# Long Short-Term Graph Memory Against Class-imbalanced Over-smoothing

Liang Yang[*]
Jiayi Wang[*]
yangliang@vip.qq.com
jayeew@qq.com
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China

Tingting Zhang
101101964@seu.edu.cn
College of Command and Control Engineering
Army Engineering University
Nanjing, China

Dongxiao He[†]
hedongxiao@tju.edu.cn
College of Intelligence and Computing
Tianjin University
Tianjin, China

Chuan Wang
wangchuan@iie.ac.cn
State Key Laboratory of Information Security
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China

Yuanfang Guo
andyguo@buaa.edu.cn
School of Computer Science and Engineering
Beihang University
Beijing, China

Xiaochun Cao
caoxiaochun@mail.sysu.edu.cn
School of Cyber Science and Technology, Shenzhen Campus
Sun Yat-sen University
Shenzhen, China

Bingxin Niu
niubingxin666@163.com
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China

Zhen Wang
w-zhen@nwpu.edu.cn
OPtics and ElectroNics (iOPEN),
School of Cybersecurity
Northwestern Polytechnical University
Xi'an, China

## ABSTRACT

Most Graph Neural Networks (GNNs) follow the message-passing scheme. Residual connection is an effective strategy to tackle GNNs' over-smoothing issue and performance reduction issue on non-homophilic networks. Unfortunately, the coarse-grained residual connection still suffers from class-imbalanced over-smoothing issue, due to the fixed and linear combination of topology and attribute in node representation learning. To make the combination flexible to capture complicated relationship, this paper reveals that the residual connection needs to be node-dependent, layer-dependent, and related to both topology and attribute. To alleviate the difficulty in specifying complicated relationship, this paper presents a novel perspective on GNNs, i.e., the representations of one node in different layers can be seen as a sequence of states. From this perspective, existing residual connections are not flexible enough for sequence modeling. Therefore, a novel node-dependent residual connection, i.e., Long Short-Term Graph Memory Network (LSTGM) is proposed to employ Long Short-Term Memory (LSTM), to model the sequence of node representation. To make the graph topology fully employed, LSTGM innovatively enhances the updated memory and three gates with graph topology. A speedup version is also proposed for effective training. Experimental evaluations on real-world datasets demonstrate their effectiveness in preventing over-smoothing issue and handling networks with heterophily.

[*]Both authors contributed equally to this research.
[†]Corresponding author.

## CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Networks → Network algorithms**.

## KEYWORDS

Graph Neural Networks, Long Short-Term Memory Networks, Deep Models

**ACM Reference Format:**
Liang Yang, Jiayi Wang, Tingting Zhang, Dongxiao He, Chuan Wang, Yuanfang Guo, Xiaochun Cao, Bingxin Niu, and Zhen Wang. 2023. Long Short-Term Graph Memory Against Class-imbalanced Over-smoothing. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612566

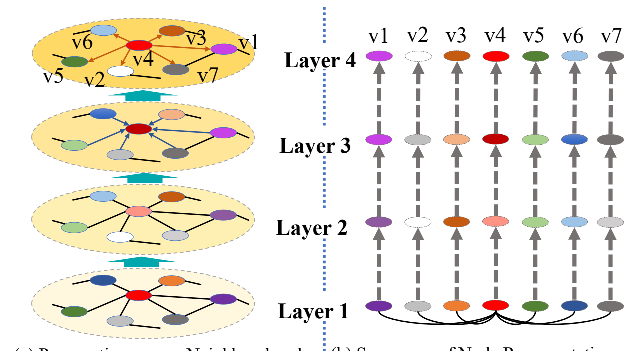

v6
v3
v1
v4
v5
v2
v7
Layer 4
Layer 3
Layer 2
Layer 1
v1 v2 v3 v4 v5 v6 v7
(a) Propagation among Neighbourhoods
(b) Sequences of Node Representations

**Figure 1: Different perspectives of GNNs. (a) Most existing GNNs focus on propagation among neighbourhoods. The obtained representations in previous layer are taken as the message to be propagated in next layer. (b) This paper alternatively considers the representations of one node in different layers as a sequence of states. Thus, the LSTM is employed to model these sequences.**

## 1 INTRODUCTION

Graph Neural Networks (GNNs) [32, 41], especially Graph Convolutional Networks (GCNs), which originated from spectral graph theory, have became a kind of powerful tools for modeling irregular data in many computer vision tasks, such as sense graph generation, skeleton-based action recognition [10], subspace clustering [15] and multi-view clustering [35]. Many variants of GCNs are designed from the spatial perspective by following the message-passing scheme [7]. These models propagate messages among neighbourhoods to achieve local smoothing. However, vanilla GCNs [16] posses two serious issues, i.e., over-smoothing issue [19, 34] by stacking multiple layers and performance reduction issue [27, 42] on non-homophilic networks.

Residual connection, which is proposed to tackle vanishing gradient problem in CNNs [11], is also effective to alleviate the two issues in GCNs [4]. Different from vanilla GCN, which takes the obtained node representations in current layer as the message to be propagated in next layer, residual connection enhances the message with representations in previous layers via simple operations, such as summation and concatenation [18, 34].

Unfortunately, the coarse-grained residual connection cannot essentially solve the two issues mentioned above. Factually, the over-smoothing issue remains, and tends to be class-imbalanced. In Figure 2, the recalls of GCN with residual connection are given in class-wise with different model depths. As the model goes deeper, the recall of Class 1 is significantly higher than others. This can be attributed to that many nodes are misclassified into this class. Therefore, different nodes possess distinguishing smoothing tends, and thus should not be handled in unified strategy.

Observations mentioned above motivate us to explore a fine-grained residual connection. Recent progress explains this enhancement as seeking representations to compromise between node attribute and graph topology from the perspective of optimization [25, 37, 43]. Actually, the relationship between topology and node attribute is complicated. Therefore, to obtain robust node [23] representation, the combination of them should be flexible and nonlinear. From the optimization perspective of GNNs [25, 37, 43], this paper reveals that the residual connection needs to be *node-dependent, layer-dependent, and related to both topology and attribute* to make the combination flexible (see Section 3 for details). Unfortunately, it is not trivial to *infer* the weights for residual connection, since it may be challenging to specify the complicated nonlinear functions between topology and attribute.

To alleviate this difficulty, this paper presents a novel perspective on GNNs, i.e., **the representations of one node in different layers can be seen as a sequence of states** as shown in Figure. 1b. Different positions on the sequence represent different hop information for the same node. Thus, the residual connection can be seen as an embedding strategy that compresses past sequential information into the current hidden state. Therefore, Long Short-Term Memory (LSTM) [12], which is proven effective in tackling vanishing and exploding gradient problems in sequence modeling, is employed to model the sequence of node representation. Benefiting from the different gates, LSTM compresses different layers of information into the current hidden state in a nonlinear and automatic way, which helps model the interaction relation among different hops.

