Deeper Insights Into Deep Graph Convolutional Networks: Stability and Generalization

Guangrui Yang, Ming Li, *Member, IEEE*, Han Feng, and Xiaosheng Zhuang

***Abstract*—Graph convolutional networks (GCNs) have emerged as powerful models for graph learning tasks, exhibiting promising performance in various domains. While their empirical success is evident, there is a growing need to understand their essential ability from a theoretical perspective. Existing theoretical research has primarily focused on the analysis of single-layer GCNs, while a comprehensive theoretical exploration of the stability and generalization of deep GCNs remains limited. In this paper, we bridge this gap by delving into the stability and generalization properties of deep GCNs, aiming to provide valuable insights by characterizing rigorously the associated upper bounds. Our theoretical results reveal that the stability and generalization of deep GCNs are influenced by certain key factors, such as the maximum absolute eigenvalue of the graph filter operators and the depth of the network. Our theoretical studies contribute to a deeper understanding of the stability and generalization properties of deep GCNs, potentially paving the way for developing more reliable and well-performing models.**

***Index Terms*—Graph convolutional networks (GCNs), generalization gap, deep GCNs, uniform stability.**

I. Introduction

GRAPH-STRUCTURED data is pervasive across diverse domains, including knowledge graphs, traffic networks, and social networks to name a few [1], [2]. Several pioneering works [3], [4] introduced the initial concept of graph neural networks (GNNs), incorporating recurrent mechanisms and necessitating neural network parameters to define contraction mappings. Concurrently, Micheli [5] introduced the neural network for graphs, commonly referred to as NN4G, over a comparable timeframe. It is worth noting that the NN4G diverges from recurrent mechanisms and instead employs a feed-forward architecture, exhibiting similarities to contemporary GNNs. In recent years, (contemporary) GNNs have gained significant attention as an effective methodology for modeling graph data [6], [7], [8], [9], [10], [11]. To obtain a comprehensive understanding of GNNs and deep learning for graphs, we refer the readers to relevant survey papers for an extensive overview [12], [13], [14], [15].

Among the various GNN variants, one of the most powerful and frequently used GNNs is graph convolutional networks (GCNs). A widely accepted perspective posits that GCNs can be regarded as an extension or generalization of traditional spatial filters, which are commonly employed in Euclidean data analysis, to the realm of non-Euclidean data. Due to its success on non-Euclidean data, GCN has attracted widespread attention on its theoretical exploration. Recent works on GCNs includes understanding over-smoothing [16], [17], [18], [19], interpretability and explainability [20], [21], [22], [23], [24], expressiveness [25], [26], [27], and generalization [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40]. In this paper, we specifically address the generalization of GCNs to provide a bound on their generalization gap.

Investigating the generalization of GCNs is essential in understanding its underlying working principles and capabilities from a theoretical perspective. However, the theoretical establishment in this area is still in its infancy. In recent work [36], Verma and Zhang provided a novel technique based on algorithmic stability to investigate the generalization capability of single-layer GCNs in semi-supervised learning tasks. Their results indicate that the stability of a single-layer GCN trained with the stochastic gradient descent (SGD) algorithm is dependent on the largest absolute eigenvalue of graph filter operators. This finding highlights the crucial role of graph filters in determining the generalization capability of single-layer GCNs, providing guidance for designing effective graph filters for these networks. On the other hand, a number of prior studies have shown that deep GCNs possess greater expressive power than their single-layer counterparts. Consequently, it is essential to extend the generalization results of single-layer GCNs to their multi-layer counterparts. This will help us understand the effect of factors (e.g., graph filters, number of layers) on the generalization capability of deep GCNs.

Received 18 June 2024; revised 11 September 2025; accepted 28 September 2025. Date of publication 6 October 2025; date of current version 9 January 2026. This work was supported in part by the National Natural Science Foundation of China under Grant U21A20473, Grant 62536006, and Grant 62172370. The work of Ming Li was supported by "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant 2024C03262. The work of Guangrui Yang was supported by Opening Project of Guangdong Province Key Laboratory of Computational Science at the Sun Yat-sen University under Grant 2024008. The work of Han Feng was supported by the Research Grants Council of Hong Kong under Grant CityU 11303821 and Grant CityU 11315522. The work of Xiaosheng Zhuang was supported in part by the Research Grants Council of Hong Kong Grant CityU 11309122, Grant CityU 11302023, Grant CityU 11301224, and Grant CityU 11300825. Recommended for acceptance by Z. Wei. *(Corresponding authors: Ming Li; Han Feng.)*

Guangrui Yang is with the Department of Mathematics, College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China (e-mail: yanggrui@mail2.sysu.edu.cn).

Ming Li is with the Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua 321017, China (e-mail: mingli@zjnu.edu.cn).

Han Feng and Xiaosheng Zhuang are with the Department of Mathematics, City University of Hong Kong, Hong Kong (e-mail: hanfeng@cityu.edu.hk; xzhuang7@cityu.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2025.3616350, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2025.3616350

In this paper, we investigate the generalization properties of deep GCNs. Building on the stability framework of [36], we analyze the uniform stability of deep GCNs in semi-supervised learning, while developing a more refined theoretical treatment suited to deep architectures. Our analysis reveals a strong connection between the generalization gap of deep GCNs and the characteristics of the graph filter, particularly the number of layers. In particular, we show that when the maximum absolute eigenvalue (or the largest singular value) of the graph filter operator remains invariant with respect to graph size, the generalization gap diminishes asymptotically at a rate of $O(1/\sqrt{m})$ as the training sample size m grows. This result explains why normalized graph filters generally outperform non-normalized ones in deep GCNs. Furthermore, our findings indicate that increasing depth can enlarge the generalization gap and consequently degrade performance, thereby offering theoretical guidance for selecting an appropriate number of layers when designing deep GCNs. We then empirically validate our theoretical results through experiments on three benchmark datasets: Cora, Citeseer, and Pubmed, demonstrating strong consistency between theory and practice. In addition, we further discuss how our theoretical framework extends to advanced architectures, including GCNII [41] and Graph Transformer [42], thereby highlighting its broader applicability and its potential to inspire future theoretical studies on more complex GNN variants.

The key contributions of our paper are as follows:

- We establish the uniform stability of deep GCNs trained with SGD, thereby extending the earlier results on single-layer GCNs presented in [36].
- We provide a rigorous upper bound for the generalization gap of deep GCNs and highlight the key factors that govern their generalization ability. Moreover, we further discuss how our theoretical framework extends naturally to advanced GNN architectures, including GCNII and Graph Transformer models.
- We conduct empirical studies on three benchmark datasets for node classification, which strongly validate our theoretical findings regarding the influence of graph filters, as well as the depth and width of deep GCNs.

The remainder of this paper is organized as follows. In Section II, an overview of prior studies on the generalization of GCNs (or generic GNNs) is presented, along with a comparative analysis highlighting the similarities and distinctions between our work and previous research. Section III offers an exposition of the essential concepts. The primary findings of this paper are given in Section IV. Experimental studies designed to validate our theoretical findings are presented in Section V. In Section VI, we discuss how our findings extend to advanced GNN architectures, including GCNII and Graph Transformer models. Section VII concludes the paper with additional remarks.

Due to space constraints and for the sake of presentation clarity, the detailed proofs of our theoretical results are deferred to the **Supplementary Material**.

II. Related Work

Theoretical studies on the generalization capability of GCNs mainly employ three methodologies: Vapnik–Chervonenkis (VC) dimension [30], [34], Rademacher complexity [31], [32], [33], [34], [35], and algorithmic stability [36], [37], [43], [44]. Other approaches include PAC-Bayesian theory [38], [39], neural tangent kernels (NTKs) [28], [40], algorithm alignment [45], [46], and methods from statistical physics and random matrix theory [47]. For a broader perspective, we refer readers to the recent survey [48], which provides a comprehensive overview of generalization theory for message-passing GNNs.

VC-Dimension and Rademacher Complexity: Scarselli et al. [30] study the generalization capability of GNNs by deriving upper bounds on the growth order of their VC-dimension. While VC-dimension is a classical tool for establishing learning bounds, it does not capture the structure of the underlying graph. Similarly, [34] provides VC-dimension–based error bounds for GNNs, but the results are trivial and fail to reflect the benefits of degree normalization. To address graph-specific effects, Esser et al. [34] analyze upper bounds using transductive Rademacher complexity (TRC), highlighting how graph convolutions and network architectures influence generalization. Tang et al. [35] establish high-probability generalization bounds for popular GNNs via TRC-based analysis of transductive SGD. However, their bounds scale with the parameter dimension, limiting tightness for large models.

Algorithmic Stability: Beyond capacity-based measures, algorithmic stability serves as an important framework for understanding GNN generalization. Building on the work of Hardt et al. [49], Verma and Zhang [36] show that one-layer GCNs exhibit uniform stability and provide generalization bounds that scale with the largest absolute eigenvalue of the graph filter operator. Extending this line, Liu et al. [43] analyze the stability of single-layer GCNs trained with an SGD-proximal algorithm under ℓ_p-regularization, yielding a more refined theoretical understanding. These studies, however, remain restricted to single-layer architectures. Cong et al. [50] examine GNNs under uniform transductive stability, showing that deeper models improve stability and reduce generalization error, whereas our work adopts a different stability formulation. Ng and Yip [37] investigate stability and generalization in two-layer GCNs under an eigen-domain formulation, relying on spectral graph convolution [51]. Because this formulation requires computationally expensive eigendecomposition of the graph Laplacian, it does not scale to large node-classification tasks. Within this methodological line, the closest studies to ours are [36] and [37], but our analysis focuses on deep GCNs without assuming a spectral-based formulation.

Other Methodologies: Alternative perspectives on GNN generalization also exist. The pioneering work of [38] introduces PAC-Bayesian analysis for GCNs and message-passing neural networks, later extended in [39] to provide tighter bounds linked to the graph diffusion matrix. The NTK framework introduced by [40] enables analysis of infinitely wide GNNs trained by gradient descent, with [28] extending this framework to multi-layer settings. However, NTK-based analyses typically focus on graph classification rather than the more challenging transductive node-classification setting. Additional work explores distinct theoretical frameworks, including topology-sampling techniques [52], analysis on large random graphs [53], and

