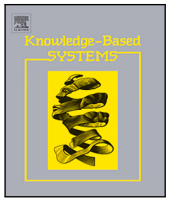
Knowledge-Based Systems 293 (2024) 111649

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Simple and deep graph attention networks

Guangxin Su [a], Hanchen Wang [b,*], Ying Zhang [c], Wenjie Zhang [a], Xuemin Lin [d]

[a] University of New South Wales, Sydney, 2052, Australia
[b] University of Technology Sydney, Ultimo, 2007, NSW, Australia
[c] Zhejiang Gongshang University, Hangzhou, 314423, China
[d] The Shanghai Jiao Tong University, Shanghai, 200240, China

ARTICLE INFO

Keywords:
Deep graph neural networks
Oversmoothing
Graph attention networks
Attention mechanism

ABSTRACT

Graph Attention Networks (GATs) and Graph Convolutional Neural Networks (GCNs) are two state-of-the-art architectures in Graph Neural Networks (GNNs). It is well known that both models suffer from performance degradation when more GNN layers are stacked, and many works have been devoted to address this problem. We notice that main research efforts in the line focus on the GCN models, and their techniques cannot well fit the GAT models due to the inherent difference between these two architectures. In GAT, the attention mechanism is limited as it ignores the overwhelming propagation from certain nodes as the number of layers increases. To sufficiently utilize the expressive power of GAT, we propose a new version of GAT named Layer-wise Self-adaptive GAT (LSGAT), which can effectively alleviate the oversmoothing issue in deep GAT and is strictly more expressive than GAT. We redesign the attention coefficients computation mechanism adaptively adjusted by layer depth, which considers both immediate neighboring and non-adjacent nodes from a global view. The experimental evaluation confirms that LSGAT consistently achieves better results on node classification tasks over relevant counterparts.

1. Introduction

Graph neural networks (GNNs) [1–3] have emerged as a promising tool for analyzing graph data, such as biochemical networks [4,5], social networks [6,7], and academic networks [8], etc.

In the realm of neural network architectures, depth, signifying the number of layers, is pivotal for performance in complex tasks, as evidenced in Convolutional Neural Networks (CNNs) applied in computer vision, which often incorporate dozens or even hundreds of layers [9]. However, GNNs utilized in various applications tend to be relatively shallow, typically comprising only a few layers. The shallow architecture could lead to the oversight of complex patterns in large-scale graphs. This limitation stems from challenges such as graph bottlenecks, over-squashing, and oversmoothing, which hinder the performance of deeper GNNs [10]. Towards the phenomenon of oversmoothing, which is axiomatically defined as the exponential convergence of suitable similarity measures on the node features, research attempts have been made to deepen current GNN architectures and understand their expressive power. Theoretical analysis investigates the deep GNNs in the perspective of expressive power [11–15], training difficulty [16–18] and generalization [19]. Deepening techniques in GNNs can be divided into three main categories [20]: those utilizing skip connections [21], graph normalization methods [16,22], and random dropping strategies [23].

Nevertheless, existing research efforts [11–15,17–22] mainly focus on the theoretical analysis and deepening techniques on GCNs, and few of them Dasoulas et al. [16] consider the attention mechanism in deep GATs. Their techniques cannot well fit the GATs models due to the inherent difference between these two architectures. Under the attention computation mechanism, GATs could compute the coefficients implicitly rather than explicitly as GCNs do, and we can use more information other than the topological information to determine each node's weight. However, we found that the performance degradation in deep GAT is caused by the overwhelming propagation from nodes with large degrees as the number of layers increases. Specifically, the feature of the node with a higher degree will be aggregated via exponentially increasing paths w.r.t. growing model depth, which will inevitably lead to the high similarity between representations of all nodes when the number of layers is large enough. The phenomenon of overwhelming propagation is emerging as a principal contributor to the oversmoothing problem. We characterize both as the exponential convergence of appropriate similarity measures applied to node features.

* Corresponding author.
E-mail addresses: guangxin.su@unsw.edu.au (G. Su), hanchen.wang@uts.edu.au (H. Wang), ying.zhang@zjgsu.edu.cn (Y. Zhang), wenjie.zhang@unsw.edu.au (W. Zhang), xuemin.lin@sjtu.edu.cn (X. Lin).

https://doi.org/10.1016/j.knosys.2024.111649
Received 27 August 2023; Received in revised form 23 January 2024; Accepted 16 March 2024
Available online 19 March 2024

To preliminarily verify the issue of overwhelming propagation in nodes with large degrees, in Section 4, we employ two intuitive yet effective neural network architectures, where nodes with a larger degree are simply reweighted with smaller weights. This approach is designed to constrain propagation in deeper layers, thereby providing a contrast to the vanilla GAT. While the results indicate a significant enhancement in performance, the overall effectiveness of these architectures does not yet fully meet our expectations.

Following the initial validation of the overwhelming propagation issues encountered in deep GAT, it remains elusive how to properly train the GAT to utilize the expressive power of the model sufficiently. Dasoulas et al. [16] has proposed a normalization technique based on Lipschitz continuity, which alleviates the gradient vanishing/explosion in deep GAT. In spite of Lipschitz normalization, the internal computing mechanism of GAT has not changed, and attention coefficients are still computed only among adjacent nodes, which means the overwhelming propagation problem still exists in deep GAT with Lipschitz normalization. Particularly, without a specifically designed framework, GAT is not capable of avoiding being affected by the oversmoothing issue due to its computation mechanism, *e.g.,* overwhelming propagation from the large-degree nodes. To better address this issue, we propose a simple and deep version of GAT named Layer-wise Self-Adaptive GAT (LSGAT). LSGAT has a newly designed attention coefficients computation mechanism, which considers the influence of both adjacent and non-adjacent nodes. LSGAT adaptively adjusts the attention coefficients with the layer depth and regularizes the coefficients for nodes with a large degree, which eventually alleviate potential oversmoothing issue significantly.

Contributions. The contributions of this paper can be summarized as follow:

- We demonstrate that overwhelming propagation from large degree nodes is the key factor to relieve the oversmoothing problem in deep GAT.
- Based on the experimental validation of the influence of overwhelming propagation problem in deep GAT, we further propose a new version of GAT named LSGAT, which considers both neighboring and non-adjacent nodes based on high-order proximity in the aggregation process, to train GAT properly and alleviate the oversmoothing issue.
- The effectiveness and versatility of LSGAT have been validated in our extensive experimental results for node classification on real-world datasets. Based on GAT, compared with the methods specifically designed for deep GAT, and multiple techniques designed for deep GNNs, LSGAT consistently exhibits superior performance. Even compared with the methods designed for relieving the oversmoothing problem equipped with GCN and ChebyNet, our LSGAT also has the best performance.

Roadmap. The rest of the paper is organized as follows: Section 2 presents the related works, and the preliminaries are introduced in Section 3. We introduce and verify the overwhelming propagation problem in Section 4. The details of proposed LSGAT are introduced in Section 5. The experimental results are reported in Section 6, and Section 7 concludes the paper.

2. Related work

In this section, we introduce the related works. Specifically, the deep graph neural network methods and graph attention network-based works are introduced.

Deep Graph Neural Networks. There has been an important line of research works that aim to relieve the oversmoothing issue. Inspired by ResNets [9], the methods that are based on skip-connection are proposed in [21,24–26] to exploit node representations from the preceding layers. Specifically, Chen et al. [27] improves the capacity of APPNP [28] by using initial residual and identity mapping in each layer. Another line is using normalization to re-scale node representations to constrain pairwise node distance [29–31]. In particularly, Zhou et al. [22] normalizes representations for nodes within the same group separately, and isolates node distributions among distinct groups to prevent oversmoothing. Random dropout methods [14,17] are connectivity-aware and graph-adaptive sampling approaches, which could address oversmoothing and over-fitting problems. Recently, Jin et al. [32] try to relieve the performance degradation problem in deep GNNs from a new perspective named overcorrelation. A series of works [14,19,33–35] have explored the underlying reasons for performance degradation towards mitigation solutions.

