without additional tricks or normalization. For example, if the ratio quickly shrinks, the over-smoothness will

[2] By this, we mean that every node in the graph can have a different aggregation extent and the total aggregation extent in the whole graph should be limited, which is quite similar to the idea of *sparse linear regression*.

[3] In addition to the three factors mentioned above, we discuss more influence factors on the proposed strategy in the supplementary material.

happen and we will tend to stop the aggregation of this layer. Formally, $\Theta^{(l)}$ can be formulated as below.

$$\Theta^{(l)} = \min\left(\left(\frac{dist^{(L_0)}}{dist^{(l)}}\right)^{\alpha}, 1\right) \in [0,1], \\ dist^{(l)} = \sum_{i=1}^{n}\left\|h_i^{(l)} - c_{z_i}^{(l)}\right\|_2^2, \quad \forall l \in [L_0, L], \tag{8}$$

where $dist^{(l)}$ refers to the objective of K-means measuring the distance between any embedding and the center of its corresponding cluster. $h_i^{(l)}$ represents the embedding of the i-th node. $\alpha \in [0,1]$ is a hyper-parameter to determine the influence of the global control coefficient.

We call this step global control since we control the aggregation extents of all nodes uniformly according to the global distribution of node embeddings.

C. Local control

Under the global control, the nodes in each layer will be treated equally (*i.e.*, their aggregation weights are the same). Considering the difference of nodes, we also introduce two types of local control coefficients, including the silhouette coefficient that evaluates the clustering quality of each node and the homophily coefficient that evaluates the local coherency of each node. These local control coefficients determine the individual weights of the aggregation for each node.

1) Silhouette-based local control: Intuitively, the Silhouette Coefficient $S_i^{(l)}$ is inversely proportional to the uncertainty of clustering for a node. When the uncertainty is high, a node is encouraged to aggregate more information to decide its corresponding cluster, and vice versa. Thus we first employ $S_i^{(l)}$ to determine how well a node is clustered and then locally adjust the distribution according to $\Phi^{(l)}$, which are defined as follows:

$$\Phi_i^{(l)} = \left(\frac{-S_i^{(l)}+1}{2}\right)^{\beta} \in [0,1], \tag{9}$$

$$S_i^{(l)} = \frac{b_i^{(l)} - a_i^{(l)}}{\max\left(a_i^{(l)}, b_i^{(l)}\right)} \in [-1,1], \quad \forall l \in [L_0, L], \tag{10}$$

$$a_i^{(l)} = \frac{1}{\left|C_{z_i^{(l)}}\right| - 1}\sum_{j \in C_{z_i^{(l)}} \setminus i}\left\|h_j^{(l)} - h_i^{(l)}\right\|_2^2, \tag{11}$$

$$b_i^{(l)} = \min_{1 \le k \le K, k \ne z_i^{(l)}} \frac{1}{|C_k|}\sum_{j \in C_k}\left\|h_j^{(l)} - h_i^{(l)}\right\|_2^2, \tag{12}$$

where a_i measures the average distance for the i-th node away from the remaining nodes from its belonged cluster. b_i calculates the distance for the i-th node away from the closest cluster that it does not belong to. $\Phi_i^{(l)}$ is the local adjustment of the node i based on the Silhouette Coefficient $S_i^{(l)}$. β is the hyper-parameter.

2) Homophily-based local control: Homophily is an essential and intrinsic characteristic of a graph and its local structures. High homophily of a graph reveals that neighbors often share the same ground-truth labels. The degree of homophily might heavily influence the aggregation behavior of a node because, just as pointed by Keriven [69], low homophily would accelerate the over-smoothing issue and beneficial smoothing never appears in the theoretical formulation of [69], which motivates our design of a homophily-based local control as follows. Specifically, when the homophily of a node is high, we encourage the node to aggregate more information from neighbors. Once the homophily is low, the node needs to aggregate less to avoid moving towards the center of different clusters, which may lead to information loss and thus aggravates the semantic over-smoothing.

However, note that we can not directly obtain the exact node homophily degrees like the latent variables in [69] due to unknown node labels. Hence, we dynamically estimate them (*i.e.*, $\zeta^{(l)}(i)$) based on clustering, assuming that a node whose neighbors belong to the same cluster (potentially with the same semantic labels) might have a large node homophily degree, and vice versa. And then the homophily-based local control coefficient of the node i (*i.e.*, $\Psi_i^{(l)}$) can be defined as below:

$$\Psi_i^{(l)} = \left(\zeta^{(l)}(i)\right)^{\gamma} \in [0,1], \quad \forall l \in [L_0, L] \tag{13}$$

$$\zeta^{(l)}(i) = \frac{1}{|\mathcal{N}_i \cup i|}\max_{k \in [1,K]} cnt_i^{(l)}(k) \in [0,1], \tag{14}$$

$$cnt_i^{(l)}(k) = \left|\left\{j | j \in \mathcal{N}_i \cup i, z_j^{(l)} = k\right\}\right|, \tag{15}$$

where $\mathcal{N}_i$ is the neighborhood of the node i and $cnt_i^{(l)}(k)$ denotes the number of the nodes in $\mathcal{N}_i$ belonging to the cluster k. γ is the hyper-parameter.

D. Discussion on Weighted Residual Connections

In this section, we explain why a sparse aggregation strategy on GNNs with weighted residual connections can favor alleviate semantic over-smoothing issues and give a conclusion on its discriminative power.