However, the simple combination of LSTM and GNN can not make graph topology fully employed. Therefore, the first two requirements of residual connection are met but not the last. To tackle this drawback, Long Short-Term Graph Memory Network (LSTGM) is proposed by enhancing the updated memory and three gates with graph topology. By doing so, the vanilla memory state is upgraded to the graph memory state. Besides, the gate mechanism can regularize the node-level optimization process, which benefits tackling the class-imbalanced over-smoothing issue. Specifically, based on the long short-term graph memory, topology-enhanced gates can stop the training of nodes belonging to the tail class, i.e. the class with fewer nodes, while keeping the training of others.

The main contributions are summarized as follows:

- We observed the existence of class-imbalanced over-smoothing in GNN with vanilla residual connection.
- We theoretically reveal that the flexible combination of topology and attribute should be met by node- and layer-dependent residual connection.
- We propose an efficient Long Short-Term Graph Memory Network (LSTGM) to enhance updated memory and three gates in LSTM with graph topology.
- We experimentally demonstrate the superiorities of the proposed LSTGM.

## 2 PRELIMINARIES

This section presents the notations used in this paper, followed by the preliminaries on graph neural networks.

### 2.1 Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with node set $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ and edge set $\mathcal{E}$, where $N$ is the number of nodes. The topology of graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A} = [a_{ij}] \in \{0,1\}^{N\times N}$, where $a_{ij} = 1$ if and only if there exists an edge $e_{ij} = (v_i, v_j)$ between nodes $v_i$ and $v_j$. The degree matrix $\mathbf{D}$ is a diagonal matrix with diagonal element $d_i = \sum_{i=1}^{N} a_{ij}$ as the degree of node $v_i$. $\mathcal{N}(v_i) = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ stands for the neighbourhoods of

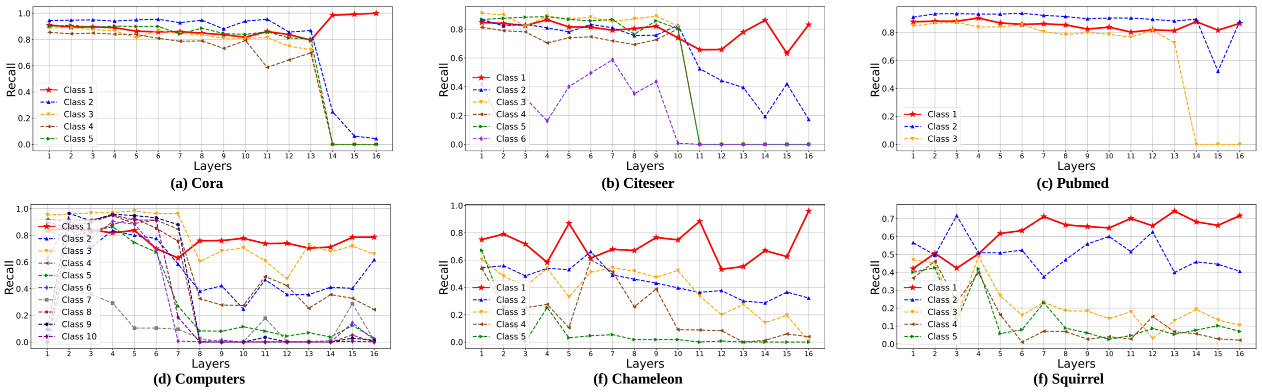


**Figure 2: Class-wise recalls of GCN with residual connection as model goes deeper. For clarity, the categories are listed in descending order of the number of nodes they contain, i.e., Class 1 (in red) is the largest class.**

node $v_i$. $\mathbf{X} \in \mathbf{R}^{N \times F}$ and $\mathbf{H} \in \mathbf{R}^{N \times F'}$ denote the collections of node attributes and representations with the $i^{th}$ rows, i.e., $\mathbf{x}_i \in \mathbb{R}^F$ and $\mathbf{h}_i \in \mathbb{R}^{F'}$, corresponding to node $v_i$, where $F$ and $F'$ stand for the dimensions of attribute and representation.

## 2.2 Graph Neural Networks

Although existing graph neural networks are proposed from the perspectives of spectral and spatial, respectively, most of them follow the message passing scheme [7] based on the connection between these two perspectives [1], such as GCN [16], SGC [31]. The graph convolutional layers of GCN, SGC are as follows.

$$\text{GCN} \quad \mathbf{H}^{(t+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(t)}\mathbf{W}), \quad \mathbf{H}^{(0)} = \mathbf{X} \tag{1}$$

$$\text{SGC} \quad \mathbf{H}^{(t+1)} = \tilde{\mathbf{A}}\mathbf{H}^{(t)}, \quad \mathbf{H}^{(0)} = \mathbf{X} \tag{2}$$

where $\tilde{\mathbf{A}} = \bar{\mathbf{D}}^{-\frac{1}{2}}\bar{\mathbf{A}}\bar{\mathbf{D}}^{-\frac{1}{2}}$ with $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the symmetric normalized adjacency matrix. However, these classical GNNs tend to be over-smoothing when multiple layers are employed. It makes the expressive ability of node representations and the performance significantly drop. Residual Connection is a widely-used strategy to tackle over-smoothing issue. Residual connection strategy takes the representations of previous layers as the input for the following layers, such as DeepGCNs [18] and JKNet [34]. Initial residual connection is a specific variant of residual connection, which feeds both representations from previous layer and original node attribute as the input of next layer, such as APPNP [17] and GCNII [4].

$$\text{APPNP} \quad \mathbf{H}^{(t+1)} = (1-\alpha)\tilde{\mathbf{P}}\mathbf{H}^{(t)} + \alpha\mathbf{X} \tag{3}$$

$$\begin{aligned}\text{GCNII} \quad \mathbf{H}^{(t+1)} = & \left((1-\alpha_t)\sigma(\tilde{\mathbf{A}}\mathbf{H}^{(t)} + \alpha_t\mathbf{X}\right) \\ & \times((1-\beta_t)\mathbf{I} + \beta_t\mathbf{W})\end{aligned} \tag{4}$$

where $\tilde{\mathbf{P}} = \bar{\mathbf{D}}^{-1}\bar{\mathbf{A}}$ is the asymmetric normalized adjacency matrix. $\beta_t$ is the weight for identity mapping to alleviate the overfitting issue. The weights for residual connections, i.e., $\alpha$ and $\alpha_t$, are identical for all.

## 2.3 Sequential modeling

Recurrent models such as LSTM are not the first time introduced to graph modelling. There are mainly two kinds of strategies to combine LSTM and GNN. The first strategy modifies the network topology to control the aggregation. GraphSAGE [9] samples and reorders neighbour nodes, then follows an LSTM unit as information aggregation. The second strategy keeps intermediate node embedding and uses LSTM as a pooling method. JKNet [34] employs bidirectional LSTM to learn layer-wise attention score, while Geniepath [22] propagates the outputs of LSTM and feeds them to the next LSTM unit. The second strategy, more similar to the proposed LSTGM, can be formulated as LSTM-regularized GNNs. Furthermore, this kind of GNN is shown in Figure. 3(c). However, the backbone GNN and the employed LSTM are two individual components, and the graph topology can not be fully employed by the LSTM.

