

SStaGCN: Simplified stacking based graph convolutional networks

Jia Cai, Zhilong Xiong, Shaogao Lv

Abstract—Graph convolutional network (GCN) is a powerful model studied broadly in various graph structural data learning tasks. However, to mitigate the over-smoothing phenomenon, and deal with heterogeneous graph structural data, the design of GCN model remains a crucial issue to be investigated. In this paper, we propose a novel GCN called SStaGCN (Simplified stacking based GCN) by utilizing the ideas of stacking and aggregation, which is an adaptive general framework for tackling heterogeneous graph data. Specifically, we first use the base models of stacking to extract the node features of a graph. Subsequently, aggregation methods such as mean, attention and voting techniques are employed to further enhance the ability of node features extraction. Thereafter, the node features are considered as inputs and fed into vanilla GCN model. Furthermore, theoretical generalization bound analysis of the proposed model is explicitly given. Extensive experiments on 3 public citation networks and another 3 heterogeneous tabular data demonstrate the effectiveness and efficiency of the proposed approach over state-of-the-art GCNs. Notably, the proposed SStaGCN can efficiently mitigate the over-smoothing problem of GCN.

Index Terms—Graph convolutional network, stacking, aggregation, heterogeneous data.

I. INTRODUCTION

RCENTLY, research with graph structural data learning has received wide considerable attention in ample fields of artificial intelligence. Graph neural networks (GNNs), particularly, graph convolutional networks (GCNs) [1]–[3] have shown remarkable success in learning graph structural data, and been applied in recommendation systems [4], computer vision [5], molecular design [6], natural language processing [7], node classification [8] and clustering tasks [9]. Despite their great success, almost all of the GCNs focus on graph data with homogeneous node embedding or bag-of-words representations. Ivanov and Prokhorenkova [10] incorporated gradient boosted decision trees (GBDT) into GNN and proposed a novel BGNN architecture to deal with heterogeneous tabular data for the first time. As is known to all, there are various types of heterogeneous data in real-world applications.

The work described in this paper was supported partially by the National Natural Science Foundation of China (11871167, 11871277), Special Support Plan for High-Level Talents of Guangdong Province (2019TQ05X571), Foundation of Guangdong Educational Committee (2019KZDZX1023, 2019GWZDXM004, 2019GWZJD003), Project of Guangdong Province Innovative Team (2020WCXTD011), Guangdong Natural Science Foundation (2019A1515011797). The corresponding author is Shaogao Lv.

Jia Cai and Zhilong Xiong are with School of Statistics and Mathematics, Guangdong University of Finance & Economics, 21 Chisha Road, Guangzhou 510320, Guangdong, P. R. China. Email: zhilongxiong98@outlook.com (Z. L. Xiong), jiacai1999@gdufe.edu.cn (J. Cai).

Shaogao Lv is with Center of Statistics and Data Science, Nanjing Audit University, Nanjing, 211815, P. R. China. Email: lvsg716@nau.edu.cn.

Can we design a general framework to handle distinct types of heterogeneous data besides heterogeneous tabular data. Moreover, as laid out by [11], graph convolution of GCN model is a special form of Laplacian smoothing, which mixes the node features from different clusters. Therefore, it brings the potential problem of over-smoothing [11]. Sun et al. [12] developed a RNN-like GCN by employing AdaBoost, which can extract knowledge from high-order neighbors of current nodes. However, $A^l = B^l$ does not imply $A = B$, one apparent simple example is considering the situation $l = 2$ (two layer), and

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

Thus, this RNN-like GCN will definitely lose the information of the original graph structure. The over-smoothing, the ability of dealing with heterogeneous data, and the working mechanisms of GCN model remain open.

Recap stacking method [13] or stacked generation is an approach to ensemble multiple different classifications, which contains of base models and meta-model. Stacking is successful in the feature extraction task of tackling different kinds of data, although the data may be disordered or irregular. This is because stacking has the following properties: (1). Combining multiple distinct learners in the base models, effective discernible features could be well learned; (2). Base models are fitted on the whole training data to compute the performance on the test data; (3). Meta-model is used to make a prediction on the test data.

Undoubtedly, we will get benefits by combining stacking and GCN model. To the best of our knowledge, there is no general framework for dealing with heterogeneous graph structural data by GCN model. In this paper, we present a novel simplified stacking based architecture for handling graph data, SStaGCN, which combines the feature extraction ability of stacking and aggregation, and GCN's ability to learn graph structure on graph data. These enable SStaGCN to inherit the advantages of the stacking method and graph convolutional network. Overall, the contributions of the paper are listed as follows:

- (1). We propose a new delicate general architecture that combines simplified stacking and GCN, which is adaptive and flexible to tackle heterogeneous graph structural data of different types. To the best of the authors' knowledge, this is the first systematic work that applies modified stacking approach to general graph structural data.
- (2). Generalization bound is addressed to highlight the role of stacking and aggregation from the viewpoint of learning

theory.

(3). Extensive evaluation of our approach against strong baselines in node prediction tasks is investigated. The experimental results indicate significant performance improvements on both homogeneous and heterogeneous node classification tasks over a variety of real-world graph structural data, the over-smoothing phenomenon can be well alleviated.

The remainder of the paper is organized as the following. In Section II, we give a brief review of related work. Section III addresses the theoretical analysis and the proposed algorithm for GCN. Experiments on 3 public citation networks and another 3 heterogeneous datasets are relegated in Section IV. Some discussions and concluding remarks go to Section V. Proof of the main result is provided in the Appendix.

II. RELATED WORK

Graph Convolutional Networks

GCNs can be routinely interpreted as extensions of traditional convolutional neural networks on the graph domain. Typically, there are two types of GCNs [14]: spatial GCNs and spectral GCNs. Spatial GCNs construct new feature vectors for each vertex using its neighborhood information, in which convolution is viewed as “patch operator”. Spectral GCNs define the convolution by decomposing a graph signal on the spectral domain, and then employing a spectral filter (Fourier or wavelet, [1], [15], [16]) on the spectral components [17], [18]. However, this model entails the computation of the Laplacian eigenvector, which is exhausted and impractical for large-scale graphs. Hammond et al. [19] used Chebyshev polynomials up to K -th order to approximate the spectral filter. Defferrard and Vandergheynst [20] constructed a K -localized ChebyNet. Kipf and Welling [8] considered the case $K = 1$ and proposed a simple but powerful model for semi-supervised classification task. Wu et al. [21] removed nonlinearities and collapsed the weight matrix between consecutive layers, achieved a simplified GCN, whilst [22], [23] considered the design of deep GCNs. Multi-scale deep GCNs were investigated in [24].