TABLE I
FREQUENTLY USED NOTATIONS

Notation	Description
$g(\mathbf{L})$	graph filter operator used in the considered deep GCNs
C_g	the 2-norm of $g(\mathbf{L})$, i.e., $C_g := \|g(\mathbf{L})\|_2$
$C_{\mathbf{X}}$	Frobenius norm of the input feature $\mathbf{X}$, i.e., $C_{\mathbf{X}} := \|\mathbf{X}\|_F$
K	number of hidden layers of the considered deep GCNs
α_σ, ν_σ	parameters w.r.t the continuity of activation function $\sigma(\cdot)$
$\nabla\sigma$	the derivative of activation function $\sigma(\cdot)$
α_ℓ, ν_ℓ	parameters w.r.t the continuity of the loss function $\ell(\cdot,\cdot)$
M	the upper bound of loss function $\ell(\cdot,\cdot)$
$\mathcal{A}_\mathcal{S}$	the learning algorithm for deep GCNs trained on dataset $\mathcal{S}$
m	the number of samples in the trained dataset $\mathcal{S}$
η	the learning rate of $\mathcal{A}_\mathcal{S}$
T	number of iterations for training $\mathcal{A}_\mathcal{S}$ using SGD
μ_m	the uniform stability of a learning algorithm $\mathcal{A}_\mathcal{S}$
$\boldsymbol{\delta}_{\mathbf{x}}$	the indicator vector with respect to node $\mathbf{x}$
$\boldsymbol{\delta}_i$	the indicator vector with respect to index i
$\mathbf{X}^{(k)}$	the output feature matrix of the k-th layer
$\triangle\mathbf{X}^{(k)}$	the variation of $\mathbf{X}^{(k)}$ in two GCNs
$\mathbf{W}^{(k)}$	the parameter matrix specific to the k-th layer
B	upper bound for 2-norm of $\{\mathbf{W}^{(1)},\dots,\mathbf{W}^{(K)},\mathbf{w}\}$
$\triangle\mathbf{W}^{(k)}$	the variation of $\mathbf{W}^{(k)}$ in two GCNs
$\triangle\theta$	$\triangle\theta = \{\triangle\mathbf{W}^{(1)},\dots,\triangle\mathbf{W}^{(K)},\triangle\mathbf{w}\}$
$\mathbf{W}_t^{(k)}$	the learnt $\mathbf{W}^{(k)}$ trained after t iterations
$\triangle\mathbf{W}_t^{(k)}$	the variation of $\mathbf{W}_t^{(k)}$ of two GCNs trained after t iterations
$\triangle\theta_t$	$\triangle\theta_t = \{\triangle\mathbf{W}_t^{(1)},\dots,\triangle\mathbf{W}_t^{(K)},\triangle\mathbf{w}_t\}$

NTK-based loss landscape analysis of wide GCNs [54]. For further perspectives, we refer readers to the survey [55], which synthesizes emerging theoretical approaches to characterizing GNN capabilities.

III. Preliminaries and Notations

In this section, we describe the problem setup considered in this paper and review fundamental concepts of uniform stability for training algorithms, which form the basis of our subsequent analysis. For clarity, we first summarize the main symbols used in this paper in the above Table I.

A. Deep Graph Convolutional Networks

Let $\mathcal{G} = (\mathcal{V},\mathcal{E},\mathbf{A})$ denote an undirected graph with a node set $\mathcal{V}$ of size N, an edge set $\mathcal{E}$ and the adjacency matrix $\mathbf{A}\in\mathbb{R}^{N\times N}$. As usual, $\mathbf{L} := \mathbf{D}-\mathbf{A}$ is denoted as its conventional graph Laplacian, where $\mathbf{D}\in\mathbb{R}^{N\times N}$ signifies the degree diagonal matrix. Furthermore, $g(\mathbf{L})\in\mathbb{R}^{N\times N}$ represents a graph filter and is defined as a function of $\mathbf{L}$ (or its normalized versions). We denote by $C_g = \|g(\mathbf{L})\|_2$ the maximum absolute eigenvalue of a symmetric filter $g(\mathbf{L})$ or the maximum singular value of an asymmetric $g(\mathbf{L})$.

We denote by $\mathbf{X} = (\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N)^\top\in\mathbb{R}^{N\times d_0}$ the input features (d_0 stands for input dimension) and $\mathbf{x}_j\in\mathbb{R}^{d_0}$ the node feature of node j, while $C_{\mathbf{X}} = \|\mathbf{X}\|_F$ represents the Frobenius norm of $\mathbf{X}$. For the input feature $\mathbf{X}$, a deep GCN with $g(\mathbf{L})$ updates the representation as follows:

$$\mathbf{X}^{(k)} = \sigma(g(\mathbf{L})\mathbf{X}^{(k-1)}\mathbf{W}^{(k)}),\ \ k = 1,2,\dots,K,$$

where $\mathbf{X}^{(k)}\in\mathbb{R}^{N\times d_k}$ is the output feature matrix of the k-th layer with $\mathbf{X}^{(0)} = \mathbf{X}$, the matrix $\mathbf{W}^{(k)}\in\mathbb{R}^{d_{k-1}\times d_k}$ represents the trained parameter matrix specific to the k-th layer. The function $\sigma(\cdot)$ denotes a nonlinear activation function applied within the GCN model. For simplicity, we set a final output in a single dimension, that is, the final output label of N nodes is given by

$$\mathbf{y} = \sigma\left(g(\mathbf{L})\mathbf{X}^{(K)}\mathbf{w}\right), \tag{1}$$

where $\mathbf{y}\in\mathbb{R}^N$ and $\mathbf{w}\in\mathbb{R}^{d_K}$.

As defined above, the deep GCN (1) with learnable parameters

$$\theta = \{\mathbf{W}^{(1)},\mathbf{W}^{(2)},\dots,\mathbf{W}^{(K)},\mathbf{w}\}$$

is a $K+1$ layers GCN with K hidden layers and a final output layer, and in the case of $K=0$, it degenerates into the single-layer GCN studied in [36].

B. The SGD Algorithm

We denote by $\mathcal{D}$ the unknown joint distribution of input features and output labels. Let

$$\mathcal{S} := \{(\mathbf{x}_j, y_j)\}_{j=1}^m$$

be the training set i.i.d sampled from $\mathcal{D}$ and $\mathcal{A}_\mathcal{S}$ be a learning algorithm for a deep GCN trained on $\mathcal{S}$. For a deep GCN model (1) with parameters $\theta = \{\mathbf{W}^{(1)},\dots,\mathbf{W}^{(K)},\mathbf{w}\}$, denote $\mathcal{A}_\mathcal{S}(\mathbf{x}) = f(\mathbf{x}|\theta_S) = \sigma(\boldsymbol{\delta}_{\mathbf{x}}^\top g(\mathbf{L})\mathbf{X}^{(K)}\mathbf{w})$ as the output of node $\mathbf{x}$, where θ_S is the corresponding learned parameter and $\boldsymbol{\delta}_{\mathbf{x}}$ is the indicator vector with respect to node $\mathbf{x}$. For a loss function $\ell:\mathbb{R}\times\mathbb{R}\to\mathbb{R}_+$, the generalization error or risk $R(\mathcal{A}_\mathcal{S})$ is defined by

$$R(\mathcal{A}_\mathcal{S}) := \mathbb{E}_{\mathbf{z}}\left[\ell(f(\mathbf{x}|\theta_S),y)\right],$$

where the expectation is taken over $\mathbf{z} = (\mathbf{x},y)\sim\mathcal{D}$, and the empirical error or risk $R_{emp}(\mathcal{A}_\mathcal{S})$ is

$$R_{emp}(\mathcal{A}_\mathcal{S}) := \frac{1}{m}\sum_{j=1}^m \ell(f(\mathbf{x}_j|\theta_S),y_j).$$

When considering a randomized algorithm $\mathcal{A}_\mathcal{S}$,

$$\epsilon_{gen}(\mathcal{A}_\mathcal{S}) := \mathbb{E}_\mathcal{A}\left[R(\mathcal{A}_\mathcal{S}) - R_{emp}(\mathcal{A}_\mathcal{S})\right] \tag{2}$$

gives the generalization gap between the generalization error and the empirical error, where the expectation $\mathbb{E}_\mathcal{A}$ corresponds to the inherent randomness of $\mathcal{A}_\mathcal{S}$.

In this paper, $\mathcal{A}_\mathcal{S}$ is considered to be the algorithm given by the SGD algorithm. Following the approach employed in [36], our analysis focuses solely on the randomness inherent in $\mathcal{A}_\mathcal{S}$ arising from the SGD algorithm, while disregarding the stochasticity introduced by parameter initialization. The SGD algorithm for a deep GCN (1) aims to optimize its empirical error on a dataset $\mathcal{S}$ by updating parameters iteratively. For $t\in\mathbb{N}$ and considering the parameters θ_{t-1} obtained after $t-1$ iterations, the t-th iteration of SGD involves randomly drawing a sample $(\mathbf{x}_t,y_t)$ from the dataset $\mathcal{S}$. Subsequently, parameters θ are iteratively updated as follows:

$$\theta_t = \theta_{t-1} - \eta\nabla_\theta\ell(f(\mathbf{x}_t|\theta_{t-1}),y_t), \tag{3}$$

with the learning rate $\eta > 0$.

C. Uniform Stability

For the sake of estimating the generalization gap $\epsilon_{gen}(\mathcal{A}_{\mathcal{S}})$ of $\mathcal{A}_{\mathcal{S}}$, we invoke the notion of uniform stability of $\mathcal{A}_{\mathcal{S}}$ as adopted in [36], [56].

Let

$$S^{\backslash i} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{i-1} \cup \{(\mathbf{x}_j, y_j)\}_{j=i+1}^{m}$$

be the dataset obtained by removing the i-th data point in $\mathcal{S}$, and

$$S^{i} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{i-1} \cup \{(\mathbf{x}'_i, y'_i)\} \cup \{(\mathbf{x}_j, y_j)\}_{j=i+1}^{m}$$

the dataset obtained by replacing the i-th data point in $\mathcal{S}$. Then, the formal definition of uniform stability of a randomized algorithm $\mathcal{A}_{\mathcal{S}}$ is given in the following.

Definition 1 (Uniform Stability [36]): A randomized algorithm $\mathcal{A}_{\mathcal{S}} = f(\mathbf{x}|\theta_S)$ is considered to be μ_m-uniformly stable in relation to a loss function ℓ when it fulfills the following condition:

$$\sup_{\mathcal{S},\mathbf{z}} \Big| \mathbb{E}_{\mathcal{A}}\left[\ell(\hat{y}, y)\right] - \mathbb{E}_{\mathcal{A}}\left[\ell(\hat{y}', y)\right] \Big| \le \mu_m, \tag{4}$$

where $\mathbf{z} = (\mathbf{x}, y) \sim \mathcal{D}$, $\hat{y} = f(\mathbf{x}|\theta_S)$ and $\hat{y}' = f(\mathbf{x}|\theta_{\mathcal{S}^{\backslash i}})$.

As shown in Definition 1, μ_m indicates a bound on how much the variation of the training set $\mathcal{S}$ can influence the output of $\mathcal{A}_{\mathcal{S}}$. It further implies the following property:

$$\sup_{\mathcal{S},\mathbf{z}} \Big| \mathbb{E}_{\mathcal{A}}\left[\ell(\hat{y}, y)\right] - \mathbb{E}_{\mathcal{A}}\left[\ell(\hat{y}', y)\right] \Big| \le 2\mu_m, \tag{5}$$

where $\mathbf{z} = (\mathbf{x}, y) \sim \mathcal{D}$, $\hat{y} = f(\mathbf{x}|\theta_S)$ and $\hat{y}' = f(\mathbf{x}|\theta_{\mathcal{S}^{i}})$.

Moreover, it is shown that the uniform stability of a learning algorithm $\mathcal{A}_{\mathcal{S}}$ can yield the following upper bound on the generalization gap $\epsilon_{gen}(\mathcal{A}_{\mathcal{S}})$.

Lemma 1 (Stability Guarantees [36]): Suppose that a randomized algorithm $\mathcal{A}_{\mathcal{S}}$ is μ_m-uniformly stable with a bounded loss function ℓ. Then, with a probability of at least $1-\delta$, considering the random draw of $\mathcal{S}, \mathbf{z}$ with $\delta \in (0,1)$, the following inequality holds for the expected value of the generalization gap:

$$\epsilon_{gen}(\mathcal{A}_{\mathcal{S}}) \le 2\mu_m + \Big(4m\mu_m + M\Big)\sqrt{\frac{\log\frac{1}{\delta}}{2m}},$$

where M is an upper bound of the loss function ℓ, i.e., $0 \le \ell(\cdot,\cdot) \le M$.

