Neurocomputing 480 (2022) 76–88

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# PLAM: A plug-in module for flexible graph attention learning

Xuran Pan [a,b], Shiji Song [a,b,*], Yiming Chen [a,b], Liejun Wang [c], Gao Huang [a,b]

[a] Tsinghua University, Beijing, China
[b] Beijing National Research Center for Information Science and Technology (BNRist), China
[c] Xinjiang University, Xinjiang, China

ARTICLE INFO

*Article history:*
Received 18 August 2021
Revised 19 December 2021
Accepted 14 January 2022
Available online 21 January 2022

*Keywords:*
Graph-based learning
Graph convolutional networks
Self-attention
Semi-supervised node classification

ABSTRACT

Graph Convolutional Networks (GCNs) are general deep representation learning models for graph-structured data. In this paper, we propose a simple *Plug-in Attention Module (PLAM)* to improve the representation power of GCNs, inspired by the recent success of the query-key mechanism in computer vision and natural language processing. With this module, our network is able to adaptively learn the weights from a node towards its neighbors. Different from existing attention-based GCNs, the proposed PLAM has several important properties. First, the parameter space for the attention module is isolated from that for feature learning. This ensures that the proposed approach can be conveniently applied to existing GCNs as a plug-in module. Second, the anchor node and neighbor nodes are treated separately when learning the attention weights, which further enhances the flexibility of our structure. Third, our attention module extracts higher-level information by computing the inner product of the features between the anchor node and neighbor nodes, leading to significantly increased representation power. Last, we take a step forward and propose a novel structural encoding technique for the graph attention module to inject local and global structure information. Although being simple, our PLAM models have achieved state-of-the-art performances on graph-structured datasets under both the transductive and inductive settings. Additionally, experiments on image and point cloud datasets show potential applications of PLAM on several computer vision tasks.



## 1. Introduction

Convolutional Neural Networks (CNNs) have proved their effectiveness in dealing with computer vision tasks [32,55,21,24]. A key component of CNNs is the convolution operation which can extract features via local filters with learned parameters. However, they are only applicable when the input data has a regular grid structure. Recently, Graph Convolutional Networks (GCNs) [27,7] are proposed as a counterpart of CNNs for graph-structured data. In each layer, GCNs learn representations of graph nodes via 1-hop neighborhood with spectral local filters followed by a fully connected layer. In specific, the spectral filters are approximated directly by the graph Laplacian along with a Chebyshev expansion method. As a result, the spectral graph convolution is essentially averaging feature vectors from the neighbors for each node. Due to their high efficiency and representation power, GCNs have attracted great research attention, and have been widely adopted in areas like social network analysis [20,66,38], biology [81], graph embedding [6,51], graph classification [61], transport [74], and natural language processing [67,70,43].

Following GCNs, several studies on graph convolution models have been made to provide deeper insights into this research field. GraphSAGE [20] extends GCNs to large-scale inductive tasks. This model proposes several neighborhood information aggregators other than the one in GCNs, such as LSTM aggregator and max pooling. More recently, Simple GCNs [66] are proposed for more efficient graph convolution with a single layer network.

Despite the significant contributions the above studies have made for GCNs, researchers have demonstrated that such graph convolution faces a dilemma of *over-smoothing* and *generalization capability* [35]. Specifically, the spectral convolution kernel weights are determined solely by the graph structure, suggesting that the information aggregation over the graph is not learnable, but predefined. In many real-world scenarios, the nodes from different categories are often connected, leading to undesirable drawbacks with deep or shallow architectures. For a deep model, the feature representations across categories may become indistinguishable as the graph convolution layers stack. On the other hand, for a shallow

* Corresponding author.

https://doi.org/10.1016/j.neucom.2022.01.045
0925-2312/

architecture, the reception field is limited within its relatively nearer neighbors, neglecting long-range dependencies.

Popular attention mechanisms have been adopted in the field of computer vision [76,15] and natural language processing [57,8] and provide insight into this issue. Transformer [57] adopts a multi-head attention module to achieve long-range dependencies in recurrent neural networks. SAGAN [76] introduces self-attention module for stable and effective training. In these studies, a query-key mechanism is adopted and the attention coefficients are computed from queries and keys extracted from the input.

Based on these observations, attention-based methods are adopted in GCN models to dynamically learn the attention coefficients between each node and its neighbors [58,56,25]. As remedies to the over-smoothing problem, these approaches have shown great potential in enhancing the model performance. However, existing attention-based GCNs still have limitations. First, the attention module is deeply entangled with the representation learning module, *i.e.,* the input feature is first served as input for the representation learning module and then adopted to compute the attention filter in a series connection structure, which makes it difficult to generalize across different model architectures. Second, the structure of the attention module in existing work is relatively simple and does not fully explore the power of the attention mechanism. Third, the attention functions in these researches are applied in a single space for both anchor and neighbor nodes. However, for each node, its importance as anchor and neighbor are not necessarily identical. Although a node is connected to others in this network, it may nonetheless be more an information provider (anchor) rather than a receiver (neighbor), or vice versa. As a result, treating each node as two roles equally while calculating its relating attentions may be unable to produce sufficient expressive power. Additionally, the receptive field in an attention layer is mostly restricted within one-hop neighborhood, which discards the structural information from the global perspective.

In this paper, we propose a flexible attention-based module, named *Plug-in Attention Module (PLAM)* (as shown in Fig. 1), to address the aforementioned weaknesses in existing approaches. First, to disentangle the parameter space of the attention module from that for feature learning, we design an independent self-attention module utilizing raw features as input. Consequently, the proposed approach can be conveniently applied to various GCN structures as a plug-in module, while attention modules from previous works are highly coupled with the feature learning layers. Second, two techniques are presented to enhance the representation capability of the attention module: (1) By computing the attention coefficients through inner product rather than linear combinations, our attention module extracts higher-level information between anchor node and neighbor nodes, leading to significantly increased representation power. (2) The anchor and neighbor nodes are learned separately before calculating the inner product, which emphasizes the different roles between these candidates and further enhances the flexibility of our structure. Last, inspired by the widely adopted positional encoding technique in the self-attention mechanism [57], we propose a novel structural encoding module by projecting several node measurements in the graph theory to the feature space. When calculating the attention weights, we conduct dot product between node features and features of node measurements, and add to the original attention matrix. By this means, we introduce the structural information from a global perspective. While being effective in enhancing the model representation capability, PLAM maintains stability in the training procedure. By bounding the spectrum of convolution filters, gradient explosion and numerical instabilities are alleviated.

The proposed PLAM can be applied in both transductive and inductive settings. We validate our method on three transductive benchmarks: *Cora, Citeseer* and *Pubmed* citation networks, four inductive benchmarks: *Reddit*, *protein–protein interaction (PPI)* and two graph-structured dataset *PATTERN, CLUSTER* generated with stochastic block model (SBM). We achieve state-of-the-art results on all of them and show the effectiveness of self-attention module on graph-structured data. As a plug-in module, our approach can also be conveniently adopted to improving the performances of various GCN-based models. Furthermore, experiments conducted on 2D image datasets and 3D point cloud dataset have shown potential applications of PLAM on several computer vision tasks.

In the remainder of this paper, we discuss related works in Section 2. In Section 3 and 4, we introduce the method and provide theoretical analysis. We show experimental results in Section 5 and conclude the paper in Section 6.

## 2. Related works

### 2.1. Graph neural networks

Early studies of Graph Neural Networks (GNNs) [19,52,36,3,46,23,13,68] have attempted to apply neural network models to the graph domain. Recent years have witnessed an upsurge of interest in extending convolution [2,22,7,27,45,44] as well as pooling [72,16,33] operations from grid-like images to graphs. Simultaneously, particular techniques have also been introduced to assist neural networks in dealing with graph-structured data [5,79,50,73]. In this paper, we focus on graph convolution, or in general, the message passing [17] and aggregation process.

Researchers proposed two types of convolution operations on graphs: non-spectral and spectral approaches. Non-spectral convolutions [10,45,44] are defined directly on the graph, typically assembling a sub-graph of the local neighborhood and feeding it into the traditional convolutional layer. Spectral approaches [2,22,7,27] implement convolution operations in the Fourier domain and the computation is based on the graph Laplacian. In [7], an approximation using Chebyshev polynomials was proposed with an efficient localized convolution framework. Graph Convolutional Networks (GCNs) proposed by [27] further restricted the graph convolution operation to the 1-hop neighborhood. More recently, MixHop [26], GDC[30], Stronger GCN[39] mixed powers of adjacency matrix to learn multi-hop information.