## 3 OBSERVATIONS AND ANALYSIS

**Observation:** In Figure 2, the recalls of GCN with residual connection are given in class-wise with different model depths. For clarity, the categories are listed in descending order of the number of nodes they contain, i.e., Class 1 is the largest class. As the model goes deeper, the recall of Class 1 is significantly higher than others. At the same time, the recalls of other classes remarkably drop. The more significant the difference in the number of nodes of each category is, the more pronounced the class-imbalanced over-smoothing is. This can be attributed to that many nodes from other classes are misclassified into Class 1. This indicates that nodes in different classes possess different smoothing trends. Therefore, different nodes possess distinguishing over-smoothing tends, and thus should not be handled in unified strategy, when their categories are unknown.

Residual connections are widely known to alleviate the over-smoothing issue slightly. However, when the superficial problem is deepened into a deep-seated one, i.e. from over-smoothing to class-imbalanced over-smoothing, it is necessary to redesign the residual connection strategy. Figure 2 reminds us that class-imbalanced phenomenon is universal in homophily and heterophily networks. Thus a good residual connection strategy should tackle the class-imbalanced over-smoothing problems and perform well both in homophily and heterophily networks. Then the direction to explore is what properties such a residual connection should be met.

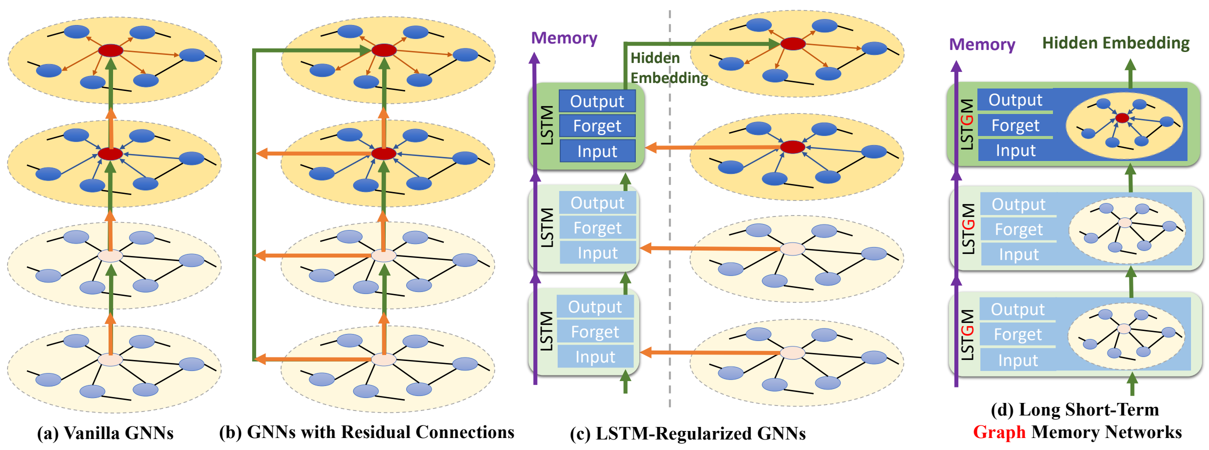

Memory
Hidden Embedding
Output
Forget
Input
LSTM
LSTGM
(a) Vanilla GNNs
(b) GNNs with Residual Connections
(c) LSTM-Regularized GNNs
(d) Long Short-Term Graph Memory Networks

**Figure 3: Comparisons between Vanilla GNNs, GNNs with residual connections, the proposed LSTM-regularized GNNs and Long Short-Term Graph Memory Networks. (a) Vanilla GNNs directly propagate the obtained node representations in last layer. (b) GNNs with residual connections combine representations in previous layers as message for all nodes in a uniform manner. (c) LSTM-regularized GNNs specify messages for different nodes from their representations in previous layers via LSTM. LSTM-regularized GNNs are equivalent to only employing topology information in the output gate in the LSTM. (d) Long Short-Term Graph Memory Networks (LSTGM) enhance LSTM-regularized GNNs by integrating topology information into input gate, forget gate and output gate. Refer to Figure. 4 for the detail.**

**Anlysis:** Recently, AUC optimization has achieved great success on long-tailed classification [38] and obtain state-of-the-art performance on complicated tasks such as adversarial training [13] and performance-constrained optimization [39]. Moreover, [3] present a first-trail to introduce topology-aware AUC optimization on Graph. Besides, some works interpret and unify graph convolutional networks from the perspective of the numeral optimization [25, 37, 43]. Specifically, they show that the graph convolution with residual connection, i.e. Eq. (3), is to minimize the following objective function via gradient descent

$$\begin{aligned} C &= ||\mathbf{X}-\mathbf{H}||_F^2 + \lambda \mathrm{tr}\left(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H}\right) \\ &= \sum_{i=1}^{N}||\mathbf{x}_i-\mathbf{h}_i||_2^2 + \frac{\lambda}{2}\sum_{i=1}^{N}\sum_{j\in N(i)}\tilde{a}_{ij}||\mathbf{h}_i-\mathbf{h}_j||_2^2, \end{aligned} \tag{5}$$

where $\tilde{a}_{ij}$ represents the element of matrix $\tilde{\mathbf{A}}$, while $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$ represents the Laplacian matrix of adjacency matrix $\tilde{\mathbf{A}}$. The first term stands for the distance between the node embeddings $\mathbf{h}_i$'s and attributes $\mathbf{x}_i$'s, while the second term stands for the distance between the representations of two connected nodes $\mathbf{h}_i$ and $\mathbf{h}_j$. The gradient of $C$ with respected to $\mathbf{h}_i$ is

$$\begin{aligned} \frac{\partial C}{\partial \mathbf{h}_i} &= \frac{\partial||\mathbf{x}_i-\mathbf{h}_i||_2^2}{\partial \mathbf{h}_i} + \frac{\lambda}{2}\frac{\partial\sum_{j\in N(i)}\tilde{a}_{ij}||\mathbf{h}_i-\mathbf{h}_j||_2^2}{\partial \mathbf{h}_i} \\ &= (\mathbf{x}_i-\mathbf{h}_i) + \lambda\left(\sum_{j\in N(i)}\tilde{a}_{ij}(\mathbf{h}_i-\mathbf{h}_j)\right). \end{aligned} \tag{6}$$

By setting the gradient as zero, the updating rule can be obtained as

$$\mathbf{h}_i^{(t+1)} = \alpha\sum_{j\in N(i)}\tilde{a}_{ij}\mathbf{h}_j^{(t)} + \beta\mathbf{x}_i, \tag{7}$$

where $\alpha$ and $\beta$ are the node-independent parameters to balance the impacts from its attribute $\mathbf{x}_i$ and representations of its neighbourhods. By comparing Eq. (7) with Eq. (3), graph convolutional operation can be seen as the gradient descent of objective function Eq. (5), and $\beta\mathbf{x}_i$ in Eq. (7) can be regarded as the initial residual connection.