IV. Main Results

This section presents an established upper bound on the generalization gap $\epsilon_{gen}(\mathcal{A}_{\mathcal{S}})$ as defined in (2) for deep GCNs trained using the SGD algorithm. Notably, this generalization bound, derived from a meticulous analysis of the comprehensive back-propagation algorithm, demonstrates the enhanced insight gained through the utilization of SGD.

A. Assumptions

First, we make some assumptions about the considered deep GCN model (1), which are necessary to derive our results.

Assumption 1: The activation function $\sigma: \mathbb{R} \to \mathbb{R}$ is assumed to satisfy the following:

1) α_σ-Lipschitz:
$$|\sigma(x) - \sigma(y)| \le \alpha_\sigma |x - y|, \ \forall x, y \in \mathbb{R}.$$
2) ν_σ-smooth:
$$|\nabla\sigma(x) - \nabla\sigma(y)| \le \nu_\sigma |x - y|, \ \forall x, y \in \mathbb{R}.$$
3) $\sigma(0) = 0$.

With these assumptions, the derivative of σ, denoted by $\nabla\sigma$, is bounded, i.e., $|\nabla\sigma(\cdot)| \le \alpha_\sigma$, and $\|\sigma(\mathbf{X})\|_F \le \alpha_\sigma \|\mathbf{X}\|_F$ holds for any matrix $\mathbf{X}$. It can be easily verified that activation functions such as ELU and tanh satisfy the above assumptions.

Assumption 2: Let $\hat{y}$ and y be the predicted and true labels, respectively. We denote the loss function $\ell: [y_{\min}, y_{\max}] \times [y_{\min}, y_{\max}] \to \mathbb{R}$ by $\ell(\hat{y}, y)$. Similar to [37], we adopt the following assumptions for ℓ.

1) The loss function ℓ exhibits continuity with respect to the variables $(\hat{y}, y)$ and possesses continuous differentiability with respect to $\hat{y}$.
2) The loss function ℓ satisfies α_ℓ-Lipschitz with respect to $\hat{y}$:
$$|\ell(\hat{y}, y) - \ell(\hat{y}', y)| \le \alpha_\ell |\hat{y} - \hat{y}'|,$$
$$\forall\, \hat{y}, \hat{y}', y \in [y_{\min}, y_{\max}].$$
3) The loss function ℓ meets ν_ℓ-smooth with respect to $\hat{y}$:
$$\left| \frac{\partial \ell}{\partial \hat{y}}(\hat{y}, y) - \frac{\partial \ell}{\partial \hat{y}}(\hat{y}', y) \right| \le \nu_\ell |\hat{y} - \hat{y}'|,$$
$$\forall\, \hat{y}, \hat{y}', y \in [y_{\min}, y_{\max}].$$

With these assumptions, $|\frac{\partial \ell}{\partial \hat{y}}(\hat{y}, y)| \le \alpha_\ell$, and ℓ is bounded, i.e., $0 \le \ell(\hat{y}, y) \le M$.

Assumption 3: The learned parameters $\{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(K)}, \mathbf{w}\}$ during the training procedure with limited iterations satisfies

$$\max\left\{ \|\mathbf{W}^{(1)}\|_2, \ldots, \|\mathbf{W}^{(K)}\|_2,\ \|\mathbf{w}\|_2 \right\} \le B.$$

B. Generalization Gap

This section presents the main results of this paper. Under the assumptions made in Section IV-A, the bound on the generalization gap of deep GCNs is provided in the following theorem.

Theorem 1 (Generalization gap for deep GCNs): Consider the deep GCN model, defined in (1), which comprises K hidden layers and utilizes $g(\mathbf{L})$ as the graph filter operator. The model is trained on $\mathcal{S}$ using SGD for T iterations. Under Assumptions 1, 2, and 3 stated in Section IV-A, the following expected generalization gap is valid with a probability of at least $1-\delta$, where $\delta \in (0,1)$:

$$\epsilon_{gen}(\mathcal{A}_{\mathcal{S}}) \le \frac{1}{\sqrt{m}} \Bigg\{ O\bigg(\Big((K+1)\eta\kappa_1 + \eta\kappa_2\Big)^T \bigg) + M\sqrt{\frac{\log\frac{1}{\delta}}{2}} \Bigg\}, \tag{6}$$

where

$$\kappa_1 := (\upsilon_\ell \alpha_\sigma^2 + \alpha_\ell \nu_\sigma)(B\alpha_\sigma C_g)^{2K} C_g^2 C_{\mathbf{X}}^2 + \alpha_\ell (B\alpha_\sigma C_g)^{K-1} \alpha_\sigma^2 C_g^2 C_{\mathbf{X}}, \tag{7}$$

$$\kappa_2 := \nu_\sigma \left(B\alpha_\sigma C_g\right)^K C_g^2 C_{\mathbf{X}}^2 \left(\sum_{j=0}^{K-1} (j+1)(B\alpha_\sigma C_g)^j\right). \tag{8}$$

A fundamental correlation between the generalization gap and the parameters governing deep GCNs is induced by Theorem 1. This correlation implies that the uniform stability of deep GCNs, trained using the SGD algorithm, exhibits an increase with the number of samples when the upper bound approaches zero as the sample size m tends to infinity. Specifically, it is observed that if the value of C_g (presenting the largest absolute eigenvalue of a symmetric $g(\mathbf{L})$ or the maximum singular value of an asymmetric $g(\mathbf{L})$) remains unaffected by the size N, a generalization gap decaying at the order of $O(1/\sqrt{m})$ is obtained. To compare with the result in [36], let us discuss at length the role of $g(\mathbf{L})$ and the hidden layer number K on the generalization gap.

According to (7) and (8), $\kappa_1 = O(C_g^{2K+2})$ and $\kappa_2 = O(C_g^{2K+1})$. Therefore, the bound on the generalization gap of deep GCNs in Theorem 1 is

$$\epsilon_{gen}(\mathcal{A}_{\mathcal{S}}) \leq \frac{1}{\sqrt{m}} \left(O\left(C_g^{2T(K+1)}\right) + M\sqrt{\frac{\log \frac{1}{\delta}}{2}} \right). \tag{9}$$

When $K = 0$, the GCN model (1) degenerates into the single-layer GCN model considered in [36]. At this point, according to (9), we have

$$\epsilon_{gen}(\mathcal{A}_{\mathcal{S}}) \leq \frac{1}{\sqrt{m}} \left(O\left(C_g^{2T}\right) + M\sqrt{\frac{\log \frac{1}{\delta}}{2}} \right), \tag{10}$$

which is the same as the result of [36].

Remarks: Based on (9), we present certain observations regarding the impact of filter $g(\mathbf{L})$ and the hidden layer number K on the generalization capacity of deep GCNs in (1).

- *Normalized vs. Unnormalized Graph Filters:* We examine the three most commonly utilized filters: 1) $g_1(\mathbf{L}) = \mathbf{A} + \mathbf{I}$, 2) $g_2(\mathbf{L}) = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} + \mathbf{I}$, and 3) $g_3(\mathbf{L}) = \mathbf{D}^{-1}\mathbf{A} + \mathbf{I}$. For the unnormalized filter g_1, its maximum absolute eigenvalue is bounded by $O(N)$. Consequently, as the value of m approaches the magnitude to N, the upper bound indicated by (9) tends towards $O(N^p)$ for some $p > 0$, leading to an impractical upper bound when N become infinitely large. On the contrary, for two normalized filters g_2 and g_3, their largest absolute eigenvalues are bounded and independent of graph size N. Therefore, both filters yield a diminishing generalization gap at a rate of $O\left(\frac{1}{\sqrt{m}}\right)$ as m goes to infinity. This discovery underscores the superior performance of normalized filters over unnormalized counterparts in deep GCNs. This observation is consistent with the findings in [36], [37].
- *Low-pass vs. High-pass Graph Filters:* Our theoretical results are not restricted to the choice of $g(\mathbf{L})$ as either a low-pass or a high-pass filter. To illustrate, consider two exponential filters with symmetric $\mathbf{L}$: i) a low-pass filter $g_{\text{low}}(\lambda) = e^{-b\lambda^2}$ and ii) a high-pass filter $g_{\text{high}}(\lambda) = 1 - e^{-a\lambda^2}$, where $a, b > 0$. In this setting, it is straightforward to verify that
$$\|g_{\text{high}}(\mathbf{L})\|_2 < \|g_{\text{low}}(\mathbf{L})\|_2 = 1.$$
Consequently, both filters lead to a vanishing generalization gap at the rate of $O(\frac{1}{\sqrt{m}})$ as $m \to \infty$.
- *The Role of Parameter K:* It is evident that, when the values of C_g and T are fixed, the upper bound (9) exhibits an exponential dependence on parameter K. This observation implies that a larger value K leads to an increase in the upper bound of the generalization gap, thereby offering valuable insights for the architectural design of deep GCNs. This finding diverges from the ones presented in [36], [37], as these studies do not account for generic deep GCNs and overlook the significance of the parameter K.

Furthermore, based on Theorem 1, we give a brief analysis of the impact of d_k (width of the k-th layer) on the generalization. Actually, the impact of d_k on the generalization is reflected in its impact on B. More specifically, let us consider the case where parameters $\{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(K)}, \mathbf{w}\}$ belong to the set $\mathcal{X}_\xi$, where

$$\mathcal{X}_\xi := \{\mathbf{W} : \|\mathbf{W}\|_\infty \leq \xi\},$$

i.e., $\mathcal{X}_\xi$ is the collection of all matrices whose elements' absolute values are all less than ξ. At this point, for $\mathbf{W}^{(k)} \in \mathbb{R}^{d_{k-1} \times d_k}$, we have

$$\sup_{\mathbf{W}^{(k)} \in \mathcal{X}_\xi} \|\mathbf{W}^{(k)}\|_2 \leq \sup_{\mathbf{W}^{(k)} \in \mathcal{X}_\xi} \|\mathbf{W}^{(k)}\|_F \leq \xi\sqrt{d_{k-1}d_k}.$$

Therefore, a larger d_k (i.e., width of the k-th layer) results in a larger upper bound of $\|\mathbf{W}^{(k)}\|_2$, which implies that a larger d_k results in a larger B (see Assumption 3 in Section IV-A). Finally, Theorem 1 indicates that a larger B leads to a larger bound on the generalization gap, thus we conclude that a larger d_k leads to a larger bound on the generalization gap. To justify this argument, we add some experimental studies in Section V. The empirical results are consistent with our analysis.

