Tackling Over-Smoothing: Graph Hollow Convolution Network with Topological Layer Fusion

Junhong Wan
Hikvision Research Institute
Hikvision
Hangzhou, China
wanjunhong@hikvision.com

Yao Fu
Hikvision Research Institute
Hikvision
Hangzhou, China
fuyao@hikvision.com

Weihao Jiang
Hikvision Research Institute
Hikvision
Hangzhou, China
jiangweihao5@hikvision.com

Shiliang Pu
Hikvision Research Institute
Hikvision
Hangzhou, China
pushiliang.hri@hikvision.com

Junlan Yu
Hikvision Research Institute
Hikvision
Hangzhou, China
yujunlan@hikvision.com

***Abstract*—In recent years, Graph Convolutional Networks (GCNs) have achieved great success in graph representations learning by incorporating features and topology information in each layer. Yet most GCNs are limited to the shallow network architecture due to over-smoothing which leads to indistinguishable nodes representations. In this paper, we provide a unique perspective on the key factor of over-smoothing which is the topological expressiveness loss when the stacked graph diffusion operators are over-densifying in deep layers. To tackle the over-smoothing issue, we propose the Graph Hollow Convolution Network (GHCN) with two key innovations. First, we design a hollow filter applied to the stacked graph diffusion operators to retain the topological expressiveness. Second, in order to further exploit the topology information, we integrate information from different layers based on graph local structures using topological layer fusion without propagating through nodes repeatedly. Extensive experiments on benchmark datasets show that our proposed method outperforms the state-of-the-art methods and effectively relieves the over-smoothing problem.**

***Index Terms*—Deep Learning, Graph Representation Learning, Graph Neural Networks**

I. Introduction

Graphs are the mathematical abstractions that characterize many real-world problems in social networks, traffic optimization, molecular chemistry, knowledge graphs, and recommendation systems. Among many attempts that utilize such abundant data structures, the Graph Convolutional Networks [1] approach has achieved tremendous success in recent years. One layer of graph convolution only considers immediate neighbors and by stacking those layers GCNs are able to aggregate information from multi-hop neighbors. However, when applying graph convolution recursively, the performance of GCNs is sharply compromised along with the increasing depths of the networks. Such performance deterioration is caused by over-smoothing that the representation of each node will eventually converge and thus become indistinguishable [2].

Recently, there have been many attempts to resolve the problem of over-smoothing in order to build deep architectures in GCNs. Several works try to add regularizing terms to relieve the problem of over-smoothing. DropEdge [3] suggests that randomly removing certain amounts of edges is able to reduce the convergence speed of over-smoothing and achieve deeper GCNs architecture. PairNorm [4] proposes a novel normalization layer that boosts the robustness of deep GCNs against over-smoothing. Both methods are broadly applicable to any GCN variants without any change in network architectures. While deeper GCNs adopting these modifications suffer far less from over-smoothing, modified deep models are still outperformed by shallow ones which makes deep architectures pointless.

On the other hand, a few works attempt to modify graph convolution operation to resolve the issue. DAGNN [5] provides a more rigorous and gentle theoretical description of the over-smoothing issue. The model decouples the representation transformation and propagation in current graph convolution operations. GCNII [6] proposes theoretical analysis for multi-layer GCN and GCNII models from the spectral aspect. The model introduces initial residual, which is constructing a connection to the input layer, and identity mapping to a GCN model to resolve the problem. However, the methods proposed do not directly resolve the issues caused by over-smoothing based on their theoretical analyses [7].

Reasonable smoothing makes the representations of nodes in the same cluster similar which greatly eases the classification tasks and is the key factor that GCNs work well in the first place [8], [9]. Over-smoothing has been studied by many works, and [5], [6], [8] provide their theoretical analyses of the

convergence of the GCNs when stacking graph convolutional layers. In theory, convergence only happens when extremely deep models are applied. But in practice, [10] finds that the consequences of over-smoothing occur before GCNs reach fairly deep depths. To answer the above contradiction, we provide a unique perspective from practical view based on empirical study that over-smoothing is caused by topological expressiveness loss as a result of over-densifying of the stacked graph diffusion operators, which are the stacked graph laplacian matrices in deep GCNs. The over-densifying is a phenomenon we first introduced in this paper which means the graph diffusion operator in deep layers is stacked in such a way that node neighbor sets explode exponentially along with network depth and the overall topological graph structure becomes more and more densely connected until fully connected. Consequently, the topology information loses its expressive power when the graph diffusion operator is over-densifying since most nodes have similar neighbor sets and become indistinguishable from the topological view, which makes incorporating such inexpressive topology information for tasks like node classification pointless. Thus, in order to prevent the topological expressiveness loss in deep GCNs that leads to over-smoothing, we want the over-densifying graph diffusion operator to be reasonably sparse.