### 2.2. Attention-based GCNs

Since the convolution filters in GCNs are dependent only on the graph structure, GCNs are restricted to transductive tasks and may cause over-smoothing. To overcome these problems and enable the networks to learn adaptive convolution weights, the concept of attention is introduced to graph neural networks. Graph Attention Networks (GAT) [58] is first proposed following the attention strategy and learn the attention weights directly from node feature vectors. The attention computation in GAT first receives the linearly transformed features which are to be aggregated as input, resulting in the entanglement with feature learning parameters. In contrast, our module has an independent parameter space and is able to generalize to various graph model architectures. Another related approach is the Disentangled GCN (DisenGCN) [40]. Both DisenGCN and PLAM models involve multiple linear transformations, but DisenGCN aims to separate different kinds of neighbors, while our PLAM models attempt to distinguish anchor and neighbor nodes. Under many circumstances, the entangled factors in learned features are indistinguishable while the directions of the message passing are more useful and reliable. Several researches have also considered to compute the attention coefficients based on other

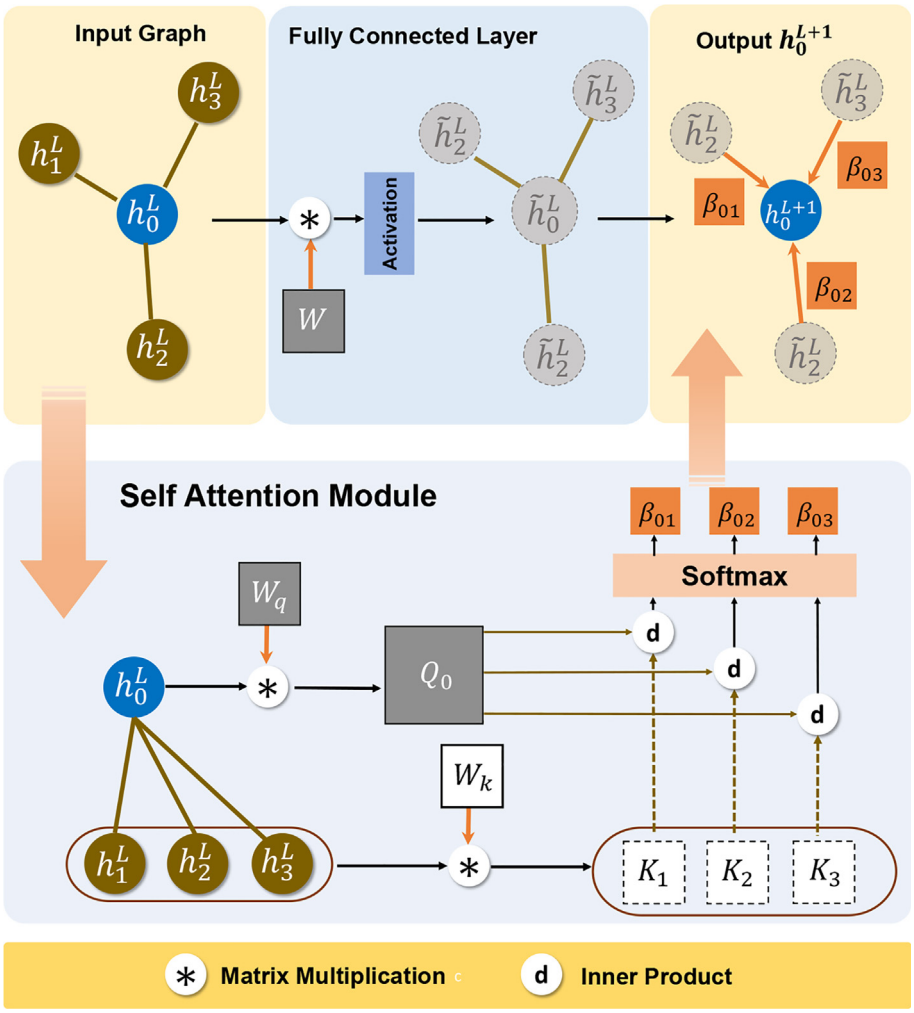



**Fig. 1.** Demonstration of the proposed self-attention module. Only the feature transformation of the anchor node $h_0$ is shown. The top row shows the transformation of a standard GCN. The bottom row represents the structure of the proposed self-attention module. The anchor features $h_0^L$ and neighbor features $h_1^L, h_2^L, h_3^L$ are transformed into the query space as $Q_0$ and key space as $K_1, K_2, K_3$, respectively. Attention coefficients $\boldsymbol{\beta}$ are computed between $Q_0$ and key vectors. Information of neighbor nodes is aggregated by the attention coefficients to obtain $h_0^L$ after a fully connected layer.

similarity metrics. In [56], attention is calculated by cosine distance. Another attention learning framework presented in [25] uses L1 distance. However, in the aforementioned methods, the attention weights are mostly learned from 1st level representation of the graph node features. PLAM is different in that it extracts higher-level information through the inner product, and thus gains higher expressive power. Besides, PLAM projects the anchor and neighbor nodes into different spaces instead of a shared one, endowing the model with better flexibility. Additionally, attention-based approaches have shown great potential in other related research areas, e.g., relational graph learning[75], knowledge graph embedding[80], mapping[77], and graph classification [33]. Graph attention models are also proved effective on vision tasks, e.g., object detection[78,64], and video object segmentation [49].

## 3. Method

In the paper, we aim to develop a plug-in module for graph neural networks with high expressive power. To achieve this goal, we introduce a novel self-attention mechanism to the traditional GCN structure. We will first define notations and give a mathematical description of GCNs. Then, we describe how our module is applied to the convolution operator. Two modes of applying the attention module in the network are presented. Lastly, we compare our module with the popular Graph Attention Networks.

### 3.1. Preliminaries

We start with an overview of GNN-based models and introduce our notation in the context of semi-supervised node classification task. Let $G = (V, A)$ denotes a graph with node feature vectors where $V$ represents the vertex set consisting of nodes $\{v_1, v_2, \ldots, v_N\}$ and $A \in \mathscr{R}^{N\times N}$ is the adjacency matrix implying the connectivity between nodes in the graph. In this paper, we mainly focus on the undirected graph with identical edge weights, which has a sparse and symmetric adjacency matrix where $a_{ij} = 1$ suggests an edge between $v_i$ and $v_j$, and $a_{ij} = 0$ otherwise. Each node has a feature of $F$ dimensions and a one-hot label vector out of $C$ classes, composing feature matrix $X \in \mathscr{R}^{N\times F}$ and label matrix $Y \in \mathscr{R}^{N\times C}$. Only the labels of a subset of nodes are available during training and other nodes are for prediction.

Modern GNN models usually follow a paradigm of message passing and updating, where we aggregate features from neighbors and update the representation of the anchor node recursively. Gen-

erally, the function between $k^{th}$ and $(k+1)^{th}$ layer can be summarized as:

$$a_v^{(k)} = \text{Aggregate}^{(k)}\left(h_u^{(k-1)} : u \in \mathcal{N}(v)\right), \tag{1}$$

$$h_v^{(k)} = \text{Update}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right), \tag{2}$$

where $h_v^{(k)}$ is the feature vector of node $v$ of the $k^{th}$ layer and $a_v^{(k)}$ is the representation integrated with received messages. In each layer, an anchor node captures the information of its direct neighbors. With the increase of network depth, representations of nodes from longer ranges will be absorbed, iteratively achieving message spreading through the whole graph along the edges.

### 3.2. Graph convolutional networks

Graph Convolutional Networks (GCNs) also follow the paradigm of message passing and updating, learning deep representation of each node over multiple layers, which is used for downstream tasks. GCNs are practically an approximation of spectral convolution operation using Chebyshev polynomials, which boils down to an efficient algorithm. Consequently, a GCN layer averages the representations of neighbors after a linear mapping, which can summarize as:

$$H^{(k+1)} = \text{ReLU}\left(\left(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)H^{(k)}W^{(k)}\right), \tag{3}$$

where $D$ is the degree matrix of $A$ and $W^{(k)} \in \mathscr{R}^{F^{(k+1)} \times F^{(k)}}$ is a trainable weight matrix. Considering that the operator $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ has unbounded eigenvalues, repeated application with stacked layers may result in numerical instabilities and erratic gradients. To alleviate this problem, a renormalization trick is introduced:

$$I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}, \tag{4}$$

with $\widetilde{A} = A + I_N$ and $\widetilde{D}$ being degree matrix of $\widetilde{A}$. By modifying the normalization method, the upper bound of operator's eigenvalue is decreased from 2 to 1, alleviating the risk of instability while preserving computation efficiency.

### 3.3. Plug-in attention module for GCNs

In the original GCNs, the convolution operator $\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ is fixed for each layer, implying that the weights shared between a node's neighbors are predefined and constant. In fact, during the update procedure, each neighbor contributes regardless of their specific features. Following GCN, attention models are proposed to learn the weights based on the node features output by the current layer. However, these approaches can not be plugged into other model architectures since the attention module is highly coupled with the feature learning layers. Besides, the structures of these models fail to extract high-level information of the graph structure for aggregation.

To overcome the aforementioned problems, we propose a *Plug-in Attention Module (PLAM)*, consisting of a novel attention module to learn the convolution weights dynamically by introducing the query-key mechanism. By isolating the parameter space of attention module from that for feature learning, PLAM can be conveniently applied to various state-of-the-art GCN structures.

Given a feature map of the $l^{th}$ layer $h^l \in \mathscr{R}^{N \times F^l}$, we aim to first transform the features into two spaces indicating the two roles each node will play: a query space $f$ and a key space $g$. Formally, we learn the two spaces as

$$f\left(h^l\right) = h^l W_q^l, \qquad g\left(h^l\right) = h^l W_k^l, \tag{5}$$

where $W_q^l, W_k^l \in \mathscr{R}^{F^l \times \widetilde{F}^l}$. $\widetilde{F}^l = F^l/C$ is generally smaller than feature dimension $F^l$, considering the computation efficiency. Then, the attention matrix $B^l = \left[\beta_{ij}^l\right]$ is calculated by dot product of query vectors and key vectors:

$$s_{ij}^l = f\left(h_i^l\right)g\left(h_j^l\right)^T, \tag{6}$$

$$\beta_{ij}^l = \begin{cases} \text{softmax}_j\left(s_{ij}^l\right) = \frac{e^{s_{ij}^l}}{\sum_{k \in \{i \cup \mathcal{N}(i)\}} e^{s_{ik}^l}}, & \text{if } \widetilde{A}_{ij} = 1, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $\widetilde{A} = A + I_N$ and $\beta_{ij}^l$ indicates the extent to which the model attends to the $j^{th}$ node when synthesizing the $i^{th}$ node. With the intention of preserving the graph's structural information, node $j$ is considered only if it is a neighbor to $i$ or node $i$ itself. Then, the output of the $l^{th}$ layer $h^{(l+1)} = \left(h_0^{(l+1)}, \ldots, h_N^{(l+1)}\right)$ is the weighted summation:

$$h_i^{(l+1)} = \text{ReLU}\left(\left(\sum_{j \in \{i \cup \mathcal{N}(i)\}} \beta_{ij}^l h_j^{(l)}\right)W^l\right), \tag{8}$$

where $W^l$ is a learnable matrix for each layer. In self-attention models, there exist three groups of features: *query*, *key* and *value*. In our graph self-attention module, $h^l W^l$ can be seen as the value matrix. Lastly, normalization is adopted for numerical stability. The final model takes the form:

$$\widehat{Y} = \text{softmax}\left(B^L \ldots \text{ReLU}\left(B^1 X W^1\right) \ldots W^L\right). \tag{9}$$

Intuitively, PLAM is able to recognize the relationships between two connected nodes and extract more powerful feature representations. The input feature map of each layer is transferred to new feature spaces in which the inner product of query and key features are trained to accurately represent the similarity between nodes. In this way, the attention weights are supposed to contain richer information about the relationship between two nodes, which is beneficial in the process of message passing and aggregation. Furthermore, the self-attention module is independent of the network layers, i.e., the module can be conveniently applied to various GCN-based models.