Table II offers a concise summary of various upper bounds on the generalization gap, derived through the application of uniform stability. From Table II, we can see that all the works derive a generalization gap decaying at the order of $O(1/\sqrt{m})$. However, compared to the other three works which only consider shallow GCNs, our work explores the case of deep GCNs. We should point out that the generalization of single-layer GCNs into deep GCNs is not trivial. To derive the results for deep GCNs, we tackle two significant challenges that arise specifically in the context of deep GCNs, which are unique to deep GCNs and are non-existent in single-layer models. **The first challenge** is the derivation of the gradient of the final output with respect to the learnable parameters across multiple layers, which requires determining how the gradient of the overall error of a GCN is shared among neurons in different hidden layers. In particular, in Appendix A.1, we provide a recursive formula to compute the related gradients. **The second challenge** is the evaluation of gradient variations between GCNs trained on

TABLE II
COMPARISON OF THE GENERALIZATION GAP ESTIMATED BASED ON UNIFORM STABILITY

Reference	Model Architecture	Estimated Upper Bound of the Generalization Gap
[36]	shallow	$\frac{1}{\sqrt{m}}\left(O\left((1+\eta\upsilon_\ell\upsilon_\sigma C_g^2)^T\right)+M\sqrt{\frac{\log\frac{1}{\delta}}{2}}\right)$
[37]	shallow	$\frac{1}{\sqrt{m}}\left(O\left(\eta\alpha_\ell\alpha_\sigma c_{2,T}\sum_{t=0}^{T-1}c_{6,t}\prod_{s=t+1}^{T-1}(1+\eta c_{5,s})\right)+M\sqrt{\frac{\log\frac{1}{\delta}}{2}}\right)$
[43]	shallow	$\frac{1}{\sqrt{m}}\left\{O\left(C_g^2\eta C_{p,\lambda}\sum_{t=1}^{T}(C_{p,\lambda}(1+(\alpha_\sigma^2+\alpha_\ell)\eta C_g^2))^{t-1}\right)+M\sqrt{\frac{\log\frac{1}{\delta}}{2}}\right\}$
Ours	deep	$\frac{1}{\sqrt{m}}\left\{O\left(\left((K+1)\eta\kappa_1+\eta\kappa_2\right)^T\right)+M\sqrt{\frac{\log\frac{1}{\delta}}{2}}\right\}$

Note: m is the number of samples in the trained dataset; M is the upper bound of loss function $\ell(\cdot,\cdot)$; $\eta > 0$ is the learning rate; $\delta \in (0,1)$; T is the number of iterations for training $\mathcal{A}_\mathcal{S}$ using SGD; C_g represents the 2-norm of filter $g(\mathbf{L})$; α_σ, υ_σ are two parameters w.r.t the continuity of activation function $\sigma(\cdot)$; α_ℓ, υ_ℓ are two parameters w.r.t the continuity of the loss function $\ell(\cdot,\cdot)$. $c_{2,t}, c_{6,t}, c_{5,t} > 0$ $(t = 0, 1, \ldots, T)$ represent some specific parameters defined in [37]. $C_{p,\lambda} = \frac{28}{p(p-1)\lambda_t}(B_\ell/\lambda)^{(3-p)/p}$, where $B_\ell > 0$ is a parameter related to loss function $\ell(\cdot,\cdot)$, $1 < p \leq 2$, $\lambda > 0$ is the regularization parameter and $\lambda_t > 0$ is another regularization parameter dependent on λ and t, as detailed in [43]. K is number of hidden layers of the considered deep GCNs; κ_1 and κ_2 are two parameters as defined in (7) and (8).

different datasets. In the single layer case, since the input feature is the same, the variation of the related gradient is only dependent on the variations of learnable parameters. While, in the case of deep GCNs, the variation of the related gradients is also dependent on the variations of the gradients of the final output with respect to the hidden layer outputs. Please see Lemma 7 and its proof for details (see Appendix C in the supplementary material).

C. Stability Upper Bound

In this subsection, we establish the uniform stability of SGD for deep GCNs, which is the key to further proving Theorem 1.

Theorem 2 (Uniform stability of deep GCNs): Consider the deep GCNs defined by (1), which are trained on a dataset $\mathcal{S}$ using the SGD algorithm for a total of T iterations and denoted as $\mathcal{A}_\mathcal{S}$. Assume that Assumptions 1, 2, and 3 stated in Section IV-A are satisfied. Then, $\mathcal{A}_\mathcal{S}$ is μ_m-uniformly stable, with μ_m satisfying the following condition:

$$\mu_m \leq \frac{C}{m}\sum_{t=1}^{T}(1+(K+1)\eta\kappa_1+\eta\kappa_2)^{t-1}, \tag{11}$$

where

$$C := (K+1)\eta\alpha_\ell^2(B\alpha_\sigma C_g)^{2K}\alpha_\sigma^2 C_g^2 C_\mathbf{X}^2,$$

κ_1 and κ_2 are defined by (7) and (8), respectively.

With a straightforward calculation, one can see that

$$\mu_m \leq \frac{1}{m}O\left(\left((K+1)\eta\kappa_1+\eta\kappa_2\right)^T\right),$$

which decays at the rate of $\frac{1}{m}$ as m tends to infinity. Together with Lemma 1, it yields the result of Theorem 1.

Proof Sketch for Theorem 2: We prove Theorem 2 in the following two steps.

- *Step 1:* We begin by bounding the stability of deep GCNs with respect to perturbations in the learned parameters caused by changes in the training set. The result is given in Lemma 2.
- *Step 2:* Next, we provide a bound for the perturbation of the learned parameters. The result is presented in Theorem 3.

Consider $\mathcal{A}_\mathcal{S}$, a set of deepGCNs defined by (1), trained on the dataset $\mathcal{S}$ using SGD for T iterations. Let $\theta_t = \{\mathbf{W}_t^{(1)}, \ldots, \mathbf{W}_t^{(K)}, \mathbf{w}_t\}$ and $\theta_t' = \{\mathbf{W}_t^{(1)'}, \ldots, \mathbf{W}_t^{(K)'}, \mathbf{w}_t'\}$ (with $\theta_0 = \theta_0'$) denote the parameters of two GCNs trained on $\mathcal{S}$ and $\mathcal{S}^i$ after t iterations, respectively. We set $\triangle\mathbf{w}_t = \mathbf{w}_t - \mathbf{w}_t'$ and $\triangle\mathbf{W}_t^{(k)} = \mathbf{W}_t^{(k)} - \mathbf{W}_t^{(k)'}$ to be the perturbation of learning parameters and define

$$\|\triangle\theta_t\|_* = \|\triangle\mathbf{w}_t\|_2 + \sum_{k=1}^{K}\|\triangle\mathbf{W}_t^{(k)}\|_2. \tag{12}$$

In the following lemma, it is shown that the stability of $\mathcal{A}_\mathcal{S}$ can be bounded by $\|\triangle\theta_T\|_*$.

Lemma 2: Let θ_t and θ_t' be the learnt parameters of two GCNs trained on $\mathcal{S}$ and $\mathcal{S}^i$ using SGD in the t-th iteration with $\theta_0 = \theta_0'$, and $\triangle\theta_t := \theta_t - \theta_t'$. Suppose that all the assumptions made in Section IV-A hold. Then, after T iterations, we have that for any $\mathbf{z} = (\mathbf{x}, y)$ taken from $\mathcal{D}$,

$$\begin{aligned}&\left|\mathbb{E}_\mathcal{A}\left[\ell(\hat{y}, y)\right] - \mathbb{E}_\mathcal{A}\left[\ell(\hat{y}', y)\right]\right|\\ &\leq \alpha_\ell B^K\alpha_\sigma^{K+1}C_g^{K+1}C_\mathbf{X}\cdot\mathbb{E}_\mathcal{A}\left[\|\triangle\theta_T\|_*\right],\end{aligned} \tag{13}$$

where $\hat{y} = f(\mathbf{x}|\theta_T)$ and $\hat{y}' = f(\mathbf{x}|\theta_T')$.

We provide the proof of Lemma 2 in Appendix B (see the supplementary material).

Combining (5) and (13), the stability of $\mathcal{A}_\mathcal{S}$ has a bound

$$\mu_m \leq \frac{\alpha_\ell B^K\alpha_\sigma^{K+1}C_g^{K+1}C_\mathbf{X}}{2}\sup_\mathcal{S}\left\{\mathbb{E}_\mathcal{A}\left[\|\triangle\theta_T\|_*\right]\right\}. \tag{14}$$

So, to estimate the uniform stability of $\mathcal{A}_\mathcal{S}$, we need to bound $\mathbb{E}_\mathcal{A}[\|\triangle\theta_T\|_*]$. Now, let us recall (3) for parameter updating, for training on $\mathcal{S}$,

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta\nabla_\mathbf{w}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t),$$

$$\mathbf{W}_t^{(k)} = \mathbf{W}_{t-1}^{(k)} - \eta\nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t),$$

$k = 1, 2, \ldots, K$, and for training on $\mathcal{S}^i$,

$$\mathbf{w}_t' = \mathbf{w}_{t-1}' - \eta\nabla_\mathbf{w}\ell(f(\mathbf{x}_t'|\theta_{t-1}'), y_t'),$$

$$\mathbf{W}_t^{(k)'} = \mathbf{W}_{t-1}^{(k)'} - \eta\nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t'|\theta_{t-1}'), y_t'),$$

$k = 1, 2, \ldots, K$, where $(\mathbf{x}_t, y_t) \in \mathcal{S}$ and $(\mathbf{x}'_t, y'_t) \in \mathcal{S}^i$ are the samples drawn at the t-th SGD iteration. Therefore, $\triangle\theta_t = \{\triangle\mathbf{W}_t^{(1)}, \ldots, \triangle\mathbf{W}_t^{(K)}, \triangle\mathbf{w}_t\}$ has the following iterations:

$$\begin{aligned}\triangle\mathbf{w}_t &= \triangle\mathbf{w}_{t-1} \\ &\quad - \eta\left(\nabla_{\mathbf{w}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{w}}\ell(f(\mathbf{x}'_t|\theta'_{t-1}), y'_t)\right),\end{aligned}$$

and for $k = 1, 2, \ldots, K$,

$$\begin{aligned}\triangle\mathbf{W}_t^{(k)} &= \triangle\mathbf{W}_{t-1}^{(k)} \\ &\quad - \eta\left(\nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}'_t|\theta'_{t-1}), y'_t)\right),\end{aligned}$$

with $\|\triangle\theta_0\|_* = 0$.

So, we need to bound

$$\nabla_{\mathbf{w}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{w}}\ell(f(\mathbf{x}'_t|\theta'_{t-1}), y'_t)$$

and

$$\nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}'_t|\theta'_{t-1}), y'_t)$$

to obtain a bound of $\|\triangle\theta_t\|_*$. There are two scenarios to consider: i) At step t, SGD picks a sample $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$ which is identical in $\mathcal{S}$ and $\mathcal{S}^i$, and occurs with probability $(m-1)/m$; and ii) At step t, SGD picks the only samples that $\mathcal{S}$ and $\mathcal{S}^i$ differ, $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$ and $\mathbf{z}'_t = (\mathbf{x}'_t, \mathbf{y}'_t)$ which occurs with probability $1/m$. We provide the results in the following Lemma 3 and Lemma 4.

Lemma 3: Consider two GCNs with parameters θ_t and θ'_t, respectively. Then, the following holds for any sample $\mathbf{z}_t = (\mathbf{x}_t, y_t)$:

$$\begin{aligned}&\left\|\nabla_{\mathbf{w}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{w}}\ell(f(\mathbf{x}_t|\theta'_{t-1}), y_t)\right\|_F \\ &\leq \kappa_1\|\triangle\theta_{t-1}\|_*, \end{aligned} \tag{15}$$

and for $k = 1, 2, \ldots, K$,

$$\begin{aligned}&\|\nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t|\theta'_{t-1}), y_t)\|_F \\ &\leq (\kappa_1 + \rho_k)\|\triangle\theta_{t-1}\|_*, \end{aligned} \tag{16}$$

where κ_1 and ρ_k are defined by (7) and (A-12) (see the supplementary material), respectively.