In this paper, we propose a novel graph convolution operation denoted as Graph Hollow Convolution that propagates information based on dynamic pruned graph diffusion operators along with network depths instead of stacking stationary graph diffusion operators. The proposed pruning function on the graph diffusion operators conceptually serves as Hollow filters on those operators as illustrated in Fig. 1. Inspired by the non-backtracking random walk which is able to explore the graph topological structure more efficiently [11], our proposed method prunes the graph diffusion operators by ignoring nodes already aggregated in the previous layer, which enables us to remove unnecessary redundancy to avoid over-densifying. Unlike the non-backtracking random walks, our approach excludes any randomness and thus does not drop any informative nodes and edges. Moreover, we proposes topological layer fusion to further exploit the topology information and incorporate with Graph Hollow Convolution. Compared to GCNs propagating through nodes repeatedly to learn local structures and identify the importance of nodes [12], our method is able to learn graph local structures and identify the importance of nodes propagating each node only once. The overall Graph Hollow Convolution Network (GHCN) framework we proposed is a combination of the two novel approaches discussed before, which are the graph hollow convolution and the topological layer fusion. The empirical study shows that our framework suffers far less from the over-smoothing issue from the deep architectures and achieves competitive performance improvement on Planetoid [13], WikipediaNetwork [14], Amazon [15], and OGB [16] datasets over the state-of-the-art methods. In a nutshell, our contributions in this paper are listed as follows:

- We provide a unique insight into the over-smoothing issue, which is caused by the topological expressiveness loss when the stacked graph diffusion operators are over-densifying in deep layers.
- We propose a novel framework Graph Hollow Convolution Network (GHCN) including novel graph hollow convolution operation to alleviate over-smoothing and topological layer fusion to further exploit the topology information.
- Extensive experiments demonstrate that our proposed method relieves over-smoothing in the deep architecture effectively and achieves the state-of-the-art results.

II. Background and Related Works

First, let us go through the notions used in the paper. An attributed graph is denoted as $G = (V, E, X)$. V is the set of nodes with n nodes and E is the set of edges with m edges. $X \in \mathbb{R}^{n \times d}$ is the collection of the features of each node, where each row $x_i \in \mathbb{R}^d$ is associated with node i and d is the dimension of the node features. In this paper, we only consider unweighted and undirected graphs. The topology information is represented by adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $a_{i,j} = 1$ if an edge exists between node i and node j, otherwise 0. The degree matrix of the adjacency matrix denotes as $D = diag(d_1, d_2, ..., d_n)$, where $d_i = \sum_j a_{i,j}$. $\tilde{A}$ is referred to the original graph with self-loop and is defined to be $\tilde{A} = A + I$, where its degree matrix is $\tilde{D} = D + I$.

A. Graph Convolutional Networks

According to the spectral graph theorem [17], full graph convolution first involves eigendecomposition of graph Laplacian matrix L, and then performs Fourier transformation on the graph which defines as the multiplication of a signal x with a filter $g_\theta = diag(\theta)$ parameterized by $\theta \in \mathbb{R}^n$ in the Fourier domain:

$$g_\theta \star x = U g_\theta(\Lambda) U^T x, \tag{1}$$

where U is the matrix of eigenvectors of the symmetric normalized graph Laplacian $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$, with a diagonal matrix of its eigenvalues Λ and $U^T x$ being the graph Fourier transform of x. ChebNet [18] suggests that $g_\theta(\Lambda)$ could be well-approximated by a Chebyshev polynomial filter so that the equation (1) can be further approximated by the k-th order Chebyshev polynomial of the graph Laplacians. GCN [1] further simplifies the ChebNet to obtain the convolution operation as:

$$g_\theta \star x \approx \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) x. \tag{2}$$

Finally by adopting the renormalization trick $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ to alleviate the problem of exploding and vanishing gradients in deep neural networks, the Graph Convolutional Layers are generalized as:

$$H_{k+1} = \sigma(\hat{A} H_k W_k), \tag{3}$$

where σ is the ReLU nonlinear activation function and W_k denotes learnable weight matrix at k-th layer.

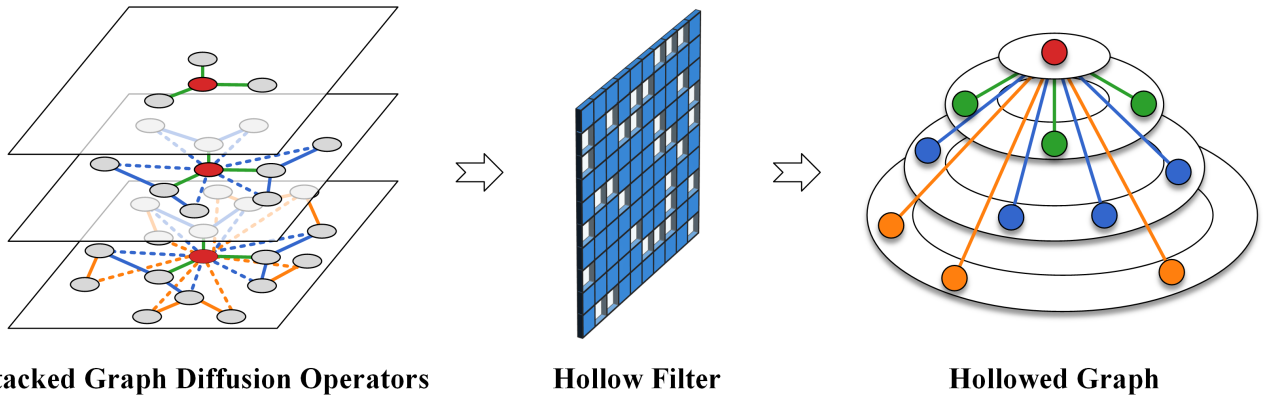

Fig. 1. Illustration of the over-densifying problem of the stacked graph diffusion operators and the concept of our proposed Graph Hollow Convolution. The red node is the center node and other colors indicate its corresponding neighbors and connections for different hops, where the recursive propagation of GCNs go through the solid lines. The stacked graph diffusion operators create redundant connections in the overall topology information, which are illustrate as the dash lines. Our method prunes such redundancy and prevents the graph diffusion operators from over-densifying.