### 3.4. Structural encoding

With the aforementioned self-attention module, the anchor node can aggregate information from neighbor nodes with dynamic weights conditioned on their features. Nevertheless, the receptive field of each node is restricted within its one-hop neighborhood, leading to natural inferiority on modelling long-range dependencies and global structural information. As a remedy, we resort to the similar technique, positional encoding, which is widely adopted in self-attention modules, and propose a novel structural encoding to integrate the graph structural into attention weight distribution.

Practically, we consider several graph measurements and reflect each node's properties from local and global perspectives respectively. We first introduce the local clustering coefficient, which quantifies the closeness of the neighborhood to a complete graph:

$$C(i) = \frac{\left|\{A_{jk} : j, k \in \mathcal{N}(i), A_{jk} = 1\}\right|}{|\mathcal{N}(i)|(|\mathcal{N}(i)| - 1)}, \tag{10}$$

where $|\mathcal{N}(i)|$ is the number of nodes in $i$'s neighborhood.

We also adopt three centrality measurements as the quantification of local and global structural. Betweenness centrality mea-

sures the importance of a node to the whole graph, and is computed as the number of shortest paths that passes through a certain node:

$$g(i) = \sum_{i \neq j \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}}, \tag{11}$$

where $\sigma_{jk}$ is the total number of shortest paths from node $j$ to $k$, and $\sigma_{jk}(i)$ represents the number of those paths that pass through node $i$. For connected graphs, closeness centrality measures the average distance of a certain node to the whole graph, which is defined as:

$$c(i) = \frac{N}{\sum_j d(j,i)}, \tag{12}$$

where $d(j,i)$ correspond to the length of shortest path from node $j$ to node $i$, and $N$ is the number of nodes in the graph. We also include the degree centrality to encode the local structure, which is defined as the degree of the node:

$$d(i) = \deg(i) = D_{ii}. \tag{13}$$

With these graph measurements, we can inject the local and global graph structure, as a complementary to the feature space. Specifically, we use a fully connected layer to encode structural information and augment attention weights:

$$r(i) = \mathrm{FC}(C(i), g(i), c(i), d(i)), \tag{14}$$

$$s_{ij}^l = f\left(h_i^l\right) g\left(h_j^l\right)^T + r(i) g\left(h_j^l\right)^T. \tag{15}$$

The computation of attention matrix and feature aggregation follow the same pipeline as Eq. (7∼9). As a consequence, $r(i)$ is independent of the node features and relies solely on the graph structural information, which emphasizes the intrinsic relationship between anchor node and neighbor node.

### 3.5. Attention modes

In this subsection, we propose two modes of applying attention matrix to the convolution layer, symmetric and asymmetric attentions. The two modes correspond respectively to graphs with undirected and directed weighted edges, endowing PLAM with better adaptability for various data.

#### 3.5.1. Symmetric attention

Since most of the input data for semi-supervised classification tasks take the shape of *undirected* graphs, the first mode we propose uses symmetric attention matrix before softmax normalization. Therefore, two connected nodes are equally important to each other. Adopting the symmetric attention matrix preserves the essential characteristic of the undirected graph and shares weights for the two directions along each edge in the message passing procedure. Specifically, we let $S_{\mathrm{sym}}^l = \left(s_{ij}^l + s_{ji}^l\right)/2$ and the convolution operator can be written as:

$$B_{\mathrm{sym}}^l = \underset{A_{ij} \neq 0}{\mathrm{softmax}}\left(S_{\mathrm{sym}}^l \odot \widetilde{A}\right), \tag{16}$$

where $\odot$ represents matrix dot product. Notice that we can also decompose the softmax function and write $B_{\mathrm{sym}}^l$ as:

$$B_{\mathrm{sym}}^l = \left(\widetilde{D}^l\right)^{-1} \widetilde{W}^l, \tag{17}$$

where

$$\widetilde{W}^l = \exp\left(S_{\mathrm{sym}}^l\right) \odot \widetilde{A}, \tag{18}$$

and $\widetilde{D}^l$ is the degree matrix of $\widetilde{W}^l$. The $\exp(\odot)$ function implies a point-wise exponential operation to the matrix.

#### 3.5.2. Asymmetric attention

The second mode views the attention coefficients as a message passing function along the *directed* edges, and thus there may exist two weights on each edge distinguished by the two directions. Therefore, attention matrix of directed graph is asymmetric. Considering that each node contains different local structural information, the asymmetry of attention coefficients can be seen as further adjustment.

We simply let $S_{\mathrm{asy}}^l = s_{ij}^l$ with

$$B_{\mathrm{asy}}^l = B^l = \underset{A_{ij} \neq 0}{\mathrm{softmax}}\left(S_{\mathrm{asy}}^l \odot \widetilde{A}\right), \tag{19}$$

which is the same form as Eq. (17).

### 3.6. Comparisons to GAT

Graph Attention Networks (GAT) [58] is an effective attention-based model, assigning different weights to different neighborhoods' features. Both GAT and PLAM introduce an attention module into the node representation learning structure, learning dynamic convolution filters based on the node features. Nevertheless, the mechanisms of the two approaches are different. GAT linearly transforms node features and concatenate anchor features with features of each of neighbors. The attention coefficients are computed as:

$$\alpha_{ij} = \mathrm{softmax}\left(a^T\left[\tilde{h}_i, \tilde{h}_j\right]\right) = \mathrm{softmax}\left(a_1^T \tilde{h}_i + a_2^T \tilde{h}_j\right), \tag{20}$$

where $a = [a_1, a_2]$ is a parameterized weight matrix and $\tilde{h}_i, \tilde{h}_j$ represent node features after the linear transformation. The attention weights in GAT models are computed as the summation of representations for the anchor node and neighbor node. Therefore, the expressive capability of the module is limited because the attentions are regressed by the first level of the node features. By comparison, as shown in Eq. (6) and (7), the self-attention module proposed in our PLAM model defines the attention weights as a second level function of the node features:

$$\begin{aligned} \beta_{ij}^l &= \mathrm{softmax}\left(h_i W_q^l \left(W_k^l\right)^T h_j^T\right) \\ &= \mathrm{softmax}\left(h_i W_{qk} h_j^T\right). \end{aligned} \tag{21}$$

Specifically, attention in GAT can be seen as particular cases of PLAM with carefully designed projection weights. By reformulating Eq. (21) as:

$$\begin{aligned} \beta_{ij}^l &= \frac{\exp\left(h_i W_{qk} h_j^T\right)}{\sum_{k \in \{i \cup \mathcal{N}(i)\}} \exp\left(h_i W_{qk} h_k^T\right)} \\ &= \frac{\exp\left(h_i W_{qk} h_j^T\right) \exp\left(a_1 h_i^T\right)}{\sum_{k \in \{i \cup \mathcal{N}(i)\}} \exp\left(h_i W_{qk} h_k^T\right) \exp\left(a_1 h_i^T\right)} \\ &= \frac{\exp\left(h_i W_{qk} h_j^T + a_1 h_i^T\right)}{\sum_{k \in \{i \cup \mathcal{N}(i)\}} \exp\left(h_i W_{qk} h_k^T + a_1 h_i^T\right)}. \end{aligned} \tag{22}$$

If we carefully set:

$$h_i W_{qk} = a_2,$$

we have

$$\beta_{ij}^{l} = \frac{\exp\left(a_2 h_j^T + a_1 h_i^T\right)}{\sum_{k\in\{i\cup\mathcal{N}(i)\}}\exp\left(a_2 h_k^T + a_1 h_i^T\right)} = \mathrm{softmax}\left(a_1 h_i^T + a_2 h_k^T\right), \tag{24}$$

which has the same formulation as with GAT where the attentions are regressed by the first level of the node features (Eq. (20)). In this way, our attention module can achieve higher representation capability which enhances generalization power of the model. Additionally, by isolating the parameter space of attention module from that for feature learning, PLAM can be conveniently applied to various state-of-the-art GCN structures while GAT is less suitable in several models.

Furthermore, we propose a structural encoding technique as a complement to the lack of structural information, which provides the model with higher capacity and flexibility.

## 4. Spectral analysis

As shown in Section 3.3, PLAM shares some similarities with traditional GCN architecture, where $B^l$ can be seen as an augmented convolution operator. A similar property is that, if the operator has an eigenvalue $\lambda_i$ satisfying $|\lambda_i| > 1$, with multiple stacked convolution layers, high power of the attention matrix will lead to feature over-amplifying, causing numerical explosion. In this section, we analyze the spectrum of $B^l$ and show that PLAM models can maintain the property of numerical stability while providing dynamic weight distributions among neighbors.

