**SStaGCN: Simplified stacking based graph convolutional networks**

Jia Cai  
*School of Digital Economy, Guangdong University of Finance & Economics*  
Guangzhou, 510320, P. R. China.  
jiacai1999@gdufe.edu.cn

Zhilong Xiong  
*School of Statistics and Mathematics, Guangdong University of Finance & Economics*  
Guangzhou, 510320, P. R. China.  
zhilongxiong98@outlook.com

Shaogao Lv*  
*Department of Statistics and Data Science, Nanjing Audit University*  
Nanjing, 211815, P. R. China.  
lvsg716@nau.edu.cn

Received (Day Month Year)  
Revised (Day Month Year)

Graph Convolutional Networks (GCNs) are powerful models extensively studied for various graph-structured data learning tasks. However, designing GCNs that effectively mitigate the over-smoothing phenomenon remains a crucial challenge. In this paper, we propose a novel Simplified Stacking-based GCN (SStaGCN) framework, leveraging stacking and aggregation techniques to address different types of graph-structured data. Specifically, we first employ stacking base models to extract node features from the graph. Next, we apply aggregation methods—such as mean, attention, and voting techniques—to further enhance feature extraction capabilities. These refined node features are then fed into a vanilla GCN model. Additionally, we provide a theoretical analysis of the generalization bound for the proposed model. Extensive experiments on three public citation networks and three heterogeneous tabular datasets demonstrate the effectiveness and efficiency of the SStaGCN approach over several state-of-the-art GCNs. Notably, SStaGCN effectively mitigates the over-smoothing issue common in GCNs.

*Keywords*: Graph convolutional network; stacking; aggregation.

Mathematics Subject Classification 2010: 68T05, 68W40

1. **Introduction**

Recent research on learning from graph-structured data has gained considerable attention across diverse fields within artificial intelligence. Graph Neural Net-
works (GNNs), particularly Graph Convolutional Networks (GCNs) [14,15], have achieved remarkable success in modeling graph-structured data and have been widely applied to recommendation systems [38], computer vision [5], molecular design [37], natural language processing [40], node classification [19], and clustering tasks [49].

Despite these successes, real-world applications present diverse types of graph-structured data, raising the question: can we design a general framework to handle distinct types of graph-structured data in a simpler, more effective manner? Additionally, as discussed in [24], the graph convolution operation in GCNs is a specific form of Laplacian smoothing, which mixes node features across different clusters and can lead to over-smoothing [24].

Sun et al. [39] addressed this issue by developing an RNN-like GCN with AdaBoost, enabling the extraction of knowledge from high-order neighbors of current nodes. However, the relation \( A^\ell = B^\ell \) does not necessarily imply \( A = B \). A simple example illustrating this is the case where \( \ell = 2 \) (two layers), with

\[
A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.
\]

As a result, this RNN-like GCN inevitably loses crucial information about the original graph structure, leaving the challenge of designing a simple GCN model to tackle the over-smoothing issue unresolved.

The stacking method [9], also known as stacked generalization, is an ensemble approach that combines multiple classifiers using base models and a meta-model. Stacking has shown success in feature extraction tasks across various data types, including disordered or irregular data, due to its advantageous properties:

- By combining distinct learners in the base models, stacking effectively captures discernible features. Base models are trained on the entire training dataset, allowing for robust performance evaluation on test data. A meta-model then aggregates these outputs to make final predictions on the test data. Combining stacking with GCN models can yield significant benefits. In this paper, we propose a novel simplified stacking-based architecture, Simplified Stacking-based GCN (SStaGCN), to address the over-smoothing issue in GCNs. SStaGCN integrates the feature extraction strengths of stacking with GCNs’ graph learning capabilities. This synergy enables SStaGCN to harness the strengths of both stacking and GCNs.

The main contributions of this paper are as follows:

- We propose a novel, versatile architecture that combines simplified stacking and GCNs, designed to adapt flexibly to diverse types of graph-structured data, including heterogeneous tabular data.
- We present a generalization bound analysis to elucidate the contributions of stacking and aggregation from a learning theory perspective.

\*Heterogeneous tabular data includes a mix of discrete and continuous variables.
We conduct extensive evaluations of our approach against strong baselines on node prediction tasks. The experimental results demonstrate significant performance improvements in both homogeneous and heterogeneous node classification tasks across various real-world graph-structured datasets, effectively alleviating the over-smoothing phenomenon.

The remainder of the paper is organized as follows. In Section 2, we provide a brief review of related work. Section 3 presents the theoretical analysis and proposed algorithm for GCNs. In Section 4, we detail experiments conducted on three public citation networks and three heterogeneous tabular datasets. Section 5 offers discussions and concluding remarks. The proof of the main result is included in the Appendix.

2. Related Work

Graph Convolutional Networks

GCNs are commonly understood as extensions of traditional convolutional neural networks applied to graph domains. Generally, there are two types of GCNs [3]: spatial GCNs and spectral GCNs. Spatial GCNs construct new feature vectors for each node by leveraging neighborhood information, where convolution acts as a "patch operator." Spectral GCNs, on the other hand, define convolution by decomposing a graph signal in the spectral domain and applying a spectral filter (such as Fourier or wavelet filters, [4,45,23]) to the spectral components [32,36]. However, spectral GCNs require computation of the Laplacian eigenvectors, which becomes impractical for large-scale graphs due to its high computational cost.

To address this, Hammond et al. [16] used Chebyshev polynomials up to the $K$-th order to approximate the spectral filter. Defferrard and Vandergheynst [8] proposed the $K$-localized ChebyNet, and Kipf and Welling [19] introduced a simpler model by setting $K = 1$, which proved effective for semi-supervised classification tasks. Wu et al. [44] further simplified GCNs by removing nonlinearities and collapsing the weight matrices between layers. In contrast, other works like [22,24] explored the development of deeper GCNs, while multi-scale deep GCNs were examined in [25]. Despite these advancements, over-smoothing remains a significant challenge for GCNs.

Various strategies have been proposed to address over-smoothing. Klicpera et al. [20] and Chien et al. [7] used PageRank and generalized PageRank, respectively, to update graph information. DropEdge [31] randomly removes edges in the graph to mitigate over-smoothing. Similarly, GRAND [10] introduced a random propagation strategy to augment graph data and applied consistency regularization. Chen et al. [6] developed GCNII, a GCN variant that uses initial residuals and identity mapping to address over-smoothing, while DGMLP [48] incorporated adaptive modes and residual connections.

This paper also adopts a decoupled approach, performing feature extraction followed by message propagation to classify nodes. However, our approach achieves
superior performance and accuracy while providing a more flexible and general framework.

**Ensemble learning based graph neural networks**

Sun et al. [39] designed an RNN-like graph structure to extract information from high-order neighbors of each node, while Ivanov and Prokhorenkova [17] integrated gradient-boosted decision trees (GBDT) into GNNs, developing BGNN to handle heterogeneous tabular data. This raises a natural question: can we design a general GCN architecture that not only accommodates diverse graph data but also addresses the over-smoothing issue? This paper seeks to investigate this challenge and provides a solution to the above question.