Theorem 2. *For an unweighted undirected graph G that is connected but without self-loops, A is its adjacency matrix and $\tilde{A} = A + I$. $\tilde{D}$ is the degree matrix of $\tilde{A}$ (i.e., $\tilde{D}_{i,i} = \sum_{j=1}^{n}\tilde{A}_{i,j}$). Denote by $\lambda_{\tilde{G}}$ the spectral gap of graph G with self-loops. Assuming the symmetrically normalized matrix of $\tilde{A}$ is $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and x is a graph signal(i.e., $x : \mathcal{V}(G) \to \mathbb{R}$), if the aggregation matrix $\left(1-\alpha^{(l)}\right)I_n + \alpha^{(l)}\hat{A}$ is used, which means:*

$$h^{(0)} = x, \quad h^{(l)} = \left(\left(1-\alpha^{(l)}\right)I_n + \alpha^{(l)}\hat{A}\right)\cdot h^{(l-1)}, \quad \forall l \ge 1 \tag{16}$$

then $h^{(k)}$ converges if $\prod_{l=1}^{k}\left(1-\alpha^{(l)}\cdot\lambda_{\tilde{G}}^2\right)$ converges to 0 as k goes to infinity and the convergence rate follows that:

$$\left|h^{(k)} - \boldsymbol{\pi}\right| \le \left(\sum_{i=1}^{n} x_i\right)\cdot\prod_{l=1}^{k}\left(1-\alpha^{(l)}\cdot\lambda_{\tilde{G}}^2\right)\cdot\mathbf{1}, \tag{17}$$

where $\boldsymbol{\pi} = \frac{\left\langle\tilde{D}^{\frac{1}{2}}\mathbf{1}, x\right\rangle}{2m+n}\cdot\tilde{D}^{\frac{1}{2}}\mathbf{1}$, and here we denote $|x|$ as element-wise absolute value if x is a vector, $x \le y$ means $\forall i\ x_i \le y_i$ and $x_i = x(i)$ is the i-th element of x.

Corollary 1. *Let $\lambda = \lambda_{\tilde{G}}^2 \in [0,1]$, $b_i = \alpha^{(i)}\cdot\lambda_{\tilde{G}}^2$. Assume that $\forall\ i \ge 1, b_i \in (0,1), \alpha^{(i)} \in [0,1]$ and $\alpha^{(1)} \ge \alpha^{(2)} \ge \cdots \ge$*

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

$\alpha^{(k)} \geq \cdots$, *then: if* $\lim_{l\to\infty} \alpha^{(l)} > 0, \sum_{l=1}^{k} \alpha^{(l)} = +\infty$, *then* $\lim_{k\to\infty} h^{(k)} = \boldsymbol{\pi}$.

According to Corollary 1, as $\alpha^{(l)}$ converges to $c \in (0, 1]$, or as it converges to 0 but slowly enough, then $h^{(k)}$ will converge absolutely. Thus, when k is sufficiently large, all node embeddings will converge to a line and the semantic structure will be corrupted completely, which means that the semantic over-smoothing issue will definitely happen. This motivates us to employ the GNN structure with a sparse aggregation strategy that tends to slow the convergence.

Theorem 3. *Using the same definition in Theorem 2, but with consideration of nonlinear activation functions and linear transformations, GCN with both layer-wise and node-wise weights can achieve* $1-$*WL discriminative power if we use* $\tilde{A}$ *and slight modification* [4]. *If combined with node-wise normalization trick* [5]*, it holds also for* $\hat{A}$ *and* $\hat{A}_{rw}$.

According to the above theorem, we demonstrate that our structure with the weighted residual connection can achieve $1-$WL discriminative power. We will elaborate the theoretical details in the supplementary material.

E. Analysis on Time and Space Complexities

In this subsection, we give a brief yet concise analysis of the overall theoretical time and space required for our method based on the specific optimized implementation. Considering the high-parallelizablity brought by GPUs, we denote by notation $\mathcal{O}_p(\cdot)$ *parallelizable time complexities* for parallelizable operations, which would be much faster in practice than those unparallelizable ones with traditional $\mathcal{O}(\cdot)$.

The updating operations in Eq. 6 require similar time and space as the chosen GNN encoder. Clustering needs overall $\mathcal{O}_p(nKd)$ time and $\mathcal{O}(Kd + nd)$ space for one iteration with $\mathcal{O}_p(nKd)$ time for both cluster centers updating and assignments calculation, after which Eq. 8 can be naturally obtained. Besides, Eq. 15 can be implemented via a simple Sum aggregation [40] with $\mathcal{O}_p\left((n+m)\,d\right)$ time and $\mathcal{O}(nd)$ space. To give an efficient, parallelable, and space-saving implementation for Silhouette Coefficient S_i in Eq. 12, let $f(i,t) = \sum_{j\in C_t} \|h_j - h_i\|_2^2 = |C_t| \cdot \|h_i\|_2^2 + S^{(norm)}(t) - 2S(t)$ where $S^{(norm)}(t) = \sum_{j\in C_t} \|h_j\|_2^2$ and $S(t) = \sum_{j\in C_t} h_j$. Then the matrix $F \in \mathbb{R}^{n\times K}$ containing all $f(i,t)$ $(\forall\ i \in [1,n], t \in [1,K])$ can be obtained with $\mathcal{O}_p(nKd)$ running time and $\mathcal{O}(nd + nK)$ space complexity where the first term $|C_t| \cdot \|h_i\|_2^2$ is a vector outer product, and all $\|h_i\|_2^2, S^{(norm)}(t), S(t)$ can be pre-processed with no more than $\mathcal{O}_p(nKd)$ time. Finally, the coefficients can be easily obtained by simply selecting the top 2 elements with $\mathcal{O}_p(nK)$ complexity. Therefore, the overall theoretical time and space complexities are $\mathcal{O}_p\left(nKd\right)$ and $\mathcal{O}\left((n+m+K)\,d + nK\right)$.

[4]The positions of linear transformations and $\sigma(\cdot)$ are slightly different.

[5]Scaling the embedding of every node by node-wise weights. Please refer to the supplementary material.

VI. Experimental Results

A. Experiment Settings

1) Datasets: In this section, we evaluate our strategy in a conventional GCN backbone with weighted residual connections. Our work focuses mainly on semi-supervised node classification tasks in graphs. We evaluate our model on public benchmarks, including three commonly used scientific citation networks, Core, Citeseer, and Pubmed, scientific co-authorship networks CS and Physics, as well as Amazon purchasing system Computers and Photo. We adopt the standard training/validation/testing split for all these datasets as prior works [10, 19].