Ensemble learning based graph neural networks

GCNs may confront with the over-smoothing problem and can not handle heterogeneous graph data. Sun et al. [12] designed a RNN-like graph structure to extract the knowledge from high-order neighbors of the current nodes, while Ivanov and Prokhorenkova [10] incorporated GBDT into GNN and developed BGNN to tackle heterogeneous tabular data. A natural question arises: Is there a general GCN architecture to handle various heterogeneous graph structural data and mitigate the over-smoothing issue? This paper aims at investigating these challenges and answers the above-mentioned questions.

III. THE PROPOSED APPROACH: SSTAGCN

A. Graph convolutional networks

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $v_i \in \mathcal{V}$, edges $(v_i, v_j) \in \mathcal{E}$. Denote $A \in \mathbb{R}^{N \times N}$ as the adjacency matrix with corresponding degree matrix $\mathbf{D}_{ii} = \sum_j A_{ij}$. Obviously, for an undirected graph, $A_{ij} = A_{ji}$. In the conventional GCN models for semi-supervised node

classification task, the graph embedding of nodes with two convolutional layers is described as the following:

$$Z = \hat{\mathbf{A}} \text{ReLU}(\hat{\mathbf{A}} X W^{(0)}) W^{(1)} \quad (1)$$

where $Z \in \mathbb{R}^{N \times K}$ is the final embedding matrix (output logits) of nodes before softmax with K the number of classes. $X \in \mathbb{R}^{N \times d}$ stands for the feature matrix with d the input dimension. $\hat{\mathbf{A}} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, where \tilde{D} is the degree matrix of \tilde{A} and $\tilde{A} = A + I$ (I stands for identity matrix). Moreover, $W^{(0)} \in \mathbb{R}^{d \times H}$ is the input-to-hidden weight matrix for a hidden layer with H feature maps, and $W^{(1)} \in \mathbb{R}^{H \times K}$ denotes the hidden-to-output weight matrix.

B. Stacking

Stacking, as a hierarchical model integration framework, is a well-known and widely used ensemble machine learning algorithm [25]. It uses a meta-learning algorithm to learn how to best combine the prediction from two or more base machine learning algorithms. Traditional stacking model involves two or more base models, and a meta-model that combines the predictions of the base models. Base models use different types of models to fit on the training data and compile the predictions. Meta-model tries to best combine the predictions of the base models, which is often simple, providing a smooth interpretation of the predictions made by the base models. Hence, linear models are often used as the meta-model, such as linear regression for regression tasks and logistic regression for classification tasks.

C. The proposed approach

As mentioned above, GCN may mix the node features from different clusters and make them indistinguishable. Therefore, it is necessary to aggregate more node information in an effective way for better predictions. Motivated by traditional stacking approach and the work in [10], [12], to reduce the computational cost, we only use base models of the stacking approach, and then aggregate the output of them to attain the node features of the graph data. Specifically, the proposed method could be addressed as follows. At first, in the first layer, we attain X' by input $X \in \mathbb{R}^{N \times d}$ (node feature matrix) through k base classifiers

$$X'_i = h_i(X), i = 1, \dots, k, \quad (2)$$

Thereafter, we get the pre-classification results X'_i through $h_i(X)$ ($i = 1, \dots, k$). Secondly, we use aggregation method to attain the final output results, i. e.,

$$\hat{X} = g(X'_1, \dots, X'_k), \quad (3)$$

where $g(\cdot)$ denotes an aggregation method. The idea of aggregation method is very simple, which aims to group attribute values by a single value. Generally, we can choose mean, attention or voting approach. Recall

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

Attention [26]: Attention mechanism has been widely used in various fields of deep learning, including but not limited

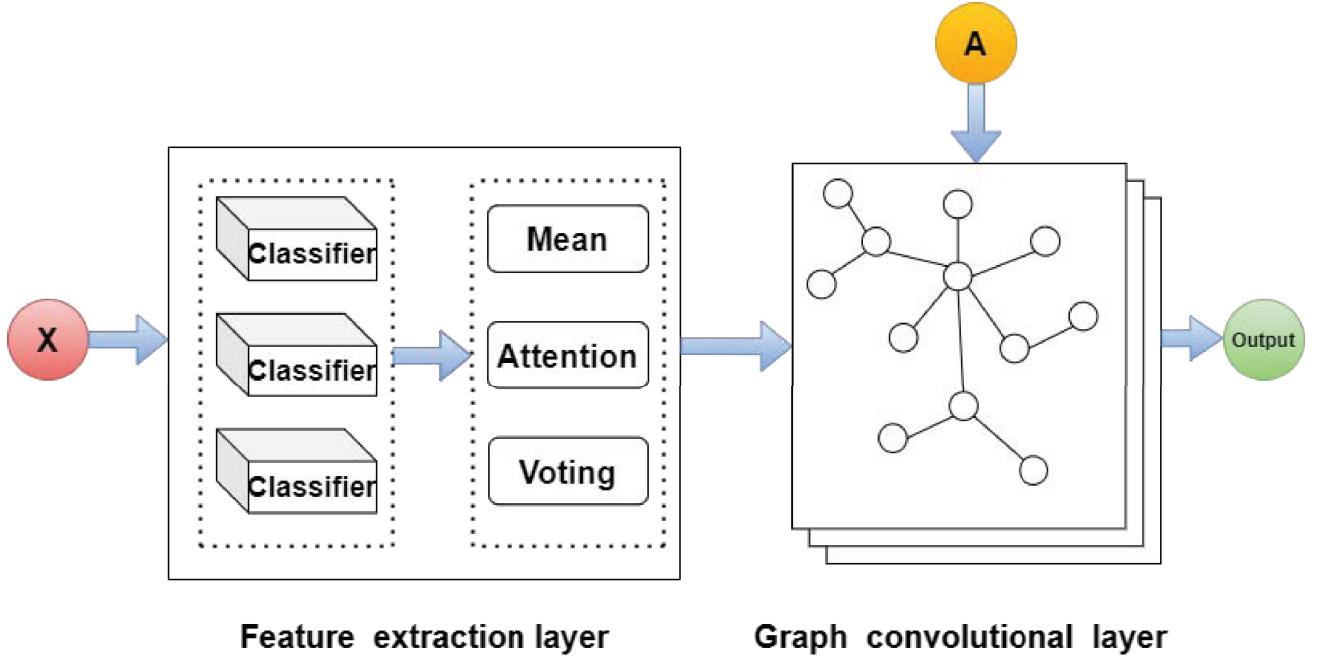


Fig. 1. Workflow of the SStaGCN model.

