Over-Smoothing Effect of Graph Convolutional Networks

Spectral and Topological Analysis with Practical Remedies

Fang Sun
Computer Science Department, UCLA
Los Angeles, CA, USA
fts@cs.ucla.edu

***Abstract*—Graph convolutional networks (GCNs) are effective for node classification but their depth is limited by over-smoothing: repeated propagation makes node features within connected components indistinguishable. We present a concise theoretical account of over-smoothing and its practical implications. From spectral and topological perspectives, we formalize smoothness and derive an upper bound that characterizes when exponential over-smoothing occurs as depth grows, in terms of the normalized Laplacian spectrum and network kernel norms. The analysis explains why several remedies—graph sparsification, depth-leveraging across hops, and architectures with initial/residual connections—alleviate over-smoothing. Empirically, on Cora, accuracy degrades beyond a small number of layers even without overfitting or vanishing gradients, consistent with the theory. Together, these results provide simple conditions for anticipating over-smoothing and a unified lens for selecting effective mitigation strategies.**

***Index Terms*—Graph neural networks; graph convolutional networks; over-smoothing; spectral graph theory; theoretical bounds; graph sparsification; node classification**

I. Introduction

Graph data are ubiquitous: from social networks and citation graphs to recommendation systems [1], [2] and molecular sciences [3], [4], they provide a natural way to represent relational information. Graph Convolutional Networks (GCNs) [5] have achieved strong performance on semi-supervised node classification, but their depth is fundamentally constrained by the over-smoothing phenomenon [6], [7]: as layers increase, node features within connected components become indistinguishable.

This paper studies over-smoothing both theoretically and empirically, and connects the analysis to practical remedies. Our contributions are threefold:

- We formalize smoothness and analyze when and why over-smoothing arises via spectral/topological views.
- We derive an upper bound characterizing when exponential over-smoothing occurs as depth grows.
- We contextualize effective mitigation techniques (e.g., sparsification, depth-leveraging, residual/initial residual designs) through the lens of our analysis.

We also report an empirical observation on Cora that accuracy degrades beyond a small number of layers, motivating the need for the theory and methods above; see Section III.

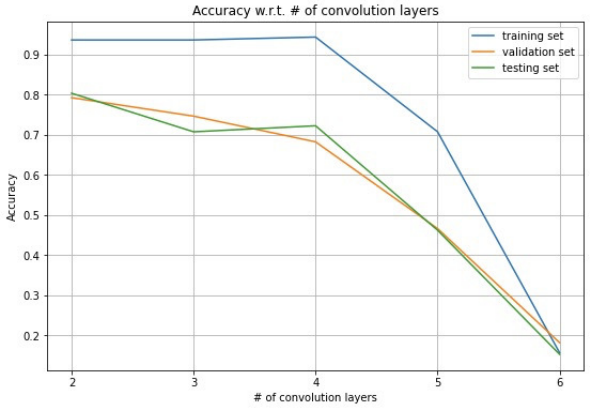


Fig. 1. Accuracy of GCN on Cora w.r.t. # of convolution layers.

II. Related Work

GCNs [5] sparked a surge of interest in scalable, effective graph learning [8]. Follow-up models explored alternative propagation and aggregation, such as attention-based message passing (GAT) [9] and decoupled diffusion with personalized PageRank (APPNP) [10]. Depth-leveraging strategies such as JKNet [11], SGC [12], and DAGNN [13] aggregate information across multiple receptive-field sizes.

The over-smoothing phenomenon was articulated in [6] and analyzed theoretically in [7], [14]. Methods to mitigate it include normalization schemes (PairNorm) [15], graph sparsification (DropEdge) [16], and architectures with initial/residual connections (GCNII) [17]. Our work complements these by providing an upper bound for when exponential over-smoothing occurs and by unifying the intuition behind successful remedies.

III. Experimental Findings

We conducted a simple study on Cora, stacking vanilla GCN layers from 2 to 6. Accuracy degrades as depth increases, despite no signs of overfitting (training and test accuracy decrease together) and a network that is too shallow to suffer vanishing gradients. This supports the hypothesis that over-smoothing, rather than optimization issues, limits depth.

IV. Preliminaries for Graph Convolution

A. Basic Architecture of GCN

a) Graph Laplacian: GCN is essentially a neighborhood-augmented MLP. Inspired by signal processing, GCN uses the Laplacian matrix to aggregate the neighborhood information. For graph G, Laplacian matrix $\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{A}$. $\boldsymbol{D}$ is the degree matrix of G, and $\boldsymbol{A}$ is the adjacency matrix of G.