3. The proposed approach: SStaGCN

3.1. **Graph convolutional networks**

Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$, let $A \in \mathbb{R}^{N \times N}$ denote the adjacency matrix, where $N$ is the total number of nodes. The corresponding degree matrix is denoted as $D$, with $D_{ii} = \sum_j A_{ij}$. For an undirected graph, it is evident that $A_{ij} = A_{ji}$.

In conventional GCN models for semi-supervised node classification, the node embeddings with two convolutional layers are computed as follows:

$$Z = \hat{A} \text{ReLU}(\hat{A}XW^{(0)})W^{(1)}.$$  \hspace{1cm} (3.1)

Here, $Z \in \mathbb{R}^{N \times K}$ represents the final embedding matrix (output logits) of the nodes prior to the softmax operation, where $K$ is the number of classes. The feature matrix $X \in \mathbb{R}^{N \times d}$ contains node features, with $d$ as the input dimension. We define $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = A + I$ (with $I$ as the identity matrix) and $\tilde{D}$ is the degree matrix of $\tilde{A}$.

Furthermore, $W^{(0)} \in \mathbb{R}^{d \times H}$ is the weight matrix for the input-to-hidden layer with $H$ hidden features, and $W^{(1)} \in \mathbb{R}^{H \times K}$ is the weight matrix for the hidden-to-output layer. Notably, the standard GCN model in Eq. (3.1) applies $\hat{A}$ to $X$ repeatedly, which leads to all node features becoming indistinguishable due to excessive smoothing.

3.2. **Stacking**

Stacking is a well-known and widely used ensemble machine learning algorithm that integrates models in a hierarchical framework [43]. It employs a meta-learning algorithm to optimally combine predictions from two or more base machine learning models. A traditional stacking model consists of multiple base models and a meta-model that integrates the base models’ predictions. Each base model is trained on the dataset and produces individual predictions. The meta-model then learns to best combine these predictions, typically using a straightforward approach to provide a cohesive interpretation of the base models’ outputs. As a result, linear
models, such as linear regression for regression tasks and logistic regression for classification tasks, are often chosen as meta-models.

3.3. The proposed approach

As noted above, GCNs may blend node features from different clusters, which can lead to suboptimal predictions. Therefore, it is essential to aggregate node information effectively to improve predictive accuracy. Inspired by the traditional stacking approach and the work in [17,39], we aim to reduce computational cost by using only the base models from the stacking method and then aggregating their outputs to derive the nodes’ attributes. Specifically, our proposed method operates as follows: in the first layer, we obtain $X'$ by passing $X \in \mathbb{R}^{N \times d}$ (the feature matrix) through $k$ base classifiers.

$$X'_i = h_i(X), \quad i = 1, \ldots, k,$$  

(3.2)

Next, we obtain the preliminary classification results $X'_i$ by passing $X$ through each base classifier $h_i(X)$ for $i = 1, \ldots, k$. Then, we apply an aggregation method to produce the final output results, i.e.,

$$\tilde{X} = g(X'_1, \ldots, X'_k),$$  

(3.3)

where $g(\cdot)$ denotes the aggregation method, which is designed to combine attribute values into a single representative value. Common aggregation methods include mean, attention, or voting approaches.

Mean: mean operator takes the element-wise mean of the components $X'_1, \ldots, X'_k$.

Attention [40]: The attention mechanism has been widely applied in various fields of deep learning, including image processing [29], speech recognition [1], and
Algorithm 1 SSStaGCN.

Input:
Feature matrix $X$, normalized adjacency matrix $\hat{A}$, graph $G$, base classifier $h_i(X), i = 1, \cdots, k$, aggregation method $g(\cdot)$;

Output:
Final predictor $f(X)$;

1: Attain $X_i'(i = 1, \cdots, k)$ via $k$ base classifiers;
$X_i' \leftarrow h_i(X), i = 1, \cdots, k$;
2: Aggregate $X_1', \cdots, X_k'$;
$\tilde{X} \leftarrow g(X_1', \cdots, X_k')$;
3: Feed $\tilde{X}$ into vanilla GCN [19];
$f(X) \leftarrow GCN(\tilde{X})$;
4: return $f(X)$;

This concept is inspired by the way humans focus attention selectively. Let the query vector $V$ represent the output of the base classifiers, and let the query vector $Y$ denote the data label. We then compute the attention coefficients between $Y$ and $V$ as follows:

$$a_i = \text{softmax}(\cos(Y, V_i)), \quad i = 1, \cdots, k, \quad (3.4)$$

where $\cos$ denotes cosine similarity. Finally, the input $\tilde{X}$ of the graph convolutional layer is aggregated by considering the following sum with attention scores $a_i, i = 1, \cdots, k$.

$$\tilde{X} = \sum_{i=1}^{k} a_i V_i. \quad (3.5)$$

Voting [33]: The voting method is the most straightforward approach in ensemble learning, aiming to select one or more “winning” predictions. In this work, our goal is to select the most common prediction among the outputs of the base classifiers, using a majority voting approach. In cases where categories appear with equal frequency, a category will be chosen at random.

This brings us to a novel GCN model, SSStaGCN, specifically designed to handle diverse types of graph-structured data by effectively integrating stacking, aggregation, and the vanilla GCN model [19]. In SSStaGCN, the first layer leverages base models from the stacking approach, while the second layer applies an aggregation method—such as mean, attention, or voting—to enhance the feature extraction capabilities of standard GCN models. The aggregated data is then used as input to the convolutional layer, which produces the final predictions. The workflow of the proposed model is presented in Algorithm 1 and Fig. 1. This prompts the question: is there a theoretical guarantee for the proposed approach?
3.4. Generalization bound

In this section, we provide a theoretical generalization analysis of the proposed approach. In the following analysis, we assume that both the adjacency matrix $A$ and the feature matrix $X$ are fixed.

In learning theory, the risk of a function $f$ over the unknown population distribution $P$ is measured by

$$E(f) := E_{X,Y}[L(y, f(\tilde{x}))],$$

where $L$ is the loss function defined as a map: $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$. Given a training data and adjacency matrix $A$, the objective is to estimate parameters $(W^{(0)}, W^{(1)})$ from model (3.1) based on empirical data. Concretely, we attempt to minimize the empirical risk functional over some function class $F$,

$$E_m(f) := \frac{1}{m} \sum_{j=1}^{m} L(y_j, f(\tilde{x}_j)),$$

where $\{(\tilde{x}_j, y_j)_{j=1}^{m}\}$ is the labeled sample achieved from the original training data $\{(x_j, y_j)_{j=1}^{m}\}$ via stacking and aggregation. Typically, the clustering algorithms will only produce discernible node features. Hence, if $\|x_i\| \leq R, i = 1, \ldots, N$, the stacking and aggregation mechanisms will not make $\tilde{x}_i, i = 1, \ldots, N$ violate the constraints, i.e., $\|\tilde{x}_i\| \leq R, i = 1, \ldots, N$. Now we are in position to present the theoretical generalization bound.