2) Comparative Baselines: In comparison experiments, ten baselines or state-of-the-art models are applied for comparison, *i.e.*, the normalization based method PairNorm [21], the skip connections based methods including GCNII [19], GPRGNN [72], APPNP [73], JKNet [66], and DAGNN [13] as well as two recenly proposed state-of-the-art models A-DGN [74] and FAGCN [75]. Note that we leave out the random dropping-based methods due to their consistently poor performance relative to others (see results reported in [17]). These aforementioned prior works all introduce additional structural tricks. For example, PairNorm employs an explicit normalization layer. GCNII employs a novel kind of initial connection followed by A-DGN. JKNet, APPNP, GPRGNN, and DAGNN are all obvious multi-scale models and utilize jump connections with different weighting schemas (see Tab. III for a clear comparison). Furthermore, FAGCN [75] utilizes a kind of sophisticated learnable signed attention mechanism as its weighting strategy with both residual and initial connections, which simultanously facilitates low-pass and high-pass filters. And by introducing a complex node-wise ordinary differential equation on the graph, A-DGN [74] can preserve long-distance dependency and alleviate both the over-smoothing and over-squashing issues. Compared to them, we choose the simplest and most commonly used structure (simple residual connections from last layer) and do not adopt any additional tricks. Our main experimental goal is not to outperform the best state-of-the-art result but to test the effectiveness of our proposed strategy. We choose this structure because: 1) They are simple enough and can clearly show the effectiveness of ours; 2) It is convenient to implement the proposed strategy because we only need to control the residual weights. One can see that those advanced structural tricks can also be incorporated to improve the performance. But as reported in the following section, we have already achieved competitive and even better performance without the tricks.

3) Implementation Details: More specifically, we employ two *fundamental* GNN models, GCN [10] and SGC [24] (with weighted residual connections), as baselines to incorporate our strategy, which is implemented by Pytorch [76] and optimized with Adam Optimizer [77]. The performance of the comparison models is evaluated in terms of accuracy. All experiments are performed on a Ubuntu system with a single GeForce RTX 3090Ti GPU (24GB Memory) and 64 AMD EPYC 7302 CPUs. For better reproduction, please see Section VI.B of

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

TABLE I
PERFORMANCE COMPARISON WITH THE STATE-OF-THE-ART METHODS ON THREE GRAPH BENCHMARKS CORA, CITESEER, AND PUBMED IN TERMS OF MEAN TEST ACCURACY, STANDARD DEVIATION AS WELL AS AVERAGE RANKING IN 5 RUNS WITH DIFFERENT INITIALIZATIONS

Model	Cora				Citeseer				Pubmed				Average
#Layers	2	8	16	32	2	8	16	32	2	8	16	32	Ranking
GCN	81.34±0.42	72.72±2.37	63.40±4.40	31.90±0.70	69.46±0.29	57.74±6.30	48.70±3.48	36.66±8.61	77.40±0.37	77.24±0.71	70.02±4.23	44.22±0.93	10.25
SGC	79.00±0.46	78.02±1.06	75.26±1.06	63.84±3.63	67.92±0.85	68.42±0.46	68.08±0.73	67.50±1.43	77.50±0.66	70.90±0.59	71.34±0.09	70.70±0.32	8.92
PairNorm	78.30±1.33	71.06±1.74	66.80±2.71	55.86±8.96	65.80±1.35	54.81±6.46	46.26±2.69	44.20±1.23	75.50±0.41	74.82±1.03	74.34±0.68	72.12±3.01	10.83
GCNII	82.19±0.77	84.23±0.42	**84.69±0.51**	**85.29±0.47**	67.81±0.89	70.62±0.63	72.97±0.71	73.24±0.78	78.05±1.53	79.34±0.51	80.03±0.50	79.81±0.27	4.00
JKNet	79.06±0.11	75.66±0.38	72.97±3.94	73.23±3.59	66.98±1.82	60.56±1.41	54.33±7.74	50.68±8.73	77.24±0.92	76.92±1.03	64.37±8.80	63.77±9.21	10.08
GPRGNN	82.53±0.49	84.19±0.40	83.69±0.55	83.13±0.60	70.49±0.95	71.47±0.58	71.39±0.73	71.01±0.79	78.73±0.63	78.90±0.47	78.78±1.02	78.46±1.03	5.42
DAGNN	80.30±0.78	**84.28±0.59**	84.14±0.59	83.39±0.59	70.91±0.68	72.44±0.54	73.05±0.62	72.59±0.54	77.74±0.57	79.68±0.37	**80.32±0.38**	**80.58±0.51**	3.58
APPNP	82.06±0.46	83.59±0.40	83.64±0.48	83.68±0.48	71.67±0.78	72.04±0.52	72.13±0.53	72.13±0.59	79.46±0.47	**80.02±0.30**	80.30±0.30	80.24±0.33	3.58
A-DGN	59.50±0.50	60.40±0.80	79.58±1.80	79.54±0.40	59.88±1.00	60.96±1.20	60.94±1.30	60.94±1.30	74.63±1.40	77.98±0.70	77.78±1.60	77.76±0.80	9.58
FAGCN	**83.34±0.50**	83.60±0.30	83.82±0.40	82.84±0.40	71.86±0.50	72.02±0.40	71.88±0.70	70.88±0.90	78.62±0.50	78.94±0.50	79.50±0.40	79.28±0.40	4.67
GCN+Ours	83.12±0.22	82.96±0.36	82.70±0.53	83.40±0.51	73.06±0.60	72.94±0.40	73.20±0.38	72.78±0.38	**80.06±0.31**	79.60±0.32	79.60±0.26	79.78±0.48	**3.25**
SGC+Ours	82.26±0.18	82.32±0.14	82.94±0.40	83.12±0.18	**73.36±0.18**	**74.0±0.29**	**73.38±0.40**	**73.64±0.31**	79.32±0.27	79.40±0.10	79.34±0.24	79.30±0.22	3.75