Graph Attention Networks. Various research works focus on designing the attention mechanism on graph neural networks for specific tasks and applications. In a synthetic issue requiring dynamic node selection, Brody et al. [36] developed a dynamic graph attention alternative which is strictly more expressive than GAT. In recommender systems, Wu et al. [37] developed dual graph attention networks to cooperatively learn representations for two-fold social impacts. Park et al. [38] proposed a new spatio-temporal graph attention paradigm with spatial attention and temporal attention for capturing the spatio-temporal dynamics in road networks. Wang et al. [39] developed multi-hop attention graph neural network, which calculated the attention between the given node and its multi-hop neighbors. Dasoulas et al. [16] tried to relieve the gradient explosion problem in deep GAT by introducing the Lipschitz normalization.

3. Preliminaries

Notations and setup. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ denotes the number of vertices, where each node $v_i \in \mathcal{V}$ is associated with a feature vector $h_i \in \mathbb{R}^d$, a layer outputs a new set of node representations $h_i' \in \mathbb{R}^{d'}$, and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $(v_j, v_i) \in \mathcal{E}$ denotes an edge from node v_j to node v_i, and its node degree is denoted as d_i. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ denote the adjacent matrix, and $\hat{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ denote the adjacent matrix with self-loop, *i.e.,* $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$.

3.1. Definition of oversmoothing

Definition 1 (*Rusch et al. [10]*)**.** Consider an undirected, connected graph $\mathcal{G}$ and let $\mathbf{X}^n \in \mathbb{R}^{v\times m}$ denote the hidden features at the nth layer of an N-layer Graph Neural Network (GNN) defined on $\mathcal{G}$. Define a **node-similarity measure** $\mu : \mathbb{R}^{v\times m} \to \mathbb{R}_{\geq 0}$ that adheres to the following axioms:

1. Constant Node Feature Axiom: There exists a constant vector $\mathbf{c} \in \mathbb{R}^m$ such that $\mathbf{X}_i = \mathbf{c}$ for all nodes i in $\mathcal{V}$ if and only if $\mu(\mathbf{X}) = 0$, for any $\mathbf{X} \in \mathbb{R}^{v\times m}$.

2. Sub-additivity Axiom: For all $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{v\times m}$, the inequality $\mu(\mathbf{X} + \mathbf{Y}) \leq \mu(X) + \mu(Y)$ holds.

Oversmoothing with respect to μ is then characterized by the layer-wise exponential convergence of the node-similarity measure μ to zero. Formally, this is defined as:

3. For $n = 0, \ldots, N$, it holds that $\mu(\mathbf{X}^n) \leq C_1 e^{-C_2 n}$ with some constants $C_1, C_2 > 0$.

3.2. Graph attention networks

Different from GCN and many other popular GNN architectures [3,27] that weigh all neighbors with equal importance (*e.g.,* mean and max-pooling as aggregation), GAT [2] enables (implicitly) specifying different weights to different nodes in a neighborhood. A scoring function e: $\mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ computes a score for every edge (v_j, v_i), which indicates the importance of the features of the neighbor v_j to node v_i:

$$e\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \text{LeakyReLU}\left(\boldsymbol{a}^{\mathrm{T}} \cdot \left[\boldsymbol{W}\boldsymbol{h}_i \parallel \boldsymbol{W}\boldsymbol{h}_j\right]\right) \tag{1}$$

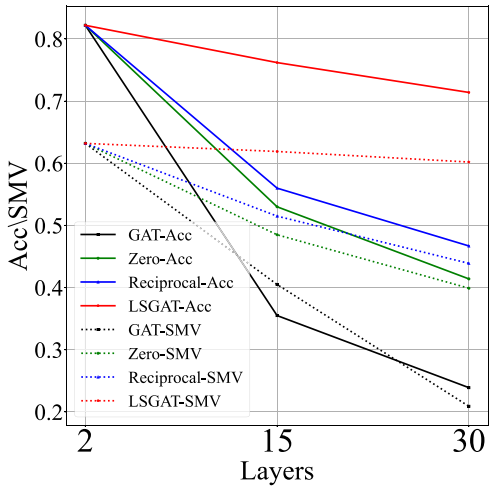


Fig. 1. Test accuracy and SMV between different architectures based on Cora dataset and equipped with GAT. Zero means Zero-GAT, Reciprocal means Reciprocal-GAT. The smaller SMV is, the smoother the node representations are.

where $\boldsymbol{a} \in \mathbb{R}^{2d'}$, $\boldsymbol{W} \in \mathbb{R}^{d' \times d}$ are trainable parameters, and $\|$ denotes vector concatenation. These attention scores are normalized across all neighbors using softmax, and the attention function is defined as:

$$\alpha_{ij} = \mathrm{softmax}_j\left(e\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right) = \frac{exp\left(e\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right)}{\sum_{v_{j'} \in \mathcal{N}_i} exp\left(e\left(\boldsymbol{h}_i, \boldsymbol{h}_{j'}\right)\right)} \tag{2}$$

We denote the coefficient matrix, whose entries are α_{ij}, if $(v_i, v_j) \in \mathcal{E}$, and 0 otherwise, as $\boldsymbol{C} \in \mathbb{R}^{n \times n}$. Then, GAT computes a weighted average of the transformed features of the neighbor nodes followed by a non-linearity σ as the new representation of v_i, using the normalized attention coefficients:

$$\boldsymbol{h}_i' = \sigma\left(\sum_{v_j \in \mathcal{N}_i} \alpha_{ij} \cdot \boldsymbol{W}\boldsymbol{h}_j\right) \tag{3}$$

where $\boldsymbol{h}_i'$ denotes the representation of node v_i in the next layer. In this paper, we refer to Eqs. (1) to (3) as the computation of each layer in an L-layer GAT.

4. Overwhelming propagation problem in deep GAT

In this section, we define and investigate the *overwhelming propagation problem* in the node classification task of deep GAT. GAT computes the attention coefficients α_{ij} for node v_i only based on the feature and topological relation with its neighbor nodes $v_j \in \mathcal{N}_i$. It is well known that the performance of GAT will decrease as the number of layers increases, and the phenomenon could be attributed to oversmoothing problem and overcorrelation problem [32].

Specifically in deeper GAT, the unique characteristics of the attention coefficients computation mechanism, which have not been considered in the previous work: the attention mechanism ignores the overwhelming propagation from nodes with large degree as model depth increases. That is, the node with higher degree will be aggregated via more paths that are exponentially increased *w.r.t.* growing model depth, and hence GAT is more likely to suffer from oversmoothing issue. Here, we hypothesize that the oversmoothing problem that happened in deep GAT could be relieved by extra consideration to constraint the propagation of nodes with large degree in the coefficients calculation mechanism, and we empirically verify that below.