**Theorem 3.1.** Suppose $\|x_i\|_2 \leq R, i = 1, \ldots, N, \|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2$. Denote $N(v)$ the number of neighbors of node $v \in \Omega$ (the set of node indices with observed labels), let $q = \max\{N(v)\}, f : \mathcal{X} \to \mathbb{R}$ be any given predictor of a class for GCNs with one-hidden layer. Assume that the loss function $L(y, \cdot)$ is Lipschitz continuous with Lipschitz constant $\alpha_L$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, we have

$$E(f) \leq E_N(f) + \frac{2\alpha_L\sqrt{2qK}B_1B_2R}{\sqrt{m}} \sum_{s=1}^{q} M_s + \sqrt{\frac{2\log(2/\delta)}{N}},$$

where $\|\cdot\|_F$ is the Frobenius norm, $M_s = \max_{i \in [N]} |\hat{A}_{iv}|$ with $v \in N(i)$.

**Remark 3.1.** Theorem 3.1 indicates that the dominant upper bound depends linearly on the maximum number of neighbors $q$, as well as on the weight bounds $B_1$ and $B_2$, which are strongly influenced by the dimension $d_1$. Clearly, setting $d_1 \ll d$ results in a tighter generalization bound. In the binary classification case ($K = 1$), the result presented here is similar to that in [26]. Overall, the conditions stated in Theorem 3.1 are mild and reasonable.
4. Experiments

4.1. Datasets

To evaluate the performance of the proposed SStaGCN model across various types of graph-structured data, we utilize six real-world datasets for a semi-supervised node classification task. This includes three commonly used citation networks—Cora, CiteSeer, and Pubmed [35]—and three heterogeneous tabular datasets: House_class, VK_class, and DBLP [17]. These heterogeneous tabular datasets differ from those in [42], containing graph data with diverse edge and node types.

In the citation networks, nodes represent documents, and undirected edges denote citation relationships between documents. Node features correspond to representative words in the documents, while the label rate indicates the percentage of node labels used for training. The Cora dataset comprises 2708 nodes, 5429 edges, 7 classes, and 1433 node features; CiteSeer has 3327 nodes, 4732 edges, 6 classes, and 3703 node features; and Pubmed includes 19717 nodes, 4438 edges, and 3 classes. We use 140, 120, and 60 nodes for training in Cora, CiteSeer, and Pubmed, respectively, and allocate 1000 nodes for testing and 500 nodes for cross-validation. This data split matches that used in GCN, Graph Attention Network (GAT, [41]), and GWNN [45].

According to [17], House_class and VK_class datasets are derived from House and VK datasets, where target labels are converted into discrete classes due to limited availability of publicly accessible heterogeneous graph-structured data. In these heterogeneous tabular datasets, features are independently defined and vary in type, scale, and meaning. The VK dataset represents a popular social network, with node features that are both numerical (e.g., last active time) and categorical (e.g., country and university affiliation). Similar to [17], we categorize age into bins: $< 20$, $20–25$, $25–30$, $\cdots$, $45–50$, $> 50$. For the House dataset, we categorize target values within the range $[1.0, 1.5, 2.0, 2.5]$, resulting in five classes for House_class and seven for VK_class. In the DBLP dataset, we construct a single graph by focusing on the APA (author-paper-author) meta-path. Each dataset is divided into training, validation, and testing splits in a 0.6/0.2/0.2 ratio across five random seeds.

Details about the citation networks and heterogeneous tabular datasets are presented in Table 1 and Table 2, respectively.

Table 1. Summary of the citation networks.

| Dataset | Cora | CiteSeer | Pubmed |
|---------|------|----------|--------|
| Nodes   | 2708 | 3327     | 19717  |
| Edges   | 5429 | 4732     | 44338  |
| Features| 1433 | 3703     | 500    |
| Classes | 7    | 6        | 3      |
| Label Rate | 5.2% | 3.6%     | 0.3%   |
Table 2. Summary of the heterogeneous tabular datasets.

| Dataset | House class | VK class | DBLP |
|---------|-------------|----------|------|
| Nodes   | 20640       | 54028    | 14475|
| Edges   | 182146      | 213644   | 40269|
| Features| 6           | 14       | 5002 |
| Classes | 5           | 7        | 4    |
| Min Target Nodes | 0.14 | 13.48 | 745  |
| Max Target Nodes | 5.00 | 118.39 | 1197 |

4.2. Baselines

We compare SStaGCN with four classical graph convolutional networks—ChebyNet, GCN [19], GAT, and APPNP [21]—as well as two ensemble-based GCN models: AdaGCN and BGNN.

4.3. Setting

As shown in Fig. 1, the proposed SStaGCN model consists of four layers, with the first and second layers referred to as the stacking and aggregation layers, respectively. The stacking layer utilizes base models from the stacking method, incorporating a combination of seven classical classifiers: KNN, Random Forest, Naive Bayes, Decision Tree, SVC [30], GBDT [12], and Adaboost [11]. These classifiers are widely used in classical machine learning for their strengths in handling different task types.

In the aggregation layer, we employ three aggregation methods: mean, attention, and voting. For the mean approach, we calculate the average output from the stacking layer, rounding it as needed. In the attention mechanism, we treat the label data as vector $Y$, use the stacking layer’s predicted output as the query vector $V$, and compute attention coefficients accordingly. For the voting method, we apply hard voting from ensemble learning [34].

The output of the aggregation layer serves as the input to the first layer of a two-layer GCN. In our configuration, the GCN has 16 hidden units and uses the Adam optimizer [18] by default, with cross-entropy as the loss function. The learning rate is set to $\gamma = 0.01$, the number of iterations $itr = 500$, weight decay at $5e^{-4}$, and a dropout rate of 0.

4.4. Results

The results of the comparative evaluation for node classification are summarized in Tables 3-11. In these tables, SStaGCN (Mean) refers to the use of the mean
Table 3. Average accuracy on 3 citation networks under 30 runs by computing the 95% confidence interval via bootstrap.

| Method      | Cora       | CiteSeer   | Pubmed     |
|-------------|------------|------------|------------|
| ChebyNet    | 81.20±0.00 | 69.80±0.00 | 74.40±0.00 |
| GCN         | 81.50±0.00 | 70.30±0.00 | 79.00±0.00 |
| GAT         | 83.00±0.70 | 72.50±0.70 | 79.00±0.30 |
| APPNP       | 85.09±0.25 | 75.73±0.30 | 79.73±0.31 |
| AdaGCN      | 85.97±0.20 | 76.68±0.20 | 79.95±0.21 |
| BGNN        | 41.97±0.19 | 30.74±0.10 | 10.32±0.10 |
| Sim_Stacking| 43.19±2.05 | 62.70±0.53 | 87.70±0.23 |
| SStaGCN (Mean) | 90.35±0.20 | 86.40±0.12 | 82.30±0.19 |
| SStaGCN (Attention) | 91.60±0.18 | 87.20±0.12 | 82.40±0.23 |
| SStaGCN (Voting) | 93.10±0.16 | 88.70±0.14 | 92.07±0.20 |