Algorithm 1 SStaGCN.

Input:

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

Output:

Final predictor $f(X)$;

- 1: Attain X'_i ($i = 1, \dots, k$) via k base classifiers;
 $X'_i \leftarrow h_i(X)$, $i = 1, \dots, k$;
 - 2: Aggregate X'_1, \dots, X'_k ;
 $\tilde{X} \leftarrow g(X'_1, \dots, X'_k)$;
 - 3: Feed \tilde{X} into vanilla GCN ;
 $f(X) \leftarrow GCN(\tilde{X})$;
 - 4: **return** $f(X)$;
-

to image processing [27], speech recognition [28], and natural language processing [29]. The idea of attention is motivated from the attention mechanism of human beings. Denote query vector V as the output of base classifiers, let query vector Y be the label of the data. Then we compute the attention coefficients between Y and V as follows:

$$a_i = \text{softmax}(\cos(Y, V_i)), i = 1, \dots, k, \quad (4)$$

where \cos denotes cosine similarity. Finally, the input \tilde{X} of the graph convolutional layer is aggregated by considering the following sum with attention score a_i

$$\tilde{X} = \sum_{i=1}^k a_i V_i. \quad (5)$$

Voting [30]: Voting method is the most intuitive way among all integrated learning methods, which aims at selecting one or more winners. In this work, our objective is to choose the most

popular prediction among the results of the base classifiers. Therefore, we will take the majority voting approach. if the number of categories appears the same, a category will be randomly selected.

Thus, we attain a novel GCN model to deal with heterogeneous graph data by elegantly combining stacking, aggregation, and vanilla GCN, namely, SStaGCN. The first layer of SStaGCN utilizes the base models of stacking approach, and the second layer of SStaGCN uses aggregation method such as mean, attention or voting, which can enhance the feature extraction ability of conventional GCN models. The aggregated data will henceforth be used as the input of conventional GCN model, and we attain the final prediction results. The workflow of the proposed model is demonstrated in Algorithm 1 and Fig. 1.

D. Generalization Bound Analysis

In this Section, we give theoretical generalization analysis of the proposed approach. In our analysis, we assume that the adjacency matrix A and the node feature matrix X are both fixed.

Theoretically, in learning theory, the risk of f over the unknown population distribution P is measured by

$$\mathcal{E}(f) := \mathbb{E}_{\mathcal{X} \times \mathcal{Y}}[\ell(y, f(\tilde{x}))],$$

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

$$\mathcal{E}_m(f) := \frac{1}{m} \sum_{j=1}^m \ell(y_j, f(\tilde{x}_j)),$$

where $\{(\tilde{\mathbf{x}}_j, y_j)_{j=1}^m\}$ is the labelled sample achieved from the original training data $\{(\mathbf{x}_j, y_j)_{j=1}^m\}$ via stacking and aggregation. Typically, the clustering algorithms will only produce discernible nodes. Hence, if $\|\mathbf{x}_i\| \leq R, i = 1, \dots, N$. It is trivial that stacking and the three aggregation methods proposed in this paper will not violate the constraints, i.e., $\|\tilde{\mathbf{x}}_i\| \leq R, i = 1, \dots, N$. Now we are in position to present the theoretical generalization bound analysis.

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

$$\begin{aligned} \mathcal{E}(f) \leq \mathcal{E}_N(f) + & \frac{2\alpha_\ell\sqrt{2q}KB_1B_2R\sum_{s=1}^q M_s}{\sqrt{m}} \\ & + \sqrt{\frac{2\log(2/\delta)}{N}}, \end{aligned}$$

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

Remark 1: Theorem 1 indicates that the dominant upper bound linearly depends on the maximum number of the neighbors of the nodes q , bounds of the weights B_1 and B_2 , which strongly depend on the dimension index d_1 . Obviously, $d_1 \ll d$ will yield a tight generalization bound. When $K = 1$ (binary classification case), the results stated here is similar to the one outlined in [31].

IV. EXPERIMENTS

A. Datasets

To evaluate the performance of the proposed SStaGCN model for distinct types of graph structural data, we utilize 6 real-world datasets for the semi-supervised node classification task, including 3 commonly used citation networks: Cora, CiteSeer, and Pubmed [32], and another 3 heterogeneous datasets: House_class, VK_class and DBLP [10]. As indicated in [10], House_class and VK_class are from House and VK datasets, respectively, where the target labels are converted into several discrete classes due to the lack of publicly available heterogeneous graph-structured data.

In the 3 citation networks, nodes represent documents, and edges (undirected) stand for the citation relationships connected to documents. The characteristics of nodes are representative words in documents, and the label rate here denotes the percentage of node tags used for training. The Cora dataset contains 2708 nodes, 5429 edges, 7 classes, and 1433 node features, the CiteSeer dataset contains 3327 nodes, 4732 edges, 6 classes, and 3703 node features, and the Pubmed dataset contains 19,717 nodes, 44,338 edges, and 3 classes. We select 140, 120, and 60 nodes for Cora, CiteSeer, and Pubmed datasets for training, respectively. Each dataset uses 1000 nodes for testing, and 500 nodes for cross-validation. The data splitting we used is the same as that of GCN, Graph Attention Network (GAT, [33]) and GWNN [15].

Details about the citation network (heterogeneous network resp.) are described in Table I (Table II resp.)

TABLE I
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 II
SUMMARY OF THE HETEROGENEOUS DATA.

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

B. Baselines

We compare SStaGCN with 4 classical graph convolutional networks: ChebyNet, GCN, GAT, and APPNP [34], and 2 ensemble learning based GCNs: AdaGCN and BGNN models.