### 4.1. Symmetric mode

In the symmetric mode, $B_{\mathrm{sym}}^l$ can be interpreted as a normalized symmetric matrix whose eigenvalues can be bounded by the properties of Laplacian matrix. We follow [27] and approximate the largest eigenvalue of the normalized Laplacian matrix $\widetilde{L} = I - \left(\widetilde{D}^l\right)^{-\frac{1}{2}}\widetilde{W}^l\left(\widetilde{D}^l\right)^{-\frac{1}{2}}$ to be 2 as we can expect that network parameters will adapt to this assumption during training. Notice that $B_{\mathrm{sym}}^l$ is a similar matrix to $\left(\widetilde{D}^l\right)^{-\frac{1}{2}}\widetilde{W}^l\left(\widetilde{D}^l\right)^{-\frac{1}{2}}$. Therefore, $B_{\mathrm{sym}}^l$ has eigenvalues in range [-1,1], i.e., norms of eigenvalues of $B_{\mathrm{sym}}^l$ are bounded by 1 and numerical explosion is alleviated.

### 4.2. Asymmetric mode

For the asymmetric mode, we refer to a corollary of the Perron-Frobenius Theorem to demonstrate the stability.

**Proposition** (Corollary of Perron-Frobenius Theorem). *Let* $B \in \mathscr{R}^{N\times N}$ *be a non-negative square matrix such that:*

(1) $\sum_j B_{ij} = 1, i = 1, 2, \ldots, N$.
(2) $B$ is irreducible.

*Then the norms of eigenvalues of B are bounded by 1.*

It is obvious that condition (1) is satisfied for the normalized $B_{\mathrm{asy}}^l$. There is a twofold discussion for condition (2). An irreducible $B_{\mathrm{asy}}^l$ indicates a strongly connected graph. Otherwise, the graph is partitioned into several sub-graphs, meaning that the corresponding reducible $B_{\mathrm{asy}}^l$ can be transformed into a block diagonal matrix through elementary transformations. Then the property can be applied to each sub-matrix independently, and the stability still holds.

### 4.3. Spectral reduction

While bounding the spectral radius of attention matrices effectively alleviates the risk of gradient explosion, numerical instabilities still exist. Some feature representations may turn to opposite signs alternatively as a result of the multiplication with negative eigenvalues, suffering from large numerical variation. Therefore, reducing the absolute values of negative eigenvalues of $B^l$[1] is an effective approach to eliminate the influence.

To tackle the numerical issue, we adopt a similar renormalization trick proposed in [66]. Notice that the convolution operator in PLAM can be decomposed to

$$B^l = \left(\widetilde{D}^l\right)^{-1}\left[\exp\left(S^l\right)\odot(A+I)\right].$$

Considering that the original adjacency matrix of a graph doesn't contain self-loops, $\exp\left(S^l\right)\odot A$ provides no attention contribution on the main diagonal. The renormalization trick adds an identity matrix to the adjacency matrix, and thus fills non-negative values to the diagonal entries of the attention matrix before normalization. This revision enforces the output features of each node to contain information from its own input instead of only neighbors.

It has been proved in [66] that by augmenting the adjacency matrix with identity matrix (e.g., $\widetilde{A} = A + \lambda I, \lambda > 0$), the spectral radius of the corresponding normalized Laplacian matrix $\widetilde{L} = I - \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ turns smaller. This is equivalent to increasing negative eigenvalues of $\tilde{\Delta} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$. Therefore, the negative eigenvalues are pulled nearer to 0, and the instability is mitigated. On this basis, we can further prove that the conclusion still stands when filling arbitrary non-negative values to the diagonal entries. Therefore, it is concluded that the renormalization trick in PLAM is helpful for maintaining numerical stability. Numerical experiments are given in Section 5.3.4 which verify the effectiveness.

## 5. Experiments

In this section, we empirically validate the proposed method on several widely used semi-supervised node classification benchmarks, including citation networks and social networks. We first evaluate the effectiveness of PLAM comparing with state-of-the-art models on both transductive and inductive tasks. Meanwhile, as a complementary component to current approaches, we implement our self-attention module on several GCN-based structures and investigate the performance. We also extend our empirical analysis to several downstream tasks in the field of computer vision, including image classification and point cloud semantic segmentation. To further demonstrate the generalization power of PLAM, we also conduct experiments on citation networks with different dataset splits. Then, we present a spectral analysis of our method to testify the stability in the training procedure. Visualization result of the clustering property in the learned feature space is also provided. For simplicity, we will refer to the models with symmetric and asymmetric attention matrices as PLAM$_u$ (indicating **u**ndirected) and PLAM$_d$ (indicating **d**irected).

### 5.1. Node classification

#### 5.1.1. Datasets and experimental setup

**Datasets.** We conduct our experiments on five real-world graph datasets and two generated graphs, whose statistics are listed in Table 1. For transductive learning, we evaluate our method on

[1] In this subsection, we unify the denotation $B_{\mathrm{sym}}^l$ and $B_{\mathrm{asy}}^l$ as $B^l, S_{\mathrm{sym}}^l$ and $S_{\mathrm{asy}}^l$ as $S^l$, for simplicity without confusion.

**Table 1**
Dataset Statistics.

| Dataset | Transductive | | | | Inductive | | |
|---|---|---|---|---|---|---|---|
| | Cora | Citeseer | Pubmed | Reddit | PPI | PATTERN | CLUSTER |
| Graphs | 1 | 1 | 1 | 1 | 24 | 14 K | 12 K |
| Classes | 7 | 6 | 3 | 41 | 121(multilabel) | 2 | 6 |
| Avg. Nodes | 2,708 | 3,327 | 19,717 | 2372 | 232,965 | 117 | 117 |
| Avg. Edges | 5,429 | 4,732 | 44,338 | 11,606,919 | 34113 | 4749 | 4302 |
| Node feature | 1,433 | 3,703 | 500 | 602 | 50 | 3 | 7 |
| Training Nodes/Graphs | 140 Nodes | 120 Nodes | 60 Nodes | 152,410 Nodes | 20 Graphs | 10000 Graphs | 10000 Graphs |
| Validation Nodes/Graphs | 500 Nodes | 500 Nodes | 500 Nodes | 23,699 Nodes | 2 graphs | 2000 Graphs | 1000 Graphs |
| Test Nodes/Graphs | 1,000 Nodes | 1,000 Nodes | 1,000 Nodes | 55,334 Nodes | 2 graphs | 2000 Graphs | 1000 Graphs |

the Cora, Citeseer, Pubmed datasets, following the experimental setup in [53]. There are 20 nodes per class with labels to be used for training and all the nodes' features are available. However, the training algorithm has access to all of the nodes- feature vectors. The predictive power of the trained models is evaluated on 1000 test nodes, and we use 500 additional nodes for validation purposes. PPI, Reddit, Pattern and Cluster datasets are adopted for inductive learning. PPI [82] is a protein–protein interaction dataset that contains 20 graphs for training, 2 for validation and 2 for testing while testing graphs remain unobserved during training. Reddit is a larger dataset and sub-graphs are sampled as mini-batches in the training and test procedure. PATTERN and CLUSTER [11] are graph datasets generated by the stochastic block model, which are designed for node-level graph pattern recognition and graph clustering tasks.

**Baselines.** Our method is compared to several baselines including state-of-the-art graph neural networks. For transductive learning on the citation networks, we compare against GCN [27], GAT [58], DGI [59], FastGCN [4], GIN [69], AdaLNet [37], AGNN [56], MixGraph[60], SSA [77], and Deep-IRTarget [78]. To demonstrate the validity of projecting anchor and neighbor nodes into separate spaces, we also consider a simple PLAM model with $W_q = W_k$ and compare it with the full versions. In addition, We implement our method on several graph convolution structures, including SGC [66], TAGCN [9], APPNP [29], GenGNN [41] and Stronger GCN [39], to further evaluate the effectiveness of self-attention module. We follow publicly released implementation on each model and almost reproduce all reported performance from the original paper. For inductive learning on PPI and Reddit, we provide comparison results with GAT, DGI, SGC, GraphSAGE [20] and FastGCN. As GraphSAGE has variations for unsupervised and supervised learning, we report highest performance.

**Experimental Setup.** To ensure a fair comparison with other methods, we implement our module based on the original GCN structure. For Cora, Citeseer, Pubmed, we use two convolution layers with hidden dimension $h = 64$. For generated dataset PATTERN and CLUSTER, we determine the model hyperparameters with a budget of 100 k [11]. We apply $L_2$ regularization with $\lambda = 0.0005$ and use dropout on both layers and the attention matrix. For computation efficiency, we adopt $C = 8$ on all the datasets. For training strategy, we initialize weights using the initialization described in [18] and adopt an early stop if validation loss does not decrease for certain consecutive epochs, following [27]. The implementations of baseline models are based on the PyTorch-Geometric library [14] and Deep Graph Library [62] in all experiments.

*5.1.2. Transductive learning*

Table 2 presents the performance of our method and several state-of-the-art graph neural networks on transductive learning datasets. It can be observed that PLAM models outperform almost all the baseline models on all datasets. Notably, PLAM models achieve consistently higher accuracy than GAT and AGNN, with same model structure and hidden dimensions, showing superiority over other attention methods. Comparing with GAT and AGNN, our method significantly improves the generalization performance of traditional GCN while it can also boil down to an individual module that can be easily implemented in many graph convolution-based layers. We conduct experiments with GCN-based models and present test accuracies with and without self-attention module in Table 3. As we can see, each model achieves considerable improvements on all datasets after applying our method. Interestingly, we can also observe that our method is more effective on the Cora dataset. A plausible explanation is that attention matrix produces a marked effect on graphs with denser edges. If we compare the sparse factor of the adjacency matrix of each dataset,

$$\delta = \frac{\#\text{non} - \text{zero elements}}{\#\text{matrix elements}}, \tag{26}$$

we can find that the Cora dataset has the largest sparse factor (nearly twice larger than Citeseer and seven times larger than Pubmed). Therefore, a wider range of neighborhoods is involved in the message passing procedure, which is more suitable for the implementation of our method. Additionally, we can observe an obvious gap between the simple and full version PLAMs, indicating the effectiveness of treating anchor and neighbor nodes respectively.