Lemma 4: Consider two GCNs with parameters θ_t and θ'_t, respectively. Then, the following holds for any two samples $\mathbf{z}_t = (\mathbf{x}_t, y_t)$ and $\mathbf{z}'_t = (\mathbf{x}'_t, y'_t)$:

$$\begin{aligned}&\|\nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}_t|\theta_{t-1}), y_t) - \nabla_{\mathbf{W}^{(k)}}\ell(f(\mathbf{x}'_t|\theta'_{t-1}), y'_t)\|_F \\ &\leq 2\alpha_\ell B^K \alpha_\sigma^{K+1} C_g^{K+1} C_{\mathbf{X}}, \end{aligned} \tag{17}$$

for $k = 1, 2, \ldots, K+1$. Note that $\mathbf{W}^{(K+1)} = \mathbf{w}$.

The proofs of Lemmas 3 and 4 are given in Appendix C (see the supplementary material). We now provide a bound for $\mathbb{E}_{\mathcal{A}}[\|\triangle\theta_T\|_*]$.

Theorem 3: Let θ_t and θ'_t be the learnt parameters of two GCNs trained on $\mathcal{S}$ and $\mathcal{S}^i$ using SGD in the t-th iteration with $\theta_0 = \theta'_0$. The assumptions made in Section IV-A hold. Then, after T iterations, $\triangle\theta_T$ satisfies

$$\mathbb{E}_{\mathcal{A}}\left[\|\triangle\theta_T\|_*\right] \leq c\sum_{t=1}^{T}\Big(1 + (K+1)\eta\kappa_1 + \eta\kappa_2\Big)^{t-1}, \tag{18}$$

TABLE III
Statistics of the Three Benchmark Datasets

	Cora	Citeseer	Pubmed
# Nodes	2,708	3,327	19,717
# Edges	5,429	4,732	44,338
# Features	1,433	3,703	500
# Classes	7	6	3
Label Rate	0.052	0.036	0.003

where $c := \frac{2(K+1)\eta\alpha_\ell B^K\alpha_\sigma^{K+1}C_g^{K+1}C_{\mathbf{X}}}{m}$, and κ_1 and κ_2 are defined by (7) and (8), respectively.

The proof of Theorem 3, using Lemmas 3 and 4, is provided in Appendix D (see the supplementary material). Combining (14) and Theorem 3, we obtain that the uniform stability μ_m of $\mathcal{A}_{\mathcal{S}}$ has a bound as

$$\begin{aligned}\mu_m &\leq \alpha_\ell B^K\alpha_\sigma^{K+1}C_g^{K+1}C_{\mathbf{X}}\sup_{\mathcal{S}}\left\{\mathbb{E}_A\left[\|\triangle\theta_T\|_*\right]\right\} \\ &\leq \frac{C}{m}\sum_{t=1}^{T}\left(1 + (K+1)\eta\kappa_1 + \eta\kappa_2\right)^{t-1},\end{aligned}$$

which completes the proof of Theorem 2.

V. Experiments

In this section, we conduct some empirical studies using three benchmark datasets commonly utilized for the node classification task, namely Cora, Citeseer, and Pubmed [57], [58]. Table III summarizes the basic statistics of these datasets.

In our experiments, we follow the standard transductive learning problem formulation and the training/test setting used in [59]. To rigorously test our theoretical insights, our experiments aim to answer the following key questions:

- Q1: How does the design of graph filters (i.e., $g(\mathbf{L})$) influence the generalization gap?
- Q2: How does the generalization gap change with the number of hidden layers (i.e., K)?
- Q3: How does the width (i.e., the number of hidden units: d) affect the generalization gap?

To address each question, we empirically estimate the generalization gap by calculating the absolute difference in loss between training and test samples. We adopt the official TensorFlow implementation (https://github.com/tkipf/gcn) for GCN [59] and the Adam optimizer with default settings. The number of iterations is fixed to $T = 200$ for all the simulations.

Results and Discussion for Q1: We analyze two types of graph filters in our study: 1) the normalized graph filter, defined as $g(\mathbf{L}) = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ (which was first employed in the vanilla GCN [59] and has subsequently become widely used in follow-up works on GCNs), and 2) the random walk filter, $g(\mathbf{L}) = \mathbf{D}^{-1}\mathbf{A} + \mathbf{I}$. To fit our theoretical finding, we compare the performance of two 5-layer GCN models (with width $d = 32$ for each layer), each employing one of these filters. Table IV presents the numerical records of $R_{emp}(\mathcal{A}_{\mathcal{S}})$, $R(\mathcal{A}_{\mathcal{S}})$, $\epsilon_{gen}(\mathcal{A}_{\mathcal{S}})$, C_g for both filters. The results indicate clearly that the 5-layer GCN with the normalized graph filter exhibits a smaller generalization gap compared to the one

TABLE IV
THE GENERALIZATION GAP WITH DIFFERENT GRAPH FILTER FOR THREE DATASETS

Dataset	Graph filter $g(\mathbf{L})$	$R_{emp}(\mathcal{A}_{\mathcal{S}})$	$R(\mathcal{A}_{\mathcal{S}})$	$\epsilon_{gen}(\mathcal{A}_{\mathcal{S}})$	C_g
Cora	$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	1.488	0.136	**1.352**	1
	$\mathbf{D}^{-1}\mathbf{A}+\mathbf{I}$	1.914	0.118	1.796	4.746
Citeseer	$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	2.896	0.235	**2.661**	1
	$\mathbf{D}^{-1}\mathbf{A}+\mathbf{I}$	3.206	0.145	3.061	4.690
Pubmed	$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	1.594	0.023	**1.571**	1
	$\mathbf{D}^{-1}\mathbf{A}+\mathbf{I}$	2.534	0.037	2.497	7.131

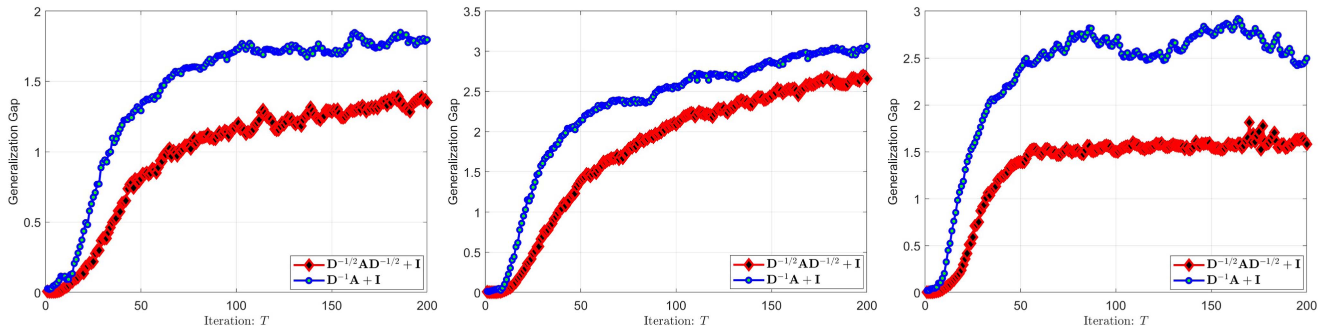

Fig. 1. Comparison of trends in the generalization gap: Cora (left), Citeseer (middle), Pubmed (right).

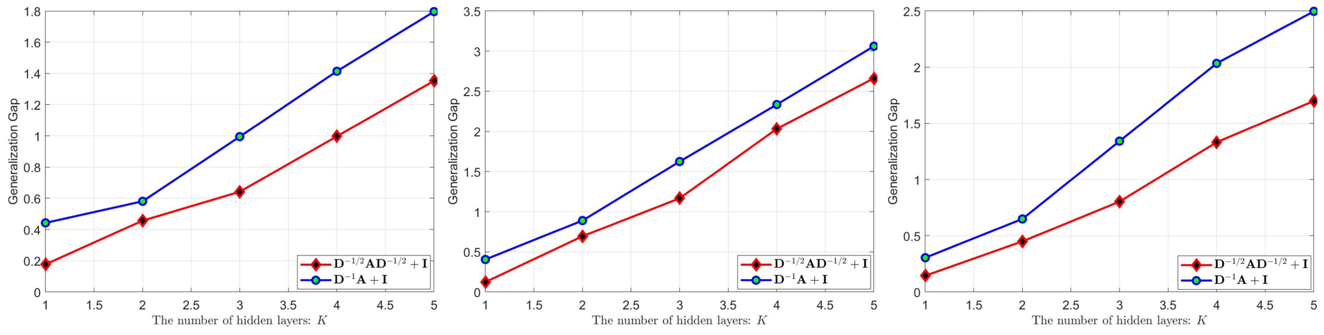

Fig. 2. Comparison of the generalization gap with different settings of network depth K: Cora (left), Citeseer (middle), Pubmed (right).

with the random walk filter. Furthermore, Fig. 1 illustrates the performance of each filter across different datasets over iterations, demonstrating the superior performance of the normalized graph filter. Overall, the empirical findings in Table IV and Fig. 1 align well with our theoretical finding regarding the impact of C_g on the generalization gap.

Results and Discussion for Q2: In this experimental study, we try different settings of K, i.e., the number of hidden layers. Specifically, for $K = \{1, 2, 3, 4, 5\}$, we compare the performance of two K-layer GCNs (with width $d = 32$ for each layer): one employing the normalized graph filter $g(\mathbf{L}) = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, and one using the random walk filter $g(\mathbf{L}) = \mathbf{D}^{-1}\mathbf{A} + \mathbf{I}$. Fig. 2 shows the performance comparison results for each K. It demonstrates clearly that, consistent with the aforementioned results for **Q1**, GCN with a normalized graph filter (with smaller C_g) consistently exhibits smaller generalization gaps compared to those with the random walk filter. Also, it is observed that the generalization gap becomes larger as K increases, further validating our theoretical assertions regarding the influence of K on the model's generalization gap.

Results and Discussion for Q3: To empirically investigate the impact of width d (i.e., the number of hidden units) on the generalization gap, we conduct additional experiments using a 5-layer GCN equipped with a normalized graph filter. The experiments specifically involve a comparison between a 5-layer GCN configured with a width of $2d$ for each layer and the previously studied model with d width ($d = 32$), as illustrated in Fig. 3. This setup allows for a direct comparison under varying network configurations, providing insights into how changes in the number of hidden units influence the generalization gap. As demonstrated in Fig. 3, across all the datasets examined, a d-width GCN consistently exhibits smaller generalization gaps compared to one with a $2d$-width. This observation is in harmony with our theoretical explanation presented after Theorem 1, that

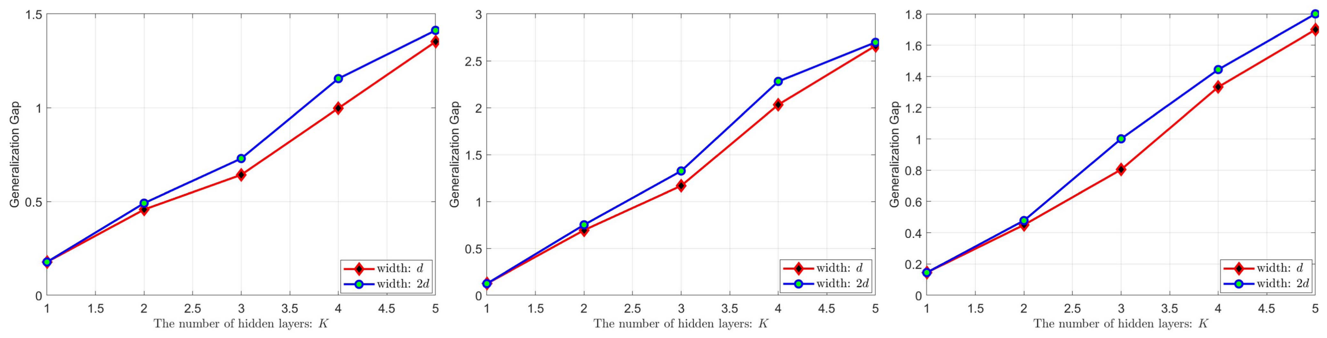

Fig. 3. Comparison of the generalization gap with different settings of network width d: Cora (left), Citeseer (middle), Pubmed (right).

is, the factor B (i.e., the upper bound of 2-norm of the parameters $\{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(K)}, \mathbf{w}\}$) directly influences factors κ_1 and κ_2 in the upper bound of the generalization gap.