C. Setting

As demonstrated in Fig. 1, the proposed SStaGCN model contains four layers, where the first and second layers are called feature extraction layer. The first layer is based on the base models of the stacking method. Here we consider the combination of 7 classical classifiers: KNN, Random Forest, Naive Bayes, Decision Tree, SVC [35], GBDT [36], and Adaboost [37] in the first layer of SStaGCN. These 7 classifiers are representative classical classifiers used in the community of machine learning, which have respective merits in dealing with distinct types of tasks. Moreover, we adopt three aggregation methods: mean, attention, and voting in the second layer. Among them, for the mean approach, we take the mean value of the output of the first layer and then round it. As for the attention mechanism, we consider the label data as vector Y , the predicted value of the feature extraction layer as the query vector V , and then compute the attention coefficients. For the voting approach, we employ the hard voting technique in ensemble learning [38].

Thereafter, the output of the second layer is considered as the input of the first layer of the GCN, which only has two layers in our setting. The GCN considered in this paper has 16 hidden units, and Adam optimizer [39] is the default optimizer, cross entropy is used as the loss function. We set learning rate $\gamma = 0.01$, number of iterations $itr = 500$, weight decay $5e-4$, and dropout rate equals 0.

D. Results

The results of the comparative evaluation for node classification are summarized in Tables III-XI, where SStaGCN (Mean)

TABLE III

AVERAGE ACCURACY ON 3 CITATION NETWORKS UNDER 30 RUNS BY COMPUTING THE 95% CONFIDENCE INTERVAL VIA BOOTSTRAP

Method	Cora	CiteSeer	Pubmed
ChebyNet	81.20 \pm 0.00	69.80 \pm 0.00	74.40 \pm 0.00
GCN	81.50 \pm 0.00	70.30 \pm 0.00	79.00 \pm 0.00
GAT	83.00 \pm 0.70	72.50 \pm 0.70	79.00 \pm 0.30
APPNP	85.09 \pm 0.25	75.73 \pm 0.30	79.73 \pm 0.31
AdaGCN	85.97 \pm 0.20	76.68 \pm 0.20	79.95 \pm 0.21
BGNN	41.97 \pm 0.19	30.74 \pm 0.10	10.32 \pm 0.10
Sim_Stacking	43.19 \pm 2.05	62.7 \pm 0.53	87.7 \pm 0.23
SStaGCN (Mean)	90.35 \pm 0.20	86.40 \pm 0.12	82.30 \pm 0.19
SStaGCN (Attention)	91.60 \pm 0.18	87.20 \pm 0.12	82.40 \pm 0.23
SStaGCN (Voting)	93.10\pm0.16	88.70\pm0.14	92.07\pm0.20

TABLE IV

AVERAGE ACCURACY ON HETEROGENEOUS DATASETS UNDER 30 RUNS BY CALCULATING THE 95% CONFIDENCE INTERVAL VIA BOOTSTRAP

Method	House_class	VK_class	DBLP
ChebyNet	54.74 \pm 0.10	57.19 \pm 0.36	32.14 \pm 0.00
GCN	55.07 \pm 0.13	56.40 \pm 0.09	39.49 \pm 1.37
GAT	56.50 \pm 0.22	56.42 \pm 0.19	76.83 \pm 0.78
APPNP	57.03 \pm 0.27	56.72 \pm 0.11	79.47 \pm 1.46
AdaGCN	26.20 \pm 0.00	46.00 \pm 0.00	10.06 \pm 0.00
BGNN	66.7 \pm 0.27	66.32 \pm 0.20	86.94 \pm 0.74
Sim_Stacking	53.89 \pm 0.29	56.64 \pm 0.10	71.58 \pm 0.64
SStaGCN (Mean)	72.35 \pm 0.05	66.62 \pm 0.17	82.31 \pm 0.20
SStaGCN (Attention)	72.40 \pm 0.12	77.64 \pm 0.08	82.51 \pm 0.22
SStaGCN (Voting)	76.13\pm0.12	87.92\pm0.07	92.60\pm0.10

TABLE V

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 \pm 0.54	63.76 \pm 0.34	77.74 \pm 0.42
GCN	82.89 \pm 0.30	70.65 \pm 0.37	78.83 \pm 0.32
GAT	83.59 \pm 0.25	70.62 \pm 0.29	77.77 \pm 0.40
APPNP	84.29 \pm 0.22	71.05 \pm 0.38	79.66 \pm 0.31
AdaGCN	79.55 \pm 0.19	63.62 \pm 0.19	78.55 \pm 0.21
BGNN	40.81 \pm 0.25	32.73 \pm 0.13	8.46 \pm 0.08
Sim_Stacking	44.02 \pm 1.61	60.86 \pm 0.56	87.31 \pm 0.13
SStaGCN (Mean)	90.66 \pm 0.18	86.42 \pm 0.12	82.30 \pm 0.19
SStaGCN (Attention)	91.69 \pm 0.14	87.24 \pm 0.14	82.45 \pm 0.23
SStaGCN (Voting)	92.76\pm0.16	88.73\pm0.14	92.07\pm0.20

TABLE VI

AVERAGE F1-SCORE(MACRO) ON HETEROGENEOUS DATASETS UNDER 30 RUNS BY CALCULATING THE 95% CONFIDENCE INTERVAL VIA BOOTSTRAP

Method	House_class	VK_class	DBLP
ChebyNet	31.34 \pm 0.12	57.44 \pm 0.27	26.84 \pm 0.62
GCN	54.95 \pm 0.14	56.52 \pm 0.09	38.5 \pm 0.97
GAT	56.54 \pm 0.68	56.41 \pm 0.07	77.1 \pm 1.86
APPNP	57.88 \pm 0.32	56.61 \pm 0.07	79.34 \pm 0.23
AdaGCN	25.01 \pm 0.00	37.03 \pm 0.00	9.60 \pm 0.00
BGNN	66.48 \pm 0.22	66.18 \pm 0.11	87.2 \pm 0.60
Sim_Stacking	53.32 \pm 0.15	56.11 \pm 0.08	71.49 \pm 0.31
SStaGCN (Mean)	72.23 \pm 0.04	66.74 \pm 0.21	82.13 \pm 0.38
SStaGCN (Attention)	72.36 \pm 0.09	77.62 \pm 0.10	82.68 \pm 0.12
SStaGCN (Voting)	75.45\pm0.82	87.84\pm0.04	92.64\pm0.06