Table 4. Average accuracy on heterogeneous tabular datasets under 30 runs by calculating the 95% confidence interval via bootstrap.

| Method      | House_class | VK_class | DBLP       |
|-------------|-------------|----------|------------|
| ChebyNet    | 54.74±0.10  | 57.19±0.36 | 32.14±0.00 |
| GCN         | 55.07±0.13  | 56.40±0.09 | 39.49±1.37 |
| GAT         | 56.50±0.22  | 56.42±0.19 | 76.83±0.78 |
| APPNP       | 57.03±0.27  | 56.72±0.11 | 79.47±1.46 |
| AdaGCN      | 26.20±0.00  | 46.00±0.00 | 10.06±0.00 |
| BGNN        | 66.70±0.27  | 66.32±0.20 | 86.94±0.74 |
| Sim_Stacking| 53.89±0.29  | 56.64±0.10 | 71.58±0.64 |
| SStaGCN (Mean) | 72.35±0.05 | 66.62±0.17 | 82.31±0.20 |
| SStaGCN (Attention) | 72.40±0.12 | 77.64±0.08 | 82.51±0.22 |
| SStaGCN (Voting) | 76.13±0.12 | 87.92±0.07 | 92.60±0.10 |

Table 5. Average F1-score (macro) on 3 citation networks under 30 runs by computing the 95% confidence interval via bootstrap.

| Method      | Cora       | CiteSeer   | Pubmed     |
|-------------|------------|------------|------------|
| ChebyNet    | 77.99±0.54 | 63.76±0.34 | 74.74±0.42 |
| GCN         | 82.89±0.30 | 70.65±0.37 | 78.83±0.32 |
| GAT         | 83.59±0.25 | 70.62±0.29 | 77.77±0.40 |
| APPNP       | 84.29±0.22 | 71.05±0.38 | 79.66±0.31 |
| AdaGCN      | 79.55±0.19 | 63.62±0.19 | 78.55±0.21 |
| BGNN        | 40.81±0.25 | 32.73±0.13 | 8.46±0.08  |
| Sim_Stacking| 44.02±1.61 | 60.86±0.56 | 87.31±0.13 |
| SStaGCN (Mean) | 90.66±0.18 | 86.42±0.12 | 82.30±0.19 |
| SStaGCN (Attention) | 91.69±0.14 | 87.24±0.14 | 82.45±0.23 |
| SStaGCN (Voting) | 92.76±0.16 | 88.73±0.14 | 92.07±0.20 |
Table 6. Average F1-score (macro) on heterogeneous tabular datasets under 30 runs by calculating the 95% confidence interval via bootstrap.

| Method         | House_class | VK_class | DBLP       |
|----------------|-------------|----------|------------|
| ChebyNet       | 31.34±0.12  | 57.44±0.27 | 26.84±0.62 |
| GCN            | 54.95±0.14  | 56.52±0.09 | 38.5±0.97  |
| GAT            | 56.54±0.68  | 56.41±0.07 | 77.1±1.86  |
| APPNP          | 57.88±0.32  | 56.61±0.07 | 79.34±0.23 |
| AdaGCN         | 25.01±0.00  | 37.03±0.00 | 9.60±0.00  |
| BGNN           | 66.48±0.22  | 66.18±0.11 | 87.2±0.60  |
| Sim_Stacking   | 53.32±0.15  | 56.11±0.08 | 71.49±0.31 |
| SStaGCN (Mean) | 72.23±0.04  | 66.74±0.21 | 82.13±0.38 |
| SStaGCN (Attention) | 72.36±0.09 | 77.62±0.10 | 82.68±0.12 |
| SStaGCN (Voting) | 75.45±0.82 | 87.84±0.04 | 92.64±0.06 |

Table 7. p-values of the paired t-test of SStaGCN (Voting) with competitors on 6 different datasets (Cora, CiteSeer, Pubmed, House_class, VK_class, and DBLP).

| Models         | Cora          | CiteSeer      | Pubmed        | House_class | VK_class | DBLP        |
|----------------|---------------|---------------|---------------|-------------|----------|-------------|
| ChebyNet       | 2.59e-06      | 2.77e-08      | 1.17e-06      | 3.51e-10    | 1.11e-11 | 6.97e-09    |
| GCN            | 4.19e-16      | 5.15e-17      | 1.84e-15      | 2.36e-08    | 2.25e-11 | 2.48e-07    |
| GAT            | 2.10e-19      | 1.54e-20      | 8.84e-19      | 3.35e-08    | 1.38e-09 | 4.95e-06    |
| APPNP          | 6.86e-42      | 7.12e-42      | 6.25e-41      | 7.05e-15    | 1.39e-21 | 2.26e-09    |
| AdaGCN         | 1.82e-19      | 1.23e-20      | 8.42e-19      | 3.05e-38    | 7.94e-43 | 1.69e-35    |
| BGNN           | 1.02e-24      | 2.33e-21      | 4.74e-22      | 6.54e-07    | 1.07e-08 | 0.11e-03    |
| Sim_Stacking   | 5.61e-12      | 3.79e-15      | 2.52e-09      | 4.56e-08    | 6.67e-11 | 1.07e-06    |

aggregation mechanism in the second layer of SStaGCN, while SStaGCN (Attention) and SStaGCN (Voting) correspond to the attention and voting mechanisms, respectively. We report the accuracy, macro F1-score, and training time on the test set for the proposed SStaGCN model and other methods. The experimental results demonstrate a significant improvement of the SStaGCN model over the baselines. Specifically, for the three public citation networks, SStaGCN (Voting) achieves an improvement in accuracy (and F1-score) of approximately 8% (8%), 12% (17%), and 13% (12%) for the Cora, CiteSeer, and Pubmed datasets, respectively. For the heterogeneous tabular datasets, SStaGCN (Voting) shows an improvement in accuracy (and F1-score) of about 9% (9%), 21% (21%), and 6% (5%) for the House_class, VK_class, and DBLP datasets, respectively.

AdaGCN performs poorly on the heterogeneous tabular datasets, possibly because AdaGCN is designed for deeper GCN architectures, which may mix node features from different clusters as the GCN layers deepen. SStaGCN, by contrast, enhances the performance of GCN, providing better results across distinct types of graph-structured data. Additionally, the paired t-test results in Table 7 indicate that the proposed SStaGCN model significantly outperforms the simplified stacking...
and other GCN models.

This impressive improvement can be explained as follows:

- The stacking and aggregation steps in SStaGCN provide a dimensionality reduction effect, making the graph data more discernible. For example, in the Cora dataset, the feature size reduces from $2708 \times 1433$ to $2708 \times 7$ after stacking and aggregation. This results in a relatively smaller $d_1$, as discussed in Remark 1, significantly enhancing both the predictive power and computational efficiency of the subsequent graph convolution model.