TABLE II
PERFORMANCE COMPARISON WITH THE STATE-OF-THE-ART METHODS ON FOUR GRAPH BENCHMARKS COAUTHORCS, COAUTHORPHYSICS, AMAZONCOMPUTERS, AND AMAZONPHOTO IN TERMS OF MEAN TEST ACCURACY, STANDARD DEVIATION AS WELL AS AVERAGE RANKING IN 5 RUNS WITH DIFFERENT INITIALIZATIONS

Model	CoauthorCS		CoauthorPhysics		AmazonComputers		AmazonPhoto		Average
#Layers	16	32	16	32	16	32	16	32	Ranking
GCN	53.19±7.23	41.29±5.11	85.23±2.18	79.87±3.86	64.02±2.38	58.30±3.35	70.81±3.48	58.47±8.80	10.50
SGC	71.75±3.65	70.52±3.96	92.34±0.20	91.46±0.48	37.48±0.07	37.44±0.12	35.64±6.11	26.08±1.39	9.86
PairNorm	75.17±5.15	63.23±5.03	90.18±1.17	88.51±0.95	77.41±1.85	68.37±4.35	82.72±1.19	71.93±5.21	7.88
GCNII	58.94±2.63	71.67±2.68	92.13±1.31	93.15±0.92	38.88±4.26	37.56±0.43	68.37±6.61	62.95±9.41	9.25
JKNet	81.31±3.21	81.82±3.32	91.24±0.97	90.92±1.61	60.76±5.10	67.99±5.07	74.86±8.45	78.42±6.95	7.88
GPRGNN	89.39±0.39	89.56±0.47	93.64±0.31	93.49±0.59	76.07±2.28	41.94±9.95	**91.55±0.43**	**91.74±0.81**	4.88
DAGNN	91.13±0.50	89.60±0.71	93.77±0.29	93.31±0.60	80.33±1.04	79.73±3.63	90.81±0.59	89.96±1.16	4.25
APPNP	91.64±0.53	91.61±0.49	93.96±0.36	93.75±0.61	39.41±5.80	43.02±10.16	64.59±20.09	59.62±23.27	6.75
A-DGN	**92.06±0.20**	91.67±0.20	88.32±2.50	87.26±1.60	**83.03±0.58**	83.41±1.30	90.40±0.60	88.65±1.40	4.50
FAGCN	81.40±5.04	83.88±2.64	85.68±1.88	86.17±2.30	69.96±3.93	73.19±1.61	78.53±2.12	80.55±3.48	7.63
GCN+Ours	92.01±0.16	**92.12±0.19**	**94.19±0.30**	93.89±0.22	82.84±1.07	83.06±0.41	90.31±0.56	90.08±0.74	2.38
SGC+Ours	91.85±0.22	91.78±0.48	94.14±0.24	**94.04±0.27**	82.54±0.55	**84.54±1.07**	90.94±0.24	90.72±0.38	**2.00**

TABLE III
THE TECHNICAL CONFIGURATION COMPARISON OF COMPARATIVE BASELINES

Model	Normalization	Skip Connection	Weighting Schema for Connections	Aggregating Schema
PairNorm	√	×	–	Standard
GCNII	×	Initial Connections	Hyper-parameters	Residual Learnable Self-Weighting
JKNet	×	Jump Connections	Concat/Max-Pooling/LSTM Attention	Standard
GPRGNN	×	Jump Connections	Exponential Moving Average	Standard
DAGNN	×	Jump Connections	Global Attention	Standard
APPNP	×	Jump Connections	Learnable	Standard
A-DGN	×	Residual Connections	Hyper-parameters	Anti-symmetric Learnable Self-Weighting
FAGCN	×	Initial & Residual Connections	Hyper-parameters	Signed Learnable Attention
Ours	×	Residual Connections	Adaptive with Proposed Sparse Aggregation Strategy	Standard

the supplementary material for the hyper-parameters tuning method and specific configurations (Tab. II).

B. Comparison with the State-of-the-arts

We report the performance results of the state-of-the-art models with varied numbers of layers on benchmarks. See Table I and Table II for detailed comparison. In Table I, we observe the performance of our model is comparable to or even better than the state-of-the-arts, considering our baseline model is GCN and SGC. With the aid of our strategy, the performance of GCN and SGC obviously increases. Especially in the Citeseer dataset, our model ranks at the top, while our models are comparable to other methods in the other two datasets. For these three datasets concerning scientific citation networks, it may not need too deep GNNs to aggregate useful information for classification. Therefore, most models suffer from performance degradation as the number of layers increases. For GCN and SGC, they suffer from severe degradation. Although we observe that there is also minor degradation in our model, considering the large drop in the performance of GCN and SGC, our strategy well stabilizes them in deep layers. Besides, in Fig. 2, we show the accuracy of our models (*i.e.*, GCN+Ours, SGC+Ours, GAT+Ours, and GIN+Ours) with different layers (*i.e.*, from 2 to 64). For comparison, we also measure the accuracy of GCN, SGC, GAT, and GIN, as well as PairNorm [21]. As observed, they all suffer from obvious performance degradation in these three datasets, which decrease by at least 15%. With the aid of our strategy, they can maintain steady performance with more