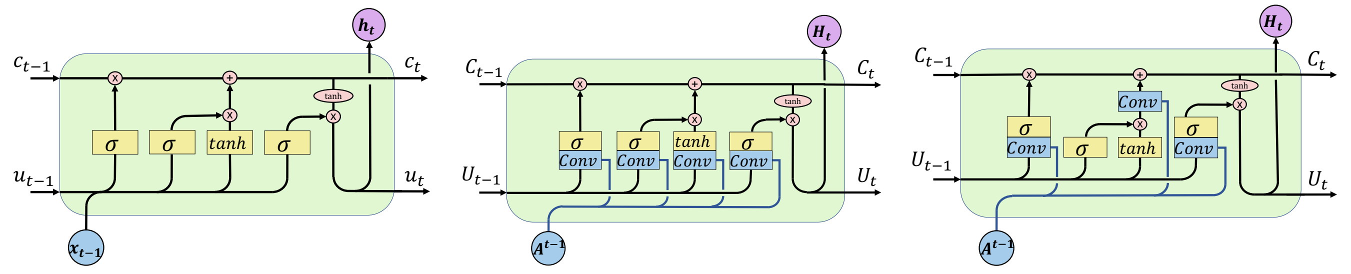
However, one remarkable drawback of the objective function in Eq. (5) is the *linear combination* of the two terms. Therefore, the widely-used graph convolutional operation also inherits this drawback. Actually, the combination of the impacts from topology and attribute may be complicated. To this end, Eq. (5) can be generalized to

$$\mathcal{F} = \sum_{i=1}^{N} f\left(||\mathbf{x}_i-\mathbf{h}_i||_2^2\right) + \frac{\lambda}{2}\sum_{i=1}^{N} g\Big(\sum_{j\in N(i)}\tilde{a}_{ij}||\mathbf{h}_i-\mathbf{h}_j||_2^2\Big), \tag{8}$$

where $f(\cdot)$ and $g(\cdot)$ represent the nonlinear function to combine the impacts from attribute and topology. The gradient of $\mathcal{F}$ with respected to $\mathbf{h}_i$ is

$$\begin{aligned} &\frac{\partial f(t_i)}{\partial t_i}\frac{\partial||\mathbf{x}_i-\mathbf{h}_i||_2^2}{\partial \mathbf{h}_i} + \frac{\partial g(s_i)}{\partial s_i}\frac{\lambda}{2}\frac{\partial\sum_{j\in N(i)}\tilde{a}_{ij}||\mathbf{h}_i-\mathbf{h}_j||_2^2}{\partial \mathbf{h}_i} \\ &\qquad = \gamma_i(\mathbf{x}_i-\mathbf{h}_i) + \lambda\eta_i\left(\sum_{j\in N(i)}\tilde{a}_{ij}(\mathbf{h}_i-\mathbf{h}_j)\right), \end{aligned}$$

where $t_i = ||\mathbf{x}_i-\mathbf{h}_i||_2^2$ represents the distance between node representation and attribute, while $s_i = \sum_{j\in N(i)}\tilde{a}_{ij}||\mathbf{h}_i-\mathbf{h}_j||_2^2$ stands for the distance of representations between nodes and its neighbourhoods. By comparing with Eq. (6), $\gamma_i = \frac{\partial f(t_i)}{\partial t_i}$ and $\eta_i = \frac{\partial g(s_i)}{\partial s_i}$ are node-dependent weights, which balance the impacts from two terms. Taking $f(x) = g(x) = \sqrt{x}$ as an example, $\gamma_i = \frac{1}{\sqrt{t_i}}$ and $\eta_i = \frac{1}{\sqrt{s_i}}$ stand for the reconstruction errors of attribute and topology from the representation of node $v_i$, respectively. Similar to



**(a) Vanilla LSTM Unit** **(b) LSTGM Unit** **(c) Speedup LSTGM Unit**

**Figure 4: From LSTM to the proposed Long Short-Term Graph Memory Networks (LSTGM). (a) The vanilla LSTM. (b) The proposed Long Short-Term Graph Memory Networks (LSTGM). LSTGM enhances input, forget and output gates with graph topology information. (c) The LSTGM speedup. Two convolutional operations are combined.**

updating rule in Eq. (7), the updating rule for Eq. (8) is

$$\mathbf{h}_i^{(t+1)} = \alpha_i^{(t)} \sum_{j \in N(i)} \tilde{a}_{ij} \mathbf{h}_j^{(t)} + \beta_i^{(t)} \mathbf{x}_i, \tag{9}$$

where $\alpha_i^{(t)}$ and $\beta_i^{(t)}$ are node- and iteration-dependent weights, which is related to reconstruction errors of $t_i^{(t)} = ||\mathbf{x}_i - \mathbf{h}_i^{(t)}||_2^2$ and $s_i^{(t)} = \sum_{j \in N(i)} \tilde{a}_{ij} ||\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)}||_2^2$. Note that the iteration in updating rule (Eq. (9)) is the layer in GNNs (Eqs. (1)-(4)). Especially, the weight $\beta_i^{(t)}$ for residual connection should possess three demanded properties.

- The weights should be node-dependent instead of identity for all nodes.
- The weights should be layer-dependent, since it is the function of representations in the previous layers.
- The weights should be related to both graph topology and node attributes.

Unfortunately, it is not trivial to *infer* the weights $\alpha_i^{(t)}$ and $\beta_i^{(t)}$ as shown above, since it may be difficult to specify the complicated nonlinear functions $f(\cdot)$ and $g(\cdot)$. To alleviate this issue, this paper tends to *learn* them via a highly expressive and robust model in next section.

## 4 METHODOLOGY

As discussed in previous section, flexible weights for residual connection may benefit modeling the complicated relationship between topology and attribute on node representation learning. To this end, this section combines Long Short-Term Memory (LSTM) in GNNs to learn the flexible residual connection.

### 4.1 Long Short-Term Graph Memory Network

As shown in Figure. 3(b), vanilla residual connection often combines representations from previous layers for propagation in next layer in a uniform manner, such as summation and concatenation. Note that **the representations of one node in different layers, i.e., $\{\mathbf{h}_i^{(t)}\}_{t=1}^T$, can be seen as a sequence of states**. From this perspective, vanilla residual connection can be regarded as a simple sequence model, which predicts next state with previous states via a fixed simple strategy. Unfortunately, existing simple strategies, such as summation and concatenation, are not flexible and robust [24] for sequence modeling. Thus, this section employs LSTM to model the sequence of node representation. LSTM is proved to be effective in tackling vanishing and exploding gradient problems in long range dependence modeling.