B. Related Works

One layer of GCN only considers one-hop neighbors and GCN is able to learn global graph structure and aggregate information from multi-hop neighbors by stacking multiple layers. However, repeatedly stacking many layers make the representations of nodes from different classes indistinguishable, which leads to over-smoothing. SGC [19] simplifies GCN by directly applying a k-th power normalized adjacency matrix which corresponds to a low-pass filter on the spectral domain and removing the nonlinear transition functions but the softmax in the last layer. SIGN [20] extends SGC and is able to combine graph convolutional filters of different types and sizes.

There have been many attempts to overcome the over-smoothing issue in order to build deep GCNs. Some take the approach to regularize the original GCNs. DropEdge [3] can be considered as a data augmentation technique and PairNorm [4] serves as a normalization layer to boost the robustness of deep GCNs. Some other works tries to bring new architecture for GCNs. JKNet [21] adaptively adjusts the influence radius for each node as it selects different layer aggregations and [5] decouples the representation transformation and nodes propagation and adaptively balances the information from different layers. Recently, a few works focus on bringing new graph filter to resolving over-smoothing issue, since the graph filter of GCNs is equivalent to a low-pass filter [19]. [10] introduces Generalized PageRank to GCN and states its adaptively learned graph filter pass relevant high and low frequencies. FAGCN [22] designs a novel frequency adaptation graph convolutional network to adaptively combine the low-frequency and high-frequency signals.

III. Graph Hollow Convolution Networks

In this section, we first introduce our unique perspective view on the over-smoothing issue and provide detailed analysis based on empirical study. Then, we introduce our proposed framework Graph Hollow Convolution Networks with two key innovations: 1) Graph Hollow Convolutional layer and 2) Topological layer Fusion.

A. Analysis of Over-smoothing

GCNs incorporate both topology and feature information at each layer. In this section, we examine how the expressive power of each component varies along with the network depths by comparing the performance of the original GCN with two of its variants. And we further investigate the over-densifying issue with the topology information. We adopt two modifications on the original GCN implementation that one only incorporates topology information and the other only incorporates feature information. In the topology only setting, we replace original features with embeddings using xavier initialization [23] to remove the influence of the feature information, while in the feature only setting we replace the adjacency matrix with an identity matrix to remove the propagation mechanism. The experiments are conducted on Cora dataset with full-supervised setting as [24] in order to solely focus on over-smoothing without influences on limited label propagation. Furthermore, for better illustration purpose on clarifying our hypothesis about over-smoothing, we retrieve the largest connected component on Cora dataset for this experiment only which contains 2485 nodes out of the original 2708 nodes since the original Cora dataset is not a connected graph.

As Fig. 2(a) shows, the performance degradation of GCN starts at layer 9 where accuracy suffers a steer drop-off, and after layer 9 the original GCN suffers heavily from over-smoothing so that the representations lose expressive power and become indistinguishable. In the version which only incorporates feature information, we can hardly tell there is a major performance drop-off and observe a slight decrease with increasing network depths due to over-fitting which is caused by excessive amounts of parameters in deep layers. In the topology only setting, we discover even more severe performance degradation from over-smoothing than GCN, which supports our hypothesis that the expressive power loss in the

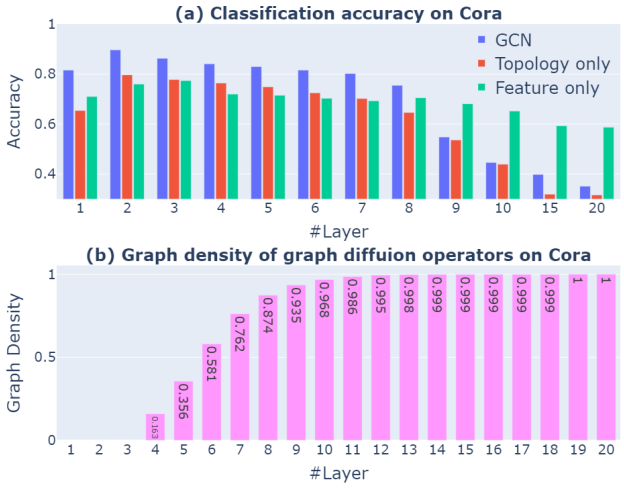

Fig. 2. (a): Test accuracy of original GCN and its two modifications with different network depths on Cora dataset. (b): Graph density of the graph diffusion operators of GCN with different network depth on Cora dataset.

topological part caused over-smoothing in GCN even with relative expressive feature information. In Fig. 2(b), we take a further look at the topology information that original GCN and the topology only version incorporated, which essentially is the graph diffusion operators. For undirected graphs, the graph density is defined as:

$$\text{Density} = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)}, \tag{4}$$

which measures the ratio of the number of edges with respect to the maximum possible edges [25]. The graph density indicates the severity of the over-densifying of the graph diffusion operators in deep layers. After layer 9, the graph diffusion operators are almost fully connected and from the topological view the nodes are already indistinguishable, which led to topological expressiveness loss. This trend of the graph density coincides with the steer performance drop of the GCN and its topology only variant. Along with the graph diffusion operators becoming over-densifying in deep layers, incorporating such inexpressive topology information in GCN causes the over-smoothing issue.