Experimental Setup. We conduct experiments to observe the test accuracy and smoothness metric SMV [40]. Specifically, SMV uses normalized node representations to compute their Euclidean distance, and measures the oversmoothing problem in GNNs, the smaller SMV is, the smoother the node representations are. Here:

$$SMV(X) = \frac{1}{N(N-1)} \sum_{i \neq j} D(X_{i,:}, X_{j,:}) \tag{4}$$

where $X_{i,:}$ denotes the ith row of the node representation matrix $X \in \mathbb{R}^{N \times d}$. $D(\cdot, \cdot)$ is the normalized Euclidean distance between two vectors:

$$D(x, y) = \frac{1}{2}\left\| \frac{x}{\|x\|} - \frac{y}{\|y\|} \right\| \tag{5}$$

To validate whether we can relieve the oversmoothing problem in GAT by constraining the propagation of the nodes with large degree, our LSGAT and two intuitive GAT variants are compared here: Zero-GAT and Reciprocal-GAT. Zero-GAT means that the top 20% largest degree nodes will not be aggregated any more from the third layer. Reciprocal-GAT means that Eq. (1) will be updated as $e\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \mathrm{LeakyReLU}\left(\boldsymbol{a}^{\mathrm{T}} \cdot \frac{1}{d_j} \cdot \left[\boldsymbol{W}\boldsymbol{h}_i \parallel \boldsymbol{W}\boldsymbol{h}_j\right]\right)$, if the given node v_i is linked with node v_j, which belongs to the top 20% largest degree nodes from the third layer. The experiment is conducted on Cora dataset [41], which follows the data split way used in [1]. The reported values are the average results of five random experiments, and the parameter tuning space is unified.

Overwhelming Propagation Problem. From Fig. 1, we can see how seriously GAT is prone to be oversmoothing by the nodes with large degree in deeper layers. Compared with the performance of GAT with two layers, both the test accuracy and SMV of GAT are dramatically decreased as the number of layers increases. The intuitive architectures named Zero-GAT and Reciprocal-GAT both evidently improve the performance of vanilla GAT. Specifically, compared with vanilla GAT, the test accuracy and SMV of Reciprocal-GAT are both improved by nearly 80% (in percentage) as shown in Fig. 1. The experiments based on intuitive models further verify the existence and influence of overwhelming propagation problem. The performance results of Zero-GAT and Reciprocal-GAT are not satisfactory enough. Following this line, it is a critical challenge to redesign the coefficients calculation mechanism in GAT.

5. Layer-wise self-adaptive GAT

We first introduce two important definitions to demonstrate our proposed LSGAT.

Definition 2 (*Cong et al. [19]*)**.** Let $\mathcal{T}_i^L$ denote the L-layer computation tree rooted at node v_i, which represents the structured L-hop neighbors of node v_i, where the children of any node v_j in the tree are the nodes in $\mathcal{N}(i)$.

Fig. 2(a) illustrates an example of computation trees $\mathcal{T}_1^L$ and $\mathcal{T}_2^L$ rooted at nodes v_1 and v_2 with $L = 2$. Please note that these computation trees consider the self-loop of nodes. The colors of nodes in Fig. 2(a) indicate the degree of nodes, *i.e.,* darker color suggests higher degree. The shape of each node illustrate the *overlap-degree* of the corresponding node which is defined as follows:

Definition 3. Overlap-degree $d_{over}^{(l)}(j)$ is the number of paths that will be affected by the node v_j in layer l through GNNs propagation processes, i.e., $\boldsymbol{D}_{over}^{(l)} = \sum_{i'=1}^{n} \hat{\boldsymbol{A}}_{i'j}^{l} = \left[d_{over}^{(l)}(j)\right]_{n \times 1}$.

5.1. Proposed technique for addressing oversmoothing

It is well known that the information can also be aggregated to the non-adjacent nodes through layers of GNNs, *e.g.,* in Fig. 2(a), v_6 is not the neighbor of v_1, but its information is propagated to v_1 after two layers via the computation path $v_6 \rightarrow v_2 \rightarrow v_1$. The overlap-degree indicates the number of such computation paths, which

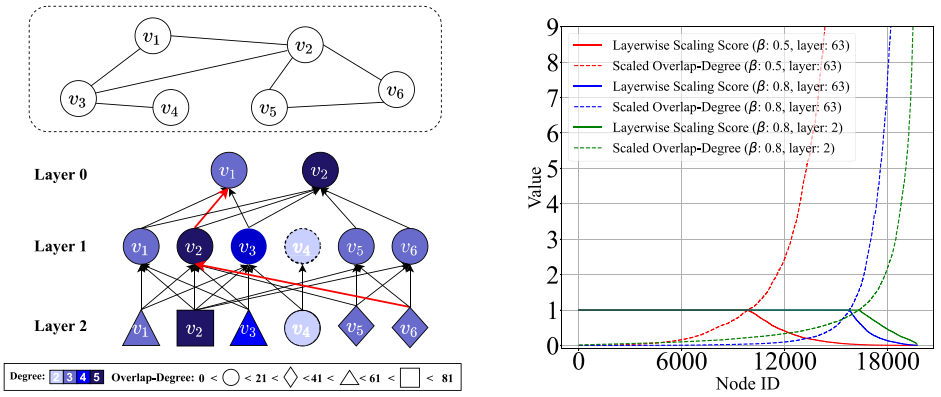


Fig. 2. (a) The computation trees rooted at v_1 and v_2 in a 2-layer GNN. Color and shape suggest the degree and *overlap-degree* of each node. (b) The layer-wise scaling score (lss) and scaled overlap-degree (d^*_{over}) varying number of layers l and hyperparamter β for nodes in Pubmed dataset.

exponentially grows *w.r.t.* node degree and number of GNN layers. For example, the overlap-degree of v_2 is 66 after two layers. The nodes with larger overlap-degree tend to be aggregated to more parent nodes via more paths in the computation tree. This phenomenon makes the representations of nodes be similar, and eventually degrade deep GNN's performance. The similar analysis and experimental results can also be found in [12,27,40].

Zero-GAT directly enforces the top 20% largest degree nodes not to be aggregated any more from the third layer, which overly restricts the propagation of the nodes with large degree. The potential disadvantages exist in Reciprocal-GAT will be discussed in Section 5.2. In this work, we aim to relieve oversmoothing problem and reduce the redundant information that come from large degree nodes by regularizing their attention coefficients. The regularization is adaptively adjusted with the depth of model to set higher importance to shallow layers. Here, we first give a self-adaptive threshold value τ to identify the nodes with higher overlap-degree.

$$\tau = \boldsymbol{D}^{(l)}_{over}{}^{(\eta^{(l)}\text{th}} \tag{6}$$

Here, τ is $\eta^{(l)}$th value of $\boldsymbol{D}^{(l)}_{over}$ in an ascending order. $\eta^{(l)}$ is defined as $\eta^{(l)} = \beta n + \frac{(1-\beta)n}{e^l}$ where $\beta \in [0,1]$ is a hyperparameter that determines the proportion of nodes to be regularized. Then, the scaled overlap-degree matrix is computed as follows:

$$\boldsymbol{D}^{(l)*}_{over} = \boldsymbol{D}^{(l)}_{over} / \tau \tag{7}$$

With the scaled overlap-degree matrix, the layer-wise scaling score, denoted as lss, is computed with the following equation:

$$lss^{(l)}_{over}(j) = \begin{cases} 1 & d^{(l)*}_{over}(j) \le 1 \\ 1/d^{(l)*}_{over}(j) & d^{(l)*}_{over}(j) > 1 \end{cases} \tag{8}$$

The layer-wise scaling scores for nodes in Pubmed dataset varying β and layer l are illustrated in Fig. 2. Then the layer-wise scaling scores are multiplied with the scores computed in Eq. (1), *i.e.*, $\boldsymbol{a}^{\mathrm{T}} \cdot \left[\boldsymbol{W}\boldsymbol{h}_i \parallel \boldsymbol{W}\boldsymbol{h}_j\right]$, followed by LeakyReLU and a Softmax function. Specifically, for a given node v_i, Softmax function is applied on values activated by LeakyReLU between all neighboring nodes of node v_i, *i.e.*,$v_j \in \mathcal{N}_i$, and v_i itself. As a result, the computation of the attention coefficient in LSGAT is formulated as follows:

$$\alpha^*_{ij} = softmax_j(\text{LeakyReLU}(lss^{(l)}_{over}(j) \cdot \boldsymbol{a}^{\mathrm{T}} \cdot \left[\boldsymbol{W}\boldsymbol{h}_i \parallel \boldsymbol{W}\boldsymbol{h}_j\right])) \tag{9}$$

Please note that the layer-wise scaling score is applied before the LeakyReLU and Softmax functions as a "soft" regulation to avoid the loss of information from the nodes with large overlap-degree. Detailed discussion is provided in Section 5.2.