However, simple combination of LSTM and GNN is imperfect. As shown in Figure.3(c), the backbone GNN and the employed LSTM are two individual components, which are bridged by the node representations. This causes the specific residual connections to be only based on the node representations in previous layers, and the graph topology can not be fully employed by the LSTM. Therefore, the first two requirements for the residual connection are met but not the third.

To alleviate this issue, Long Short-Term Graph Memory Network (LSTGM) is proposed by enhancing the updated memory and three gates with graph topology as shown in Figure.3(d). The vanilla LSTM unit shown in Figure. 4a consists of a memory component $\mathbf{c}^{(t)}$ and a hidden component $\mathbf{u}^{(t)}$. Then the architecture of LSTGM unit is shown in Figure. 4b. Different from LSTM, which models i.i.d. sequences, the proposed LSTGM tends to model the sequence of graph data. To this end, LSTGM takes adjacency matrices from different orders, i.e., $\mathbf{A}^t$, as input. Thus, the three gates and updated memory can be enhanced with these adjacency matrices as

$$\begin{aligned} \mathbf{F}^{(t)} &= \sigma(\mathbf{A}^t \mathbf{U}^{(t-1)} \mathbf{W}_f), & (10)\\ \mathbf{I}^{(t)} &= \sigma(\mathbf{A}^t \mathbf{U}^{(t-1)} \mathbf{W}_i), & (11)\\ \mathbf{O}^{(t)} &= \sigma(\mathbf{A}^t \mathbf{U}^{(t-1)} \mathbf{W}_o), & (12)\\ \tilde{\mathbf{C}}^{(t)} &= \tanh(\mathbf{A}^t \mathbf{U}^{(t-1)} \mathbf{W}_c), & (13) \end{aligned}$$

where $\mathbf{F}^{(t)}$, $\mathbf{I}^{(t)}$, $\mathbf{O}^{(t)}$ are the three gates for all nodes, while $\tilde{\mathbf{C}}^{(t)}$ is the updated memory for all nodes. Based on this, the memory and hidden state can be formulated as

$$\begin{aligned} \mathbf{C}^{(t)} &= \mathbf{F}^{(t)} \otimes \mathbf{C}^{(t-1)} + \mathbf{I}^{(t)} \otimes \tilde{\mathbf{C}}^{(t)}, & (14)\\ \mathbf{U}^{(t)} &= \mathbf{O}^{(t)} \otimes \mathbf{C}^{(t)}, & (15) \end{aligned}$$

where $\otimes$ denotes element-wise product. LSTGM needs four graph convolution operations and some of them are redundancy. Therefore, we give a speedup version for efficient.

**Speedup:** To update the memory, both $\mathbf{I}^{(t)}$ and $\tilde{\mathbf{C}}^{(t)}$ needs perform graph convolution. Therefore, by exchanging the order of nonlinear mapping and graph convolution, Eqs. (11), (13) and (14)

Table 1: Datasets statistics

| Dataset | Cora | Citeseer | Pubmed | Computers | Photo | Chameleon | Squirrel | Actor | Texas | Cornell |
|---|---|---|---|---|---|---|---|---|---|---|
| # Nodes | 2,708 | 3,327 | 19,717 | 13,752 | 7650 | 2,277 | 5,201 | 7,600 | 183 | 183 |
| # Edges | 5,429 | 4,732 | 44,338 | 245,861 | 119,081 | 36,101 | 217,073 | 33,544 | 309 | 295 |
| # Features | 1,433 | 3,703 | 500 | 767 | 745 | 2,325 | 2,089 | 931 | 1,703 | 1,703 |
| # Classes | 7 | 6 | 3 | 10 | 8 | 5 | 5 | 5 | 5 | 5 |
| Homphily Rate | 0.83 | 0.71 | 0.79 | 0.79 | 0.84 | 0.25 | 0.22 | 0.24 | 0.06 | 0.11 |

Table 2: Large-scale graph statistics

| Dataset | Ogbn-products | Ogbn-mag |
|---|---|---|
| # Nodes | 2,449,029 | 1,939,743 |
| # Edges | 61,859,140 | 21,111,007 |
| # Features | 100 | 128 |
| # Classes | 47 | 349 |
| # Train/Val/Test | 10%/2%/88% | 85%/9%/6% |

can be reformulated as

$$\mathbf{I}^{(t)} = \sigma(\mathbf{U}^{(t-1)}\mathbf{W}_i), \tag{16}$$

$$\tilde{\mathbf{C}}^{(t)} = \tanh(\mathbf{U}^{(t-1)}\mathbf{W}_c), \tag{17}$$

$$\mathbf{C}^{(t)} = \mathbf{F}^{(t)} \otimes \mathbf{C}^{(t-1)} + \mathbf{A}^{t}(\mathbf{I}^{(t)} \otimes \tilde{\mathbf{C}}^{(t)}). \tag{18}$$

The architecture for speedup LSTGM is shown in Figure. 4c. It only needs three graph convolution operations. In practice, the first and third convolution operation from left to right have the same input, hence the convolution operation can be further reduced to two. Furthermore, the feature transformation in convolution can be removed for simplicity.

**Remark:** The proposed LSTGM is very different from Spatial Temporal Graph Neural Networks (STGNNs), such as Graph-WaveNet [33], CGCN [40], ST-GCN [36], TSSRGCN [5] and ASTGCN [8] from both solved problem and date structure. LSTGM focuses on static graph and tends to alleviate the weakness in residual connection, while STGNNs focus on dynamic graph and model spatial temporal changes. To the best of our knowledge, LSTGM is the first to model representation in layers via sequence model.

## 5 EVALUATIONS

### 5.1 Experimental Setup

**Real-world Datasets.** The proposed LSTGM is validated on 10 networks, whose statistics are shown in Table 1. These 10 networks can be categorized into 4 categories. **Citation networks**: Cora, Citeseer, and Pubmed are the standard citation network benchmark datasets [26, 29]. **WebKB webpage networks**: Cornell and Texas are the webpage networks.**Co-occurrence network**: Actor network contains the co-occurrences of actors in films**Wikipedia networks**: Chameleon and Squirrel are the webpages extracted from different topics in Wikipedia [28]. In Table 1, the network homophily rates [27] are provided. **Large-scale graph.** Ogbn-products and Ogbn-mag [14]. The statistics about these 2 datasets are summarized in Table 2.

**Baseline Methods.** To verify the effectiveness LSTGM, 13 baseline methods are employed.They are divided into 3 categories: **Classic GNNs** for node classification task include vanilla GCN [16], GAT [30], and GraphSAGE [9].**GNNs with residual connection** include JKNet [34], APPNP [17], and GCNII [4]. **GNNs for networks with heterophily** include Geom-GCN [27], H2GCN [42], GPRGNN [6], and FAGCN [2]. **GNNs for large-scale graphs** include LINKX [20] and GloGNN [21].