stands for mean mechanism is utilized in the second layer of SStaGCN, SStaGCN (Attention) and SStaGCN (Voting) have similar meanings. We report the accuracy, F1-score (macro), and training time on the test set between the proposed SStaGCN model and other methods. Experimental results successfully demonstrate significant improvement of SStaGCN model over the baselines. Specifically, for 3 public citation

networks, SStaGCN (Voting) achieves almost 8% (8%), 12% (17%), and 13% (12%) improvement of the accuracy (resp. F1-score) for Cora, CiteSeer and Pubmed datasets, respectively. For heterogeneous datasets, SStaGCN (Voting) obtains nearly 9% (9%), 21% (21%), and 6% (5%) improvement of the accuracy (resp. F1-score) for House_class, VK_class, and DBLP datasets, respectively. AdaGCN performs bad on heterogeneous dataset. This maybe due to the reason that AdaGCN aims at designing deep GCNs, which may mix the node features of different clusters when the layers of GCNs go deeper. Intuitively, SStaGCN is able to enhance the performance of GCN, and provides better qualitative results for distinct types of graph structural data.

This impressive improvement can be explained as follows:

(1). The feature extraction step of SStaGCN can achieve a dimensionality reduction effect and make the graph data more discernible. For instance, the size of Cora dataset reduces to 2708×7 from 2708×1433 after conducting feature extraction, which means we attain a relatively smaller d_1 as discussed in Remark 1, and this greatly improves the prediction ability and computation efficiency in the subsequent graph convolution model.

(2). In the aggregation step of SStaGCN model, the mean and attention mechanisms destroy the pre-classification results to some extent, which is inappropriate for feature extraction, while voting mechanism does not. Therefore, experimental results demonstrate that SStaGCN (Voting) is more efficient in our datasets.

(3). Simplified stacking can extract efficient node features but ignore the graph structure information, while GCN model is weak at extracting node features. Hence, the SStaGCN model inherits the merits of simplified stacking and GCN, not only achieves higher classification accuracy, but also reduces the cost of computation time.

Tables VIII and IX indicate the comparison of training time between SStaGCN and other methods. We can see that BGNN runs faster on 3 citation networks followed by our SStaGCN method, whilst the proposed SStaGCN runs faster on heterogeneous datasets except on DBLP dataset. We think the reason is that the feature extraction step takes extra computation time but yields more efficiency when the output of features is fed into the GCN model.

To express the effect of feature extraction step of the proposed model, we provide a visualization with t-SNE [40] as shown in Figs. 2 and 3. Figs. 2 and 3 indicate that the combination of stacking and aggregation could well extract the node features and make the graph data more discriminative. Moreover, the paired t-test in Table VII demonstrates that the proposed SStaGCN model is significantly different from the simplified stacking and other GCN models.

Table X indicates that we do not need all the seven classifiers. Taking Cora dataset as an example, we can observe that KNN, Random Forest, and Naive Bayes is the best combination, which attains the highest accuracy value without much cost in computation time (only 3 seconds). Therefore, this demonstrates that the 7 classifiers have their own merits in handling different specific tasks.

TABLE VII
P-VALUES OF THE PAIRED T-TEST OF SStAGCN (VOTING) WITH COMPETITORS ON 6 DIFFERENT DATA SETS (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	6.07e-11	1.07e-06

TABLE VIII

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

TABLE IX

AVERAGE TRAINING TIME(S) ON HETEROGENEOUS 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

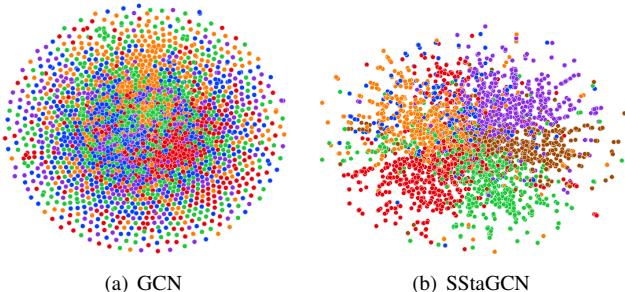


Fig. 2. Visualization of classification features by the GCN (left) and the features after conducting feature extraction step in the SStaGCN model (right) on CiteSeer dataset, node colors denote classes.

Furthermore, to demonstrate the effect of simplified stacking to the over-smoothing problem, we also add an experiment on the over-smoothing discussion. In Fig. 4, we can observe that conventional GCN may mix the features of vertices from different clusters when increasing the layers of GCN. However, as demonstrated in Fig. 5 and Table XI¹, the proposed SStaGCN could effectively ameliorate the over-smoothing phenomenon and improve the accuracy.

¹here the number of layers do not contain the number of layers included in feature extraction part (only 2 layers) of SStaGCN.

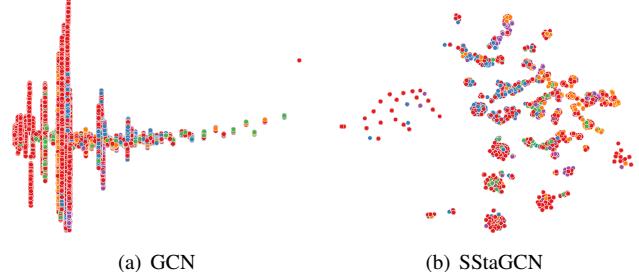


Fig. 3. Visualization of classification features by the GCN (left) and the features after conducting feature extraction step in the SStaGCN model (right) on DBLP dataset, node colors denote classes.

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

E. Visualization

To further demonstrate the performance of SStaGCN, we plot the final classification features via GCN, AdaGCN, BGNN, and our SStaGCN. Fig. 6 (Fig. 7 resp.) displays the final classification features of relevant methods on CiteSeer (DBLP resp.) dataset. From Figs. 6 and 7, we can observe that relatively smaller points are misclassified by the proposed SStaGCN, whilst many classes are wrongly predicted and classified by GCN, AdaGCN and BGNN.