The spectral convolution of GCN is presented as:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{\Theta}^{(l)}\right). \tag{1}$$

$\boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l+1)}$ are the outputs of the previous/present layer, $\boldsymbol{\Theta}^{(l)}$ is the tune-able convolution kernel, σ is the non-linear activation function (ReLU).

b) Derivation of GCN Model: The GCN model is derived via 4 steps of approximation, as shown in Equantions (2)–(6) at the top of next page, where re-normalization trick is $\boldsymbol{I}_N + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} \rightarrow \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$. $\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$ can be viewed as the normalized Laplacian matrix of graph $\widetilde{G}$, i.e. G with self-loop. The implication behind taking 1st order Chebyshev approximation is that, in each layer of convolution, the model only considers the 1st order neighbor of each node. Nevertheless, higher orders of neighbor information can be aggregated via stacking more convolution layers.

B. Laplacian Smoothing is the Key Power of GCN

[6] proposes that Laplacian smoothing is central to GCN's power in classification tasks. The layer-wise propagation rule of the simplest fully-connected networks (FCNs) is

$$\boldsymbol{H}^{(l+1)} = \sigma(\boldsymbol{H}^{(l)}\boldsymbol{\Theta}^{(l)}). \tag{7}$$

We observe that the sole difference between GCN and FCN is the **normalized Laplacian matrix** $\boldsymbol{S} = \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$. By comparison, even a 1-layer GCN can out-perform a 1-layer FCN by a large margin. This is because Laplacian smoothing makes the output features of nodes in the same cluster more similar and eases the classification task.

The aggregating abilities of Laplacian smoothing is further demonstrated by Simple Graph Convolution (SGC) [12]:

$$\hat{\boldsymbol{Y}}_{\mathrm{SGC}} = \mathrm{softmax}\left(\boldsymbol{S}^K\boldsymbol{X}\boldsymbol{\Theta}\right), \tag{8}$$

where K is the number of Laplacian matrices stacked. SGC shows that even if we remove the redundant ReLU (non-linearity) and MLP layers between aggregators, the multi-layer Laplacian smoothing yields the same degree of accuracy with GCN.

Yet, by applying Laplacian smoothing many times, the feature of nodes in the same connected component will converge to the same value and thus become indistinguishable. As is shown in **Figure 2**, while the two types of points are well-separable under the 2-layer scenario, they all become squashed up in the 5-layer GCN.

Thus, we give the natural definition of **over-smoothing**.

a) Definition 1: **Over-smoothing** is the effect that node features become indistinguishable after multiple rounds of Laplacian smoothing.

V. Deeper Insight into Over-smoothing via Mathematical Formulation

A. Spectral Analysis of GCN

Recent works addressing the over-smoothing issue tend to regard GCN as low-pass filtering [14], inspired by signal processing. The spectral analysis on GCN has yielded some qualitative insight into the issue.

a) Theorem 1: Given a connected graph G, for the normalized Laplacian $\boldsymbol{S}$,

$$\lim_{k\to\infty} \boldsymbol{S}^k = \boldsymbol{\Pi}, \tag{9}$$

where $\boldsymbol{\Pi} = \Phi\left(\widetilde{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{e}^\top\right)\left(\Phi\left(\widetilde{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{e}^\top\right)\right)^\top$, $\Phi(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||}$.

Proof: Because $\boldsymbol{S}$ is symmetric, we orthogonally diagonalize $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$. Thus,

$$\boldsymbol{S}^k = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\cdots\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top = \boldsymbol{Q}\boldsymbol{\Lambda}^k\boldsymbol{Q}^\top = \sum_{i=1}^{k}\lambda_i^n\boldsymbol{v}_i\boldsymbol{v}_i^\top, \tag{10}$$

where $\boldsymbol{v}_i$ is the normalized eigenvector of λ_i. Laplacian $\boldsymbol{S}$ always has an eigenvalue 1 with unique associated eigenvector $\widetilde{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{e}^\top$, and all other eigenvalues λ satisfy $|\lambda| < 1$. Thus, as $k\to\infty$, $\boldsymbol{S}^k \rightarrow \Phi\left(\widetilde{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{e}^\top\right)\left(\Phi\left(\widetilde{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{e}^\top\right)\right)^\top = \boldsymbol{\Pi}$.