We also investigate the training and test consumption of PLAM and compare with several representative baseline models from Table 2 and Table 3. To empirically testify the computation efficiency, we conduct experiments on Cora and report the training and test time of on a single RTX 2080 Ti GPU. The results are shown in Table 4. As we can observe, when combining with vanilla GCNs, the training and test time of our model is similar to GAT and AGNN and faster than APPNP. When combining with APPNP, the additional training and test time is minor when comparing to the original model.

**Table 2**
Test accuracy (%) on transductive learning datasets. We report mean values and standard deviations of the test accuracies in 30 independent experiments. The best results are highlighted with **boldface** and the asterisk (*) marker denotes that the difference is statistically significant by a t-test at significance level 0.05.

| Dataset | Citeseer | Cora | Pubmed |
|---|---|---|---|
| GCN [27] | 70.3 ± 0.4 | 81.5 ± 0.5 | 79.0 ± 0.4 |
| GIN [69] | 66.1 ± 0.9 | 77.6 ± 1.1 | 77.0 ± 1.2 |
| AdaLNet [37] | 68.7 ± 1.0 | 80.4 ± 1.1 | 78.1 ± 0.4 |
| FastGCN [4] | 68.8 ± 0.6 | 79.8 ± 0.3 | 76.8 ± 0.6 |
| DGI [59] | 71.8 ± 0.7 | 82.3 ± 0.6 | 76.8 ± 0.6 |
| MixGraph [60] | 72.8 ± 0.4 | 83.3 ± 0.6 | 79.0 ± 0.2 |
| GAT [58] | 72.5 ± 0.7 | 83.0 ± 0.6 | 78.5 ± 0.3 |
| AGNN [56] | 71.6 ± 0.5 | 82.7 ± 0.4 | 78.9 ± 0.4 |
| SSA [77] | 72.36 ± 0.45 | 83.22 ± 0.29 | 79.02 ± 0.11 |
| Deep-IRTarget [78] | 72.62 ± 0.71 | 82.91 ± 0.47 | 78.77 ± 0.56 |
| simple PLAM | 72.87 ± 0.38 | 83.75 ± 0.64* | 79.40 ± 0.44* |
| **PLAM$_u$** | **73.36 ± 0.24*** | 83.66 ± 0.42* | 79.06 ± 0.37 |
| **PLAM$_d$** | 73.25 ± 0.46* | **84.01 ± 0.23*** | **79.41 ± 0.65*** |

**Table 3**
Test accuracies (%) of state-of-the-art GCN-based models with and without self-attention module on citation datasets.

| Dataset | Citeseer | Cora | Pubmed |
|---|---|---|---|
| APPNP [29] | 70.53 ± 0.87 | 82.69 ± 0.82 | 79.41 ± 0.62 |
| **APPNP + PLAM** | **70.82 ± 0.52** | **83.31 ± 0.81*** | **79.61 ± 0.41** |
| TAGCN [9] | 71.01 ± 0.80 | 82.72 ± 0.54 | 79.42 ± 0.56 |
| **TAGCN + PLAM** | **71.77 ± 0.68*** | **83.30 ± 0.51*** | **79.61 ± 0.33** |
| SGC [66] | 71.9 ± 0.1 | 81.0 ± 0.0 | 78.9 ± 0.0 |
| **SGC + PLAM** | **72.19 ± 0.52*** | **82.22 ± 0.50*** | **79.60 ± 0.36*** |
| GenGNN [41] | 74.5 ± 0.1 | 82.9 ± 0.3 | 78.4 ± 0.6 |
| **GenGNN + PLAM** | **74.89 ± 0.29*** | **83.67 ± 0.33*** | **79.04 ± 0.56*** |
| Stronger GCN [41] | 73.9 ± 0.4 | 83.2 ± 0.5 | 80.1 ± 0.6 |
| **Stronger GCN + PLAM** | **74.11 ± 0.2*** | **83.92 ± 0.6*** | **80.20 ± 0.2** |

**Table 4**
Training and test time on Cora. We report mean values in 5 independent experiments.

| Method | Training Time(s) | Test Time(ms) | Accuracy(%) |
|---|---|---|---|
| GCN [27] | 1.8 | 1.9 | 81.5 ± 0.5 |
| GAT [58] | 5.4 | 3.3 | 83.0 ± 0.6 |
| AGNN [56] | 5.3 | 3.2 | 82.7 ± 0.4 |
| APPNP [29] | 9.8 | 13.6 | 82.7 ± 0.8 |
| GCN + PLAM (ours) | 5.7 | 3.2 | 84.0 ± 0.2 |
| APPNP + PLAM (ours) | 10.3 | 15.6 | 83.3 ± 0.8 |

#### 5.1.3. Inductive learning

Table 5 and Table 6 presents the comparison results on inductive learning datasets. It can be seen that PLAM models compare favourably with all the competitive baselines. On both PPI and Reddit dataset, PLAM models achieve 0.5%-1% higher on test Micro-F1 score. On the generated datasets, PLAM models achieve higher accuracy on CLUSTER and competitive results on PATTERN, where models are restricted with budgets. Notably, PLAM is 7% and 1.3% higher than GAT, demonstrating the effectiveness of the proposed attention structure.

Besides, it is worth noting that attention-based methods significantly outperform other methods on PPI dataset with a gap of more than 20%. Comparing with methods that depend heavily on the structure information of graph, attention-based methods further focus on the relationship between neighbors extracted from feature space, which is more robust among different graphs.

We also present the experimental results of combining our self-attention module with state-of-the-art inductive learning models, GraphSAGE and SGC, in Fig. 2. For GraphSAGE, we adopt the mean aggregator. The Micro F1 scores on Reddit during the training procedure of 50 epochs are reported. Both PLAM models achieve noticeable improvements, comparing with standard situations. We can also observe that applying self-attention module provides help on faster convergence.

**Table 5**
Test Micro-F1 Score on inductive learning datasets. We report the mean values and standard deviations of the test error in 5 independent experiments.

| Dataset | PPI | Reddit |
|---|---|---|
| GAT [58] | 97.3 ± 0.2 | 92.9 ± 0.1 |
| SGC [66] | 66.4 ± 0.0 | 94.9 ± 0.1 |
| GraphSAGE [20] | 61.2 ± 0.2 | 95.4 ± 0.2 |
| FastGCN [4] | – | 93.7 ± 0.1 |
| DGI [59] | 63.8 ± 0.2 | 94.0 ± 0.1 |
| **PLAM$_u$** | 97.69 ± 0.32 | **96.08 ± 0.04*** |
| **PLAM$_d$** | **98.23 ± 0.08*** | 95.9 ± 0.42* |

**Table 6**
Test accuracies (%) on generated graph datasets. We report mean values and standard deviations of the test accuracies in 5 independent experiments.

| Dataset | Pattern | | Cluster | |
|---|---|---|---|---|
| | #Param | Acc | #Param | Acc |
| GCN [27] | 100,923 | 74.36 ± 1.59 | 101,655 | 47.82 ± 4.91 |
| GIN [69] | 100,884 | **98.25 ± 0.38** | 103,544 | 52.54 ± 1.03 |
| GraphSAGE [20] | 98,607 | 81.25 ± 3.84 | 99,139 | 53.90 ± 4.12 |
| GAT [58] | 109,936 | 90.72 ± 2.04 | 110,700 | 54.12 ± 1.21 |
| GatedGCN [1] | 104,003 | 97.24 ± 1.19 | 104,355 | 54.20 ± 3.58 |
| **PLAM** | 100,421 | 98.18 ± 0.41 | 88,303 | **55.45 ± 0.93*** |

### 5.2. Downstream tasks

We extend our empirical evaluation to several downstream applications - 2D image classification and 3D point cloud semantic segmentation. Experiment results show great potential of PLAM on computer vision tasks.

#### 5.2.1. Image classification

**Datasets and experimental setup.** We use the popular MNIST and CIFAR10 image classification datasets. The original images are converted to graphs using super-pixels extracted with the SLIC technique [31], and the nodes are connected with the $k$-nearest neighbor algorithm. We follow the setup in [11] and train the models with Adam optimizer with learning rate decay strategy. Similar to the generated datasets, we restrict model parameters with a budget of 100 k. Data statistics for two datasets are presented in Table 7. Edge features are not applied in this comparison.

**Performances.** Table 8 shows experimental results on MNIST and CIFAR10 datasets. As we can observe, PLAM achieves higher performances on all the settings and shows a significant improvement of 2% on CIFAR10 datasets, which demonstrates the efficacy of the proposed method.

Comparing to citation and social networks, PLAM seems to have a greater advantage on the 2D image datasets. We argue that the reason is twofold: 1) Image classification is essentially a graph classification task, where a Readout function is adopted to acquire the global information. Therefore, the quality of the global feature is greatly compromised when representations of different nodes become indistinguishable. By applying PLAM, the over-smoothing problem is alleviated and therefore enhance the model performance. 2) In citation networks, edges represent the academic connection between the nodes, which means that the connected nodes belong to the same class with high probability and message passing along these edges are reliable. However, in the image classification task, the useful information is largely represented in the *foreground* of the input image. Nodes on the edges of the foreground will be inevitably connected with useless *background* nodes. Traditional GCNs treat all neighbors with equal status in the aggregation function, leading to the spread of background information. With the self-attention mechanism, connections within foreground nodes will be further addressed.

An interesting observation is that the performances of PLAM models are slightly better without residual connection in the network structure. In fact, if we set the attention coefficients between the anchor node and neighbor nodes to be 0 and the anchor node towards itself to be 1, the PLAM layer corresponds to a residual connection. Therefore, information from the previous layer has already been considered in the original structure and the extra residual connection is not necessarily required.