In this paper, we denote the matrix of the attention coefficients α^*_{ij} as $\boldsymbol{C}^{*(l)} \in \mathbb{R}^{n\times n}$ at lth layer. Finally, the representation computation of node v_i in each layer with LSGAT is as:

$$\boldsymbol{h}'_i = \sigma\left(\sum_{v_j \in \mathcal{N}_i} \alpha^*_{ij} \cdot \boldsymbol{W}\boldsymbol{h}_j\right) \tag{10}$$

In matrix format, LSGAT can also be formulated as $\boldsymbol{H}^{(l+1)} = \sigma(\boldsymbol{C}^{*(l)} \boldsymbol{W}^{(l)} \boldsymbol{H}^{(l)})$.

5.2. Discussion

In this section, we would like to provide further discussions about the characteristics of LSGAT and the comparisons with the existing works.

Choice and utilization of overlap-degree. One may wonder why the overlap-degree is chosen in our model rather than degree. The layer-wise aggregations in GNNs can enlarge the influence of the nodes from deeper layers. If only consider degree of nodes layer-by-layer, the global high-order information cannot be fully utilized in the computation. As the number of layers increases, the extent of overwhelming propagation influence is divergent and should be treated differently and adaptively with the number of layers, and $\eta^{(l)}$ is used in LSGAT. Furthermore, the layer-wise scaling scores based on the overlap-degree could be smoother in terms of continuity as shown in Fig. 2 than that based on degree. For example, the proportion of the largest group in Pubmed dataset [41] with the same degree is 46%, which means a large number of nodes will share the same scaling score in each aggregation. As a comparison, the layer-wise scaling scores based on the overlap-degree are different from each other and more helpful for training a discriminative model. Besides, the layer-wise scaling scores can also be directly multiplied with the attention coefficient, *i.e.*, $\alpha^*_{ij} = \alpha_{ij} \cdot lss^{(l)}_{over}(j)$. However, we found that this variant of our model would significantly ignore the information from the nodes with large overlap-degree which are typically of great importance in the graph. Therefore, we provide a "soft" regulation of the attention coefficients based on overlap-degree for these nodes as shown in Eq. (9). The comparison in Fig. 1 also verifies that the performance of LSGAT is much better than the performance of Reciprocal-GAT.

Comparison with prior works. Zhou et al. [22] shared the similar intuition as ours. They claim that most studies focus on performance degradation problem based on immediate neighboring relationships, but ignore the global graph structural information in each layer. Zhou et al. [22] tackles the oversmoothing problem by making the representations similar for nodes that are in the same class and differentiating for the nodes that are not. However, the information of nodes in a

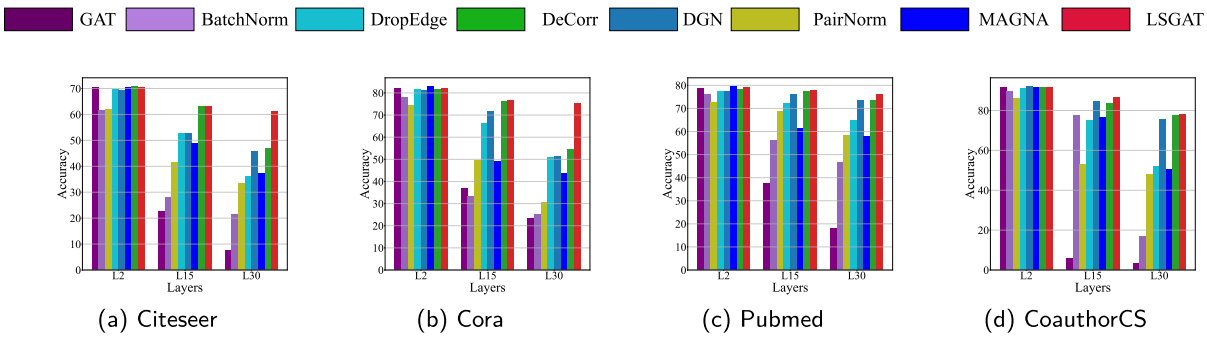


Fig. 3. The comparison of test accuracies between LSGAT and other general deep graph neural network methods, which are equipped with Graph Attention Networks (GAT).

Table 1
Dataset statistics.

Dataset	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19,717	44,338	500	3
Physics	34,493	247,962	8415	5
CoauthorCS	18,333	81,894	6805	15
Ogbn-Arxiv	169,343	1,166,243	128	40

fully connected graph could be aggregated to all other nodes in GNNs if the network is deep enough, regardless the nodes belonging to the same class or not. Thus, in our work, we utilize global information to regularize the coefficient during aggregation.

The method in [23] is based on the random drop of edges, which effectively prevents oversmoothing in deep GNNs. Compared with the random dropout, LSGAT is based on the premise of retaining as much information of nodes as possible, while weakens the feature expression of the nodes that are more likely to suffer from the oversmoothing problem, and hence improves the generalization ability of the model.

Versatility of our proposed mechanism. Our proposed LSGAT conducts the regulation on the aggregation process without modifying the model architecture, which allows LSGAT to be integrated with various existing GNN deepening techniques such as graph normalization [22], GAT-Lip [16], and random dropping [23] to further improve the performance of deep GNNs. In our experiments, we show that with skip-connection and identity mapping [27,42], LSGAT achieves the state-of-the-art performance for node classification task.

6. Experiment

In this section, we evaluate the performance of our LSGAT in real-world benchmarks under semi-supervised learning mainly on the following research questions:

RQ1. Whether LSGAT outperforms other state-of-the-art deep GAT models?

RQ2. When applied on GAT, how is the performance comparison between LSGAT and other deep GNN techniques?

RQ3. When applied on GCN and ChebyNet, could LSGAT outperform other algorithms, which are specifically designed to alleviate the oversmoothing issue that happened in deep GCNs?

RQ4. How good is the versatility of our proposed LSGAT?

Specifically, we first introduce the experimental settings in Section 6.1. To answer **RQ1**, in Section 6.2, we compare LSGAT with the state-of-the-art models which are specifically designed for deep GATs on node classification tasks. To answer **RQ2**, in Section 6.3, we compare LSGAT with proposed deep GNNs techniques and multi-hop methods for node classification tasks, which are all applied on GAT. The comparisons between LSGAT and other deep GNNs methods which are designed to alleviate the oversmoothing issue based on GCN and ChebyNet have also been included in this section to answer **RQ3**. To validate the versatility of LSGAT (**RQ4**), in Section 6.4, we report the state-of-the-art performance of LSGAT with skip-connection and identity mapping. To verify whether LSGAT and GAT-Lip could improve the performance at the same time, the combined performance of LSGAT and GAT-Lip has been analyzed in this section. Furthermore, the hyper-parameter study of LSGAT has been reported in Section 6.5.

6.1. Experiment setup

All the codes are implemented in Python 3 and Pytorch 1.11.0, and running on one NVIDIA Quadro RTX 2080Ti GPUs and one NVIDIA Quadro RTX A6000 GPUs serve with CUDA 11.6. All models are trained with a maximum of 1000 epochs using the Adam optimizer [43] and early stopping. Weights in GATs models are initialized with Glorot algorithm [44].