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

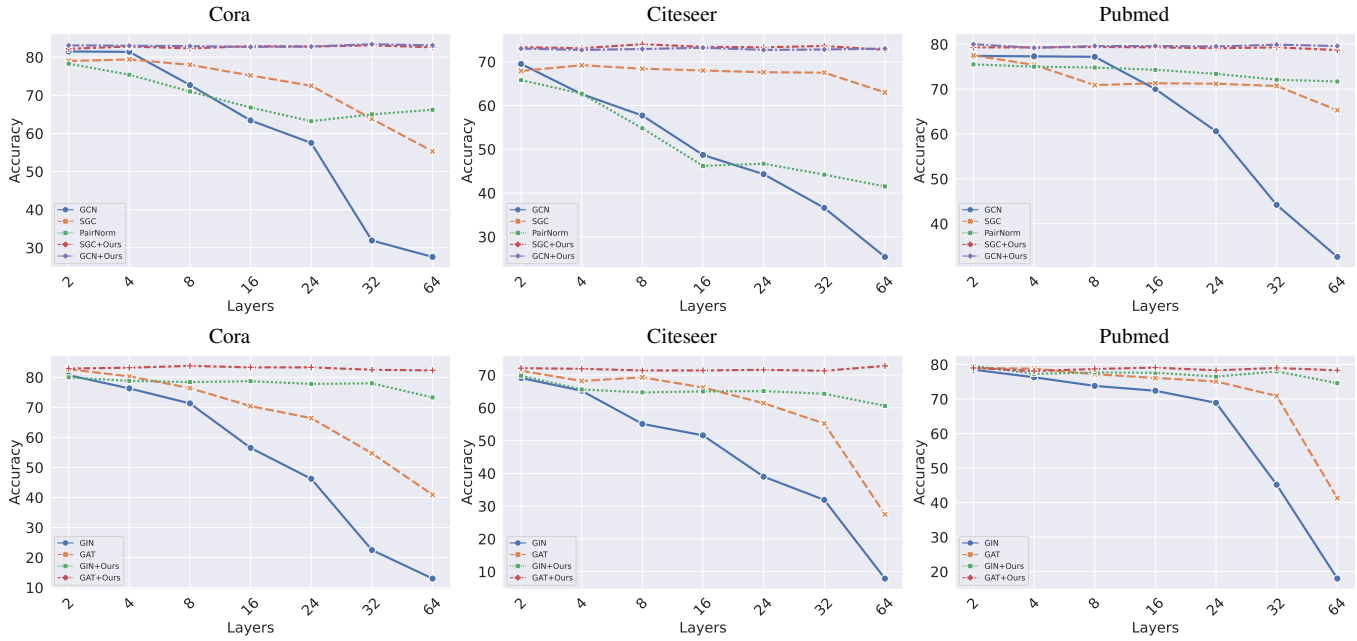

Fig. 2. Accuracy of the models with different layers, including the comparisons with GCN, SGC, and Pairnorm in the first line and with GIN and GAT in the second line.

than 80% accuracy.

In Table II, our model generally achieves state-of-the-art performance against the competing methods in these benchmarks. These datasets contain more complex graph data, so deep GNNs demonstrate excellent performance. Impressively, the performance gains of our strategy over GCN and SGC are significant, which reflects the effectiveness of our strategy. Furthermore, we visualize two examples of the distributions of the embeddings produced by GCN and the one enhanced by our strategy in Fig. 3. In the first column of the figure, we show the initial distribution of clusters for the embeddings on the first layer. In the second column, we observe the over-smoothing phenomenon where the embeddings converge into lines or curves, leading to indistinguishable clusters produced by the GCN with 32 layers. From the 32-layer GCN equipped with our proposed strategy, as shown in the third column, we do not observe over-smoothing. Besides, the embeddings are generally well clustered, and their structural information remains complete.

C. Ablation and Hyper-parameter Studies

Our strategy is mainly determined by the global and local control coefficients including silhouette-based and homophily-based coefficients. As shown in Fig. 4, we evaluate the variants of our proposed strategy with silhouette-based local coefficient only, homophily-based local coefficient only, global coefficient only, and our complete strategy. The evaluation is performed on three datasets, Cora, Citeseer, and Pubmed. We show the accuracy results of the model with different layers (*i.e.*, from 2 to 32 layers). As observed, combining the local and global coefficients produce the optimal results, better than the other variants. Moreover, to study the effect of the hyper-parameter K (see Sec. V-B) on the performance, we vary it from 5 to 40 and test the method on Cora, Citeseer, and Pubmed (Fig. 7). The results clearly show that our method is strongly robust to the selection of K, revealing that reducing K is a viable solution for time-intensive situations.

D. Node Classification With Noisy Features

Inspired by the prior works (*e.g.*, [21]), we set up a challenging task to show the superiority of deep GNN models compared to their shallow counterparts. In particular, for a graph dataset with a semi-supervised node classification task, we substitute all node features by noise sampled from a standard normal distribution (*i.e.*, $\mathcal{N}(0,1)$). In this way, only the structural information of graphs remains. Different from the task in [21], our task is more difficult since we conduct the feature substitution for all nodes instead of the nodes out of the training set only. In the meantime, we contaminate the features with noise instead of replacing them with zeros. Thus, this task forces GNN models to learn useful and valid structural information to overcome the negative impact of noise. Therefore, our proposed task is able to reflect the superiority of GNN models. We show the results in Fig. 5.

From Fig. 5, we observe that the performance of two baseline models degrades from around $12-14$ layers due to numerical over-smoothing, while the performance of Pairnorm [21] degrades from around $22-25$ layers due to the semantic over-smoothing issue. On the contrary, our method can consistently improve performance as layers go deep. When models are deep enough, we can achieve better results compared to all the competing methods, which implies that deep structural information and distant messages can be helpful for graph representation learning.