In shallow layers, both feature and topology information have their own unique expressive power. Incorporating them into a unified framework boosts the performance considerably. This is why GCNs achieves such success in the first place. However, the shallow architecture of GCNs limit topological information learned from only local graph structure rather than global, which impedes GCNs expressive power. Along with the increasing network depth, the topology information eventually loses its expressive power since most of the nodes have similar neighbor sets and become indistinguishable from the topological view. Incorporating topology information obtained from the over-densifying graph diffusion operators worsens the performance of GCN in deep layers. In order to tackle over-smoothing, we need topology information to retain its expressive power in deep layers by preventing the stacked graph diffusion operator from over-densifying. Given the fact that GCNs usually achieved the best performance at layer 2 or 3 with low graph density of the graph diffusion operators, we want the graph diffusion operators in deep layers to stay reasonably sparse in deep layers, too.

Algorithm 1 GHCN graph diffusion operator pruning algorithm

Input: Graph $G = (V, E)$ and corresponding adjacency matrix A; depth k; pruning function $\Phi(\cdot)$
Output: Pruned Graph diffusion operators $\hat{A}_{pruned}$

for $k = 0, 1, \ldots$ **do**
  Calculating k-hop adjacency matrix A^k.
  Get neighbor set $\mathcal{N}^k$ of A^k.
  if $k = 0$ **then**
    $\hat{A}^0_{pruned} = I$
  else
    $\hat{A}^k_{pruned} = \Phi(\hat{A}^k)$.
  end if
end for
return $\{\hat{A}^0_{pruned}, \hat{A}^1_{pruned}, \ldots, \hat{A}^k_{pruned}\}$

B. Graph Hollow Convolution

[3] proposes that randomly removing certain amounts of edges in each convolutional layer is able to reduce the convergence speed of over-smoothing and it also can be viewed as a method that preventing over-densifying, which verifies our hypothesis. Randomly removing edges prevents deep GCNs from over-smoothing, but the shallow models still outperform the deep ones. [21] proves that a k-layer GCN simulates k-step random walks on graph G with self-loop and adds residual connections in GCN approximately correspond to lazy random walks in which each step has a higher probability of staying at the current node. The over-smoothing issue of the stacking graph diffusion operator is equivalent to the random walks distribution of infinite step random walks converges to a stationary distribution [26]. While it has been proven that adding residual connections slows down the speed of convergence, theoretically over-smoothing still happens when k goes to infinity [6].

In order to overcome over-smoothing, we propose a novel graph convolutional layer denoted as Graph Hollow Convolution, which prunes the graph diffusion operators and directly applies large filters to aggregate from k-hop neighbors. We first define the pruning function $\Phi(\cdot)$ of the graph diffusion operator that prevents nodes that already propagated in the previous layer from propagating in the current layer:

$$A^k_{pruned} = \Phi(A^k) = \begin{cases} a^k_{i,j} & \text{if } \{j \in \mathcal{N}^k_i \mid j \notin \mathcal{N}^{k-1}_i\} \\ 0 & \text{else} \end{cases}, \tag{5}$$

where $\mathcal{N}^k_i$ represents the neighbor set of node i in graph A^k. Our pruning function can also be viewed as a way to remove

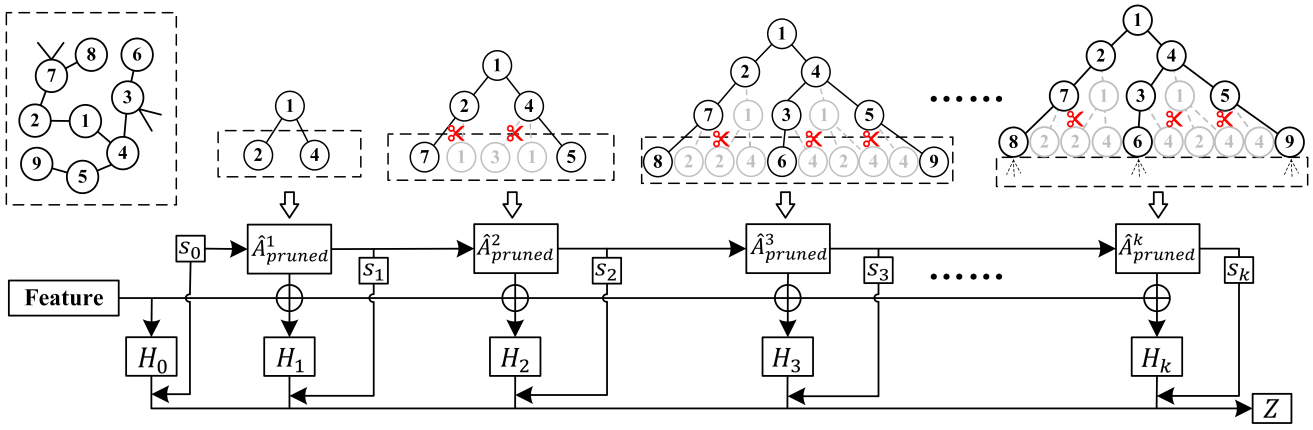

Fig. 3. The overall framework of our proposed GHCN with the illustration of pruning function on the graph diffusion operators on the top and the topological layer fusion at the bottom.