Datasets. Joining the practice of previous work, we evaluate GNN models by performing the node classification tasks on six real-world datasets: Cora, Citeseer, Pubmed [41], CoauthorPhysics, CoauthorCS [45], and Ogbn-Arxiv [46]. The statistics of these datasets and data splits could be found in Table 1.

Baseline Methods and Models. To our best knowledge, GAT-Lip is the only work proposed to improve the performance in deep GAT. The methods try to relieve oversmoothing problem in deep GNNs include: PairNorm [30], BatchNorm [29], DGN [22], and DropEdge [23]. As a variant of GATs, MAGNA [39] incorporates multi-hop context information into every layer of attention computation. DeCorr [32] is proposed to help enable deeper GNNs from a feature overcorrelation perspective.

We consider three basic GNN models, GAT [2], GCN [1], and ChebyNet [47]. Besides, as one of the state-of-the-art GNN variants, it is meaningful to see whether GCNII will perform better based on our LSGAT. Specifically, in GCNII, the computation in lth layer is defined as $\boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{A}}\tilde{\boldsymbol{W}})$, where $\tilde{\boldsymbol{A}} = \left(\left(1-\alpha_l\right)\hat{\boldsymbol{A}}_{\odot}\boldsymbol{H}^{(l)} + \alpha_l\boldsymbol{H}^{(0)}\right)$, $\hat{\boldsymbol{A}}_{\odot} = \hat{\boldsymbol{D}}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I}_n)\hat{\boldsymbol{D}}^{-\frac{1}{2}}$, $\tilde{\boldsymbol{W}} = \left(\left(1-\beta_l\right)\boldsymbol{I}_n + \beta_l\boldsymbol{W}^{(l)}\right)$, α_l and β_l are two hyperparameters that adjust initial residual and identity mapping respectively.

Implementations. We have strictly followed the experiment settings of previous works in each comparison section. Specifically, in Section 6.2, we follow the setting used in [16], and we directly report the best performance of GAT and GAT-Lip shown in its work. The results of Section 6.3 are based on the best performance and setting reported in [22,32,39]. Furthermore, the techniques such as ResNet [9] and LayerNorm [48] which were used in MAGNA are removed for a fair comparison. In Section 6.4, we follow the setting used in [27] and report the best results shown in the work.

GAT-Lip Settings. We evaluate the performance of GNN models with respect to increasing model depth for the node classification task. We used again the Adam optimizer [43] with a weight decay $L = 5\times10^{-4}$ and the initial learning rate was set in $\{0.1, 0.01, 0.005, 0.001\}$.

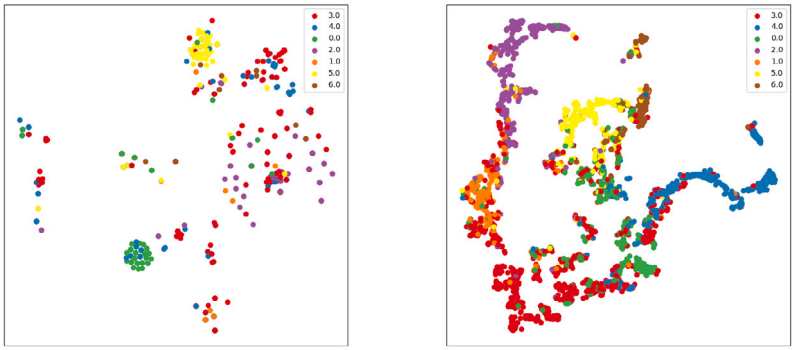


Fig. 4. (a) The t-SNE visualization of node representation for GAT on Cora dataset in 30 layers (node colors represent classes). (b) The t-SNE visualization of node representation for LSGAT on Cora dataset in 30 layers (node colors represent classes).

- **Model Selection.** We performed for all models and datasets cross-validation with predefined train/validation/test splits and reported the best-achieved validation accuracy.
- **Model Depth.** In order to examine the model behavior under the depth increase, for each architecture we used models consisting of l GNN layers, where $l \in \{2, 5, 10, 15, 20, 25, 30\}$. We run each experiment 5 times and we keep the configuration with the best average accuracy.
- **GAT Hyper-parameter tuning.** For each model depth and GNN model, we performed grid search for hyper-parameter tuning. The hyper-parameters of GAT were tuned as the following: The dimensionality of the hidden units was set in $\{8, 16, 64, 128\}$. The number of attention heads was selected between $\{1, 2, 4, 8\}$ and we experimented over two standard aggregators of the attention heads: (a) concatenation and (b) averaging of the attention heads. The dropout of the attention weights was set in $\{0, 0.2, 0.5\}$.

Experiments settings of Section 6.3 are strictly followed the work [32]. Specifically, for BatchNorm, PairNorm and DGN, we reuse the performance reported in [22] for GCN and GAT. For ChebyNet, we use their best configuration to run the experiments. For DropEdge, we tune the sampling percent from $\{0.1, 0.3, 0.5, 0.7\}$, weight decay from $\{0, 5e{-}4\}$ dropout rate from $\{0, 0.6\}$ and fix the learning rate to be 0.005. For DeCorr, we reuse the best results shown in [32]. For MAGNA, we use their basic configuration to run the experiments, however, for a fair comparison, we remove the resnet [9] and LayerNorm used its original work, we also limited the maximum hidden number to 128, which is used in other works under attention mechanism. For our work, We set the number of hidden units as $\{8, 16, 32, 64, 128\}$. The number of attention heads was selected between $\{1, 2, 4, 8\}$. We tune the hyperparameters for all datasets from the following sets: $\{0, 0.1, \ldots, 0.6\}$ (dropout rate), $5{\times}10^{\{-3,-4\}}$ (learning rate), $5{\times}10^{\{-3,-4,-5\}}$ (L2 regularization).

For experiments in Section 6.4, we directly use the best performance and settings of related works reported in GCNII [27].

GCNII Settings. 0.1 (α_l for initial residual), 5×10^{-4} (L2 regularization), and other hyperparameters are tuned by grid search. The experiments are randomly repeated for ten times, and the average accuracy and the standard deviation are reported.

6.2. Comparison with GAT-based algorithms

We evaluate the performance of the proposed LSGAT and existing deep GATs methods *w.r.t.* the increasing number of layers for node classification on kinds of real-world datasets. The mean values of results with standard deviations are shown in Table 2, where the "Variation" denotes the accuracy gap between models with 2 and 30 layers. The underlined values are the best results through all models and layers on the specific dataset. From Table 2, it can be observed from variation that LSGAT has remarkably alleviated the oversmoothing issue that happened in GAT as the number of layers increased. Simultaneously, LSGAT could achieve the best performance among all layers compared with vanilla GAT and GAT-Lip in most comparisons. The accuracy of LSGAT is greatly higher than other methods as depth equals to 30 in all datasets. Particularly, on CoauthorPhysics dataset, the accuracy of our method LSGAT is 87.0% which is much better than the previous best performance by GAT-Lip (63.9%). Furthermore, LSGAT performs the best through almost all corresponding numbers of layers, specifically in Citeseer, Pubmed, CoauthorPhysics, and Ogbn-Arxiv datasets. At the same time, LSGAT consistently demonstrated improved performance with an increased number of layers across four out of five datasets. Particularly noteworthy is its performance on the Pubmed dataset. Here, it was observed that LSGAT consistently outperformed other models when the number of layers was fewer than 25. This trend highlights the efficacy of LSGAT in leveraging deeper architectures for enhanced performance in most cases. In conclusion, considering both the stability and accuracy through all layers, LSGAT significantly outperforms the baseline methods, especially when the number of layers is large enough. What is more, as a new version of GAT, we can see that the performance gap between LSGAT and GAT has dramatically increased as the number of layers increases. Intuitive classification results between vanilla GAT and LSGAT with 30 layers can be found in Fig. 4.