Theorem 1 demonstrates that over-smoothing is inevitable in very deep models, where $\boldsymbol{S}^k$ converges to $\boldsymbol{\Pi}$. In this scenario, only the degree information of graph G is retained.

Another important thing to consider is the convergence rate. From the above deduction, the convergence rate is associated with the largest eigenvalue of $\boldsymbol{S}$ other than 1. If we view from another angle and look at how the features of each node $\boldsymbol{v}_i$ in GCN is aggregated with its local neighbors:

$$\mathbf{h}_i^{(k)} \leftarrow \frac{1}{d_i+1}\mathbf{h}_i^{(k-1)} + \sum_{i=1}^{N}\frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}}\mathbf{h}_j^{(k-1)}. \tag{11}$$

The above propagation suggests that the higher the node degree d_i is, the quicker feature $\boldsymbol{h}_i$ would converge. Thus we have the following claim:

b) Claim 1: Nodes with higher degree d_i are more likely to suffer from over-smoothing.

B. Quantifying Smoothness: a Topological View

In addressing the over-smoothing effect, many papers have proposed their own metric for smoothness, either to quantify and prove their hypothesis, or to validate the effectiveness of their method. JKNet [11] defined Influence Score to measure the sensitivity of node x to node y, and uses the Influence Distribution to capture the relative influences of all other nodes. PairNorm [15] focuses on node-wise smoothing and feature-wise smoothing, and defined two metrics for smoothness: rol-diff and col-diff. [18] proposes Mean Average Distance

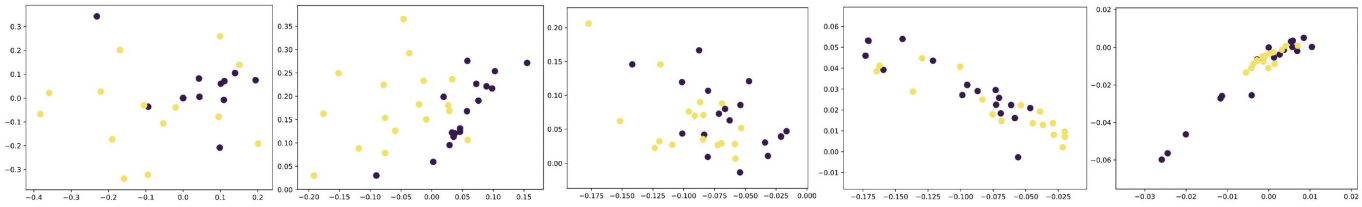
$$g_\theta \star \boldsymbol{x} = g_\theta \boldsymbol{U}^\top \boldsymbol{x} \qquad \text{(spectral convolution)} \tag{2}$$

$$\approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\boldsymbol{L}}) \boldsymbol{x} \qquad (k\text{-th order Chebyshev apprm.}) \tag{3}$$

$$\approx \theta'_0 \boldsymbol{x} + \theta'_1 \left(\boldsymbol{L} - \boldsymbol{I}_N\right) \boldsymbol{x} \qquad (k=1, \text{considering only 1st order neighbor}) \tag{4}$$

$$\approx \theta \left(\boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}\right) \boldsymbol{x} \tag{5}$$

$$\approx \theta \left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\right) \boldsymbol{x} \qquad \text{(the re-normalization trick)} \tag{6}$$



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

Fig. 2. Vertex embeddings of Zachary's karate club network with GCNs of 1,2,3,4,5 layers.

(MAD). MAD reflects the smoothness of graph representation by calculating the mean of the average distances between nodes. However, these metrics are generally task-specific and incompatible to further theoretical analysis.

By contrast, the metric proposed by [7] provides a general framework for measuring smoothness, which solely relies on the topological information of the underlying graph G. Denote the maximum singular value of convolution kernel $\boldsymbol{\Theta_l}$ by s_l and set $s := \sup_{l \in \mathbb{N}_+} s_l$. Denote the distance induced as the Frobenius norm from $\boldsymbol{X}$ to $\mathcal{M}$ by $d_{\mathcal{M}}(\boldsymbol{X}) := \inf_{\boldsymbol{Y} \in \mathcal{M}} \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}$, where $\mathcal{M} := \left\{\boldsymbol{EC} \mid \boldsymbol{C} \in \mathbb{R}^{M \times C}\right\}$, and $\boldsymbol{E}$ is the eigenspace associated with $\lambda_{N-M}, \lambda_{N-M+1}, \cdots, \lambda_N$. We define the ϵ-smoothing metric:

a) Definition 1: (ϵ-smoothing) If there exists a layer L, such that for any hidden layer l beyond L, output feature $\boldsymbol{H}^{(l)}$ has a distance smaller than ϵ w.r.t. subspace $\mathcal{M}$, we call the GCN suffers from ϵ-smoothing, i.e.,

$$\exists L, \forall l \geq L, d_{\mathcal{M}}(\boldsymbol{H^{(l)}}) < \epsilon. \tag{12}$$