#### 5.2.2. Point cloud semantic segmentation

**Datasets and experimental setup.** We evaluate PLAM on the S3DIS Semantic Segmentation dataset and apply the module to a ResGCN-28 structure proposed by [34]. The model contains a GCN backbone block, a fusion block and a MLP prediction block,

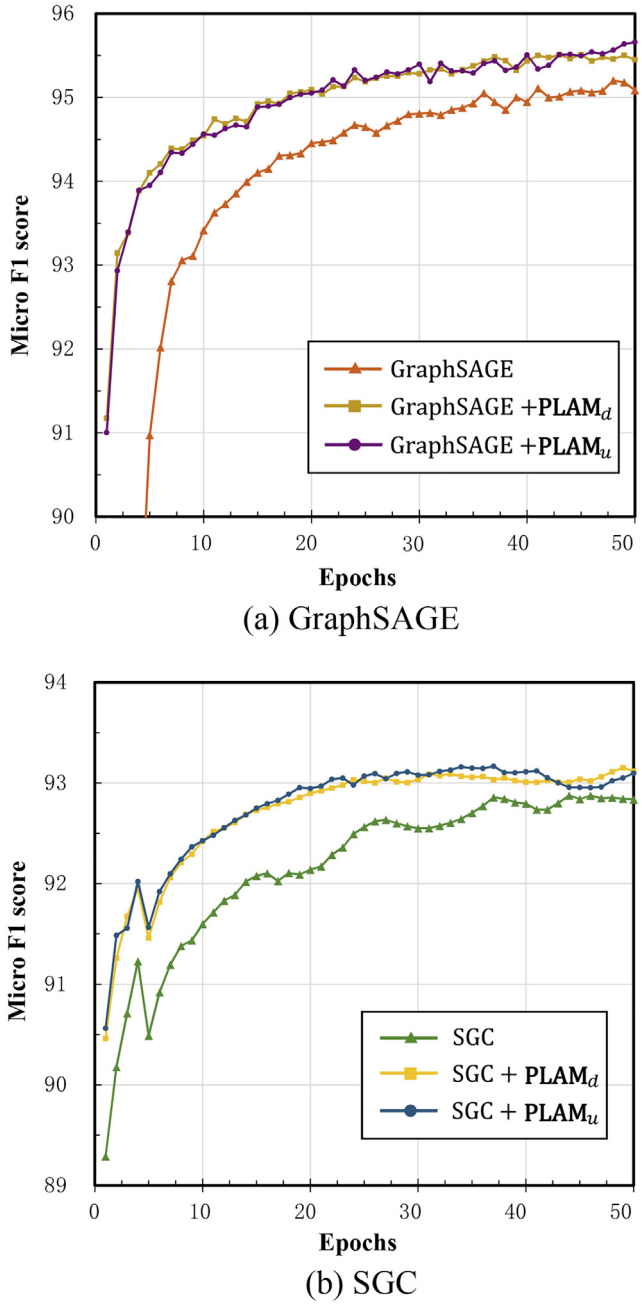



**Fig. 2.** Performances of directed and undirected self-attention modules (abbreviated as PLAM$_d$ and PLAM$_u$) on Reddit combined with (a) GraphSAGE and (b) SGC.

where we substitute the original GCN layers with PLAM layers. The training configurations are identical with [34]. For simplicity, we refer to our model as ResPLAM.

**Performances.** We compare ResPLAM to state-of-the-art baselines and Table 9 shows the empirical results on S3DIS datasets. As we can see, PLAM effectively enhances the performance based on ResGCN-28 and outperforms other GNN-based models. Additionally, PLAM achieves the highest result in 8 of 13 classes. These experimental results prove that our PLAM can be implemented in deeper and wider GCN-based networks, showing the potential of applying PLAM on more sophisticated tasks.

**Table 8**
Test accuracies (%) on image classification datasets. We report mean values and standard deviations of the test accuracies in 5 independent experiments.

| Dataset | Model | #Param | Acc (Residual) | Acc (No Residual) |
|---|---|---|---|---|
| MNIST | GCN [27] | 101,365 | 89.99 ± 0.15 | 89.05 ± 0.21 |
| | GraphSAGE [20] | 102,691 | 97.09 ± 0.02 | 97.20 ± 0.17 |
| | GAT [58] | 110,400 | 95.62 ± 0.13 | 95.56 ± 0.16 |
| | GatedGCN [1] | 104,217 | 97.37 ± 0.06 | 97.47 ± 0.13 |
| | **PLAM** | 108,697 | **97.98 ± 0.10*** | **97.99 ± 0.11*** |
| CIFAR10 | GCN [27] | 101,657 | 54.46 ± 0.10 | 51.64 ± 0.45 |
| | GraphSAGE [20] | 102,907 | 65.93 ± 0.30 | 66.08 ± 0.24 |
| | GAT [58] | 110,704 | 65.40 ± 0.38 | 65.48 ± 0.33 |
| | GatedGCN [1] | 104,357 | 69.19 ± 0.28 | 69.37 ± 0.48 |
| | **PLAM** | 108,697 | **71.49 ± 0.37*** | **71.94 ± 0.28*** |

### 5.3. Discussion

#### 5.3.1. Random splits

As illustrated in [54], using the same train/validation/test splits of the same datasets precludes a fair comparison of different architectures. Therefore, we follow the setup in [54] and evaluate the performance of PLAM on three citation networks with random splits. Empirically, for each dataset, we use 20 labelled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the test set. For every model, we pick the hyperparameter that achieves the best average accuracy on Cora and CiteSeer datasets and applied to the Pubmed dataset.

Table 10 shows the results on three citation networks under the random split setting. As we can observe, PLAM consistently achieves higher performances on all the datasets. On Citeseer, the test accuracy of PLAM is nearly comparable to the original split, while most of the baselines suffer from a serious decline.

#### 5.3.2. Ablation – module parts

Practically, three parts in our PLAM contributes to the final performance: (1) separate query and key projection weights; (2) the inner-product attention computation to achieve higher-level information; (3) the structural encoding technique. To better show the effectiveness of each part, we perform ablations on transductive datasets, Citeseer, Cora, Pubmed and summarize the results in Table 11. As we can observe, all three parts contribute to the better performance. Comparably, leveraging higher-level information are proved to be most effective, where a large performance drop can be observed without it.

#### 5.3.3. Ablation – C

As we have illustrated in Section 3.3, hyperparameter C is adopted to reduce the dimension of the attention learning matrix $W_q, W_k$ and consequently reduce the additional parameters of PLAM. To fully investigate the influence of C, we conduct experiments on PLAM models with different Cs on the citation networks. As we can observe in Fig. 5, even with a large value of C, PLAM still consistently outperforms GAT. Meanwhile, the improvement

**Table 7**
Dataset Statistics.

| Dataset | #Graphs | #Classes | Avg. Nodes | Avg. Edges | Node feat. (dim) | Edge feat. (dim) |
|---|---|---|---|---|---|---|
| MNIST | 70,000 | 10 | 70.57 | 564.53 | Pixel + Coord (3) | Node Dist(1) |
| CIFAR10 | 60,000 | 10 | 117.63 | 941.07 | Pixel[RGB]+Coord (5) | Node Dist(1) |

**Table 9**
Experiments of ResPLAM-28 with state-of-the-art methods on S3DIS Semantic Segmentation. ResPLAM model follows the architecture adopted in [34] and substitute the GCN layers with PLAM layers. We report per-class results and mean IoU (mIoU) score.

| Method | mIOU | ceiling | floor | wall | beam | column | window | door | table | chair | sofa | bookcase | board | clutter |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PointNet [47] | 47.6 | 88.0 | 88.7 | 69.3 | 42.4 | 23.1 | 47.5 | 51.6 | 54.1 | 42.0 | 9.6 | 38.2 | 29.4 | 35.2 |
| MS + CU [12] | 47.8 | 88.6 | 95.8 | 67.3 | 36.9 | 24.9 | 48.6 | 52.3 | 51.9 | 45.1 | 10.6 | 36.8 | 24.7 | 37.5 |
| G + RCU [12] | 49.7 | 90.3 | 92.1 | 67.9 | **44.7** | 24.2 | 52.3 | 51.2 | 58.1 | 47.4 | 6.9 | 39.0 | 30.0 | 41.9 |
| PointNet++ [48] | 53.2 | 90.2 | 91.7 | 73.1 | 42.7 | 21.2 | 49.7 | 42.3 | 62.7 | 59.0 | 19.6 | 45.8 | 48.2 | 45.6 |
| 3DRNN + CF [71] | 56.3 | 92.9 | 93.8 | 73.1 | 42.5 | 25.9 | 47.6 | 59.2 | 60.4 | **66.7** | 24.8 | 57.0 | 36.7 | 51.6 |
| DGCNN [65] | 56.1 | – | – | – | – | – | – | – | – | – | – | – | – | – |
| ResGCN-28 [34] | 60.0 | 93.1 | 95.3 | **78.2** | 33.9 | 37.4 | 56.1 | 68.2 | **64.9** | 61.0 | 34.6 | 51.5 | **51.1** | 54.4 |
| **ResPLAM-28** | **61.3** | **93.5** | **98.0** | 73.3 | 30.0 | **38.5** | **65.3** | **70.2** | 62.5 | 63.9 | **36.6** | **60.9** | 46.7 | **57.0** |

**Table 10**
Test accuracy (%) on transductive learning datasets with random slits. We report mean values and standard deviations of the test accuracies over 100 train/validation/test splits.

| Dataset | Citeseer | Cora | Pubmed |
|---|---|---|---|
| GCN [27] | 71.9 ± 1.9 | 81.5 ± 1.3 | 77.8 ± 2.9 |
| GAT [58] | 71.4 ± 1.9 | 81.8 ± 1.3 | 78.7 ± 2.3 |
| MoNet [44] | 71.2 ± 2.0 | 81.3 ± 1.3 | 78.6 ± 2.3 |
| GraphSAGE [20] | 71.6 ± 1.9 | 79.2 ± 7.7 | 77.4 ± 2.2 |
| **PLAM** | **72.6 ± 1.9** | **82.3 ± 1.4** | **78.9 ± 1.7** |

**Table 11**
Ablation study on each part of PLAM. We report mean values and standard deviations of the test accuracies in 30 independent experiments.

| Dataset | Citeseer | Cora | Pubmed |
|---|---|---|---|
| w/o separate qk weights | 72.87 ± 0.38 | 83.75 ± 0.64 | 79.40 ± 0.44 |
| w/o higher-level information | 72.62 ± 0.40 | 83.66 ± 0.36 | 79.14 ± 0.42 |
| w/o structural encoding | 73.03 ± 0.51 | 83.72 ± 0.48 | 79.22 ± 0.37 |
| **PLAM** | **73.25 ± 0.46** | **84.01 ± 0.23** | **79.41 ± 0.65** |

achieved by decreasing C gradually decreases and C = 8 makes a good balance between effectiveness and efficiency.