6.3. Comparison with other deep GNN algorithms

To further validate the enhanced performance of LSGAT, which is developed with a focus on mitigating the overwhelming propagation issue, we first analyze the comparison between the state-of-the-art algorithms designed for deep GNNs and our LSGAT based on GAT to verify the performance. The deep GNNs algorithms compared here include: PairNorm [30], BatchNorm [29], DGN [22], DropEdge [23], MAGNA [39] and DeCorr [32]. As shown in Fig. 3, LSGAT consistently exhibits superior performance. Especially the results with mainly compared layers: fifteen layers and thirty layers, among sixty-four cases based on eight algorithms, our LSGAT achieves the best performance in sixty-three cases (63/64). Specifically, in the examination of the Citeseer and Cora datasets, particularly at the 30th layer, the performance of our model (denoted as LSGAT) surpasses the second-best method by a significant margin, approximately 15%–20%. The comparison implies that with the proper consideration of overwhelming propagation, GAT itself could be effective in both shallow and deep layers.

Additionally, we compare LSGAT with other advanced deep GNN methods aimed at addressing the oversmoothing issue, such as Pair-

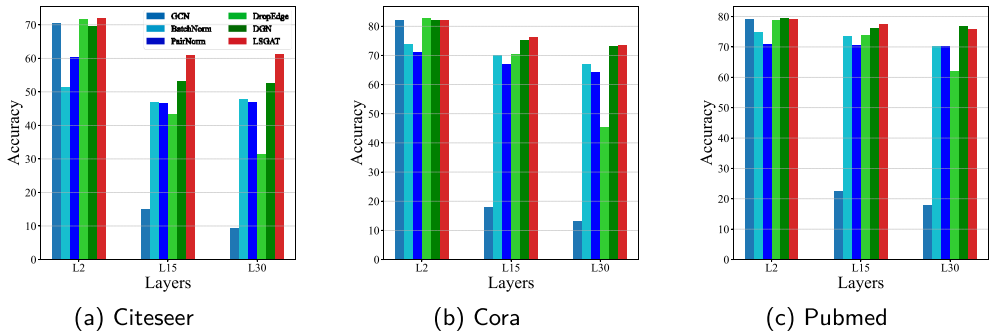


Fig. 5. The comparison of test accuracies between LSGAT and other general deep graph neural network methods, which are equipped with Graph Convolutional Neural Network (GCN).

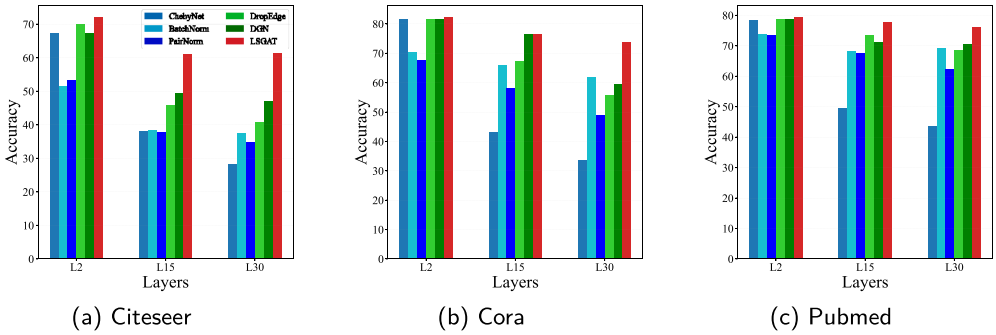


Fig. 6. The comparison of test accuracies between LSGAT and other general deep graph neural network methods, which are equipped with ChebyNet.

Table 2
Summary of classification accuracy (%) results among deep GAT based methods.

Dataset	Method	Layers							Variation
		2	5	10	15	20	25	30	
Cora	GAT	**82.2 ± 1.1**	78.9 ± 1.0	57.8 ± 0.6	35.5 ± 1.1	32.2 ± 1.1	30.0 ± 1.2	**23.9 ± 0.6**	58.3
	GAT-Lip	**82.2 ± 0.6**	83.3 ± 2.2	80.7 ± 1.1	78.8 ± 1.2	76.6 ± 0.6	71.6 ± 1.1	**68.8 ± 2.0**	13.4
	LSGAT	**82.2 ± 1.0**	79.1 ± 1.3	77.5 ± 0.6	76.4 ± 1.1	76.2 ± 0.8	74.8 ± 1.5	**73.5 ± 0.8**	8.70
Citeseer	GAT	**66.8 ± 0.4**	65.0 ± 1.1	62.9 ± 0.7	61.2 ± 2.8	60.9 ± 1.1	59.9 ± 2.1	**56.1 ± 3.1**	10.7
	GAT-Lip	**67.1 ± 0.8**	65.9 ± 1.6	62.6 ± 1.8	62.1 ± 1.7	60.1 ± 1.5	60.9 ± 2.5	**59.4 ± 3.8**	7.70
	LSGAT	**67.6 ± 0.8**	68.1 ± 0.9	63.8 ± 0.8	60.9 ± 2.5	62.4 ± 1.3	62.5 ± 0.4	**61.2 ± 1.4**	6.40
Pubmed	GAT	**76.3 ± 1.9**	78.1 ± 1.1	64.5 ± 1.0	57.4 ± 0.5	51.5 ± 1.4	48.8 ± 1.4	**29.5 ± 1.1**	46.8
	GAT-Lip	**77.6 ± 1.2**	78.2 ± 0.7	75.4 ± 0.8	72.4 ± 0.5	73.2 ± 0.8	67.7 ± 1.1	**65.0 ± 1.5**	12.6
	LSGAT	**76.5 ± 0.5**	77.0 ± 0.8	78.6 ± 1.2	77.6 ± 0.8	77.4 ± 0.7	76.2 ± 1.5	**72.8 ± 1.4**	3.70
Physics	GAT	**93.2 ± 0.6**	91.0 ± 0.1	88.3 ± 0.1	77.0 ± 10	50.0 ± 16	15.3 ± 0.3	**13.6 ± 0.2**	79.6
	GAT-Lip	**93.4 ± 0.2**	91.6 ± 1.2	90.4 ± 0.8	84.2 ± 4.3	72.6 ± 8.7	71.7 ± 7.6	**63.9 ± 1.3**	29.5
	LSGAT	**93.7 ± 0.4**	92.1 ± 0.6	91.7 ± 0.3	91.5 ± 0.4	91.2 ± 1.0	91.0 ± 0.4	**87.0 ± 2.7**	6.70
Ogbn-arxiv	GAT	**72.2 ± 2.4**	72.5 ± 2.0	67.8 ± 2.6	59.5 ± 0.7	53.9 ± 0.8	52.9 ± 0.3	**31.4 ± 2.1**	40.8
	GAT-Lip	**72.0 ± 2.0**	72.3 ± 4.4	72.4 ± 2.4	69.7 ± 1.7	67.3 ± 2.1	66.8 ± 2.5	**62.2 ± 1.8**	9.80
	LSGAT	**72.2 ± 2.2**	72.7 ± 3.5	71.8 ± 2.7	70.5 ± 0.4	67.9 ± 2.1	67.3 ± 3.1	**64.4 ± 1.2**	7.80

Norm [30], BatchNorm [29], DGN [22], and DropEdge [23], which are primarily based on the GCN and ChebyNet models. The results, depicted in Figs. 5 and 6, clearly demonstrate that LSGAT consistently outperforms these five algorithms. This is particularly evident in the Citeseer dataset, where LSGAT's performance consistently surpasses the best outcomes of other baseline methods by about 10%, especially notable at fifteen and thirty layers. The findings indicate that LSGAT markedly outperforms other techniques specifically tailored for GCN and ChebNet models. This positions LSGAT as an advantageous choice for future research and practical applications, particularly for complex tasks requiring substantial graph models with deep layers.