From [7], we have the following lemma:

b) Lemma 1: Let $\lambda_1 \leq \cdots \leq \lambda_N$ be the eigenvalues of graph Laplacian $\boldsymbol{S}$, sorted in ascending order. Suppose the multiplicity of the largest eigenvalue $\lambda_N = 1$ is $M(\leq N)$, i.e. $\lambda_{N-M} < \lambda_{N-M+1} = \cdots = \lambda_N = 1$. The second largest eigenvalue is defined as

$$\lambda := \max_{n=1}^{N-M} |\lambda_n| < |\lambda_N|. \tag{13}$$

Then we have $\lambda < \lambda_N = 1$, and

$$d_{\mathcal{M}}\left(\boldsymbol{H}^{(l)}\right) \leq s_l \lambda d_{\mathcal{M}}\left(\boldsymbol{H}^{(l-1)}\right). \tag{14}$$

If all the kernel $\boldsymbol{\Theta_l}$ have been initialized such that $s_l < 1$, we have $s_l \lambda < 1$, and the output feature $\boldsymbol{H}^{(l)}$ **exponentially** approaches $\mathcal{M}$ w.r.t. layer depth l. We derive **Theorem 2** from **Lemma 1**:

c) Theorem 2: If $s\lambda < 1$, then ϵ-smoothing would happen whenever layer depth l satisfies

$$l \geq \hat{l} = \left\lceil \frac{\log \frac{\epsilon}{d_{\mathcal{M}}(\boldsymbol{H^{(0)}})}}{\log(s\lambda)} \right\rceil. \tag{15}$$

Proof: From **Lemma 1**, we have

$$d_{\mathcal{M}}\left(\boldsymbol{H}^{(l)}\right) \leq s_l \lambda d_{\mathcal{M}}\left(\boldsymbol{H}^{(l-1)}\right) \tag{16}$$

$$\leq \left(\prod_{i=1}^{l} s_i\right) \lambda^l d_{\mathcal{M}}(\boldsymbol{H^{(0)}}) \tag{17}$$

$$\leq s^l \lambda^l d_{\mathcal{M}}(\boldsymbol{H^{(0)}}) \tag{18}$$

$$< \epsilon. \tag{19}$$

The inequality is equivalent to

$$l > \frac{\log \frac{\epsilon}{d_{\mathcal{M}}(\boldsymbol{H^{(0)}})}}{\log(s\lambda)}. \tag{20}$$

Taking the ceiling on RHS, we have $l \geq \hat{l}$.

VI. Main Result: Factors Contributing to Over-smoothing

A. Upper Bound for the Occurrence of Over-smoothing

When would the exponential over-smoothing occur? According to **Theorem 2**, We only need to guarantee that $s\lambda < 1$. [7] has studied the issue on Erdos–Renyi graph $G_{N,p}$. Here we study the issue in a more generalized setting.

a) Theorem 3: For N-order graph G with no isolated nodes, denote its largest node degree as d_{max}, and denote its diameter as D. The GCN satisfies the condition of **Theorem 2**, i.e. $s\lambda < 1$, providing that

$$s < (1 - \frac{4}{NDd_{max}})^{-1}. \tag{21}$$

Proof: We carry out our discussion on graph $\widetilde{G}$, which adds a self-loop to each node in G.

First, we consider the smallest eigenvalue other than 0 of the unnormalized Laplacian $\boldsymbol{L} = \widetilde{\boldsymbol{D}} - \widetilde{\boldsymbol{A}}$, denoted as $\lambda(\widetilde{G})$. $\lambda(\widetilde{G})$ is the famous algebraic connectivity (Fiedler eigenvalue, [19]). According to [20], p. 25, $\lambda(\widetilde{G})$ is bounded by

$$\lambda(\widetilde{G}) \geq \frac{4}{ND}. \tag{22}$$

Then we consider the relation between $\lambda(\widetilde{G})$ and λ. According to [21], eigenvalues of the normalized Laplacian $\boldsymbol{S} = \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$ is bounded by their corresponding eigenvalues in $\boldsymbol{L}$ and d_{max}:

$$\lambda_k(\boldsymbol{S}) \leq 1 - \frac{\lambda_k(\boldsymbol{L})}{d_{max}}, \tag{23}$$

where $\lambda_k(\boldsymbol{S})$ is the k-th largest eigenvalue of $\boldsymbol{S}$, and $\lambda_k(\boldsymbol{L})$ is the k-th smallest eigenvalue of $\boldsymbol{L}$. Because λ and $\lambda(\widetilde{G})$ are corresponding eigenvalues, take them into the above inequality to derive

$$\lambda \leq 1 - \frac{\lambda(\widetilde{G})}{d_{max}} \tag{24}$$

$$\leq 1 - \frac{4}{NDd_{max}}. \tag{25}$$

It suffices to show $s\lambda < 1$ if we set

$$s < (1 - \frac{4}{NDd_{max}})^{-1}. \tag{26}$$

Thus, the condition of **Theorem 2** is satisfied, and exponential over-smoothing could happen in this scenario.

B. What Factors Contribute to Over-smoothing?

a) Large and Dense Graphs: Large and dense graphs suffer from over-smoothing. The conclusion is in line with [7], which formulates the issue on Erdos–Renyi graphs. It also confirms the sensibility of graph sparsification methods for combating over-smoothing, e.g. DropEdge [16].

b) Small-World Graphs: Small-world graphs, with $D \propto log\ N$, have already achieved relatively high performance on GCNs with only $2 \sim 3$ layers, since these $2 \sim 3$ hops are sufficient to aggregate neighboring information from a large portion of the whole graph. By contrast, tasks like Point Cloud Classification and molecular dynamics simulation [4], [22] require deeper convolutions to capture long-range information.

c) GCN with Residual Connection: In theory, adding residual connections alone cannot address the over-smoothing issue. If we regard graph convolution as a Markov process [7], the residual connection only leads to a lazy version of the Markov process. The graph Laplacian would still converge, as is shown in **Theorem 1**. Effective versions of residual connections will be discussed in the next section.

VII. Methods for Alleviating Over-smoothing

A. Leveraging between Different Convolution Depths

a) DAGNN: [13] This SGC-based [12] work is straightforward, simple and elegant. With insight from **Claim 1** that node features are smoothed at different rates w.r.t. node degree, DAGNN stacks up the features output from different convolution depths. By adaptively adjusting these features, DAGNN exploits the advantage of deeper Laplacian convolutions without suffering from performance degradation. The adaptive adjustment process of DAGNN is shown above, where $\boldsymbol{s}$ is a trainable projection vector.

$$\boldsymbol{Z} = \text{MLP}(\boldsymbol{\Theta}); \tag{27}$$

$$\boldsymbol{H}_l = \boldsymbol{S}^l\boldsymbol{Z}, l = 1, 2, \cdots, k; \tag{28}$$

$$\boldsymbol{H} = \text{stack}(\boldsymbol{Z}, \boldsymbol{H}_1, \cdots, \boldsymbol{H}_k); \tag{29}$$

$$\boldsymbol{S} = \sigma(\boldsymbol{H}\boldsymbol{s}). \tag{30}$$

B. Graph Sparsification

a) DropEdge: [16] As has been discussed in **Section 4.2**, graph sparsification can slow down the convergence rate of over-smoothing by reducing information passage between layers. DropEdge randomly removes a certain number of edges from the input graph at each training epoch, and can be equipped to many other backbone models. The removal of edges in DropEdge is dynamic and layer-wise:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\mathfrak{N}\left(\boldsymbol{A} \odot \boldsymbol{Z}^{(l)}\right)\boldsymbol{H}^{(l)}\boldsymbol{\Theta}^{(l)}\right), \tag{31}$$

where $\boldsymbol{Z}^{(l)}$ is the binary random mask, and $\mathfrak{N}(\cdot)$ is the normalization operator, i.e., $\mathfrak{N}(\boldsymbol{A}) = \boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$.

Other 'smarter' ways of dropping edges include Graph DropConnect (GDC) [23], which drops edges both layer-wise and channel-wise. Also, NeuralSparse [24] uses neural networks to drop out edges.