- In the aggregation step of the SStaGCN model, the mean and attention mechanisms somewhat disrupt the pre-classification results, making them less suitable for feature extraction, whereas the voting mechanism preserves these results. Consequently, experimental results demonstrate that SStaGCN (Voting) is more effective across the six datasets.

- Simplified stacking can effectively extract useful attributes but overlooks graph structure information, while GCN models are limited in feature extraction. Therefore, the SStaGCN model combines the strengths of simplified stacking and GCN, achieving both higher classification accuracy and reduced computational cost.

Tables 8 and 9 present a comparison of training times between SStaGCN and other methods. BGNN achieves the fastest runtime on the three citation networks, followed closely by our SStaGCN method. However, SStaGCN runs faster than other methods on the heterogeneous tabular datasets, with the exception of the DBLP dataset. We attribute this to the extra computation time required for the stacking and aggregation steps, which ultimately enhance efficiency when feeding the feature outputs into the GCN model.

### Table 8. Average training time(s) on 3 citation networks by computing the 95% confidence interval via bootstrap.

| Method          | Cora        | CiteSeer    | Pubmed      |
|-----------------|-------------|-------------|-------------|
| ChebyNet        | 22.74±1.24  | 30.87±1.21  | 124.99±1.77 |
| GCN             | 13.41±0.16  | 99.21±0.98  | 55.61±0.73  |
| GAT             | 20.98±0.46  | 30.74±1.40  | 126.33±1.80 |
| APPNP           | 203.75±0.15 | 55.40±0.40  | 457.62±12.77|
| AdaGCN          | 772.26±83.56| 2129.02±148.97| 2098.10±275.88|
| BGNN            | 1.33±0.00   | 2.40±0.00   | 2.54±0.00   |
| Sim_Stacking    | 11.9±0.08   | 27.9±0.24   | 79.1±1.40   |
| SStaGCN (Mean)  | 10.9±0.13   | 17.2±0.13   | 89.6±2.20   |
| SStaGCN (Attention) | 11.2±0.24   | 17.6±0.41   | 87.6±0.96   |
| SStaGCN (Voting) | 16.2±0.19   | 29.6±1.40   | 13.1±2.61   |

To demonstrate the effect of stacking and aggregation steps in the proposed model, we provide a t-SNE visualization in Figs. 2 and 3. These figures show
that the combination of stacking and aggregation effectively extracts features, enhancing the discriminative power of the graph data.

Table 9. Average training time(s) on heterogeneous tabular datasets by calculating the 95% confidence interval via bootstrap.

| Method           | House_class | VK_class   | DBLP        |
|------------------|-------------|------------|-------------|
| ChebyNet         | 833.82±0.00 | 1394.68±0.00 | 8890.74±0.00 |
| GCN              | 46.06±0.80  | 120.1±3.35  | 268.5±5.35  |
| GAT              | 197.6±3.08  | 410.9±9.95  | 205.3±2.00  |
| APPNP            | 129.7±3.26  | 383.8±12.02 | 176.8±5.58  |
| AdaGCN           | 607.31±0.00 | 511.41±0.00 | 590.90±0.00 |
| BGNN             | 26.37±1.65  | 93.47±6.23  | 50.25±3.04  |
| Sim_Stacking     | 16.41±0.36  | 43.58±2.78  | 380.8±12.82 |
| SStaGCN (Mean)   | 52.94±0.51  | 132.9±1.61  | 188.7±0.54  |
| SStaGCN (Attention) | 57.38±1.75 | 133.2±1.11  | 246.2±0.37  |
| SStaGCN (Voting) | 59.69±0.82  | 154.1±1.02  | 310.7±0.49  |

Table 10 suggests that using all seven classifiers is unnecessary. For example, on the Cora dataset, the combination of KNN, Random Forest, and Naive Bayes achieves the highest accuracy with minimal computational cost (only 3 seconds). This indicates that each of the seven classifiers has unique strengths suited to different tasks. However, selecting the optimal combination of base models remains an open area for theoretical analysis. To further illustrate the impact of simplified stacking on the over-smoothing problem, we conduct an additional experiment on this topic. In Fig. 4, we observe that a conventional GCN tends to blend node features from different clusters as the number of convolutional layers increases.
Fig. 3. Visualization of classification features by the GCN (left) and the features after conducting stacking and aggregation steps in the SStaGCN model (right) on DBLP dataset, node colors denote classes.

Table 10. Accuracy and training time(s) on Cora dataset by combinations of different classifiers based on SStaGCN model.

| KNN   | Random Forest | Naive Bayes | Decision Tree | GBDT | Adaboost | SVC | Accuracy | Training Time |
|-------|---------------|-------------|---------------|------|----------|-----|----------|--------------|
| √     | √             |             |               |      |          |     | 91.2     | 13.90        |
| √     | √             | √           |               |      |          |     | 84.2     | 567.9        |
| √     | √             | √           | √             |      |          |     | 92.9     | 144.7        |
| √     | √             | √           | √             |      |          |     | 84.1     | 723.4        |
| √     | √             | √           | √             |      |          |     | 92.8     | 15.90        |
| √     | √             | √           | √             |      |          |     | 93.4     | 18.80        |
| √     | √             | √           | √             |      |          |     | 92.9     | 568.3        |
| √     | √             | √           | √             |      |          |     | 92.9     | 570.7        |
| √     | √             | √           | √             |      |          |     | 92.8     | 708.5        |

However, as shown in Fig. 3 and Table 10, the proposed SStaGCN effectively mitigates the over-smoothing phenomenon and enhances accuracy.

Table 11. Accuracy comparison between GCN and SStaGCN models on Cora dataset using distinct number of layers.

| Method | 2-layer | 3-layer | 4-layer | 5-layer | 6-layer | 7-layer |
|--------|---------|---------|---------|---------|---------|---------|
| GCN    | 80.5    | 80.4    | 75.8    | 71.9    | 72.6    | 60.8    |
| SStaGCN| 93.3    | 88.8    | 87.5    | 86.4    | 84.8    | 84.3    |

Overall, these experiments demonstrate the superiority of SStaGCN model over competitors.

The number of layers excludes those included in the stacking and aggregation parts of SStaGCN, which consist of only two layers.
**4.5. Visualization**

To further illustrate the performance of SSStaGCN, we plot the final classification features of GCN, AdaGCN, BGNN, and our SSStaGCN. Fig. 6 (for CiteSeer) and Fig. 7 (for DBLP) display the final classification features of each method on these datasets. As shown in these figures, the proposed SSStaGCN misclassifies relatively...
fewer points, while GCN, AdaGCN, and BGNN result in more misclassified classes.

5. Conclusion

Traditional GCNs often face the over-smoothing problem. In this work, we propose a novel GCN architecture, SStaGCN, which leverages stacking and aggregation techniques to capture pre-classified data features, followed by GCN for predictions on distinct graph-structured data. SStaGCN effectively explores and utilizes features for heterogeneous graph data in a stacked manner. By incorporating classical machine learning methods, we design a GCN model that provides a versatile framework for handling diverse types of graph-structured data, offering new insights into understanding GCNs. Extensive experiments demonstrate that the proposed model outperforms several state-of-the-art competitors in terms of accuracy, F1-score, and training time. The framework presented here could also be extended to regression tasks. Additionally, we believe this method can address various types of heterogeneous graph data beyond tabular data. A promising future research direction is to investigate deeper GCNs within our framework, as suggested by [39]. The source code of SStaGCN will be issued soon.