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

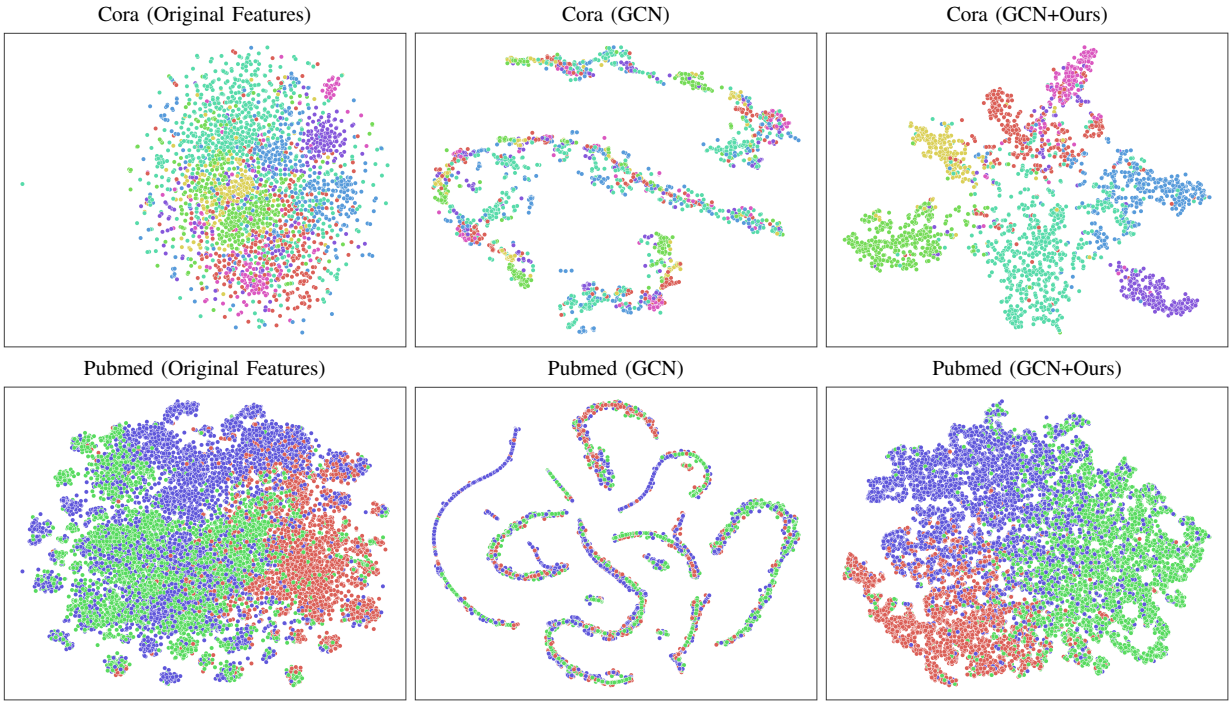

Fig. 3. Scatterings of embeddings produced by 32-layer GCN and GCN+Ours on Cora and Pubmed.

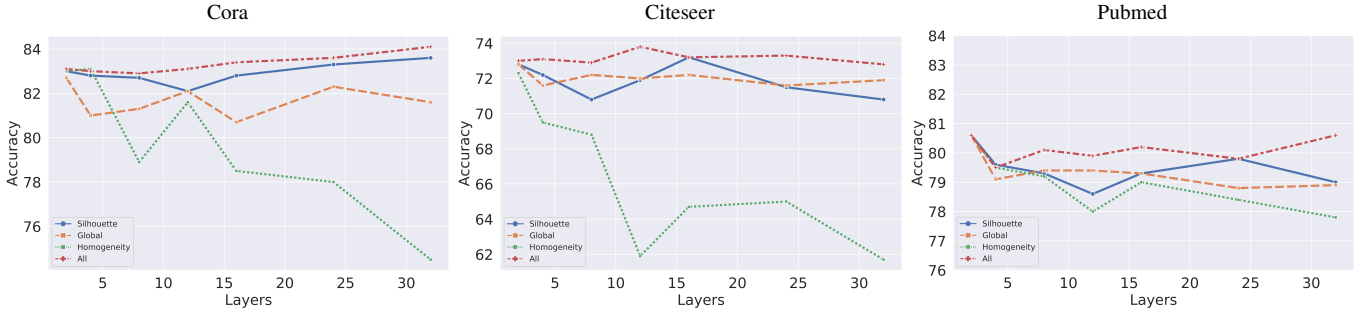

Fig. 4. Ablation studies on the global and local control coefficients

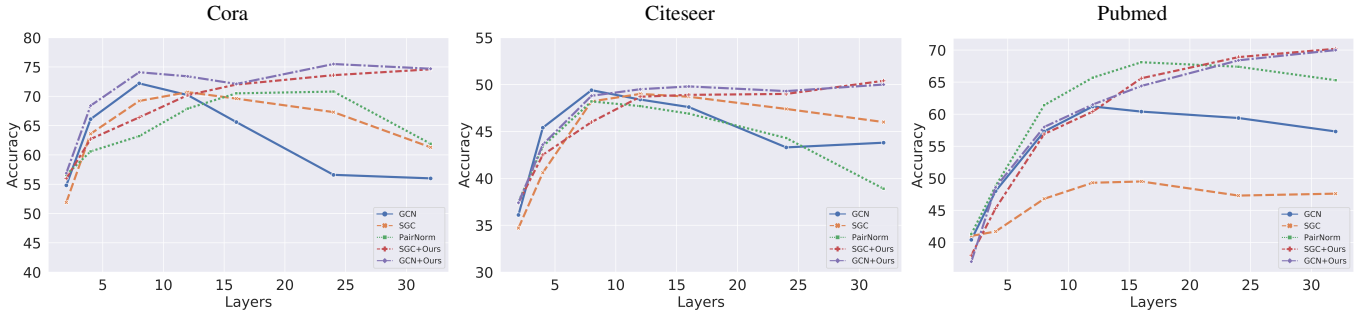

Fig. 5. The performance of different methods on Noisy Features

TABLE IV
THE COMPARISON OF THE RUNNING TIME (PER EPOCH) OF DIFFERENT METHODS

32-layer models with 128 hidden units	Cora	Citeseer	Pubmed	CoauthorCS	AmazonPhoto	AmazonComputer	CoauthorPhysics
A-DGN	0.04s	0.04s	0.2s	0.30s	0.32s	0.55s	0.88s
FAGCN	0.06s	0.06s	0.2s	0.25s	0.37s	0.57s	0.71s
GCN+Ours	0.47s	0.46s	0.61s	0.64s	0.52s	0.68s	0.92s
SGC+Ours	0.44s	0.45s	0.59s	0.62s	0.59s	0.66s	0.90s