edges from the adjacency matrix at each layer. Compared to [3] removing certain amounts of edges randomly at each layer, our method serves as a more rigorous strategy based on topology information which removes edges in the current layer only if such connections exist in the previous layer. According to Algorithm 1, we are able to pre-compute the pruned graph diffusion operators served as the mechanisms of propagation outside the training process instead of computing with the training process that GCN operates on. And we adopt the same renormalization trick as GCN that $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. To retain topological expressiveness, our proposed pruning function forces the over-densifying graph diffusion operator to be reasonably sparse by deliberately filtering out the nodes propagated previously, which serves as a hollow filter.

[19] proves that stacking graph convolutional layers with local filters is practically equivalent to directly applying large filters and all non-linearities can be removed with the exception of the last layer. The Graph Hollow Convolution propagates information directly on k-hop neighbors with the pruned graph diffusion operator:

$$H_k = \hat{A}^k_{pruned} X W_k, \tag{6}$$

where W_k is the trainable weight matrix at k-layer. The final representations of nodes incorporate information from immediate neighbors to k-hop neighbors:

$$Z = \sigma([H_0, H_1, H_2, \ldots, H_k]), \tag{7}$$

where σ denotes non-linear activation function.

C. Topological Layer Fusion

Incorporating information effectively from different layers of GCNs and determining how much each layer contributes to the final representations are still open challenges. The original GCN only utilizes the output of the last layer as the final representations and work fine with shallow architecture. However, the approach of GCN fails to properly leverage information from different layers. The incapability of GCN when it comes to benefit from deep architecture is not only because of over-smoothing but also because of its improper strategy to incorporating information from different layers.

[21] first attempt to build deep GCNs with fusion strategies such as concatenation, max-pooling, and LSTM, which combine aggregations from different layers. [2], [6] both identify and prove through empirical studies that adding simple residual connections between layers relieves the problem effectively, which forces deep layers to maintain certain amounts of information from shallow layers. [5] implements an adaptive adjustment fusion strategy that enables the model to learn and adaptively balance the information from the local and global neighborhoods.

While the above fusion strategies have made adjustments to deep architecture, we propose that each node at each layer should have a different and adaptive contribution to the final representation:

$$\begin{aligned} Z &= \sigma(\mathrm{diag}(\alpha_0)H_0 + \cdots + \mathrm{diag}(\alpha_k)H_k) \\ \alpha_k &= \mathrm{softmax}(s_k) = \frac{\exp(s_k)}{\sum_k \exp(s_k)}, \end{aligned} \tag{8}$$

where $\alpha^k \in \mathbb{R}^{1\times n}$ denotes the normalized contribution coefficients for the k-th layer as diagonal matrix and $s^k \in \mathbb{R}^{1\times n}$ denotes as the score vector for each layer which is driven from the pruned graph diffusion operator at the k-th layer. The contribution of each node in each layer can be obtained by an initial score vector passed along with the pruned graph diffusion operators:

$$s_k = s_0 \prod_{k=0}^{k} \hat{A}^k_{pruned}, \tag{9}$$

where s_0 is the initial score vector. The initial score vector reflects the influence of each node that is usually set as all one vector $[1, 1, \ldots, 1] \in \mathbb{R}^{1\times n}$ which indicates equal influence at the start or is set manually according to the priori knowledge.

GCNs propagate nodes repeatedly to learn local structures and identify the importance of the neighbor nodes [12], while our proposed Graph Hollow Convolution only propagates each node only once. The key reason that topological layer fusion

TABLE I
THE STATISTICS OF THE DATASETS.

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2708	5429	1433	7
CiteSeer	3327	4723	3703	6
PubMed	19717	44338	500	3
Chameleon	2277	36101	2325	4
Amazon Computers	13381	245778	767	10
Amazon Photo	7487	119043	745	8
ogbn-arixv	169343	1166243	128	40

was proposed to incorporate with graph hollow convolution is that it was enabled to learn local structures and identify the importance of nodes based on the topological structure of the graph without propagating nodes repeatedly. The k-th hidden layer incorporates information from original features and the k-th pruned graph diffusion operator, where the pruned graph diffusion operators can be viewed as transition matrices that transform original features into different states according to k-hop neighborhoods propagated in the current layer. The influence score vector at each layer reflects how the influence scores of nodes pass along through the topological information incorporated in each layer. After topological layer fusion, we strengthen the final representations of nodes by balancing information from each layer based on the influence scores transited along with the topological graph structure. The final output of our model is:

$$Y = \mathrm{softmax}(\mathrm{MLP}(Z)), \tag{10}$$

which output probabilities of each class according to the downstream tasks. Note that our GHCN model is trained end-to-end, and all of the above steps are optimized together.