*https://github.com/dragon0916/SStaGCN.
Appendix

In this part, we provide a detailed proof of Theorem 3.1. Before proceeding with the proof, we introduce several supporting lemmas. First, we present the contraction inequality for Rademacher complexity in vector form.

**Lemma 5.1.** [28] Let $X$ be any set, $(x_1, \ldots, x_N) \in X$ and $F$ be a class of functions $f : X \to \mathbb{R}^K$ and let $\tau_i : \mathbb{R}^K \to \mathbb{R}$ have Lipschitz constant $M$. Then

$$E \sup_{f \in F} \sum_{i=1}^{N} \sigma_i \tau_i f(x_i) \leq \sqrt{2M} E \sup_{f \in F} \sum_{i=1}^{N} \sum_{k=1}^{K} \sigma_{ik} f_k(x_i),$$

where $\sigma_{ik}$ is an independent doubly indexed Rademacher sequence and $f_k(x_i)$ is the $k$-th component of $f(x_i)$.

**Lemma 5.2.** [2] Consider a loss function $L : X \times Y \to \mathbb{R}^+$. Denote $H = \{L(y, f(\cdot)), f \in F\}$, and let $(x_i, y_i)_{i=1}^{N}$ be independently selected according to the probability measure $P$. Then for any $0 < \delta < 1$, with probability at least $1 - \delta$,

$$\mathcal{E}(f) \leq \mathcal{E}_N(f) + 2R(H) + \frac{r_2 \log(2/\delta)}{N}, \quad \forall f \in H.$$

We first give a lemma which plays essential role in the proof of Theorem 3.1.

**Lemma 5.3.** Let $q = \max \{ N(v) \}$ for each node $v \in \Omega$, then

$$\max_v \| \sum_{j \in N(v)} \hat{A}_{vj} x_j \|_2^2 \leq R^2 q.$$

**Proof.** Denote $\hat{A}_v \in \mathbb{R}^{n \times q_v}$ as the sub-matrix of $\hat{A}$ whose row and column indices belong to the set $j \in N(v)$. Hence the size of $\hat{A}_v$ depends on the node $v$. Obviously, $q = \max q_v$ for $v \in \Omega$. Let $\tilde{X}_v = (x_1^T, \ldots, x_q^T)^T \in \mathbb{R}^{q \times d}$ be the feature matrix of the nodes in $\tilde{G}_v$ (subgraph of $G$). Hence,

$$\max_v \left\| \sum_{j \in N(v)} \hat{A}_{vj} x_j \right\|_2^2 = \max_v \| \hat{A}_v \tilde{X}_v \|_2^2 \leq \max_v \| \hat{A}_v \|_2^2 \| \tilde{X}_v \|_2^2,$$

where $\hat{A}_v$ is the $v$-th row of the matrix $\hat{A}$ with column index $j$ belong to the set $N(v)$. Notice that

$$\| \tilde{X}_v \|_2 \leq \sup_{\|t\|=1} \| \tilde{X}_v t \|_2 \leq \sqrt{\sum_{s=1}^{q_v} \| x_s \|_2^2} \leq R \sqrt{q_v} \leq R \sqrt{q},$$

and $\| \hat{A} \|_2 \leq 1$. Therefore,

$$\max_v \left\| \sum_{j \in N(v)} \hat{A}_{vj} x_j \right\|_2^2 \leq \| \hat{A} \|_2^2 \| \tilde{X}_v \|_2^2 \leq R^2 q.$$

\[\qed\]
Now we are in position to give the proof of Theorem 3.1

**Proof.** [Proof of Theorem 3.1] To allow a slight abuse of notations, we will use $X_j$ to denote $\tilde{X}_j$ due to the explanation on page 3. Denote $h(W^{(0)}) = \text{ReLU}(\hat{A}XW^{(0)})$, $f(W^{(1)}) = \text{softmax}(h(W^{(0)})W^{(1)}) = (f_1(W^{(1)}), \cdots, f_m(W^{(1)}))^T$ with $\hat{h}(W^{(0)}) = \hat{A}h(W^{(0)})$. Applying Proposition 4 in [13] to the case $\lambda = 1$, we can attain the Lipschitz constant for standard softmax function is $M = 1$. Let $A_i$ stands for the $i$-th row of the matrix $\hat{A}$, for function set

$$\mathcal{F}_{B_1, B_2} = \{f_i = \text{softmax}(A_i \text{ReLU}(\hat{A}XW^{(0)})W^{(1)}), i = 1, \cdots, m, \|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2\},$$

the empirical Rademacher complexity is defined as

$$\hat{R}(\mathcal{F}_{B_1, B_2}) = \mathbb{E}_\sigma \left[ \frac{1}{m} \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \sum_{i=1}^{m} \sigma_i f_i(W^{(1)}) \right],$$

where $\{\sigma_i\}_{i=1}^{m}$ is an i.i.d. family of Rademacher variables independent of $x_i$. By the contraction property of Rademacher complexity,

$$\hat{R}(\mathcal{H}) \leq \alpha \hat{R}(\mathcal{F}_{B_1, B_2}),$$

and notice Lemma 5.2, we only need to bound $\hat{R}(\mathcal{F}_{B_1, B_2})$. Therefore, we have the following estimate by utilizing Lemma 5.1

$$\hat{R}(\mathcal{F}_{B_1, B_2}) = \mathbb{E}_\sigma \left[ \frac{1}{m} \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \sum_{i=1}^{m} \sigma_i f_i(W^{(1)}) \right] \leq \sqrt{\frac{2}{m}} \mathbb{E}_\sigma \left[ \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \sum_{i=1}^{m} \sum_{k=1}^{K} \sigma_{ik} \hat{h}_i(W^{(0)})w^{(1)}_k \right],$$

where $W^{(1)} = (w_1^{(1)}, \cdots, w_K^{(1)})$, notice the property of inner product, the above estimate can be further bounded as

$$\hat{R}(\mathcal{F}_{B_1, B_2}) \leq \sqrt{\frac{2}{m}} \mathbb{E}_\sigma \left[ \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \sum_{k=1}^{K} \max_{k \in [K]} \|w_k^{(1)}\|_2 \| \sum_{i=1}^{m} \sigma_{ik} \hat{h}_i(W^{(0)}) \|_2 \right],$$

$$\leq \sqrt{\frac{2}{m}} \mathbb{E}_\sigma \left[ \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \|W^{(1)}\|_F \sum_{k=1}^{K} \| \sum_{i=1}^{m} \sigma_{ik} \hat{h}_i(W^{(0)}) \|_2 \right],$$