6.4. Combining with other deep GNN methods

Combining with GCNII. As one of the state-of-the-art deep GCN variants, the authors of GCNII suggested in [27] to have a try of new version of GCNII, which includes attention mechanism. In this section, we further investigate the performance of LSGAT with initial residual and identity mapping introduced in GCNII, named as LSGAT-GCNII. The results are demonstrated in Table 3, where parentheses include the number of layers of the model that achieves the best performance. As shown in Table 3, LSGAT-GCNII further enhances the performance with new state-of-the-art results on several datasets. As highlighted in Section 4 and mentioned in GCNII, the nodes with large degrees are more likely to suffer from oversmoothing problem. With redesigned self-attention mechanism, the component $\hat{\boldsymbol{A}}_{\odot}\boldsymbol{H}^{(l)}$ in GCNII is replaced by $\boldsymbol{C}^{*}\boldsymbol{H}^{(l)}$ in LSGAT-GCNII, which is much more sensitive to the nodes with high overlap-degrees during aggregation, and better relieve the oversmoothing problem. LSGAT-GCNII fills the gap and improves the performance of GCNII based on the theory that nodes with high degrees are more likely to lead to the oversmoothing problem, which is not addressed in GCNII. Therefore, through all datasets, LSGAT-GCNII

Table 3
Comparison results of test accuracy (%) between GCNII and LSGAT-GCNII.

Dataset	Method	Layers						Best
		2	4	8	16	32	64	
Cora	GCNII	82.2	82.6	84.2	84.6	85.4	85.5	85.5 ± 0.5(64)
	LSGAT-GCNII	83.8	83.9	84.5	85.0	85.6	85.5	85.6 ± 0.7(32)
	Improvement	+1.6	+1.3	+0.3	+0.4	+0.2	+0.0	+0.1
Citeseer	GCNII	68.2	68.9	70.6	72.9	73.4	73.4	73.4 ± 0.6(32)
	LSGAT-GCNII	71.6	72.3	73.3	73.2	73.4	72.4	73.4 ± 0.8(32)
	Improvement	+3.4	+3.4	+2.7	+0.3	+0.0	−1.0	+0.0
Pubmed	GCNII	78.2	78.8	79.3	80.2	79.8	79.7	80.2 ± 0.4(16)
	LSGAT-GCNII	79.2	79.3	79.4	79.6	79.8	80.4	80.4 ± 0.5(64)
	Improvement	+1.0	+0.5	+0.1	−0.6	+0.0	+0.7	+0.2

Table 4
Test accuracies (%) of LSGAT based on different β w/wo GAT-Lip.

Dataset	Method	Layers						
		2	5	10	15	20	25	30
Cora	LSGAT 0.2	80.5 ± 1.1	79.1 ± 1.3	77.5 ± 0.6	76.1 ± 1.6	75.8 ± 1.8	74.8 ± 1.5	73.5 ± 0.8
	LSGAT + Lip 0.2	80.5 ± 0.8	79.1 ± 1.3	77.1 ± 0.6	76.0 ± 1.3	75.8 ± 1.5	74.8 ± 1.5	69.3 ± 6.2
	LSGAT 0.4	80.5 ± 0.4	78.9 ± 1.0	77.2 ± 0.5	76.3 ± 0.5	75.5 ± 0.4	74.6 ± 2.2	69.2 ± 1.6
	LSGAT + Lip 0.4	80.1 ± 0.6	78.9 ± 0.5	77.6 ± 1.0	76.4 ± 1.4	75.3 ± 1.8	74.2 ± 0.8	72.4 ± 2.7
	LSGAT 0.6	80.6 ± 1.3	78.7 ± 1.0	77.2 ± 1.6	76.4 ± 1.1	76.2 ± 0.8	74.5 ± 2.3	71.2 ± 3.0
	LSGAT + Lip 0.6	80.1 ± 0.3	78.7 ± 1.2	77.3 ± 1.0	76.8 ± 0.7	75.3 ± 1.0	74.0 ± 1.5	70.2 ± 3.1
	LSGAT 0.8	80.4 ± 1.0	78.6 ± 0.7	77.2 ± 0.7	76.2 ± 0.7	75.9 ± 0.6	74.3 ± 2.5	71.4 ± 2.1
	LSGAT + Lip 0.8	80.2 ± 0.4	78.8 ± 0.8	77.1 ± 0.8	77.0 ± 1.9	75.2 ± 1.1	75.9 ± 1.3	73.0 ± 3.4
Pubmed	LSGAT 0.2	76.1 ± 1.5	76.1 ± 0.7	76.3 ± 0.5	77.3 ± 1.1	76.9 ± 0.9	75.9 ± 1.9	72.8 ± 1.4
	LSGAT + Lip 0.2	76.2 ± 0.8	76.0 ± 1.1	76.4 ± 1.6	77.0 ± 0.6	76.9 ± 1.0	76.2 ± 1.6	71.9 ± 4.4
	LSGAT 0.4	76.3 ± 0.5	76.9 ± 1.1	75.9 ± 1.5	76.8 ± 1.4	77.4 ± 0.7	75.7 ± 2.0	73.2 ± 8.0
	LSGAT + Lip 0.4	76.5 ± 0.8	76.7 ± 1.6	76.8 ± 1.2	76.8 ± 1.5	78.0 ± 0.8	76.4 ± 1.0	75.7 ± 3.0
	LSGAT 0.6	76.2 ± 0.7	76.7 ± 0.8	76.6 ± 1.1	77.3 ± 0.3	77.0 ± 1.3	75.7 ± 0.8	72.1 ± 2.4
	LSGAT + Lip 0.6	76.3 ± 1.8	76.5 ± 0.8	76.8 ± 2.9	76.9 ± 1.0	77.5 ± 1.1	76.1 ± 0.8	72.0 ± 2.6
	LSGAT 0.8	76.5 ± 0.5	77.0 ± 0.8	76.9 ± 1.2	77.6 ± 0.8	76.7 ± 1.0	76.2 ± 1.5	71.3 ± 3.4
	LSGAT + Lip 0.8	76.4 ± 1.2	76.7 ± 0.5	77.0 ± 0.6	77.0 ± 1.0	77.7 ± 0.5	76.7 ± 0.9	71.7 ± 2.2

Table 5
Test accuracies (%) of LSGAT based on different β.

Dataset	Method	Layers		
		10	20	30
Cora	LSGAT 0.2	77.5 ± 0.6	75.8 ± 1.8	73.5 ± 0.8
	LSGAT 0.4	77.2 ± 0.5	75.5 ± 0.4	69.2 ± 1.6
	LSGAT 0.6	77.2 ± 1.6	76.2 ± 0.8	71.2 ± 3.0
	LSGAT 0.8	77.2 ± 0.7	75.9 ± 0.6	71.4 ± 2.1
Pubmed	LSGAT 0.2	78.3 ± 0.5	76.9 ± 0.9	72.8 ± 1.4
	LSGAT 0.4	77.9 ± 1.5	77.4 ± 0.7	73.2 ± 8.0
	LSGAT 0.6	78.6 ± 1.2	77.0 ± 1.3	72.1 ± 2.4
	LSGAT 0.8	77.9 ± 1.2	76.7 ± 1.0	71.3 ± 3.4

generally outperforms GCNII. Specifically, the improvement brought by LSGAT can be up to 3.4% on Citeseer. Notably, with no more than sixteen layers, the improvement of LSGAT-GCNII is significant. As number of layers increases, the improvement generally shrinks, whose reason is in the computation process of GCNII. As introduced in Section 6.1, because of the initial residual connection, the fraction of information from lth layer, *i.e.,* $\boldsymbol{C}^{*}\boldsymbol{H}^{(l)}$ in LSGAT-GCNII, dramatically reduces when l increases. The performance improvement brought by LSGAT reduces accordingly.