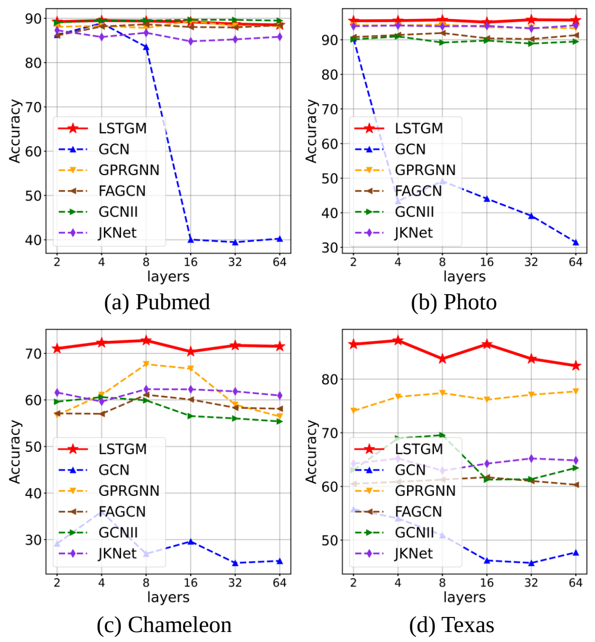


Figure 5: Node classification results with various model depths.

**Parameter Setting.** For datasets in Table. 1, we randomly split nodes of each class into 60%, 20% and 20% for training, validation and testing. All results are obtained by computing over 10 random splits, as suggested in [27]. The hyperparameters, including weight decay, dropout, and learning rate, are tuned on validation set.

### 5.2 Results Analysis

*5.2.1 Results on Real-World Datasets.* The results are shown in Table 3, where the bold and the underlined indicate the best and the second best performances, respectively.

On small homophilic datasets, i.e., Cora, Citeseer, and Pubmed, LSTGM achieves comparable performances, which are slightly lower than SOTA, and the differences are very tiny. It can be observed that LSTGM achieves new remarkable SOTA results on large homophilic datasets Computer and Photo, compared to all other baselines. It demonstrates that the node-dependent weights for residual connections are indispensable.

On heterophilic datasets, LSTGM is compared with Geom-GCN, GPRGNN, FAGCN, and H2GCN, which are all the GNNs designed

Table 3: Classification Accuracy (Bold indicates the best, underlined indicates the second best).

| **Methods** | Cora | Citeseer | Pubmed | Computer | Photo | Chameleon | Squirrel | Actor | Texas | Cornell |
|---|---|---|---|---|---|---|---|---|---|---|
| GCN | 85.77±0.25 | 73.68±0.31 | 88.13±0.28 | 82.52±0.32 | 90.54±0.21 | 35.99±2.58 | 34.02±1.34 | 26.97±1.49 | 55.68±9.61 | 55.14±7.57 |
| GAT | 86.37±0.30 | 74.32±0.27 | 87.62±0.26 | 81.95±0.38 | 90.09±0.27 | 60.26±2.50 | 40.72±1.55 | 27.44±0.89 | 58.38±4.45 | 58.92±3.32 |
| GraphSAGE | 87.77±1.04 | 71.09±1.30 | 88.42±0.50 | 83.11±0.23 | 90.51±0.25 | 58.73±1.68 | 41.61±0.74 | 34.23±0.99 | 82.43±6.14 | 75.95±5.01 |
| MLP | 74.82±2.22 | 70.94±0.39 | 63.76±0.78 | 70.48±0.28 | 78.69±0.30 | 46.21±2.99 | 28.77±1.56 | 36.53±0.70 | 81.89±4.78 | 81.08±6.37 |
| APPNP | 87.87±0.85 | 76.53±1.33 | 89.40±0.61 | 81.99±0.26 | 91.11±0.26 | 54.30±0.34 | 33.29±1.72 | 31.71±0.70 | 82.43±1.72 | 82.16±3.83 |
| GCNII | 88.49±2.78 | 77.08±1.21 | 89.57±1.56 | 86.13±0.51 | 90.98±0.93 | 60.61±2.00 | 37.85±2.76 | 36.18±0.61 | 69.46±1.86 | 74.86±2.73 |
| JKNet | **88.93±1.35** | 74.37±1.53 | 87.68±0.30 | 77.80±0.97 | 94.13±0.70 | 62.31±2.76 | 44.24±2.11 | 36.47±0.51 | 65.35±4.86 | 56.49±3.22 |
| DAGNN | 87.40±0.72 | 74.67±1.31 | 84.84±0.35 | 88.35±0.24 | 94.36±0.25 | 53.79±1.29 | 37.68±0.63 | 30.50±0.59 | 79.19±2.42 | 74.32±3.47 |
| Geom-GCN-I | 85.19±1.13 | 77.99±1.23 | **90.05±0.90** | NA | NA | 60.31±1.77 | 33.32±1.59 | 29.09±0.86 | 57.58±1.97 | 56.76±3.17 |
| Geom-GCN-P | 84.93±0.51 | 75.14±1.50 | 88.09±1.37 | NA | NA | 60.90±1.13 | 38.14±1.23 | 31.63±0.98 | 67.57±1.13 | 60.81±2.21 |
| GPRGNN | 88.65±1.37 | **77.99±1.64** | 89.18±0.61 | 89.43±0.86 | 94.76±0.20 | 67.48±1.98 | 49.93±1.34 | 36.58±1.04 | 77.84±2.78 | 79.73±3.91 |
| FAGCN | 87.77±1.69 | 74.66±2.27 | 88.60±0.64 | 86.09±0.40 | 91.96±0.71 | 61.12±1.95 | 40.88±2.02 | 36.81±0.26 | 61.82±8.71 | 67.95±10.02 |
| H2GCN | 86.92±1.37 | 76.88±1.77 | 89.40±0.34 | 86.67±0.32 | 93.91±0.48 | 59.39±1.98 | 37.90±2.02 | 35.62±1.30 | 84.86±4.32 | 82.16±3.27 |
| LSTM-GNNs | 83.98±1.15 | 73.94±1.24 | 88.78±0.23 | 89.90±0.81 | 94.81±0.17 | 69.66±0.49 | 56.55±1.65 | 36.51±0.65 | 80.54±5.65 | 81.62±4.09 |
| LSTGM | 86.72±0.55 | 76.41±0.90 | 89.93±0.16 | **90.85±0.32** | **95.68±0.29** | **73.77±1.01** | **61.36±0.91** | **37.26±0.57** | **85.41±4.18** | **84.59 ± 4.17** |

Table 4: Classification Accuracy on Large-scale Graph(Bold indicates the best, underlined indicates the second best).