V. CONCLUSION

Traditional GCNs could not well deal with heterogeneous graph structural data. In this work, we propose a novel GCN architecture, namely, SStaGCN. SStaGCN first takes advantages of stacking method and aggregation technique to attain pre-classified data features, and then utilizes GCN to conduct prediction for heterogeneous graph data. Our approach SStaGCN can effectively explore and exploit node features for heterogeneous graph data in a stacking way. Our work paves a way towards better combining classical machine learning methods to design GCN models, proposes a general framework for handling distinct types of graph structural data, and will definitely give insights to a better understanding of GCN. Extensive experiments demonstrate that the proposed model is superior to state-of-the-art competitors in terms of accuracy, F1-score, and training time. The proposed framework here could be generalized to the regression setting. Furthermore, we believe the proposed method can tackle various distinct types of heterogeneous graph data, although experiments are

TABLE X
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
	✓	✓			✓	✓	93.6	16.60
		✓	✓	✓			84.2	567.9
		✓	✓			✓	92.9	144.7
		✓	✓			✓	93.1	149.5
✓	✓	✓	✓				92.8	15.90
✓	✓	✓	✓		✓		93.4	18.80
✓	✓	✓	✓	✓	✓		92.9	568.3
✓	✓	✓	✓	✓	✓	✓	92.9	570.7
✓	✓	✓	✓	✓	✓	✓	92.8	708.5

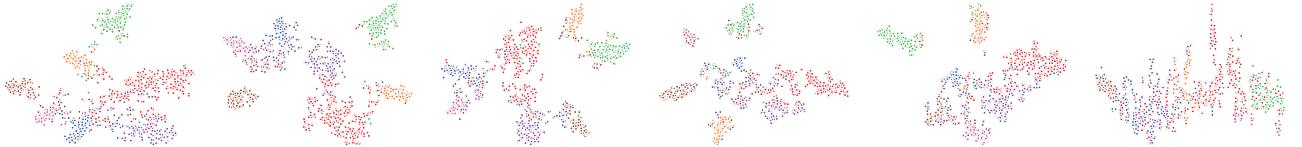


Fig. 4. Visualization of final classification features via GCN on Cora dataset with 2, 3, 4, 5, 6, 7 layers, node colors denote classes.

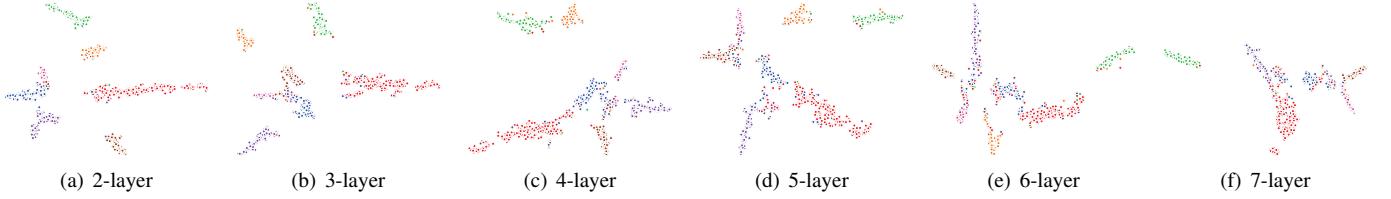


Fig. 5. Visualization of final classification features via SStAGCN on Cora dataset with 2, 3, 4, 5, 6, 7 layers, node colors denote classes.

TABLE XI
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

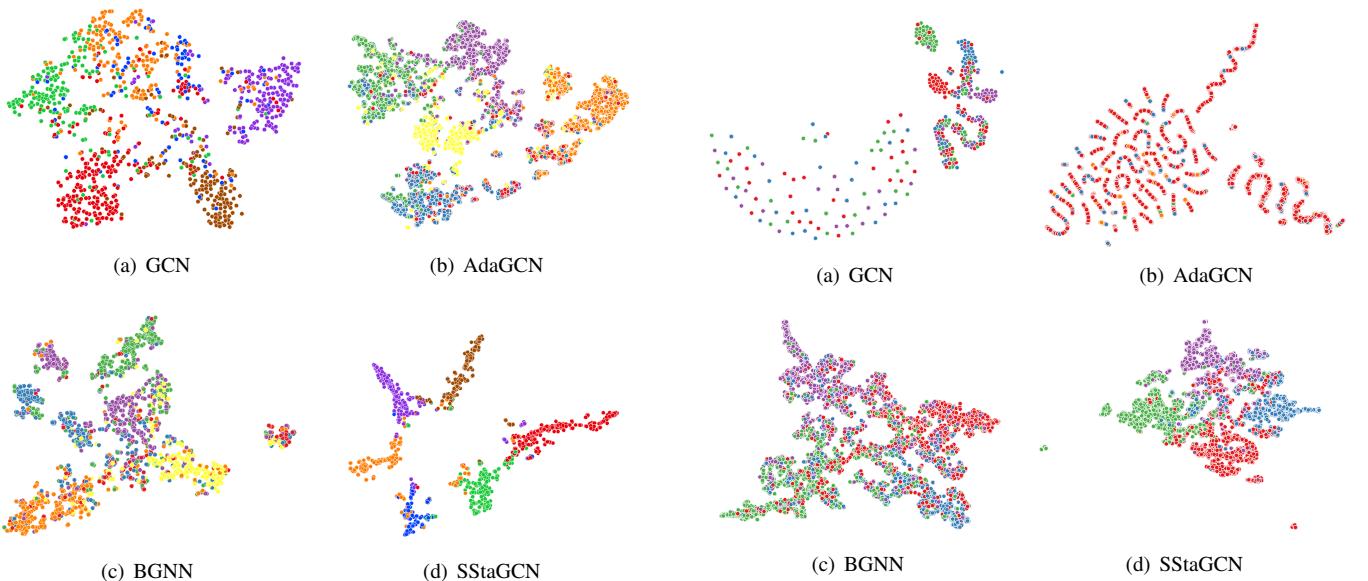


Fig. 6. Visualization of final classification features via (a). GCN , (b). AdaGCN , (c). BGNN , and (d). SStAGCN model on Citeseer dataset, node colors denote classes.

Fig. 7. Visualization of final classification features via (a). GCN , (b). AdaGCN , (c). BGNN , and (d). SStAGCN model on DBLP dataset, node colors denote classes.

conducted on tabular data. A promising future research direction is to investigate deeper GCNs in our setting as discussed by [12].