### 5.3.4. Spectral analysis

To testify the effectiveness of the renormalization trick, we conduct experiments on the eigenvalues of convolution operators. In Fig. 3, we show all the eigenvalues of the convolution operator from the hidden layer. The two curves represent eigenvalues with and without the trick, respectively. As we can observe, the negative eigenvalues are significantly increased. Apart from bounding the norm of spectral radius to 1, it also shrinks the smallest eigenvalue from −1 to approximately −0.5, which contributes to eliminating the effect of negative coefficients and improve the numerical stability. Besides, eigenvalues with the renormalization trick seem to have lower variances. (See Fig. 4).

### 5.3.5. Visualization

To analyze the effectiveness of the proposed method qualitatively, we follow [58] and provide the 2D t-SNE [42] visualization of Cora dataset feature representations from the first hidden layer

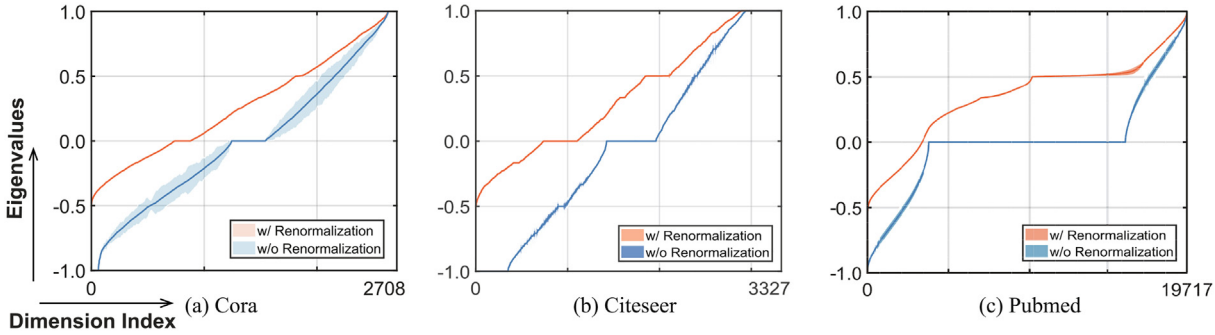



**Fig. 3.** Sorted eigenvalues of the hidden layer and their standard deviations on citation networks with and without renormalization trick.

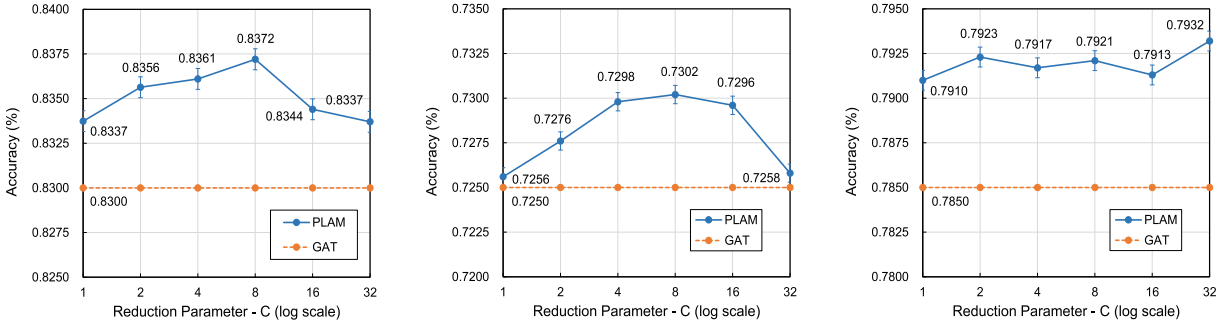
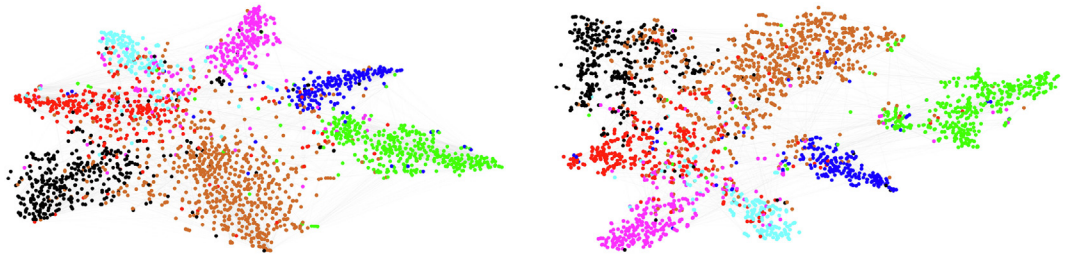



**Fig. 4.** Performances of PLAM models with varying reduction parameter Cs. Use GAT for comparison.



(a) PLAM (b) GAT

**Fig. 5.** A t-SNE plot of the feature representations of a PLAM model's hidden layer on the Cora dataset. Node colours denote classes, and the edge thickness indicates the normalized undirected attention coefficients.

of PLAM and GAT models(Fig. 5). In the meantime, we visualize the strengths of the attention coefficients according to the augmented convolution operator as the line thickness. As we can observe, by adopting the novel self-attention module, clusters corresponding to different classes are generally separated and the attentions are more significant within clusters, indicating the discriminative capability of the proposed model.

## 6. Limitations

In this paper, we develop a flexible attention-based module to address the limitations in existing graph neural networks. Although the proposed module can effectively improve the performances of graph models and alleviate over-smoothing, the dot-product attention mechanism inevitably introduces additional computational cost. Researches have also investigated the possibility of efficient attention modes, e.g., Linformer[63], Reformer[28], which may serve as the substitution of current attention formulation. In conclusion, improving the efficiency of the training process will be an interesting future direction.

## 7. Conclusion

In this paper, we have presented the *Plug-in Attention Module (PLAM)*, a novel network architecture learning dynamic convolution filters through the self-attention mechanism. To acquire a conveniently applied module with strong representation power, we introduce the query-key mechanism which transfers the nodes to two different sub-spaces respectively. A novel structural encoding technique is proposed to inject local and global structure information. Our PLAM models have achieved state-of-the-art performances on both transductive and inductive tasks and enhance the performances on other GCN-based models as well. We have also demonstrated that PLAM maintains the stability in the training procedure and alleviates numerical variation through the renormalization trick. Furthermore, experiments on 2D image datasets and point cloud dataset have shown the potential applications of PLAM on computer vision tasks.

Note that, the form of inner production between query-key pairs adopted in our model is not unique. In the future, we will discuss the differences of various attention calculation models. Also, we can explore the possibility of extending the self-attention mechanism to graph classification tasks as well as more complex data such as hyper-graphs.

## CRediT authorship contribution statement

**Xuran Pan:** Conceptualization, Methodology, Software, Writing - original draft. **Shiji Song:** Supervision, Writing - original draft. **Yiming Chen:** Software, Validation. **Liejun Wang:** Supervision, Writing - review & editing. **Gao Huang:** Supervision, Writing - review & editing.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

This work is supported in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grants 2018AAA0101604, the National Natural Science Foundation of China under Grants 61906106 and 62022048, and the Guoqiang Institute of Tsinghua University.

## References

[1] X. Bresson, T. Laurent, Residual gated graph convnets, CoRR abs/1711.07553 (2017).
[2] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: International Conference on Learning Representations, 2014.
[3] H. Cai, V.W. Zheng, K.C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (2018) 1616–1637.
[4] J. Chen, T. Ma, C. Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, International Conference on Learning Representations (2018).
[5] J. Chen, J. Zhu, L. Song, Stochastic training of graph convolutional networks with variance reduction, in: International Conference on Machine Learning, 2017.
[6] S. Das, S. Chakravarthy, Duplicate reduction in graph mining: Approaches, analysis, and evaluation, IEEE Trans. Knowl. Data Eng. 30 (2018) 1454–1466.
[7] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Advances in Neural Information Processing Systems (2016) 3844–3852.
[8] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018).
[9] J. Du, S. Zhang, G. Wu, J.M. Moura, S. Kar, Topology adaptive graph convolutional networks, CoRR abs/1710.10370 (2017).
[10] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, Advances in Neural Information Processing Systems (2015) 2224–2232.
[11] V.P. Dwivedi, C.K. Joshi, T. Laurent, Y. Bengio, X. Bresson, Benchmarking graph neural networks, 2020. arXiv preprint arXiv:2003.00982..
[12] F. Engelmann, T. Kontogianni, A. Hermans, B. Leibe, Exploring spatial context for 3d semantic segmentation of point clouds, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 716–724.
[13] F. Feng, X. He, J. Tang, T. Chua, Graph adversarial training: Dynamically regularizing based on graph structure, IEEE Trans. Knowl. Data Eng. (2019), 1-1.
[14] M. Fey, J.E. Lenssen, Fast graph representation learning with pytorch geometric, CoRR abs/1903.02428 (2019).
[15] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, H. Lu, Dual attention network for scene segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3146–3154.