| **Dataset** | Ogbn-products | | Ogbn-mag | |
|---|---|---|---|---|
| | Val Accuracy | Test Accuracy | Val Accuracy | Test Accuracy |
| GCN | 92.00±0.03 | 75.64±0.21 | 39.66±0.18 | 39.02±0.16 |
| GraphSAGE | 92.24±0.07 | 80.50±0.14 | 43.68±0.06 | 42.28±0.21 |
| APPNP | 91.84±0.07 | 79.71±0.50 | 42.81±0.15 | 42.50±0.45 |
| GPRGNN | 93.02±0.13 | 81.95±0.14 | 50.24±0.30 | 49.48±0.34 |
| LINKX | 93.31±0.03 | 83.30±0.25 | 53.62±0.06 | 53.14±0.19 |
| GloGNN | 93.42±0.14 | 83.28±0.51 | 54.77±0.15 | 53.86±0.13 |
| LSTGM | 93.05±0.02 | **84.42±0.22** | 54.84±0.62 | **55.11±0.27** |

Table 5: Model Efficiency: average total running time (s)

| **Model** | GCNII | GPRGNN | H2GCN | LSTGM |
|---|---|---|---|---|
| Pubmed | 31.81 | 22.39 | 46.29 | 65.18 |
| Computer | 18.15 | 12.52 | 33.50 | 40.84 |
| Actor | 11.35 | 10.36 | 20.97 | 29.17 |
| Squirrel | 9.11 | 6.17 | 15.39 | 21.06 |

for handling networks with heterophily. This demonstrates that topology and node attributes-related adaptive residual is critical in heterophilic networks on account of the complicated topology information and poor predictability attributes are widespread in networks with heterophily. In addition, by sequence modeling, layer-dependent residual connections benefit LSTGM by capturing different order information.

LSTGM can handle large-scale graphs. As shown in Table 4, LSTGM achieves the best results on Ogbn-products and Ogbn-mag. These results inspire a broader industrial application for LSTGM.

## 5.3 Preventing Over-smoothing Issue

To prove that the proposed LSTGM can alleviate over-smoothing issue, five baselines such as GCN, GCNII, FAGCN, GPRGNN and JKNet, which perform well as deep models, are compared with the proposed LSTGM with varying numbers of layers on 8 networks mentioned in Table 1. The results are shown in Figure 5.

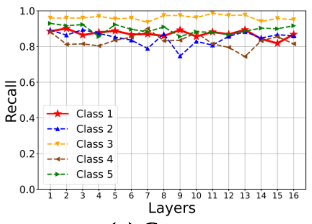
As shown in Figure 5(a)-(b), the proposed LSTGM keeps the performance stable or continues to rise with the number of layers increasing. This phenomenon is mainly attributed to the adaptive residual connections, which enable fine-grained consideration. For different nodes and layers, the residual information regularized by the LSTGM is different, ensuring that redundant information will not be propagated frequently, thus alleviating the over-smoothing. As shown in Figure 5(c)-(d), the proposed LSTGM outperforms other deep models, which are specially designed for heterophilic networks. This phenomenon proves that a more comprehensive and appropriate residual design is vital for heterophilic networks.

Compared to the case of GCN with residual connection in Figure 2, Figure 6 provides the class-wise recalls of LSTGM as model goes deeper. It can be observed that LSTGM overcome the class-imbalanced over-smoothing issue, due to its node- and layer-dependent fine-grained residual connection in LSTGM.

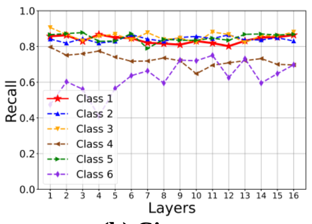
## 5.4 Efficiency study

In this section, we study efficiency. We compare the average training time for effective methods on relatively large datasets for fairness. We use the same training set for all these methods on each dataset and run the experiments for 500 epochs.

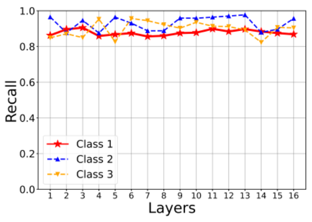
As shown in 5, though the average total training time of LSTGM is larger than others, the gap is minimal. GPRGNN get the fewest running time, but the classification accuracy on heterophily networks is far behind LSTGM as shown in 3. GCNII and H2GCN introduce new trainable parameters for every layer, while LSTGM's parameters are agnostic of layers and reduce the time cost consequently. In general, the additional time cost is acceptable and LSTGM is efficient.



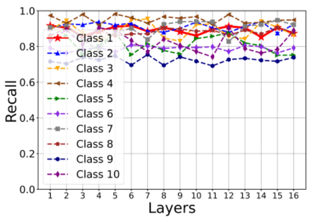
**(a) Cora**



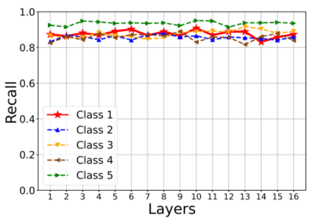
**(b) Citeseer**



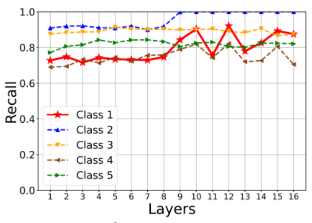
**(c) Pubmed**



**(d) Computers**

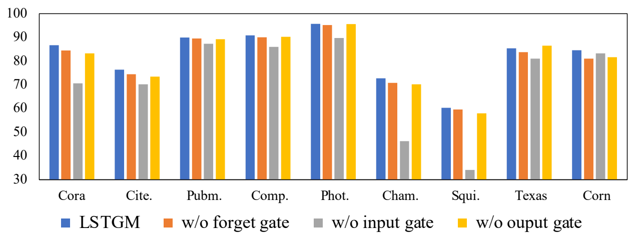


**(e) Chameleon**

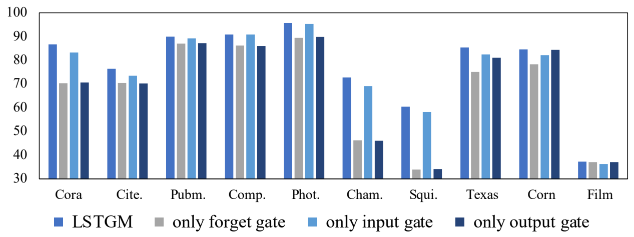


**(f) Squirrel**

**Figure 6: Class-wise recalls of LSTGM as model goes deeper.**



(a) Classification accuracy results when removing one gate



(b) Classification accuracy results when living one gate

**Figure 7: The accuracies of LSTGM with different parts.**

## 5.5 Ablation Study

This section performs the ablation study.In the first experiment, one of the three gates are kept. As shown in Figure 7(a), dramatic performance degradation can be observed once we remove the input gate in most datasets.Forget and output gate contribute more to the long-term dependencies, which tell nodes what they should be forgotten and should be passed. In the second experiment, one of three gates is removed respectively. Figure 7(b) also intuitively shows the importance of different gates on various datasets. They come to a similar conclusion as Figure 7(a).