IV. EXPERIMENT

In this section, we conduct extensive and detailed experiments to evaluate our proposed framework GHCN. In the first place, we introduce the datasets chosen and the implementation setup for our experiments. Then, we compare our method to the state-of-the-art methods to illustrate the effectiveness of GHCN. Finally, we provide a detailed investigation on how GHCN relieves the issue of over-smoothing in deep architecture.

A. Datasets and Baselines

We conduct extensive experiments on 7 datasets of varied sizes and different domains. Cora, CiteSeer, and PubMed in Planetoid dataset [13] are representative citation network datasets where nodes and edges denote documents and their citation relationships, respectively. Chameleon in WikipediaNetwork dataset [14] is web network where nodes and edges represents web pages and hyperlinks, respectively. Computers and Photo in Amazon dataset [15] are segments of the Amazon co-purchase network where nodes are goods and edges denote that two goods are frequently bought together. Ogbn-arxiv in OGB dataset [16] is citation network between all Computer Science arXiv papers where each node is an arXiv paper and each edge indicates that one paper cites another one. The statistics of the datasets are detailed in Tab. 1.

We compare our method to the following baselines:

GCN [1] uses a layer-wise propagation that is based on a first-order approximation of spectral convolutions on graphs.

GAT [27] learns the edges weights at each layer based on node features.

SGC [19] simplifies GCN by directly applying a k-th power normalized adjacency matrix.

SIGN [20] combines graph convolution filters of different types and sizes.

DAGNN [5] decouples transformation and propagation in current graph convolution operations and adaptively balances the information from different layers.

GCNII [6] applies initial residual connection and identity mapping to GCN to deal with the over-smoothing problem.

FAGCN [22] learns the weight of an edge as the difference in the proportion of low-frequency and high-frequency signals.

B. Experimental Setup

We implement our method GHCN using Pytorch [28] and Pytorch Geometric [29]. In order to provide rigorous and fair comparisons, we follow the same dataset splits and training strategies. We use Adam SGD optimizer [30] with learning rate of 0.01 and early stopping on validation accuracy with a patience 100 epochs and we perform a grid search to tune hyperparameters for our model as follows: $k \in \{1 \sim 20\}$, dropout $\in \{0, 0.5, 0.6, 0.7, 0.8\}$, and weight decay $\in \{0, 5\mathrm{e}-3, 5\mathrm{e}-4, 5\mathrm{e}-5, 5\mathrm{e}-6\}$. Specifically, for Cora/CiteSeer/PubMed/Chameleon/Computers/Photo/ogbn-arxiv, the best performance of our model is obtained by: k = $8/20/9/10/15/10/16$, dropout = 0.5/0.8/0.5/0.6/0.6/0.6/0.6, and weight decay = $5\mathrm{e}{-4}/5\mathrm{e}{-3}/5\mathrm{e}{-5}/5\mathrm{e}{-6}/5\mathrm{e}{-5}/5\mathrm{e}{-5}/0$. The hyperparameters of baselines are set according to their original papers or their GitHub repository.

For the train/val/test split, we adopt the full-supervised setting in Planetoid datasets as [24] and Chameleon dataset as [31], and the semi-supervised settings in Amazon datasets as [15] and ognb-arxiv dataset as [16]. We report the mean classification accuracy of each methods after 10 runs.

C. Results

The results in Tab. 2 demonstrate that our method clearly outperforms the state-of-the-art methods across all datasets and achieves the new state-of-the-art performance. SGC and SIGN, which directly applying large graph filters instead of stacking graph convolutional layers, have similar performance as more traditional approaches like GCN and GAT. This also proves that stacking graph convolutional layers with local filters is practically equivalent to directly applying large filters. Three recent deep methods that aim to resolve the over-smoothing issue, DAGNN, GCNII, and FAGCN, do offer competitive advantages over the shallow architecture of other models, which indicates that there are still valuable information in further hop neighbors and GCNs could still benefit from deep

TABLE II
RESULTS OF CLASSIFICATION ACCURACY (%)

Model	Cora	CiteSeer	PubMed	Chameleon	Amazon Computers	Amazon Photo	ogbn-arxiv
GCN	85.7	73.7	88.1	28.1	82.6	91.2	71.7
GAT	86.4	74.3	87.6	42.9	78.0	85.7	73.2
SGC	86.7	75.6	68.9	47.2	83.4	91.5	69.3
SIGN	87.4	75.8	88.8	50.0	68.8	85.6	71.9
DAGNN	89.2	77.3	90.0	49.5	84.5	92.0	72.1
GCNII	88.5	77.1	90.3	62.5	78.2	86.8	72.7
FAGCN	89.0	80.1	90.4	59.2	82.5	92.5	OOM
GHCN	**90.4**	**80.9**	**91.3**	**69.7**	**87.1**	**93.1**	**73.8**
GHCN w/o prune	88.7	79.1	90.2	63.1	84.1	91.6	72.6
GHCN w/o fusion	89.9	80.5	90.9	66.8	84.8	92.0	73.0