VI. Theoretical Implications

Our work establishes a theoretical framework for analyzing the generalization gap of traditional deep GCNs, which further provides insights into extending the analysis to other classes of graph neural networks, including Graph Transformers. As illustrative examples, we briefly discuss how the theoretical proof methodology developed in our framework can be applied to GCNII and Graph Transformer, which are representative models of more advanced GNNs, thereby demonstrating the broader applicability of our theoretical framework.

A. Extension to GCNII

With input features $\mathbf{X}^{(0)} = \mathbf{X} \in \mathbb{R}^{N\times d}$, GCNII defines its k-th layer as

$$\mathbf{X}^{(k)} = \sigma\Bigg(\Big((1-a_k)g(\mathbf{L})\mathbf{X}^{(k-1)} + a_k\mathbf{X}^{(0)}\Big) \cdot \Big((1-b_k)\mathbf{I}_d + b_k\mathbf{W}^{(k)}\Big) \Bigg),$$

for $k = 1, 2, \ldots, K$, where $a_k, b_k \in (0,1)$ are two hyperparameters, $\mathbf{X}^{(k)}$ is the output feature matrix of the k-th layer, $\mathbf{W}^{(k)}$ is the trained parameter matrix specific to the k-th layer, graph filter $g(\mathbf{L}) = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, and $\mathbf{I}_d$ is the $d\times d$ identity matrix. The output for node $\mathbf{x}$ is

$$f(\mathbf{x}|\theta) = \sigma\left(\boldsymbol{\delta}_{\mathbf{x}}^{\top}\left((1-a_{K+1})g(\mathbf{L})\mathbf{X}^{(K)} + a_{K+1}\mathbf{X}^{(0)}\right)\mathbf{w}\right),$$

where $\theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(K)}, \mathbf{w}\}$ (all trainable parameters, with $\mathbf{w} \in \mathbb{R}^d$ the output layer parameter); $\boldsymbol{\delta}_{\mathbf{x}} \in \mathbb{R}^N$ is the indicator vector for node $\mathbf{x}$; $a_{K+1} \in (0,1)$ is a hyperparameter for the output layer residual connection. Let θ_t and θ'_t be the learnt parameters of two GCNs trained on $\mathcal{S}$ and $\mathcal{S}^i$ using SGD in the t-th iteration with $\theta_0 = \theta'_0$, and $\triangle\theta_t := \theta_t - \theta'_t$.

For each layer k, the perturbation of layer outputs $\|\triangle\mathbf{X}^{(k)}\|_F = \|\mathbf{X}^{(k)} - \mathbf{X}^{(k)\prime}\|_F$ satisfies the recursive bound:

$$\|\triangle\mathbf{X}^{(k)}\|_F \le c_1^{(k)}\|\triangle\mathbf{X}^{(k-1)}\|_F + c_2^{(k)}\|\triangle\mathbf{W}^{(k)}\|_2, \tag{19}$$

where $c_1^{(k)} = (1-a_k)(1-b_k+b_kB)\alpha_\sigma C_g$ and $c_2^{(k)} = \alpha_\sigma b_k((1-a_k)C_gB_{\mathbf{X}}^{(k-1)} + a_kC_{\mathbf{X}})$ with $B_{\mathbf{X}}^{(k-1)}$ the bound of $\|\mathbf{X}^{(k-1)}\|_F$ (see (A.22) in the Appendix E). The first term on the right side of the iterative formula captures propagation of perturbations from the previous layer, while the second term captures perturbation from $\mathbf{W}^{(k)}$.

By induction, it yields that

$$\|\triangle\mathbf{X}^{(k)}\|_F \le e^{(k)}\left(\sum_{j=1}^{k}\|\triangle\mathbf{W}^{(k)}\|_2\right), \tag{20}$$

where $e^{(k)} = \max\{c_1^{(k)}e^{(k-1)}, c_2^{(k)}\}$ with $e^{(0)} = 0$. We provide the proof of (19) and (20) in Appendix E. Then, combining layer-wise bounds and using the Lipschitz property of σ, one can have the output perturbation $|f(x|\theta) - f(x|\theta')|$ bounded by the total parameter perturbation $\|\Delta\theta\|_* = \sum_{j=1}^{K}\|\mathbf{W}^{(j)} - \mathbf{W}^{(j)\prime}\|_2 + \|\mathbf{w}-\mathbf{w}'\|_2$ (see Appendix E for technical details) as

$$|f(\mathbf{x}|\theta) - f(\mathbf{x}|\theta')| \le \alpha_\sigma \cdot \varrho\|\triangle\theta\|_*, \tag{21}$$

where $\varrho = \max\{(1-a_{K+1})BC_g\cdot e^{(K)}, (1-a_{K+1})C_gB_{\mathbf{X}}^{(K)} + a_{K+1}C_{\mathbf{X}}\}$. Then,

$$\begin{aligned}&\Big|\mathbb{E}_{\mathcal{A}}\left[\ell(\hat{y},y)\right] - \mathbb{E}_{\mathcal{A}}\left[\ell(\hat{y}',y)\right]\Big| \\ &= \Big|\mathbb{E}_{\mathcal{A}}\left[\ell\left(f(\mathbf{x}|\theta_T),y\right) - \ell\left(f(\mathbf{x}|\theta'_T),y\right)\right]\Big| \\ &\le \alpha_\ell\mathbb{E}_{\mathcal{A}}\left[\left|f(\mathbf{x}|\theta_T) - f(\mathbf{x}|\theta'_T)\right|\right] \le \varrho\alpha_\ell\cdot\mathbb{E}_{\mathcal{A}}\left[\|\triangle\theta_T\|_*\right].\end{aligned}$$

This implies that the stability of $\mathcal{A}_{\mathcal{S}}$ for GCNII has a bound

$$\mu_m \le \frac{\varrho\alpha_\ell}{2}\sup_{\mathcal{S}}\left\{\mathbb{E}_{\mathcal{A}}[\|\triangle\theta_T\|_*]\right\}.$$

Note that when $a_k = 0, b_k = 1$ for all k, GCNII degenerates into the traditional GCN, we have $\varrho = B^K\alpha_\sigma^K C_g^{K+1}C_{\mathbf{X}}$, and thus

$$\mu_m \le \frac{\alpha_\ell B^K\alpha_\sigma^K C_g^{K+1}C_{\mathbf{X}}}{2}\sup_{\mathcal{S}}\left\{\mathbb{E}_{\mathcal{A}}[\|\triangle\theta_T\|_*]\right\},$$

which is consistent with (14).

To further bound $\|\triangle\theta_T\|_*$, the crucial step is to bound the perturbation of the gradient of $f(\mathbf{x}|\theta)$ with respect to the parameters $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_K, \mathbf{w}\}$ and obtain the result similar to Lemma 7 in Appendix C, which can be achieved by following the technique in our paper. Here, we provide the result for $\|\nabla_{\mathbf{w}} f(\mathbf{x}|\theta) - \nabla_{\mathbf{w}} f(\mathbf{x}|\theta')\|_F$:

$$\|\nabla_{\mathbf{w}} f(\mathbf{x}|\theta) - \nabla_{\mathbf{w}} f(\mathbf{x}|\theta')\|_F \leq \Big(\nu_\sigma \varrho \cdot \big((1-a_{K+1})C_g B_{\mathbf{X}}^{(K)} + a_{K+1}C_{\mathbf{X}}\big) + \alpha_\sigma \cdot (1-a_{K+1})C_g e^{(K)}\Big) \cdot \|\triangle\theta\|_*, \qquad (22)$$

where $\varrho = \max\{(1-a_{K+1})BC_g \cdot e^{(K)}, (1-a_{K+1})C_g B_{\mathbf{X}}^{(K)} + a_{K+1}C_{\mathbf{X}}\}$. Note that when $a_k = 0, b_k = 1$ for all k, GCNII degenerates into the traditional GCN, we have $\varrho = B^K \alpha_\sigma^K C_g^{K+1} C_{\mathbf{X}}$, $B_{\mathbf{X}}^{(K)} = B^K \alpha_\sigma^K C_g^K C_{\mathbf{X}}$ and $e^{(K)} = B^{K-1}\alpha_\sigma^K C_g^K C_{\mathbf{X}}$. At this point,

$$\big\|\nabla_{\mathbf{w}} f(\mathbf{x}|\theta) - \nabla_{\mathbf{w}} f(\mathbf{x}|\theta')\big\|_F \leq \left(\upsilon_\sigma B^{2K}\alpha_\sigma^{2K} C_g^{2K+2} C_{\mathbf{X}}^2 + B^{K-1}\alpha_\sigma^{K+1} C_g^{K+1} C_{\mathbf{X}}\right)\|\triangle\theta\|_*,$$

which is consistent with (A.10) in Appendix C. For the bound of $\|\nabla_{\mathbf{W}^{(k)}} f(\mathbf{x}|\theta) - \nabla_{\mathbf{W}^{(k)}} f(\mathbf{x}|\theta')\|_F$, we refer the readers to the proof process of (A.27) in Appendix E.

Finally, these structured analysis results can lead to the results corresponding Lemma 3 and Lemma 4, and thus enable bounding the stability of GCNII.

B. Extension to Graph Transformer

To extend our theoretical framework to more complex models like Graph Transformer, the key is to bound the generalization gap of Graph Transformer by quantifying how perturbations in the training set (e.g., removing or replacing a node) propagate to changes in model outputs. Graph Transformer introduce new learnable parameters: query ($\mathbf{W}_Q$), key ($\mathbf{W}_K$), and value ($\mathbf{W}_V$) projection matrices, alongside attention scalers and feed-forward layers, for which a self-attention layer is defined [42] as

$$F(\mathbf{x}_n) = \mathbf{a}^\top \operatorname{Relu}\left(\mathbf{W}_O \sum_{i\in\mathcal{T}^n} \mathbf{W}_V \mathbf{x}_i \cdot \operatorname{softmax}_n \left((\mathbf{W}_K \mathbf{x}_i)^\top \mathbf{W}_Q \mathbf{x}_n\right)\right),$$

where $\mathbf{x}_i$ denotes features of node i, $\mathcal{T}^n$ is the set of nodes for the aggregation computing of node n, and $\operatorname{softmax}_n(h(i,n)) = \exp(h(i,n))/\sum_{j\in\mathcal{T}^n}\exp(h(j,n))$. Despite their architectural complexity (e.g., self-attention mechanisms, query/key/value projections), gradient decomposition still remains to be conducted via the product rule and chain rule, accounting for the propagation of attention-weight variations to the final output. Besides, a Lipschitz-type inequality for softmax may be critically needed, for which we claim that for $\mathbf{z} = (z_1, z_2, \ldots, z_p)$, $\mathbf{z}' = (z_1', z_2', \ldots, z_p')$ with $\|\mathbf{z} - \mathbf{z}'\|_\infty \leq 1$,

$$\|\operatorname{softmax}(\mathbf{z}) - \operatorname{softmax}(\mathbf{z}')\|_1 \leq 2e\|\mathbf{z} - \mathbf{z}'\|_\infty. \qquad (23)$$

Actually, the proof is not hard to set up by straight forward boundedness and the mean value theorem of exponential functions (see the technical details in Appendix F in the Supplementary Material).