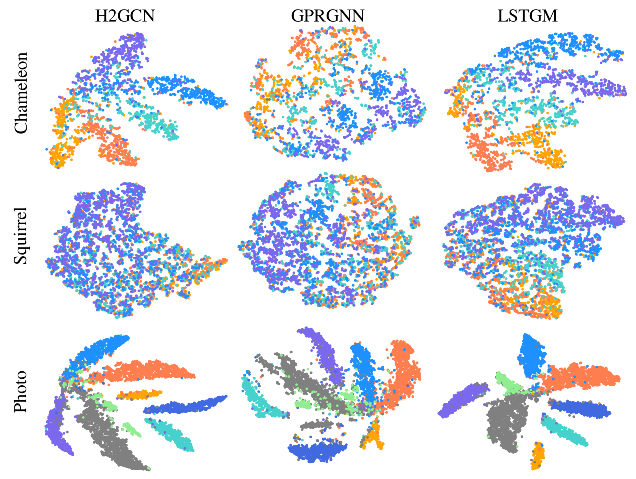


**Figure 8: t-SNE of the node representations.**

## 5.6 Visualizations

To provide an intuitive understanding of LSTGM, the t-SNE visualizations of the node embeddings obtained from different GNNs, including GPRGNN, H2GCN, and the proposed LSTGM on the Chame-leon, Squirrel, and Photo, are shown in Figure. 8. The colors of nodes represent their labels. The proposed LSTGM occurs relatively little overlapping phenomenon on heterophilic datasets. Besides, the representations on homophilic networks are more discriminative than those from other methods. Therefore, the expressive power of LSTM-GNN is high.

## 6 CONCLUSIONS

The coarse-grained node-independent residual connection still suffers from class-imbalanced over-smoothing issue, due to the fixed and linear combination of topology and attribute in node representation learning. To make the combination flexible to capture complicated relationship, this paper reveals that the residual connection needs to be node-dependent, layer-dependent, and related to both topology and attribute, and implements these requirements via LSTM by considering the representations of one node in different layers as a sequence of states. Furthermore, Long Short-Term Graph Memory Network (LSTGM) is proposed by updating memory to graph memory and enhancing the three gates with graph topology. Experimental evaluations demonstrate that LSTGM possesses the attractive characteristics including overcome the class-imbalanced over-smoothing and superior performance on real-world networks including homophilic, heterophilic and large networks.

## 7 ACKNOWLEDGMENTS

This work was supported in part by the National Science Fund for Distinguished Young Scholarship of China (No. 62025602), in part by the National Natural Science Foundation of China (No. 61972442, 62102413, 62276187, 62272020, U1936210, U1936208, U22B2036, 11931915), in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Fok Ying-Tong Education Foundation China (No. 171105), the Tencent Foundation and XPLORER PRIZE, and in part by the Fundamental Research Funds for the Central Universities.

## REFERENCES

[1] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. 2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*.

[2] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. (2021), 3950–3957.

[3] Junyu Chen, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. A Unified Framework against Topology and Class Imbalance. In *ACM International Conference on Multimedia*. 180–188.

[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*. 1725–1735.

[5] Xu Chen, Yuanxing Zhang, Lun Du, Zheng Fang, Yi Ren, Kaigui Bian, and Kunqing Xie. 2020. TSSRGCN: Temporal Spectral Spatial Retrieval Graph Convolutional Network for Traffic Flow Forecasting. In *ICDM*. 954–959. https://doi.org/10.1109/ICDM50108.2020.00108

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

[7] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.

[8] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *AAAI*. 922–929. https://doi.org/10.1609/aaai.v33i01.3301922

[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[10] Xiaoke Hao, Jie Li, Yingchun Guo, Tao Jiang, and Ming Yu. 2021. Hypergraph Neural Network for Skeleton-Based Action Recognition. *IEEE TIP* 30 (2021), 2263–2275. https://doi.org/10.1109/TIP.2021.3051495

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[13] Wenzheng Hou, Qianqian Xu, Zhiyong Yang, Shilong Bao, Yuan He, and Qingming Huang. 2022. AdAUC: End-to-end Adversarial AUC Optimization Against Long-tail Problems. In *International Conference on Machine Learning*. PMLR, 8903–8925.

[14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.

[15] Zhao Kang, Zhiping Lin, Xiaofeng Zhu, and Wenbo Xu. 2022. Structured Graph Learning for Scalable Subspace Clustering: From Single View to Multiview. *IEEE Trans. Cybern.* 52, 9 (2022), 8976–8986. https://doi.org/10.1109/TCYB.2021.3061660

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[17] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

[18] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *ICCV*. 9266–9275. https://doi.org/10.1109/ICCV.2019.00936

[19] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. 3538–3545.

[20] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. 2022. Finding Global Homophily in Graph Neural Networks When Meeting Heterophily. In *ICML*. 13242–13256.

[21] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. 2021. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In *NeurIPS*. 20887–20902.

[22] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4424–4431.

[23] Ke Ma, Qianqian Xu, Jinshan Zeng, Xiaochun Cao, and Qingming Huang. 2022. Poisoning Attack Against Estimating From Pairwise Comparisons. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 6393–6408.

[24] Ke Ma, Qianqian Xu, Jinshan Zeng, Guorong Li, Xiaochun Cao, and Qingming Huang. 2023. A Tale of HodgeRank and Spectral Method: Target Attack Against Rank Aggregation is the Fixed Point of Adversarial Game. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2023), 4090–4108.

[25] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2020. A Unified View on Graph Neural Networks as Graph Signal Denoising. arXiv:2010.01777 [cs.LG]

[26] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. 2012. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*.

[27] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[28] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. *arXiv:1909.130* (2019).

[29] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[30] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[31] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*. 6861–6871.

[32] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *TNNLS* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[33] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*. 1907–1913. https://doi.org/10.24963/ijcai.2019/264

[34] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*. 5449–5458.

[35] Zhe Xue, Junping Du, Hai Zhu, Zhongchao Guan, Yunfei Long, Yu Zang, and Meiyu Liang. 2022. Robust Diversified Graph Contrastive Network for Incomplete Multi-view Clustering. In *ACM MM*. 3936–3944. https://doi.org/10.1145/3503161.3547894

[36] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*. 7444–7452. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17135

[37] Liang Yang, Chuan Wang, Junhua Gu, Xiaochun Cao, and Bingxin Niu. 2021. Why Do Attributes Propagate in Graph Convolutional Neural Networks?. In *AAAI*. 4590–4598.

[38] Zhiyong Yang, Qianqian Xu, Shilong Bao, Xiaochun Cao, and Qingming Huang. 2021. Learning with Multiclass AUC: Theory and Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[39] Zhiyong Yang, Qianqian Xu, Shilong Bao, Yuan He, Xiaochun Cao, and Qingming Huang. 2022. Optimizing Two-way Partial AUC with an End-to-end Framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[40] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*. 3634–3640. https://doi.org/10.24963/ijcai.2018/505

[41] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

[42] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*.

[43] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *WWW*. 1215–1226.