TABLE III
RESULTS OF CLASSIFICATION ACCURACY (%) WITH DIFFERENT DEPTHS

Methods	Cora				CiteSeer				PubMed			
	#Layer/Hop											
	5	10	15	20	5	10	15	20	5	10	15	20
GCN	**85.2**	84.4	31.9	24.5	**72.5**	71.3	25.5	24.3	**86.0**	44.6	42.8	45.1
GAT	**86.0**	82.8	31.9	31.9	**74.0**	18.1	23.1	26.6	**87.2**	85.2	84.4	84.0
SGC	**86.6**	86.1	85.8	85.1	**75.5**	74.6	74.7	75.0	**68.9**	67.4	66.3	67.2
SIGN	86.6	86.9	**87.4**	87.0	**75.8**	74.9	74.1	73.7	88.6	**88.7**	86.6	84.6
DAGNN	88.5	**89.1**	88.0	88.2	**77.3**	77.2	77.0	76.6	88.3	**89.8**	87.9	87.8
GCNII	87.8	88.0	88.3	88.4	75.7	76.8	77.0	**77.1**	89.1	89.4	**90.0**	89.9
FAGCN	**89.0**	88.6	88.4	88.5	80.0	80.0	**80.1**	79.7	**90.2**	88.6	88.4	89.0
GHCN	88.7	89.6	**90.2**	89.8	80.3	80.6	80.7	**80.9**	90.0	90.6	90.5	**91.0**

Methods	Chameleon				Amazon Computers				Amazon Photo			
	#Layer/Hop											
	5	10	15	20	5	10	15	20	5	10	15	20
GCN	**27.8**	25.7	24.3	24.1	**77.7**	66.3	41.7	41.8	**89.5**	48.1	43.0	41.9
GAT	**42.7**	40.2	38.1	33.2	**75.4**	53.9	48.8	39.0	**85.1**	81.3	58.3	40.9
SGC	**46.8**	43.1	43.5	43.2	**81.3**	77.7	77.1	69.3	**89.7**	84.9	83.9	79.7
SIGN	48.6	**50.0**	49.3	48.4	69.5	**71.4**	70.5	69.6	84.3	84.6	85.3	**85.6**
DAGNN	**49.3**	48.0	47.8	47.4	**84.5**	83.9	83.4	83.3	**92.0**	90.3	90.1	89.7
GCNII	**62.4**	61.2	60.5	59.0	75.3	76.3	**77.9**	77.1	**86.6**	85.1	81.6	78.1
FAGCN	57.9	**59.1**	57.2	58.3	**82.5**	81.9	82.4	81.2	**92.3**	92.2	91.4	91.3
GHCN	67.1	**68.9**	68.6	67.8	86.0	86.7	**86.9**	86.3	92.6	**93.1**	92.4	91.5

architectures. Notably, our method achieves a 10% improvement on the Chameleon dataset and have better performance improvement on datasets with higher average degree, which verifies the effectiveness of our approach to resolving over-smoothing by preventing the graph diffusion operators from over-densifying.

D. Ablation Study

We conduct ablation study on two variants of GHCN according to the bottom part of Tab. 2. In the first version, we keep the original graph diffusion operators and implement topological layer fusion only. The results indicate that our fusion strategy had clear margin of improvement over the fusion strategy of GCNs and the concatenation of SIGN. The second variant still achieves competitive edge over all other baselines, which only implements Graph Hollow Convolution and uses concatenation instead of topological layer fusion. None of these two aspects of our proposed framework consistently outperform all baselines, while the variant only with graph hollow convolution has slightly better performance than the variant with topological layer fusion since the latter does not adjust to over-smoothing. With the collaboration of graph hollow convolution that retains topology expressiveness and topological layer fusion that learns graph local structure via the importance scores of nodes without propagating nodes repeatedly, our proposed framework is able to achieve the new state-of-the-art performance.

E. Depths Analysis

Furthermore, we investigate how over-smoothing affects each model along with different depths on all datasets with the same hyperparameters setting as above section except networks depths. Since some baselines have scalability issue for large graph datasets after layer 5, we exclude the ogbn-arxiv dataset in this section. As Tab. 3 shows, the performance of the models without adjustment to over-smoothing including

Fig. 4. Heatmap of the final representations of GCN, DAGNN and GHCN. The similarity in color indicates the similarity of the value. The x axis indicates dimension and y axis stands for node index.

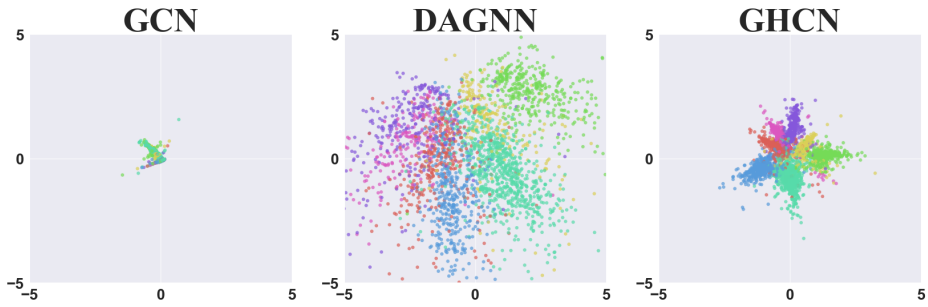

Fig. 5. MDS visualizations of the GCN, DAGNN, and GHCN. The colors represent different classes.