For trainable parameters $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$, set the attention output is:

$$F(\mathbf{x}_n) = \mathbf{a}^\top \operatorname{ReLu}\left(\mathbf{W}_O \sum_{i\in\mathcal{T}^n} \mathbf{W}_V \mathbf{x}_i \cdot \operatorname{Attn}(\mathbf{x}_n)_i\right),$$

where $S_{i,n} = (\mathbf{W}_K\mathbf{x}_i)^T(\mathbf{W}_Q\mathbf{x}_n)$ is the scaled dot-product score, $A_{i,n} = \operatorname{softmax}_n(S_{i,n})$ are attention weights, and $\operatorname{Attn}(\mathbf{x}_n) = \sum_{i\in\mathcal{T}^n}\mathbf{W}_V\mathbf{x}_i \cdot A_{i,n}$ the attention output. Then the gradient decomposition with respect to $\mathbf{W}_K$ is given by

$$\nabla_{\mathbf{W}_K} F(\mathbf{x}_n) = \underbrace{\nabla_{\operatorname{ReLU}(\mathbf{Z})} F(\mathbf{x}_n)}_{①} \cdot \underbrace{\nabla_{\mathbf{Z}}\operatorname{ReLU}(\mathbf{Z})}_{②} \cdot \underbrace{\nabla_{\operatorname{Attn}(\mathbf{x}_n)}\mathbf{Z}}_{③} \cdot \underbrace{\nabla_{\mathbf{A}}\operatorname{Attn}(\mathbf{x}_n)}_{④} \cdot \underbrace{\nabla_S \mathbf{A}}_{⑤} \cdot \underbrace{\nabla_{\mathbf{W}_K} S}_{⑥}$$

where $\mathbf{Z} = \mathbf{W}_O \cdot \operatorname{Attn}(\mathbf{x}_n)$, $\mathbf{A} = \{A_{i,n}\}$, and $\mathbf{S} = \{S_{i,n}\}$. Then calculating each item gives that

$$\nabla_{\mathbf{W}_K} F(\mathbf{x}_n) = \mathbf{a}^\top \mathbb{I}_{\geq 0}(\mathbf{W}_O \cdot \operatorname{Attn}(\mathbf{x}_n)) \cdot \mathbf{W}_Q \cdot \mathbf{W}_V \cdot \left(\sum_{i\in\mathcal{T}^n} \mathbf{A}_{i,n}(\mathbf{x}_i - \bar{\mathbf{x}}_n)\mathbf{x}_i^\top\right) \cdot (\mathbf{W}_Q\mathbf{x}_n)^\top.$$

By leveraging the Lipschitz continuity of the gradient with respect to its trainable parameters, it can lead to bounding the gradient perturbation in terms of the total parameter perturbation $\|\triangle\theta\|_* = \|\mathbf{W}_K - \mathbf{W}_K'\|_2 + \|\mathbf{W}_V - \mathbf{W}_V'\|_2 + \|\mathbf{W}_O - \mathbf{W}_O'\|_2 + \|\mathbf{W}_Q - \mathbf{W}_Q'\|_2 + \|\mathbf{a} - \mathbf{a}'\|_2$ by

$$\|\nabla_{\mathbf{W}_K} F(\mathbf{x}_n|\theta) - \nabla_{\mathbf{W}_K} F(\mathbf{x}_n|\theta')\|_2 \leq 2eK_{\max}B^3 C_{\mathbf{X}}^3 \|\triangle\theta\|_*, \qquad (24)$$

where $K_{\max} \geq |\mathcal{T}^n|$ is the maximum neighborhood size, B is the upper bound of weight matrices (technical details in Appendix F). It mirrors the Lemma 7 in our approach for deep GCNs, where we recursively decomposed gradients across layers (see Lemma 7 in the Supplementary Material). For Graph Transformer, similar recursive relations can be derived for attention layers, with additional terms capturing interactions between $\mathbf{W}_Q\mathbf{X}, \mathbf{W}_K\mathbf{X}, \mathbf{W}_V\mathbf{X}$. For GCNs, we bounded gradient variations using norms of graph filters and layer parameters (e.g., $\|g(\mathbf{L})\|_2$, $\|\mathbf{W}^{(k)}\|_2$). For Graph Transformer, this will be extended to: singular values of $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ (analogous to C_g in GCNs), as they control the "strength" of feature projections and Lipschitz constants of softmax and feed-forward activations (replacing α_σ for GCN activations, and leads to an analogous to Theorem 2 for deep GCNs.

VII. Conclusion and Further Remarks

This paper explores the generalization of deep GCNs by providing an upper bound on their generalization gap. Our generalization bound is obtained based on the algorithmic stability of deep GCNs trained by the SGD algorithm. Our analysis demonstrates that the algorithmic stability of deep GCNs is contingent upon two factors: the largest absolute eigenvalue (or maximum singular value) of graph filter operators and the number of layers utilized. In particular, if the aforementioned eigenvalue (or singular value) remains invariant regardless of changes in the graph size, deep GCNs exhibit robust uniform stability, resulting in an enhanced generalization capability. Additionally, our results suggest that a greater number of layers can increase the generalization gap and subsequently degrade the performance of deep GCNs. This provides guidance for designing well-performing deep GCNs with a proper number of layers [60]. Most importantly, the result of single-layer GCNs in [36] can be regarded as a special case of our results in deep GCNs without hidden layers.

While our study is primarily focused on exploring the fundamental principles of generalizability and stability in the context of a simple deep GCN model framework, the theoretical insights obtained here can also offer preliminary perspectives on several research topics that have drawn increasing attention in the graph neural network community. These include, among others, the over-smoothing problem in deep architectures [61], [62], the design of models tailored for heterophilic graphs [63], [64], and the emerging topic of graph out-of-distribution (OOD) generalization [65], [66]. Our theoretical study can provide potential hints toward these directions, but more fine-grained and comprehensive work is still needed to fully address them. Below, we elaborate on these aspects in turn, aiming to clarify their conceptual connections with our work, outline possible directions for extending our theoretical framework, and highlight three open and challenging questions that can serve as seeds for future exploration.

How can the impact of over-smoothing in deep GCNs be mitigated? We first note that, given a trivial deep GCN model characterized by over-smoothed node embeddings (which typically result in significant training errors), our theoretical upper bound still holds — that is, for a given graph filter, an increase in layers could potentially increase this upper bound in a probabilistic sense. This also motivates the exploration of advanced deep GCN models that incorporate mechanisms to counteract over-smoothing, such as the skip connection technique used in GCNII [41] and its follow-up works. As detailed in Section V, our theoretical results can in fact be extended to the setting of GCNII, thereby providing analytical support for architectures that integrate skip connections. In both theory and practice, reducing the maximum absolute eigenvalue of graph filter operators is achievable through the strategic implementation of skip connections across layers, which can potentially reduce the generalization gap. From this perspective, our findings may inspire further studies into sophisticated deep GCN architectures designed to mitigate over-smoothing, offering a promising direction for both theoretical and practical advancements.

What is the role of heterophily in GCN generalization? It is also valuable to consider extending our theoretical analysis to models specifically designed for heterophilic graphs, where nodes often connect to neighbors with dissimilar labels. This would require incorporating the homophily/heterophily ratio of the input graph signal into the upper bound estimation, thereby capturing how graph signal characteristics influence generalization. Although our empirical study here considers two types of low-pass filters on homophilic benchmark datasets (Cora, Citeseer, Pubmed), our theoretical framework is not restricted to low-pass scenarios alone. As remarked in Section IV-B, the analysis framework is in principle applicable to a broader range of filtering schemes; however, the derivations in our proofs do not explicitly examine the impact of specific quantities such as the homophily/heterophily ratio, leaving this as an open aspect for further refinement. To ensure a consistent and fair empirical evaluation, as demonstrated in [36], we adopt homophilic datasets that are standard in prior stability and generalization analyses of GCNs. For analyses involving high-pass filters, it would be appropriate to engage with heterophilic benchmark datasets (e.g., Texas, Wisconsin, Cornell). Relevant to this discussion is the recent work [47], which employs analytical tools from statistical physics and random matrix theory to precisely characterize generalization in simple GCNs on the contextual stochastic block model (CSBM). Such studies, although based on specific graph signal assumptions, could inspire refinements to our theoretical framework by jointly considering graph signal characteristics (homophily/heterophily) and model complexities (filter types, depth, and width).

Can insights from in-distribution generalization inform OOD generalization? Beyond the above considerations, another relevant line of research that has recently attracted considerable attention is graph out-of-distribution (OOD) generalization [65], [66]. It is worth clarifying that the problem setting and theoretical assumptions in OOD generalization are distinct from those in the in-distribution generalization framework considered in this work. In-distribution generalization focuses on scenarios where both training and test data are drawn from the same underlying distribution, enabling rigorous analysis under well-defined stochastic assumptions, such as those adopted in our stability-based framework. In contrast, OOD generalization addresses cases involving distribution shifts, which often require additional modeling principles (e.g., invariance to spurious correlations, causal structure modeling, or domain adaptation techniques) and seek performance guarantees that hold across domains. Despite these differences, the two areas can be mutually beneficial: in-distribution analyses, such as our characterization of bias–variance trade-offs and the influence of spectral properties of graph filters on generalization, may offer insights for developing more OOD-robust architectures; conversely, OOD-oriented approaches, such as invariant risk minimization or causal subgraph intervention, may inspire new regularization schemes or architectural components that also enhance in-distribution performance. Related to this discussion, the authors in [67] analyze a one-layer GCN trained on the CSBM via logistic regression, providing theoretical insights into improved linear separability and out-of-distribution generalization in semi-supervised node classification. Extending the current stability-based framework to accommodate mild forms of distribution shift thus presents an appealing research direction that could bridge these two lines of work and advance the understanding of generalization in graph neural networks.

Taken together, these discussions highlight that our theoretical framework, while developed under a specific in-distribution setting, has the potential to be extended and adapted to address a broader range of challenges in graph learning.

Building on the above open questions, which outline core challenges for future exploration, it is also important to consider more concrete research directions and methodological extensions. For example, the theoretical analysis presented in this study could be extended to encompass other commonly used learning algorithms in graph neural networks, moving beyond the scope of SGD. Our theoretical results may also inform the exploration of strategies to enhance the generalization capability of deep graph neural networks, such as investigating the efficacy of regularization techniques, conducting advanced network architecture searches, or developing adaptive graph filters. In addition, establishing the potential connection between model stability, generalization, and the issues of over-smoothing and over-squashing represents another promising avenue. Understanding these interrelationships could contribute to the development of novel techniques and algorithms that address these challenges, thereby complementing the broader problem-oriented directions discussed above and improving the overall effectiveness of deep graph neural networks in dealing with more complex tasks.