C. Adding Residual Connections

a) GCNII: [17] This is the first work that successfully trains deep GCNs on knowledge graphs, with up to 64 layers of Laplacian. The propagation rule of GCNII is

$$\begin{aligned}\boldsymbol{H}^{(\ell+1)} = \sigma\Big(&\big((1-\alpha_\ell)\boldsymbol{S}\boldsymbol{H}^{(\ell)} + \alpha_\ell\boldsymbol{H}^{(0)}\big)\\ &\big((1-\beta_\ell)\boldsymbol{I}_n + \beta_\ell\boldsymbol{\Theta}^{(\ell)}\big)\Big)\end{aligned} \tag{32}$$

The identity mapping $((1-\beta_\ell)\boldsymbol{I}_n + \beta_\ell\boldsymbol{\Theta}^{(\ell)})$ resembles that of ResNet, yet the initial residual connection $((1-\alpha_\ell)\boldsymbol{S}\boldsymbol{H}^{(\ell)} + \alpha_\ell\boldsymbol{H}^{(0)})$ is the highlight. By integrating the most 'unsmooth' layer $\boldsymbol{H}^{(0)}$ during each round of propagation, GCNII circumvents the pitfall described in **Theorem 1**. In fact, the output feature can still carry information from both the input feature and the graph structure, even as $K \to \infty$, which is guaranteed by **Theorem 4**.

b) Theorem 4: A K-layer GCNII can express a K order polynomial filter $\left(\sum_{\ell=0}^{K}\theta_\ell\tilde{\boldsymbol{L}}^\ell\right)\boldsymbol{x}$ with arbitrary coefficients.

According the above theorem, by fine-tuning the hyper-parameters α_ℓ and β_ℓ, GCNII can well preserve node features even at high depths. The tuning process would be tedious for a deep network, though.

VIII. Future Work

Although **Theorem 3** has yielded much theoretical insight into over-smoothing, a tighter bound w.r.t. node features like number of nodes, diameter and sparsity is direly needed. This objective can be better served with comprehensive experiments measuring the effects of those factors on over-smoothing. Also, the interesting properties of residual architectures like GCNII call for further theoretical analysis.

References

[1] Y. Qin, Y. Wang, F. Sun, W. Ju, X. Hou, Z. Wang, J. Cheng, J. Lei, and M. Zhang, "DisenPOI: Disentangling sequential and geographical influence for point-of-interest recommendation," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM)*, 2023.

[2] Y. Wang, Y. Qin, F. Sun, B. Zhang, X. Hou, K. Hu, J. Cheng, J. Lei, and M. Zhang, "DisenCTR: Dynamic graph-based disentangled representation for click-through rate prediction," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2022.

[3] F. Sun, Z. Zhan, H. Guo, M. Zhang, and J. Tang, "GraphVF: Controllable protein-specific 3D molecule generation with variational flow," *arXiv preprint arXiv:2304.12825*, 2023.

[4] F. Sun, Z. Huang, H. Wang, H. Tang, X. Luo, W. Wang, and Y. Sun, "Graph Fourier neural ODEs: Modeling spatial-temporal multi-scales in molecular dynamics," *Transactions on Machine Learning Research*, 2025.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[6] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

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

[8] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao, J. Yang, J. Yuan, Y. Zhao, Y. Wang, X. Luo, and M. Zhang, "A comprehensive survey on deep graph representation learning," *Neural Networks*, vol. 173, p. 106207, 2024.

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

[10] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations (ICLR)*, 2019.

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

[12] F. Wu, A. H. Souza Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *International Conference on Machine Learning (ICML)*, 2019, pp. 6861–6871.

[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 and Data Mining (KDD)*, 2020.

[14] N. T. Hoang and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv preprint arXiv:1905.09550*, 2019.

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

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

[17] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning (ICML)*, 2020, pp. 1725–1735.

[18] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[19] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[20] B. Mohar, "The laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*. Wiley, 1991, vol. 2, pp. 871–898.

[21] M. S. Cavers, "The normalized laplacian matrix and general randić index of graphs," Ph.D. dissertation, University of Regina, Saskatchewan, Canada, 2010.

[22] F. Sun, Z. Huang, Y. Cao, X. Luo, W. Wang, and Y. Sun, "DoMiNO: Down-scaling molecular dynamics with neural graph ordinary differential equations," in *International Conference on Learning Representations (ICLR)*, 2025.

[23] A. Hasanzadeh, E. Hajiramezanali, S. Boluki, M. Zhou, N. Duffield, K. Narayanan, and X. Qian, "Bayesian graph neural networks with adaptive connection sampling," in *International Conference on Machine Learning (ICML)*, 2020.

[24] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, "Robust graph representation learning via neural sparsification," in *International Conference on Learning Representations (ICLR)*, 2020.