Combining with GAT-Lip. To address the gradient explosion issue that happened in deep GAT, Dasoulas et al. [16] proposed GAT-Lips, which could be applied to GAT and our LSGAT. From the results shown in Table 4, we can see that both algorithms contribute to the improvement of performance, and the performance is further significantly improved on the original GAT and LSGAT.

6.5. Parameter study

In this subsection, we take a deeper look at the proposed LSGAT to find whether there exists the best proportion (β in Section 5) between large degree nodes and small degree nodes in each setting. As shown in Table 5, the performances based on different proportion are all good enough compared with vanilla GAT ($\beta = 0.0$). This further verifies that the performance of deep GAT could be improved by constraining the nodes with a large degree in deeper layers. Simultaneously, our analysis reveals no positive correlation between the value of β and overall outcomes, indicating the necessity of adjusting β as a hyperparameter in future experiments.

7. Conclusion

In this paper, we investigated that oversmoothing problem happened in deep GAT could be relieved by considering overwhelming propagation caused by the nodes with large degree.

Then, we propose a novel and versatile coefficient computation mechanism LSGAT to properly train GAT. This mechanism could rescale the propagation influence based on overlap-degree from adjacent and non-adjacent nodes adaptively with the number of layers, and specifically limit the propagation of nodes with large degrees to relieve the oversmoothing problem in deep GAT. Specifically, LSGAT does not change the architecture of GAT, and our layer-wise scaling scores could be calculated offline and easily applied to GAT in the training phase. The results of extensive experiments on various real-world datasets show the advantage of our proposed method over the baselines. Specifically, with initial residual and identity mapping, our proposed LSGAT achieves the state-of-the-art performance as a deep GNN.

CRediT authorship contribution statement

Guangxin Su: Writing – original draft, Validation, Software, Conceptualization. **Hanchen Wang:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Ying Zhang:** Writing – review & editing. **Wenjie Zhang:** Writing – review & editing, Supervision. **Xuemin Lin:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

Ying Zhang is supported by ZJNSF LY21F020012, and Wenjie Zhang is supported by ARC FT210100303.

References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[3] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[4] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, Y. Shen, Graph contrastive learning with augmentations, Adv. Neural Inf. Process. Syst. 33 (2020) 5812–5823.
[5] H. Wang, D. Lian, Y. Zhang, L. Qin, X. Lin, GoGNN: graph of graphs neural network for predicting structured entity interactions, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 1317–1323.
[6] X. Huang, Q. Song, Y. Li, X. Hu, Graph recurrent networks with attributed random walks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 732–740.
[7] X. Dong, B. Jin, W. Zhuo, B. Li, T. Xue, Improving sequential recommendation with attribute-augmented graph neural networks, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2021, pp. 373–385.
[8] H. Gao, Z. Wang, 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.
[9] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[10] T.K. Rusch, M.M. Bronstein, S. Mishra, A survey on oversmoothing in graph neural networks, 2023, arXiv preprint arXiv:2303.10993.
[11] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, D. Koutra, Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks, 2021, arXiv preprint arXiv:2102.06462.
[12] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[13] K. Oono, T. Suzuki, Graph neural networks exponentially lose expressive power for node classification, in: International Conference on Learning Representations, 2019.
[14] W. Huang, Y. Rong, T. Xu, F. Sun, J. Huang, Tackling over-smoothing for general graph convolutional networks, 2020, arXiv preprint arXiv:2008.09864.
[15] C. Cai, Y. Wang, A note on over-smoothing for graph neural networks, 2020, arXiv preprint arXiv:2006.13318.
[16] G. Dasoulas, K. Scaman, A. Virmaux, Lipschitz normalization for self-attention layers with application to graph neural networks, in: International Conference on Machine Learning, PMLR, 2021, pp. 2456–2466.
[17] W. Huang, Y. Li, W. Du, R.Y. Da Xu, J. Yin, L. Chen, M. Zhang, Towards deepening graph neural networks: A GNTK-based optimization perspective, 2021, arXiv preprint arXiv:2103.03113.
[18] S. Luan, M. Zhao, X.-W. Chang, D. Precup, Training matters: Unlocking potentials of deeper graph convolutional neural networks, 2020, arXiv preprint arXiv:2008.08838.
[19] W. Cong, M. Ramezani, M. Mahdavi, On provable benefits of depth in training graph convolutional networks, Adv. Neural Inf. Process. Syst. 34 (2021).
[20] T. Chen, K. Zhou, K. Duan, W. Zheng, P. Wang, X. Hu, Z. Wang, Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study, IEEE Trans. Pattern Anal. Mach. Intell. (2022).
[21] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 5453–5462.
[22] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, X. Hu, Towards deeper graph neural networks with differentiable group normalization, Adv. Neural Inf. Process. Syst. 33 (2020) 4917–4928.
[23] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: Towards deep graph convolutional networks on node classification, 2019, arXiv preprint arXiv:1907.10903.
[24] G. Li, M. Muller, A. Thabet, 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.
[25] H. Zhang, T. Yan, Z. Xie, Y. Xia, Y. Zhang, Revisiting graph convolutional network on semi-supervised node classification from an optimization perspective, 2020, arXiv preprint arXiv:2009.11469.
[26] S. Luan, M. Zhao, X.-W. Chang, D. Precup, Break the ceiling: Stronger multi-scale deep graph convolutional networks, Adv. Neural Inf. Process. Syst. 32 (2019).
[27] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 1725–1735.
[28] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, 2018, arXiv preprint arXiv:1810.05997.
[29] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[30] L. Zhao, L. Akoglu, Pairnorm: Tackling oversmoothing in gnns, 2019, arXiv preprint arXiv:1909.12223.
[31] K. Zhou, Y. Dong, K. Wang, W.S. Lee, B. Hooi, H. Xu, J. Feng, Understanding and resolving performance degradation in graph convolutional networks, 2020, arXiv preprint arXiv:2006.07107.
[32] W. Jin, X. Liu, Y. Ma, C. Aggarwal, J. Tang, Feature overcorrelation in deep graph neural networks: A new perspective, 2022, arXiv preprint arXiv:2206.07743.
[33] A. Loukas, What graph neural networks cannot learn: depth vs width, 2019, arXiv preprint arXiv:1907.03199.
[34] H. Zeng, M. Zhang, Y. Xia, A. Srivastava, A. Malevich, R. Kannan, V. Prasanna, L. Jin, R. Chen, Deep graph neural networks with shallow subgraph samplers, 2020, arXiv preprint arXiv:2012.01380.
[35] G. Li, C. Xiong, A. Thabet, B. Ghanem, Deepergcn: All you need to train deeper gcns, 2020, arXiv preprint arXiv:2006.07739.
[36] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks? 2021, arXiv preprint arXiv:2105.14491.
[37] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, G. Chen, Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems, in: The World Wide Web Conference, 2019, pp. 2091–2102.
[38] C. Park, C. Lee, H. Bahng, Y. Tae, S. Jin, K. Kim, S. Ko, J. Choo, ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1215–1224.
[39] G. Wang, R. Ying, J. Huang, J. Leskovec, Multi-hop attention graph neural network, 2020, arXiv preprint arXiv:2009.14332.
[40] M. Liu, H. Gao, 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.
[41] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008) 93.
[42] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: European Conference on Computer Vision, Springer, 2016, pp. 630–645.
[43] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[44] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[45] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, in: Relational Representation Learning Workshop, NeurIPS 2018, 2018.
[46] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, Adv. Neural Inf. Process. Syst. 33 (2020) 22118–22133.
[47] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016).
[48] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016, arXiv preprint arXiv:1607.06450.