$$\leq \sqrt{\frac{2}{m}} B_2 \mathbb{E}_\sigma \left[ \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \sum_{k=1}^{K} \| \sum_{i=1}^{m} \sigma_{ik} \hat{h}_i(W^{(0)}) \|_2 \right],$$

$$\leq \sqrt{\frac{2}{m}} B_2 \mathbb{E}_\sigma \left[ \sum_{k=1}^{K} \sup_{\|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2} \| \sum_{i=1}^{m} \sigma_{ik} \hat{h}_i(W^{(0)}) \|_2 \right].$$
the last inequality follows by the property that \( \sup(\sum_s a_s) \leq \sum_s \sup(a_s) \). Now the key point is how to estimate the term \( \sup_{\|W(0)\|_F \leq B_1} \left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_t(W(0)) \right\|_2 \). We will employ the idea introduced in [26] (in the proof of Theorem 1) to remove the “sup” term. Let \( h_v(W(0)) = \left( h^1_v(w^{(0)}_1), h^2_v(w^{(0)}_2), \cdots, h^H_v(w^{(0)}_H) \right) \) with \( W(0) = (w^{(0)}_1, \cdots, w^{(0)}_H) \) and notice that \( \hat{h}(W(0)) = \hat{A} h(W(0)) \), then we have

\[
\left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_t(W(0)) \right\|_2^2 = \sum_{t=1}^H \left( \sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h^t_v(w^{(0)}_i) \right)^2 \\
= \sum_{i=1}^H \|w^{(0)}_i\|_2^2 \left( \sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h^0_v(w^{(0)} / \|w^{(0)}_i\|_2) \right)^2.
\]

By the definition of Frobenius norm \( \|W\|_F^2 = \sum_{t=1}^H \|w_t\|_2^2 \), the supremum of the above quantity under the constraint \( \|W\|_F \leq R \) must be obtained when \( \|w_{t_0}\|_2 = B_1 \) for some \( t_0 \in [H] \), and \( \|w_t\|_2 = 0 \) for all \( t \neq t_0 \). Hence

\[
\sup_{\|W(0)\|_F \leq B_1} \left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_t(W(0)) \right\|_2 = \sup_{\|w\|_2 = B_1} \left( \sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h_v(w) \right).
\]

Let \( n_s(j) \) be the \( s \)-th neighbor number of node \( j \) \( (s \in [q], j \in [m]) \). Recall \( q := \max_{j \in [m]} |N(j)|, j \in [m], M_s = \max_{i \in [m]} |\hat{A}_{iv}| \) with \( v \in N(i) \), therefore

\[
\mathbb{E}_\sigma \left[ \sum_{k=1}^K \sup_{\|W(0)\|_F \leq B_1} \left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_t(W(0)) \right\|_2 \right] \\
= \mathbb{E}_\sigma \left[ \sum_{k=1}^K \sup_{\|w\|_2 = B_1} \left( \sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h_v(w) \right) \right] \\
\leq \mathbb{E}_\sigma \left[ \sum_{k=1}^K \sup_{\|w\|_2 = B_1} \left( \sum_{s=1}^q M_s \sum_{i=1}^m |\sigma_{ik} h_{n_s(i)}(w)| \right) \right].
\]

Applying the conclusion \( \sup(\sum_s a_s) \leq \sum_s \sup(a_s) \) and contraction property of
Rademacher complexity again, we have
\[
\mathbb{E}_\sigma \left[ \sum_{k=1}^K \sup_{\|w\|_2 = B_1} \left( \sum_{s=1}^q M_s \sum_{i=1}^m \sigma_{ik} h_{n_s(i)}(w) \right) \right] \\
\leq \mathbb{E}_\sigma \left[ \sum_{k=1}^K \sum_{s=1}^q M_s \sup_{\|w\|_2 = B_1} \sum_{i=1}^m \sigma_{ik} h_{n_s(i)}(w) \right] \\
\leq \sum_{s=1}^q M_s \mathbb{E}_\sigma \left[ \sum_{k=1}^K \sup_{\|w\|_2 = B_1} \sum_{i=1}^m \sigma_{ik} \left( \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right) \right] \\
= \sum_{s=1}^q M_s \mathbb{E}_\sigma \left[ \sum_{k=1}^K \sup_{\|w\|_2 = B_1} \left( \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right) \right] \\
\leq B_1 \sum_{s=1}^q M_s \mathbb{E}_\sigma \left[ \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2 \right].
\]

Therefore, we only need to estimate the term
\[
\mathbb{E}_\sigma \left[ \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2 \right].
\]

Applying Cauchy-Schwartz inequality yields that
\[
\mathbb{E}_\sigma \left[ \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2 \right] \leq \sqrt{\mathbb{E}_\sigma \left( \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2 \right)^2} \\
\leq \mathbb{E}_\sigma K \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2 \\
\leq K \sum_{i=1}^m \left\| \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2,
\]

where the last inequality is due to the i.i.d condition of Rademacher sequences. Plugging the conclusion of Lemma 5.3 into the above term leads to
\[
\mathbb{E}_\sigma \left[ \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)}x_j \right\|_2^2 \right] \leq KR\sqrt{qm},
\]

and
\[
\hat{R}(\mathcal{F}_{B_1, B_2}) \leq \frac{\sqrt{2qKB_1B_2R} \sum_{s=1}^q M_s}{\sqrt{m}}.
\]

This completes the proof by combining with Lemma 5.2.
Acknowledgement

The work described in this paper was supported partially by the National Natural Science Foundation of China (12271111, 11871277), Special Support Plan for High-Level Talents of Guangdong Province (2019TQ05X571), Guangdong Basic and Applied Basic Research Foundation (2022A1515011726). The authors would like to thank Prof. Hong Chen from Huazhong Agricultural University for useful discussions about the theoretical analysis, which have helped to improve the presentation of the paper.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, in *Int. Conf. on Learning Representations (ICLR)*, San Diego, United States (May 2015).

[2] P. L. Bartlett and S. Mendelson, Rademacher and Gaussian complexities: risk bounds and structural results, in *Int. Conf. on Computational Learning Theory & European Conference on Computational Learning Theory*, Amsterdam, Netherlands, (July 2001), pp. 224-240.

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, Geometric deep learning: going beyond euclidean data, *IEEE Sigal Proc. Mag.* 34(4) (2017) 18-42.

[4] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, Spectral networks and locally connected networks on graphs, in *Int. Conf. on Learning Representations (ICLR)*, Banff, Canada (April 2014).

[5] S. Casas, C. Gulino, R. Liao, and R. Urtasun, Spagnn: spatially-aware graph neural networks for relational behavior forecasting from sensor data, in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Paris, France (May 2020), pp. 9491-9497.

[6] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, Simple and deep graph convolutional networks, in *Int. Conf. on Machine Learning (ICML)*, Virtual, (July 2020), pp. 1725-1735.

[7] E. Chien, J. Peng, P. Li, and O. Milenkovic, Adaptive universal generalized pagerank graph neural network, in *Int. Conf. on Learning Representations (ICLR)*, Virtual (May 2021).