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2024.3402317

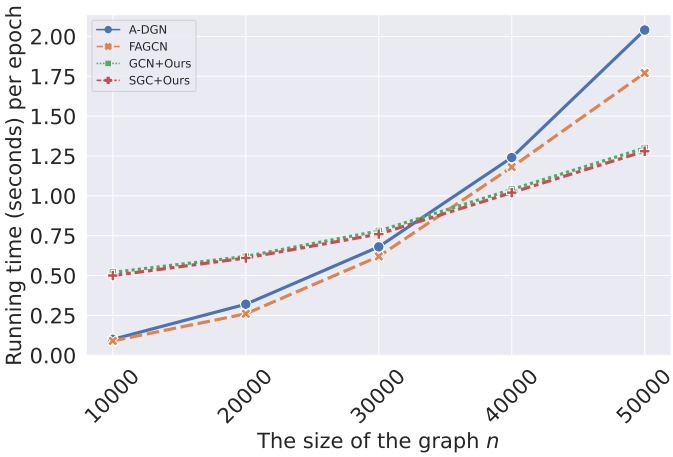

Fig. 6. Running time comparison on some synthetic graphs with increasing sizes (see the main text for the specific and detailed graph generating methods)

E. Running Time Comparison and Analysis

To provide a better understanding of the trade-off between our performance and computational burden, we give the running time (per epoch) of the proposed methods (*GCN+Ours* and *SGC+Ours*) and some recently proposed state-of-the-art counterparts (A-DGN [74] and FAGCN [75]) on all graph benchmarks in Tab. IV. From Tab. IV, we found that our method might need more running time due to clustering multiple times. Nevertheless, the time consumption is comprehensively acceptable for immediate-size graphs such as Pubmed, CS, and Physics. In Tab. IV, we sort the columns, *i.e.*, the names of these benchmarks, with increasing sizes (the numbers of contained nodes and edges), from which we can easily find that though the counterparts can often be more efficient than ours, however, the running time gaps are gradually bridged as the graph becomes large. Please note that some datasets with less nodes might need more running time due to significantly more edges (*e.g.*, AmazonComputer and Pubmed have much more edges than Pubmed though their sizes, *i.e.*, numbers of nodes, are relatively smaller). To more clearly and intuitively show the tendency, we construct some synthetic random (connected) graphs with increasing sizes n [6], and conduct the efficiency comparison on them, whose results are plotted in Fig. 6 where we can observe a clear gap alleviation process and gradually surpassing tendency showing the acceptance of the efficiency of ours. Moreover, in practice, choosing more efficient clustering methods (*e.g.*, DBSCAN [79]) may potentially accelerate the whole framework, which we will leave as future work to explore in detail.

VII. Conclusion and Limitations

Conclusion In this paper, we investigate the over-smoothing issue in deep GNNs. We observe that over-smoothing not only results in indistinguishable embeddings of graph nodes, but also corrupts their semantic structures, denoted as semantic over-smoothing. In order to alleviate the concern, we propose a simple yet effective cluster-keeping sparse aggregation strategy to preserve the semantic structure of embeddings in deep GNNs, which can be easily implemented as a plug-and-play structure of GNNs via weighted residual connections. Empirically, our method generally achieves state-of-the-art performance against the competing models in seven real-world benchmarks.

Limitations Due to the required frequent clustering, our strategy may risk high computational complexity. More efficient and sophisticated strategies are left for future work, *e.g.*, considering edge-wise sparse aggregation. On the other hand, our strategy relies on the quality of clustering, so better clustering methods might be beneficial. Also, in future work, we will explore why and how deep GNNs can alleviate over-fitting issues, especially considering the possible noise in graph structures, *i.e.*, noisy connections.

[6]We directly utilize ***gnp_random_graph(n, p)***, a stochastic generator provided by NetworkX python library [78], to generate random graphs with $n \in [10^4, 5 \times 10^4]$ and $p = 10^{-3}$ (see Supplementary Materials for the detailed configuration and numerical results in Tab. IV).

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[2] W. Zhang, K. Shu, H. Liu, and Y. Wang, "Graph neural networks for user identity linkage," *arXiv preprint arXiv:1903.02174*, 2019.

[3] Y. Shen, Y. Wu, Y. Zhang, C. Shan, J. Zhang, B. K. Letaief, and D. Li, *How Powerful is Graph Convolution for Recommendation?* New York, NY, USA: Association for Computing Machinery, 2021, p. 1619–1629.

[4] Z. Liang, M. Yang, L. Deng, C. Wang, and B. Wang, "Hierarchical depthwise graph convolutional neural network for 3d semantic segmentation of point clouds," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8152–8158.

[5] K. Han, B. Lakshminarayanan, and J. Z. Liu, "Reliable graph neural networks for drug discovery under distributional shift," in *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[6] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.

[7] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 729–734.

[8] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[9] C. Gallicchio and A. Micheli, "Graph echo state networks," in *The 2010 international joint conference on neural networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR*

Fig. 7. Performance of GCN+Ours with different layers against varied hyper-parameter K

2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1263–1272.

[12] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *International Conference on Learning Representations*, 2020.

[13] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 338–348.

[14] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 6437–6449.

[15] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," in *International Conference on Learning Representations*, 2019.

[16] C. Yang, R. Wang, S. Yao, S. Liu, and T. Abdelzaher, "Revisiting over-smoothing in deep gcns," *arXiv preprint arXiv:2003.13663*, 2020.

[17] T. Chen, K. Zhou, K. Duan, W. Zheng, P. Wang, X. Hu, and Z. Wang, "Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2022.

[18] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," in *International Conference on Learning Representations*, 2022.

[19] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1725–1735.