Acknowledgment

The authors would like to express their sincere gratitude to the Associate Editor and the anonymous reviewers for their valuable comments and constructive suggestions, which have greatly contributed to enhancing the quality of this manuscript. The authors also wish to thank Dr. Yi Wang (City University of Hong Kong, Hong Kong SAR, China) and Dr. Xianchen Zhou (National University of Defense Technology, China) for their insightful discussions and dedicated assistance with the experimental studies.

References

[1] Y. Liang, F. Meng, Y. Zhang, Y. Chen, J. Xu, and J. Zhou, "Emotional conversation generation with heterogeneous graph neural network," *Artif. Intell.*, vol. 308, 2022, Art. no. 103714.

[2] Y. Ma and J. Tang, *Deep Learning on Graphs*. Cambridge, U.K.: Cambridge Univ. Press, 2021.

[3] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2005, pp. 729–734.

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[5] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 498–511, Mar. 2009.

[6] K. Yao, J. Liang, J. Liang, M. Li, and F. Cao, "Multi-view graph convolutional networks with attention mechanism," *Artif. Intell.*, vol. 307, 2022, Art. no. 103708.

[7] W. L. Hamilton, *Graph Representation Learning*. San Rafael, CA, USA: Morgan & Claypool, 2020.

[8] L. Wu, P. Cui, J. Pei, and L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Berlin, Germany: Springer, 2022.

[9] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional ARMA filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3496–3507, Jul. 2022.

[10] B. Jiang, B. Wang, S. Chen, J. Tang, and B. Luo, "Graph neural network meets sparse representation: Graph sparse neural networks via exclusive group lasso," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12692–12698, Oct. 2023.

[11] H. Zhang, Y. Zhu, and X. Li, "Decouple graph neural networks: Train multiple simple GNNs simultaneously instead of one," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 11, pp. 7451–7462, Nov. 2024.

[12] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Netw.*, vol. 129, pp. 203–221, 2020.

[13] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[14] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[15] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

[16] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.

[17] L. Zhao and L. Akoglu, "PairNorm: Tackling oversmoothing in GNNs," in *Proc. Int. Conf. Learn. Representations*, 2020.

[18] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *Proc. Int. Conf. Learn. Representations*, 2020.

[19] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. Int. Conf. Learn. Representations*, 2020.

[20] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards model-level explanations of graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 430–438.

[21] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 12241–12252.

[22] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5782–5799, May 2023.

[23] T. Schnake et al., "Higher-order explanations of graph neural networks via relevant walks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7581–7596, Nov. 2022.

[24] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 657–668, Jan. 2023.

[25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019.

[26] Z. Chen, S. Villar, L. Chen, and J. Bruna, "On the equivalence between graph isomorphism testing and function approximation with GNNs," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 15868–15876.

[27] N. Dehmamy, A.-L. Barabási, and R. Yu, "Understanding the representation power of graph neural networks in learning graph topology," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 15413–15423.

[28] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5723–5733.

[29] S. Zhang, M. Wang, S. Liu, P.-Y. Chen, and J. Xiong, "Fast learning of graph neural networks with guaranteed generalizability: One-hidden-layer case," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 11268–11277.

[30] F. Scarselli, A. C. Tsoi, and M. Hagenbuchner, "The Vapnik-Chervonenkis dimension of graph and recursive neural networks," *Neural Netw.*, vol. 108, pp. 248–259, 2018.

[31] V. Garg, S. Jegelka, and T. Jaakkola, "Generalization and representational limits of graph neural networks," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 3419–3430.

[32] K. Oono and T. Suzuki, "Optimization and generalization analysis of transduction through gradient boosting and application to multi-scale graph neural networks," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 18917–18930.

[33] S. Lv, "Generalization bounds for graph convolutional neural networks via rademacher complexity," 2021, *arXiv:2102.10234*.

[34] P. Esser, L. Chennuru Vankadara, and D. Ghoshdastidar, "Learning theory can (sometimes) explain generalisation in graph neural networks," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 27043–27056.

[35] H. Tang and Y. Liu, "Towards understanding the generalization of graph neural networks," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 33674–33719.

[36] S. Verma and Z.-L. Zhang, "Stability and generalization of graph convolutional neural networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1539–1548.

[37] M. K. Ng and A. Yip, "Stability and generalization of graph convolutional networks in eigen-domains," *Anal. Appl.*, vol. 21, no. 03, pp. 819–840, 2023.

[38] R. Liao, R. Urtasun, and R. Zemel, "A PAC-Bayesian approach to generalization bounds for graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2021.

[39] H. Ju, D. Li, A. Sharma, and H. R. Zhang, "Generalization in graph neural networks: Improved PAC-Bayesian bounds on graph diffusion," in *Proc. 26th Int. Conf. Artif. Intell. Statist.*, 2023, pp. 6314–6341.

[40] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8580–8589.

[41] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1725–1735.

[42] H. Li, M. Wang, T. Ma, S. Liu, Z. Zhang, and P.-Y. Chen, "What improves the generalization of graph transformers? A theoretical dive into the self-attention and positional encoding," in *Proc. 41th Int. Conf. Mach. Learn.*, 2024, pp. 28784–28829.

[43] S. Liu, L. Wei, S. Lv, and M. Li, "Stability and generalization of ℓ_p-regularized stochastic learning for GCN," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, 2023, pp. 5685–5693.

[44] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2751–2768, Mar. 2023.

[45] K. Xu, J. Li, M. Zhang, S. S. Du, K.-I. Kawarabayashi, and S. Jegelka, "What can neural networks reason about?," in *Proc. Int. Conf. Learn. Representations*, 2020.

[46] K. Xu, M. Zhang, J. Li, S. S. Du, K.-I. Kawarabayashi, and S. Jegelka, "How neural networks extrapolate: From feedforward to graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2021.

[47] C. Shi, L. Pan, H. Hu, and I. Dokmanić, "Homophily modulates double descent generalization in graph convolution networks," in *Proc. Nat. Acad. Sci.*, vol. 121, no. 8, 2024, Art. no. e2309504121.

[48] A. Vasileiou, S. Jegelka, R. Levie, and C. Morris, "Survey on generalization theory for graph neural networks," 2025, *arXiv:2503.15650*.

[49] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1225–1234.

[50] W. Cong, M. Ramezani, and M. Mahdavi, "On provable benefits of depth in training graph convolutional networks," in *Proc. 35rd Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 9936–9949.

[51] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014.

[52] H. Li, M. Wang, S. Liu, P.-Y. Chen, and J. Xiong, "Generalization guarantee of training graph convolutional networks with graph topology sampling," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 13014–13051.

[53] N. Keriven, A. Bietti, and S. Vaiter, "Convergence and stability of graph convolutional networks on large random graphs," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21512–21523.

[54] X. Zhou, K. Hu, and H. Wang, "A tighter generalization error bound for wide GCN based on loss landscape," *Appl. Comput. Harmon. Anal.*, vol. 78, 2025, Art. no. 101777.

[55] S. Jegelka, "Theory of graph neural networks: Representation and learning," 2022, *arXiv:2204.07697*.

[56] A. Elisseeff, T. Evgeniou, M. Pontil, and L. P. Kaelbing, "Stability of randomized learning algorithms," *J. Mach. Learn. Res.*, vol. 6, no. 1, pp. 55–79, 2005.

[57] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.

[58] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 40–48.

[59] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[60] G. Li, C. Xiong, G. Qian, A. Thabet, and B. Ghanem, "DeeperGCN: Training deeper GCNs with generalized aggregation functions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13024–13034, Nov. 2023.

[61] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023, *arXiv:2303.10993*.

[62] T. Chen et al., "Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2769–2781, Mar. 2023.

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

[64] J. Zhu, Y. Yan, M. Heimann, L. Zhao, L. Akoglu, and D. Koutra, "Heterophily and graph neural networks: Past, present and future," *IEEE Data Eng. Bull.*, vol. 47, no. 2, pp. 10–32, Feb. 2023.

[65] H. Li, X. Wang, Z. Zhang, and W. Zhu, "OOD-GNN: Out-of-distribution generalized graph neural network," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 7328–7340, Jul. 2023.

[66] H. Li, X. Wang, Z. Zhang, and W. Zhu, "Out-of-distribution generalization on graphs: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 11, pp. 10490–10512, Nov. 2025.

[67] A. Baranwal, K. Fountoulakis, and A. Jagannath, "Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization," 2021, *arXiv:2102.06966*.

Guangrui Yang received the BS and PhD degrees in mathematics from Sun Yat-sen University, Guangzhou, China, in 2015 and 2021, respectively. He was a postdoctoral fellow with the Department of Mathematics, City University of Hong Kong, Hong Kong, China. He is currently with the Department of Mathematics, College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His research interests include graph signal processing, graph neural networks, and computational harmonic analysis, etc.

Ming Li (Member, IEEE) received the PhD degree from the Department of Computer Science and IT with La Trobe University, Australia, in 2017. He is currently a "Shuang Long Scholar" Distinguished Professor with the Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, China. He is the Director of GraphME Lab (**Graph M**achine Learning and Intelligent **E**ducation). He completed two postdoctoral fellowship positions with the Department of Mathematics and Statistics, La Trobe University, Australia, and the Department of Information Technology in Education, South China Normal University, China, respectively. He has published more than 100 papers in top-tier journals and conferences, including *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *Artificial Intelligence*, *IEEE Transactions on Knowledge and Data Engineering*, NeurIPS, ICML, IJCAI, AAAI, IJCAI, etc. He, as a leading guest editor, organized a special issue "*Deep Neural Networks for Graphs: Theory, Models, Algorithms and Applications*" in IEEE TNNLS, and a special session on "*Recent Advances in Deep Learning for Graphs*" in LOD2022. He is a Member of IEEE Task Force on Learning For Graphs, an Associate Editor for *Neural Networks*, *Applied Intelligence*, *Alexandria Engineering Journal*, *Network: Computation in Neural Systems*, *Soft Computing*, *Neural Processing Letters*. His research interests include graph neural networks, graph representation learning, hypergraph learning, learning theory for neural networks, etc.

Han Feng received the BS degree in mathematics from Beijing Normal University, China, in 2011, and the MSc and PhD degrees in mathematics from University of Alberta, Canada, in 2013 and 2016, respectively. She is currently an associate professor with the Department of Mathematics, City University of Hong Kong. Her research interests include graph neural networks, graph signal processing, spherical approximation theory, learning theory, and deep neural networks.

Xiaosheng Zhuang received the bachelor's and master's degree in mathematics from Sun Yat-Sen (Zhongshan) University, China, in 2003 and 2005, respectively, and the PhD degree in applied mathematics from the University of Alberta, Edmonton, AB, Canada, in 2010. He was a postdoctoral fellow with the Universität Osnabrück in 2011 and Technische Universität Berlin in 2012. He is a professor with the Department of Mathematics, City University of Hong Kong, which he joined since 2012. His research interests include graph neural networks, applied and computational harmonic analysis, sparse approximation and directional multiscale representation systems, deep and machine learning, and signal/image processing.