VI. APPENDIX

In this part, we give the detailed proof of Theorem 1. Before addressing the proof, we give several lemmas. At first, let us present the contraction inequality of Rademacher complexity in the vector form.

Lemma 1: [41] Let \mathcal{X} be any set, $(\mathbf{x}_1, \dots, \mathbf{x}_N) \in \mathcal{X}$ and \mathcal{F} be a class of functions $f : \mathcal{X} \rightarrow \mathbb{R}^K$ ad let $h_i : \mathbb{R}^K \rightarrow \mathbb{R}$ have Lipschitz constant L . Then

$$\mathbb{E} \sup_{f \in \mathcal{F}} \sum_{i=1}^N \sigma_i h_i f(\mathbf{x}_i) \leq \sqrt{2} L \mathbb{E} \sup_{f \in \mathcal{F}} \sum_{i=1}^N \sum_{k=1}^K \sigma_{ik} f_k(\mathbf{x}_i),$$

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

Lemma 2: [42] Consider a loss function $\ell : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$. Denote $\mathcal{H} = \{\ell(y, f(\cdot)), f \in \mathcal{F}\}$, and let $(\mathbf{x}_i, y_i)_{i=1}^N$ be independently selected according to the probability measure \mathbb{P} . Then for any $0 < \delta < 1$, with probability at least $1 - \delta$,

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

We first give a lemma which plays essential role in the proof of Theorem 1.

Lemma 3: Let $q = \max\{N(i)\}$ for each node $v_i \in \Omega$, then

$$\left\| \sum_{j \in N(v)} \hat{A}_{vj} \mathbf{x}_j \right\|_2^2 \leq R^2 q.$$

Proof 1: Denote $\hat{A}_v \in \mathbb{R}^{q \times q}$ as the sub-matrix of \hat{A} whose row and column indices belong to the set $j \in N(v)$. Let $\tilde{X}_v = (\mathbf{x}_1^T, \dots, \mathbf{x}_q^T)^T \in \mathbb{R}^{q \times d}$ be the feature matrix of the nodes in \mathcal{G}_v (subgraph of \mathcal{G}). Hence,

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

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

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

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

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

Now we are in position to give the proof of Theorem 1.

Proof 2: To allow a slight abuse of notations, we will use \mathbf{x}_j to denote $\tilde{\mathbf{x}}_j$ due to the explanation on page 3. Denote $h(W^{(0)}) = \text{ReLU}(\hat{A}XW^{(0)})$, $f(W^{(1)}) = \text{softmax}(\hat{h}(W^{(0)})W^{(1)}) = (f_1(W^{(1)}), \dots, f_m(W^{(1)}))^T$ with $\hat{h}(W^{(0)}) = \hat{A}h(W^{(0)})$. Applying Proposition 4 in [43] to the case $\lambda = 1$, we can attain the Lipschitz constant for standard

softmax function is $L = 1$. Let \hat{A}_i stands for the i -th row of the matrix \hat{A} , for function set

$$\begin{aligned} \mathcal{F}_{B_1, B_2} = \{f_i = \text{softmax}(\hat{A}_i \cdot \text{ReLU}(\hat{A}XW^{(0)})W^{(1)}), \\ i = 1, \dots, m, \|W^{(0)}\|_F \leq B_1, \|W^{(1)}\|_F \leq B_2\}, \end{aligned}$$

the empirical Rademacher complexity is defined as

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

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

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

and notice Lemma 2, we only need to bound $\hat{\mathcal{R}}(\mathcal{F}_{B_1, B_2})$. Therefore, we have the following estimate by utilizing Lemma 1.

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

where $W^{(1)} = (\mathbf{w}_1^{(1)}, \dots, \mathbf{w}_K^{(1)})$, notice the property of inner product, the above estimate can be further bounded as

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

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 $\sup_{\|W^{(0)}\|_F \leq B_1} \left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_i(W^{(0)}) \right\|_2$. We will employ the idea introduced in [31] (in the proof of Theorem 1) to remove the “sup” term. Let $h_v(W^{(0)}) = (h_v^1(\mathbf{w}_1^{(0)}), h_v^2(\mathbf{w}_2^{(0)}), \dots, h_v^K(\mathbf{w}_K^{(0)}))$ with $W^{(0)} = (\mathbf{w}_1^{(0)}, \dots, \mathbf{w}_K^{(0)})$ and notice that

$\hat{h}(W^{(0)}) = \hat{A}h(W^{(0)})$, then we have

$$\begin{aligned} & \left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_i(W^{(0)}) \right\|_2^2 \\ &= \sum_{t=1}^H \left(\sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h_v^t(\mathbf{w}_t^{(0)}) \right)_2^2 \\ &= \sum_{t=1}^H \|\mathbf{w}_t^{(0)}\|_2^2 \left(\sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h_v(\mathbf{w}_t^{(0)}) \right)^2. \end{aligned}$$

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

$$\begin{aligned} & \sup_{\|W^{(0)}\|_F \leq B_1} \left\| \sum_{i=1}^m \sigma_{ik} \hat{h}_i(W^{(0)}) \right\|_2 \\ &= \sup_{\|\mathbf{w}\|_2 = B_1} \left(\sum_{i=1}^m \sigma_{ik} \sum_{v \in N(i)} \hat{A}_{iv} h_v(\mathbf{w}) \right). \end{aligned}$$

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

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

Applying the conclusion $\sup(\sum_s a_s) \leq \sum_s \sup(a_s)$ and contraction property of Rademacher complexity again, we have

$$\begin{aligned} & \mathbb{E}_\sigma \left[\sum_{k=1}^K \sup_{\|\mathbf{w}\|_2 = B_1} \left(\sum_{s=1}^q M_s \sum_{i=1}^m \sigma_{ik} h_{n_s(i)}(\mathbf{w}) \right) \right] \\ &\leq \mathbb{E}_\sigma \left[\sum_{k=1}^K \sum_{s=1}^q M_s \sup_{\|\mathbf{w}\|_2 = B_1} \sum_{i=1}^m \sigma_{ik} h_{n_s(i)}(\mathbf{w}) \right] \\ &\leq \sum_{s=1}^q M_s \mathbb{E}_\sigma \left[\sum_{k=1}^K \sup_{\|\mathbf{w}\|_2 = B_1} \sum_{i=1}^m \sigma_{ik} \left(\sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \langle \mathbf{x}_j, \mathbf{w} \rangle \right) \right] \\ &= \sum_{s=1}^q M_s \mathbb{E}_\sigma \left[\sum_{k=1}^K \sup_{\|\mathbf{w}\|_2 = B_1} \left\langle \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j, \mathbf{w} \right\rangle \right] \\ &\leq B_1 \sum_{s=1}^q M_s \mathbb{E}_\sigma \left[\sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2 \right]. \end{aligned}$$