[20] H. Zhang, T. Yan, Z. Xie, Y. Xia, and Y. Zhang, "Revisiting graph convolutional network on semi-supervised node classification from an optimization perspective," *arXiv preprint arXiv:2009.11469*, 2020.

[21] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *International Conference on Learning Representations*, 2020.

[22] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Tackling over-smoothing for general graph convolutional networks," *arXiv preprint arXiv:2008.09864*, 2020.

[23] T. Fang, Z. Xiao, C. Wang, J. Xu, X. Yang, and Y. Yang, "Dropmessage: Unifying random dropping for graph neural networks," *ArXiv*, vol. abs/2204.10037, 2022.

[24] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6861–6871.

[25] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, and B. Long, "Graph neural networks for natural language processing: A survey," *arXiv preprint arXiv:2106.06090*, 2021.

[26] U. Nazir, H. Wang, and M. Taj, "Survey of image based graph neural networks," *arXiv preprint arXiv:2106.06307*, 2021.

[27] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *arXiv preprint arXiv:2101.11174*, 2021.

[28] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu, "A survey on deep graph generation: Methods and applications," *arXiv preprint arXiv:2203.06714*, 2022.

[29] N. Yang, H. Wu, J. Yan, X. Pan, Y. Yuan, and L. Song, "Molecule generation for drug design: a graph learning perspective," *arXiv preprint arXiv:2202.09212*, 2022.

[30] C. Zhang, K. Ding, J. Li, X. Zhang, Y. Ye, N. V. Chawla, and H. Liu, "Few-shot learning on graphs: A survey," *arXiv preprint arXiv:2203.09308*, 2022.

[31] A. Demir, T. Koike-Akino, Y. Wang, M. Haruna, and D. Erdogmus, "Eeg-gnn: Graph neural networks for classification of electroencephalogram (eeg) signals," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2021, pp. 1061–1067.

[32] G. Dong, M. Tang, Z. Wang, J. Gao, S. Guo, L. Cai, R. Gutierrez, B. Campbell, L. E. Barnes, and M. Boukhechba, "Graph neural networks in iot: A survey," *arXiv preprint arXiv:2203.15935*, 2022.

[33] Z. Chen, F. Chen, L. Zhang, T. Ji, K. Fu, L. Zhao, F. Chen, L. Wu, C. Aggarwal, and C.-T. Lu, "Bridging the gap between spatial and spectral domains: A survey on graph neural networks," *arXiv preprint arXiv:2002.11867*, 2020.

[34] J. M. Thomas, A. Moallemy-Oureh, S. Beddar-Wiesing, and C. Holzhüter, "Graph neural networks designed for different graph types: A survey," *arXiv preprint arXiv:2204.03080*, 2022.

[35] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[36] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[37] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[38] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *International Conference on Learning Representations*, 2021.

[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[41] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.

[42] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[43] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1416–1424.

[44] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.

[45] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, 2020, pp. 2704–2710.

[46] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and P. S. Yu, "A survey on heterogeneous graph embedding: methods," *Techniques, Applications and Sources*, 2020.

[47] C. Chen, W. Ma, M. Zhang, Z. Wang, X. He, C. Wang, Y. Liu, and S. Ma, "Graph heterogeneous multi-relational recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 3958–3966.

[48] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Comput. Surv.*, mar 2022, just Accepted.

[49] Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 349–357.

[50] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[51] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph neural networks for graphs with heterophily: A survey," *arXiv preprint arXiv:2202.07082*, 2022.

[52] R. Sato, "A survey on the expressive power of graph neural networks," *arXiv preprint arXiv:2003.04078*, 2020.

[53] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine, "Analyzing the expressive power of graph neural networks in a spectral perspective," in *International Conference on Learning Representations*, 2021.

[54] G. A. D'Inverno, M. Bianchini, M. L. Sampoli, and F. Scarselli, "A unifying point of view on expressive power of gnns," *arXiv preprint arXiv:2106.08992*, 2021.

[55] F. Geerts and J. L. Reutter, "Expressiveness and approximation properties of graph neural networks," in *International Conference on Learning Representations*, 2022.

[56] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. P. Silva, "The logical expressiveness of graph neural networks," in *International Conference on Learning Representations*, 2020.

[57] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *arXiv preprint arXiv:2012.15445*, 2020.

[58] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 12 241–12 252.

[59] J. Xia, Y. Zhu, Y. Du, and S. Z. Li, "A survey of pretraining on graphs: Taxonomy, methods, and applications," *arXiv preprint arXiv:2202.07893*, 2022.

[60] Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *arXiv preprint arXiv:2103.00111*, 2021.

[61] K. Zhou, Y. Dong, K. Wang, W. S. Lee, B. Hooi, H. Xu, and J. Feng, "Understanding and resolving performance degradation in deep graph convolutional networks," *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*,

2016, pp. 770–778.

[63] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.

[64] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[65] S. Luan, M. Zhao, X.-W. Chang, and D. Precup, "Break the ceiling: Stronger multi-scale deep graph convolutional networks," *Advances in neural information processing systems*, vol. 32, 2019.

[66] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.

[67] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[68] M. Tiezzi, G. Marra, S. Melacci, and M. Maggini, "Deep constraint-based propagation in graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 727–739, 2021.

[69] N. Keriven, "Not too little, not too much: a theoretical analysis of graph (over) smoothing," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2268–2281, 2022.

[70] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision gnn: An image is worth graph of nodes," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8291–8303, 2022.

[71] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *International Conference on Learning Representations*, 2020.

[72] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *International Conference on Learning Representations*, 2021.

[73] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Combining neural networks with personalized pagerank for classification on graphs," in *International Conference on Learning Representations*, 2019.

[74] A. Gravina, D. Bacciu, and C. Gallicchio, "Anti-symmetric DGN: a stable architecture for deep graph networks," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=J3Y7cgZOOS

[75] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 3950–3957.

[76] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style,high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8024–8035, 2019.

[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[78] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[79] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, 2014, pp. 232–238.