GCN and GAT takes a steer dive around layer 10 due to over-smoothing. Other baselines that make adjustments to over-smoothing have much better performance in deep architectures. Our method achieves the best performance in relative deep layers compared all baselines with a clear margin of improvement and is able to extract more information from deeper architecture. Note that for the Chameleon and Amazon dataset with much higher average degree than the other three datasets, all models are limited in relative shallow architecture which aligns with the hypothesis proposed by [5], [6] that nodes with higher degrees suffer more from over-smoothing. Our method also strives in learning representations with high expressive power in limited depths compared to other methods when over-smoothing happened fairly early on the densely connected graphs.

F. Case Study

To further demonstrate the performance of GHCN against the over-smoothing issue, we compare GHCN with other graph convolutional models in this section and provide detailed interpretation from the visualizations on their nodes representations and their corresponding labels separations . In particular, we train GHCN with the Cora dataset and utilize the embedding of the last layer as the node representations of each model. In addition to GCN, we select DAGNN from the models that deal with over-smoothing for comparison since it outperforms other models on Cora dataset.

1) Heat Map: In this part, we compare the embeddings of GHCN with other models in a more detailed perspective according to the heat map. 10 relatively separated nodes from different classes are selected according to the distance of representations from the 15 layer GCN. The colors of the heat maps represent the corresponding values of the embeddings, and the stronger diversification of the colors indicates better expressiveness of the model. Even though we select nodes that are relatively remote from the nodes representations of GCN, the heat map of GCN shows that its most separate nodes still have strong similarity in each dimension which is a typical sign of over-smoothing. The separations in node representations of DAGNN and GHCN according to the heat maps suggest that both methods alleviate the over-smoothing problem. However, DAGNN is inexpressive in certain dimensions which may compromise the performance. The representations of the GHCN are diversified in almost all dimensions and thus richer in expressive power.

2) Multi-dimensional Scaling: In the final part, we make use of Multi-dimensional Scaling (MDS), a method that visualizes the level of similarity of individuals in a dataset. It maps data from high dimension space to low dimension space and preserves the distance of each paired data as well as possible, which enables us to show the dispersion of node representations on a two-dimensional plane. The classes of the nodes are indicate by different colors in Fig. 5. Specifically, we show the MDS visualization of three models with 15 layers. We can observe that the node representations of GCN are all squeezed together and are difficult to separate, which is a phenomenon of over-smoothing. Though the representations of DAGNN seem to be scattered with larger distance between nodes compared to GHCN, the boundaries between each class are less clear than our method. Overall, our method not only overcomes the over-smoothing issue but also eases the job for the classifier.

V. Conclusion

In this paper, we further investigate the performance degradation of deep GCNs and identify the key factor of over-smoothing which is the topological expressiveness loss when the stacked graph diffusion operators are over-densifying in

deep layers. According to our analysis, we propose Graph Hollow Convolution Network with two key innovations: 1) a dedicatedly designed convolutional layer that applies a hollow filter on the stacked graph diffusion operator to retain the expressiveness of the topological information in deep layers; 2) the topological layer fusion that further exploits topology information without propagating nodes repeatedly. Extensive experiments on benchmark datasets prove the effectiveness of our method.

REFERENCES

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

[2] G. Li, M. Müller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *The IEEE International Conference on Computer Vision*, 2019.

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

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

[5] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[6] Z. W. Ming Chen, B. D. Zengfeng Huang, and Y. Li, "Simple and deep graph convolutional networks," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[7] L. Yang, C. Wang, J. Gu, X. Cao, and B. Niu, "Why do attributes propagate in graph convolutional neural networks?" in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[8] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

[10] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *International Conference on Learning Representations*, 2021.

[11] O. Angel, J. Friedman, and S. Hoory, "The non-backtracking spectrum of the universal cover of a graph," *Transactions of the American Mathematical Society*, 2008.

[12] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

[13] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33nd International Conference on Machine Learning*, 2016.

[14] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-Scale Attributed Node Embedding," *Journal of Complex Networks*, vol. 9, no. 2, 2021.

[15] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2019.

[16] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[17] D. A. Spielman, "Spectral graph theory and its applications," in *48th Annual IEEE Symposium on Foundations of Computer Science FOCS 2007*. IEEE Computer Society, 2007.

[18] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016.

[19] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[20] F. Frasca, E. Rossi, D. Eynard, B. Chamberlain, M. Bronstein, and F. Monti, "Sign: Scalable inception graph neural networks," in *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

[21] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[22] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[24] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," in *6th International Conference on Learning Representations*, 2018.

[25] T. F. Coleman and J. J. Moré, "Estimation of sparse jacobian matrices and graph coloring blems," *SIAM Journal on Numerical Analysis*, vol. 20, no. 1, pp. 187–209, 1983.

[26] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *7th International Conference on Learning Representations*, 2019.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[29] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[31] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *8th International Conference on Learning Representations*, 2020.