[8] M. Defferrard, X. Bresson, and P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in *Advances in Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain (December 2016).

[9] S. Dżeroski and B. Ženko, Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.* 54 (2004) 255-273.

[10] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, Graph random neural networks for semi-supervised learning on graphs, in *Advances in Neural Information Processing Systems (NeurIPS)*, Virtual (December 2020), pp. 22092-22103.

[11] Y. Freund and R. E. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence* 14 (1999) 771-780.

[12] J. Friedman, Greedy function approximation: a gradient boosting machine, *Ann. Stat.* 29 (2001) 1189-1232.

[13] B. Gao and L. Pavel, On the properties of the softmax function with application in game theory and reinforcement learning, Technical Report, [arXiv:1704.00805](https://arxiv.org/abs/1704.00805) (2017).
[14] M. Gori, G. Monfardini, and F. Scarselli, A new model for learning in graph domains, in *Int. Joint Conf. on Neural Networks (IJCNN)*, Montreal, Canada (July 2005).

[15] W. L. Hamilton, Z. Ying, and J. Leskovec, Inductive representation learning on large graphs, in *Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, United States (December 2017).

[16] D. K. Hammond, P. Vandergheynst, and R. Gribonval, Wavelets on graphs via spectral graph theory, *Appl. Comput. Harmon. A.*, 30 (2011) 129-150.

[17] S. Ivanov and L. Prokhoronenkova, Boost then convolve: gradient boosting meets graph neural networks, in *Int. Conf. on Learning Representations (ICLR)*, Virtual (May 2021).

[18] D. Kingma and J. Ba, Adam: a method for stochastic optimization, in *Int. Conf. on Learning Representations (ICLR)*, San Diego, United States (May 2015).

[19] T. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *Int. Conf. on Learning Representations (ICLR)*, Toulon, France (April 2017).

[20] J. Klicpera, A. Bojchevski, and S. Günnemann, Predict then propagate: graph neural networks meet personalized pagerank, in *Int. Conf. on Learning Representations (ICLR)*, New Orleans, United States (April 2019).

[21] J. Klicpera, A. Bojchevski, and S. Günnemann, Predict then propagate: graph neural networks meet personalized pagerank, in *Int. Conf. on Learning Representations (ICLR)*, New Orleans, United States (April 2019).

[22] G. Li, M. Mueller, A. Thabet, and B. Ghanem, Deepgcn: can gcns go as deep as cnns? in *Int. Conf. on Computer Vision (ICCV)*, Seoul, Korea (October 2019).

[23] M. Li, Z. Ma, Y. G. Wang, and X. Zhuang, Fast haar transforms for graph neural networks, *Neural Networks*, 128 (2020) 188-198.

[24] Q. Li, Z. Han, and X. M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in *AAAI Conf. on Artificial Intelligence*, New Orleans, United States (February 2018).

[25] S. Luan, M. Zhao, X. W. Chang, and D. Precup, Break the ceiling: stronger multiscale deep graph convolutional networks, in *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada (December 2019).

[26] S. Lv, Generalization bounds for graph convolutional neural networks via Rademacher complexity, Technical Report, arXiv:2102.10234 (2021).

[27] L. V. D. Maaten and G. E. Hinton, Visualizing data using t-sne, *J. Mach. Learn. Res.* 9 (2008) 2579-2605.

[28] A. Maurer, A vector-contraction inequality for Rademacher complexities, in *Int. Conf. on Algorithmic Learning Theory*, Bari, Italy (October 2016).

[29] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, Recurrent models of visual attention, in *Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, Canada (December 2014).

[30] K. Pal and B. Patel, Data classification with k-fold cross validation and holdout accuracy estimation methods with 5 different machine learning techniques, in *2020 fourth Int. Conf. on Computing Methodologies and Communication (ICCMC)*, Erode, India (March 2020), pp. 83-87.

[31] Y. Rong, W. Huang, T. Xu, and J. Huang, Dropedge: towards deep graph convolutional networks on node classification, in *Int. Conf. on Learning Representations (ICLR)*, Virtual, (April 2020).

[32] A. Sandryhaila and J. Moura, Discrete signal processing on graphs, *IEEE Trans. Signal Processing* 61(7) (2013) 1644-1656.

[33] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, Boosting the margin: a new
explanation for the effectiveness of voting methods, in *Int. Conf. on Machine Learning (ICML)*, San Francisco, United States (July 1997).

[34] F. Schwenker, Ensemble methods: foundations and algorithms, *IEEE Comput. Intell. M.* 8(2013) 77-79.

[35] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, Collective classification in network data, *AI Mag.*, 29(2008) 93-106.

[36] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains, *IEEE Signal Proc. Mag.* 30(3) (2013) 83-98.

[37] J. M. Stokes, K. Yang, K. Swanson, W. Jin, and J. J. Collins, A deep learning approach to antibiotic discovery, *Cell* 180(4) (2020) 688-702.e13.

[38] J. Sun, W. Guo, D. Zhang, Y. Zhang, F. Regol, Y. Hu, H. Guo, R. Tang, H. Yuan, X. He, and M. Coates, A framework for recommending accurate and diverse items using bayesian graph convolutional neural networks, in *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, New York, United States (July 2020).

[39] K. Sun, Z. Lin, and Z. Zhu, Adagcn: adaboosting graph convolutional networks into deep models, in *Int. Conf. on Learning Representations (ICLR)*, Virtual (May 2021).

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, United States (December 2017), pp. 6000-6010.

[41] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio’, and Y. Bengio, Graph attention networks, in *Int. Conf. on Learning Representations (ICLR)*, Vancouver, Canada (April 2018).

[42] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, Heterogeneous graph attention network, in *The World Wide Web Conference*, San Francisco, United States (May 2019), pp. 2022-2032.

[43] D. Wolpert, Stacked generalization, *Neural Netw.* 5(1992) 241-259.

[44] F. Wu, T. Zhang, A. Souza, C. Fifty, T. Yu, and K. Q. Weinberger, Simplifying graph convolutional networks, in *Int. Conf. on Machine Learning (ICML)*, Long Beach, United States (June 2019).

[45] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, Graph wavelet neural network, in *Int. Conf. on Learning Representations (ICLR)*, New Orleans, United States (May 2019).

[46] L. Yao, C. Mao, and Y. Luo, Graph convolutional networks for text classification, in *AAAI Conf. on Artificial Intelligence*, Honolulu, USA (January 2019).

[47] W. Yin, H. Schütze, B. Xiang, and B. Zhou, Abcnn: attention-based convolutional neural network for modeling sentence pairs, *Transactions of the Association for Computational Linguistics*, 4(2016) 259-272.

[48] W. Zhang, Z. Sheng, Y. Jiang, Y. Xia, J. Gao, Z. Yang, and B. Cui, Evaluating deep graph neural networks, *Technical Report* [arXiv:2108.00955] (2021).

[49] J. Zhu, Max-margin nonparametric latent feature models for link prediction, in *Int. Conf. on Machine Learning (ICML)*, Edinburgh, UK (June 2012).