[16] H. Gao, S. Ji, Graph u-nets, in: International Conference on Machine Learning, 2019.
[17] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, International Conference on Machine Learning (2017) 1263–1272.
[18] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[19] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2005, pp. 729–734..
[20] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in Neural Information Processing Systems (2017) 1024–1034.
[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[22] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, CoRR abs/1506.05163 (2015).
[23] L. Hong, L. Zou, X. Lian, P.S. Yu, Subgraph matching with set similarity in a large graph database, IEEE Trans. Knowl. Data Eng. 27 (2015) 2507–2521.
[24] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[25] B. Jiang, Z. Zhang, D. Lin, J. Tang, B. Luo, Semi-supervised learning with graph learning-convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11313–11320.
[26] A. Kapoor, A. Galstyan, B. Perozzi, G.V. Steeg, H. Harutyunyan, K. Lerman, N. Alipourfard, S. Abu-El-Haija, Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing, in: International Conference on Machine Learning, 2019.
[27] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.
[28] Kitaev, N., Kaiser, Ł., Levskaya, A., 2020. Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451.
[29] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, in: International Conference on Learning Representations, 2018.
[30] J. Klicpera, S. Weißenberger, S. Günnemann, Diffusion improves graph learning, 2019. http://arxiv.org/abs/1911.05485arXiv:1911.05485..
[31] B. Knyazev, G.W. Taylor, M. Amer, Understanding attention and generalization in graph neural networks, Advances in Neural Information Processing Systems (2019) 4204–4214.
[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems (2012) 1097–1105.
[33] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in: International Conference on Machine Learning, 2019.
[34] G. Li, M. Muller, A. Thabet, B. Ghanem, Deepgcns: Can gcns go as deep as cnns?, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp 9267–9276.
[35] Q. Li, Z. Han, X.M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
[36] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, in: International Conference on Learning Representations, 2016.
[37] R. Liao, Z. Zhao, R. Urtasun, R.S. Zemel, Lanczosnet: Multi-scale deep graph convolutional networks, CoRR abs/1901.01484 (2019).
[38] G. Liu, Y. Liu, K. Zheng, A. Liu, Z. Li, Y. Wang, X. Zhou, Mcs-gpm: Multi-constrained simulation based graph pattern matching in contextual social graphs, IEEE Trans. Knowl. Data Eng. 30 (2018) 1050–1064.
[39] S. Luan, M. Zhao, X.W. Chang, D. Precup, Break the ceiling: Stronger multi-scale deep graph convolutional networks, Advances in neural information processing systems (2019) 10945–10955.
[40] J. Ma, P. Cui, K. Kuang, X. Wang, W. Zhu, Disentangled graph convolutional networks, International Conference on Machine Learning (2019) 4212–4221.
[41] J. Ma, W. Tang, J. Zhu, Q. Mei, A flexible generative framework for graph-based semi-supervised learning, CoRR abs/1905.10769 (2019).
[42] L.V.D. Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[43] M.T. Mills, N.G. Bourbakis, Graph-based methods for natural language processing and understanding-a survey and analysis, IEEE Trans. Syst., Man, Cybern.: Syst. 44 (2014) 59–71, https://doi.org/10.1109/TSMCC.2012.2227472.
[44] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model cnns, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5115–5124.
[45] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, International Conference on Machine Learning (2016) 2014–2023.
[46] S. Pan, J. Wu, X. Zhu, Cogboost: Boosting for fast cost-sensitive graph classification, IEEE Trans. Knowl. Data Eng. 27 (2015) 2933–2946.
[47] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.
[48] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, Advances in neural information processing systems (2017) 5099–5108.
[49] S. Qi, W. Wang, B. Jia, J. Shen, S.C. Zhu, Learning human-object interactions by graph parsing neural networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 401–417.
[50] M. Qu, Y. Bengio, J. Tang, Gmnn: Graph markov neural networks, in: International Conference on Machine Learning, 2019.
[51] R.A. Rossi, R. Zhou, N.K. Ahmed, Deep inductive graph representation learning, IEEE Trans. Knowl. Data Eng. 32 (2020) 438–452.
[52] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Networks 20 (2008) 61–80.
[53] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (2008), 93-93.
[54] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, CoRR abs/1811.05868 (2018).
[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[56] K.K. Thekumparampil, C. Wang, S. Oh, L.J. Li, Attention-based graph neural network for semi-supervised learning, CoRR abs/1803.03735 (2018).
[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in Neural Information Processing Systems (2017) 5998–6008.
[58] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[59] P. Veličković, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, International Conference on Learning Representations (2019).
[60] V. Verma, M. Qu, A. Lamb, Y. Bengio, J. Kannala, J. Tang, Graphmix: Regularized training of graph neural networks for semi-supervised learning, 2019. arXiv preprint arXiv:1909.11715..
[61] H. Wang, J. Wu, X. Zhu, Y. Chen, C. Zhang, Time-variant graph classification, IEEE Trans. Syst., Man, Cybern.: Syst. 50 (2020) 2883–2896, https://doi.org/10.1109/TSMC.2018.2830792.
[62] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A.J. Smola, Z. Zhang, Deep graph library: Towards efficient and scalable deep learning on graphs, ICLR Workshop on Representation Learning on Graphs and Manifolds (2019), URL: https://arxiv.org/abs/1909.01315.
[63] S. Wang, B.Z. Li, M. Khabsa, H. Fang, H. Ma, Linformer: Self-attention with linear complexity, 2020. arXiv preprint arXiv:2006.04768..
[64] W. Wang, X. Lu, J. Shen, D.J. Crandall, L. Shao, Zero-shot video object segmentation via attentive graph neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9236–9245.
[65] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Transactions on Graphics (TOG) 38 (2019) 1–12.
[66] F. Wu, T. Zhang, A.H.d. Souza Jr, C. Fifty, T. Yu, K.Q. Weinberger, Simplifying graph convolutional networks, in: International Conference on Machine Learning, 2019.
[67] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 346–353.
[68] W. Wu, B. Li, L. Chen, X. Zhu, C. Zhang, k -ary tree hashing for fast graph classification, IEEE Trans. Knowl. Data Eng. 30 (2018) 936–949.
[69] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, CoRR abs/181000826 (2018).
[70] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 7370–7377.
[71] X. Ye, J. Li, H. Huang, L. Du, X. Zhang, 3d recurrent neural networks with context fusion for point cloud semantic segmentation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 403–417.
[72] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, Advances in Neural Information Processing Systems (2018) 4800–4810.
[73] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: International Conference on Machine Learning, 2019.
[74] X. Yuan, J. Guo, X. Hao, H. Chen, Traffic sign detection via graph-based ranking and segmentation algorithms, IEEE Trans. Syst., Man, Cybern.: Syst. 45 (2015) 1509–1521, https://doi.org/10.1109/TSMC.2015.2427771.
[75] S. Yun, M. Jeong, R. Kim, J. Kang, H.J. Kim, Graph transformer networks, Advances in Neural Information Processing Systems (2019) 11983–11993.
[76] H. Zhang, I. Goodfellow, D. Metaxas, A. Odena, Self-attention generative adversarial networks, in: International Conference on Machine Learning, 2019.
[77] Q. Zhang, L. Ge, R. Zhang, G.I. Metternicht, Z. Du, J. Kuang, M. Xu, Deep-learning-based burned area mapping using the synergy of sentinel-1&2 data, Remote Sens. Environ. 264 (2021) 112575.
[78] R. Zhang, L. Xu, Z. Yu, Y. Shi, C. Mu, M. Xu, Deep-irtarget: An automatic target detector in infrared imagery using dual-domain feature extraction and allocation, IEEE Trans. Multimedia (2021).

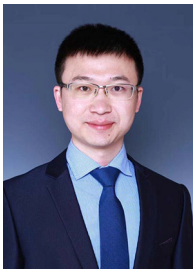
[79] Y. Zhang, S. Pal, M. Coates, D. Ustebay, Bayesian graph convolutional neural networks for semi-supervised classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 5829–5836.
[80] K. Zhou, Q. Song, X. Huang, D. Zha, N. Zou, X. Hu, Multi-channel graph convolutional networks, 2019. arXiv preprint arXiv:1912.08306..
[81] M. Zitnik, M. Agrawal, J. Leskovec, Modeling polypharmacy side effects with graph convolutional networks, Bioinformatics 34 (2018) i457–i466.
[82] M. Zitnik, J. Leskovec, Predicting multicellular function through multi-layer tissue networks, Bioinformatics 33 (2017) i190–i198.

**Xuran Pan** received his B.S. degrees in Department of Automation, Tsinghua University in 2018. From 2018 he started his Ph.D. at Institute of System Integration, Department of Automation, Tsinghua University, China. His main research interests include deep learning, especially in computer vision and graph neural network.

**Shiji Song** received the Ph.D. degree in mathematics from the Department of Mathematics, Harbin Institute of Technology, Harbin, China, in 1996. He is currently a Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored over 180 research papers. His current research interests include system modeling, optimization and control, computational intelligence, and pattern recognition.

**Yiming Chen** received his B.S. degrees in Department of Automation, Tsinghua University in 2015. From 2015 he started his Ph.D. at Institute of System Integration, Department of Automation, Tsinghua University, China. His main research interests include deep learning, especially in computer vision and graph convolutional network.

**Liejun Wang** received the Ph.D. degree from the School of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an, China, in 2012. He is currently a Professor with Xinjiang University, Ürümqi, China. His current research interests include computer vision, natural language processing, and wireless sensor.



**Gao Huang** received the B.S. degree from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2009, and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, in 2015. He was a Visiting Research Scholar with the Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA, in 2013 and was a Post-Doctoral Researcher with Department of Computer Science, Cornell University, Ithaca, USA from 2015 to 2018. He is currently an assistant professor at the Department of Automation, Tsinghua University. His research interests include machine learning and deep learning.