Therefore, we only need to estimate the term

$$\mathbb{E}_\sigma \left[\sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2 \right].$$

Applying Cauchy-Schwartz inequality yields that

$$\begin{aligned} & \mathbb{E}_\sigma \left[\sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2 \right] \\ &\leq \sqrt{\mathbb{E}_\sigma \left(\sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2 \right)^2} \\ &\leq \sqrt{\mathbb{E}_\sigma K \sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2^2} \\ &\leq K \sqrt{\sum_{i=1}^m \left\| \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2^2}, \end{aligned}$$

where the last inequality is due to the i.i.d condition of Rademacher sequences. Plugging the conclusion of Lemma 3 into the above term leads to

$$\begin{aligned} & \mathbb{E}_\sigma \left[\sum_{k=1}^K \left\| \sum_{i=1}^m \sigma_{ik} \sum_{j \in N(n_s(i))} \hat{A}_{n_s(i)j} \mathbf{x}_j \right\|_2 \right] \\ &\leq KR\sqrt{qm}, \end{aligned}$$

and

$$\hat{\mathcal{R}}(\mathcal{F}_{B_1, B_2}) \leq \frac{\sqrt{2q}KB_1B_2R\sum_{s=1}^q M_s}{\sqrt{m}}.$$

This completes the proof by combining with Lemma 2.

REFERENCES

- [1] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in *ICLR*, 2014.
- [2] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *IJCNN*, 2005.
- [3] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NIPS*, 2017.
- [4] 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,” *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.
- [5] S. Casas, C. Gulino, R. Liao, and R. Urtasun, “Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data,” *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9491–9497, 2020.
- [6] J. M. Stokes, K. Yang, K. Swanson, W. Jin, and J. J. Collins, “A deep learning approach to antibiotic discovery,” *Cell*, vol. 180, no. 4, pp. 688–702.e13, 2020.
- [7] L. Yao, C. Mao, and Y. Luo, “Graph convolutional networks for text classification,” in *AAAI*, 2019.
- [8] T. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *ICLR*, 2017.
- [9] J. Zhu, “Max-margin nonparametric latent feature models for link prediction,” in *ICML*, 2012.
- [10] S. Ivanov and L. Prokhorenko, “Boost then convolve: Gradient boosting meets graph neural networks,” in *ICLR*, 2021.
- [11] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in *AAAI*, 2018.
- [12] K. Sun, Z. Lin, and Z. Zhu, “Adagcn: Adaboosting graph convolutional networks into deep models,” in *ICLR*, 2021.
- [13] S. Džeroski and B. Ženko, “Is combining classifiers with stacking better than selecting the best one?” *Machine Learning*, vol. 54, pp. 255–273, 2004.
- [14] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: Going beyond euclidean data,” *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.
- [15] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, “Graph wavelet neural network,” in *ICLR*, 2019.

- [16] M. Li, Z. Ma, Y. G. Wang, and X. Zhuang, “Fast haar transforms for graph neural networks,” *Neural Networks*, vol. 128, pp. 188–198, 2020.
- [17] A. Sandryhaila and J. Moura, “Discrete signal processing on graphs,” *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.
- [18] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.
- [19] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.
- [20] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *NIPS*, 2016.
- [21] F. Wu, T. Zhang, A. Souza, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” in *ICML*, 2019.
- [22] Q. Li, Z. Han, and X. M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in *AAAI*, 2018.
- [23] G. Li, M. Mueller, A. Thabet, and B. Ghanem, “Deepgcns: Can gcns go as deep as cnns?” in *ICCV*, 2019.
- [24] S. Luan, M. Zhao, X. W. Chang, and D. Precup, “Break the ceiling: Stronger multi-scale deep graph convolutional networks,” in *NIPS*, 2019.
- [25] D. Wolpert, “Stacked generalization,” *Neural Networks*, vol. 5, pp. 241–259, 1992.
- [26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NIPS*, 2017, pp. 6000–6010.
- [27] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent models of visual attention,” in *NIPS*, 2014.
- [28] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *ICLR*, 2015.
- [29] 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*, vol. 4, pp. 259–272, 2016.
- [30] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, “Boosting the margin: A new explanation for the effectiveness of voting methods,” in *ICML*, 1997.
- [31] S. Lv, “Generalization bounds for graph convolutional neural networks via rademacher complexity,” 2021.
- [32] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, “Collective classification in network data,” *AI Mag.*, vol. 29, pp. 93–106, 2008.
- [33] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio’, and Y. Bengio, “Graph attention networks,” in *ICLR*, 2018.
- [34] J. Klicpera, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized pagerank,” in *ICLR*, 2019.
- [35] K. Pal and B. Patel, “Data classification with k-fold cross validation and holdout accuracy estimation methods with 5 different machine learning techniques,” *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 83–87, 2020.
- [36] J. Friedman, “Greedy function approximation: A gradient boosting machine.” *Annals of Statistics*, vol. 29, pp. 1189–1232, 2001.
- [37] Y. Freund and R. E. Schapire, “A short introduction to boosting,” *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771–780, 1999.
- [38] F. Schwenker, “Ensemble methods: Foundations and algorithms,” pp. 77–79, 2013.
- [39] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- [40] L. V. D. Maaten and G. E. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
- [41] A. Maurer, “A vector-contraction inequality for rademacher complexities,” in *International Conference on Algorithmic Learning Theory*, 2016.
- [42] P. L. Bartlett and S. Mendelson, “Rademacher and gaussian complexities: Risk bounds and structural results,” in *Conference on Computational Learning Theory & and European Conference on Computational Learning Theory*, 2001.
- [43] B. Gao and L. Pavel, “On the properties of the softmax function with application in game theory and reinforcement learning,” 2017.