Knowledge-Based Systems 279 (2023) 110923

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

SGCN: A scalable graph convolutional network with graph-shaped kernels and multi-channels

Zhenhua Huang [a,1], Wenhao Zhou [a,1], Kunhao Li [a,1], Zhaohong Jia [a,b,*]

[a] School of Internet, Anhui University, Feixi Road, Hefei, 230039, China
[b] Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, Jiulong Road, Hefei, China

ARTICLE INFO

Article history:
Received 2 June 2023
Received in revised form 14 August 2023
Accepted 17 August 2023
Available online 20 August 2023

Keywords:
Graph neural network
Graph-shaped kernels
Node classification
Large graph

ABSTRACT

Graph neural networks (GNNs) have demonstrated great success in graph processing. However, current message-passing-based GNNs have limitations in terms of feature aggregations and update mechanisms that rely on a fixed mode, resulting in inadequate representations of the neighborhood structure's richness. Furthermore, the convolution layer in most GNNs lacks flexibility and multiple channels compared to convolutional neural networks (CNNs), which employ multiple channels, where different kernels capture shared patterns of images. To address these limitations, we propose a novel scalable graph convolution network (SGCN) to enhance structural information learning by introducing an expressive feature aggregation mechanism. Drawing inspiration from the design of CNNs, the SGCN includes graph-shaped kernels that perform multichannel convolutions on the subgraph structure. A sampler based on degree centrality is employed to simplify the computation costs, and extensive experiments demonstrate that the SGCN achieved state-of-the-art performances on graph datasets with various scales (with accuracy improvements of up to 5.38%).

1. Introduction

Graph neural networks (GNNs), which are deep learning frameworks for processing graph-structured data, have been successfully applied to solve different tasks, such as network analysis [1,2], recommendation systems [3,4], biochemistry [5,6], traffic prediction [7,8], location prediction [9], knowledge tracing [10], computer vision [11,12], and natural language processing [13,14].

Message passing is a widely adopted aggregation architecture in various classic and advanced GNNs [15], such as the GCN [16], which utilizes a simple and efficient graph convolution operation. The GAT [17] introduces attention mechanisms for more expressive representations, GraphSAGE [18] employs neighborhood sampling to handle large-scale graphs, and LightGCN [19] focuses on collaborative filtering-based recommendation tasks. Other works based on message passing also include UniMP [20], FusedGAT [21], and the Anti-Symmetric DGN [22]. GNNs based on message passing aggregate and update nodes' representation vectors by considering information from their neighbors and achieving satisfactory performances for node representation. Fig. 1 illustrates that message passing operates on the features of nodes within their respective neighborhoods. However, the existing graph models suffer from a limitation in that they fail to capture the common structural patterns inherent in graphs within their parameters. Consequently, this limitation hampers their ability to effectively learn and incorporate information about the underlying graph structure [23].

In contrast to GNNs, convolutional neural networks (CNNs) [24] are designed to capture the local structure in images using convolution kernels, which refer to a pattern [25]. CNNs offer the advantage of flexible kernel design, where different kernels are used to capture distinct patterns in different channels. Each channel shares the same kernel, allowing for efficient parameter sharing and feature extraction. Existing message-passing-based GNNs face challenges in directly learning and leveraging such patterns in graphs despite the presence of shared meaningful substructures or patterns. For instance, the GAT overcomes the problem of single-channel convolution by utilizing attention mechanisms to aggregate features from subgraphs, while the pattern is singular across the neighbors. Nevertheless, the concept of flexible convolutional kernels and multiple channels in CNNs can inspire the design of novel GNN frameworks [26]. By incorporating the idea of flexible convolutional kernels and multiple channels from CNNs, we explore new possibilities for enhancing GNN architectures to learn patterns within graphs.

Furthermore, message-passing-based GNNs face significant challenges in terms of computation and storage when processing large graphs. Several models have been proposed for large

* Corresponding author.
E-mail addresses: zhhuangscut@gmail.com (Z. Huang), wenhaozhou1112@gmail.com (W. Zhou), kunhomlihf@gmail.com (K. Li), zhjia@mail.ustc.edu.cn (Z. Jia).
[1] Equal First Authors.

https://doi.org/10.1016/j.knosys.2023.110923
0950-7051/

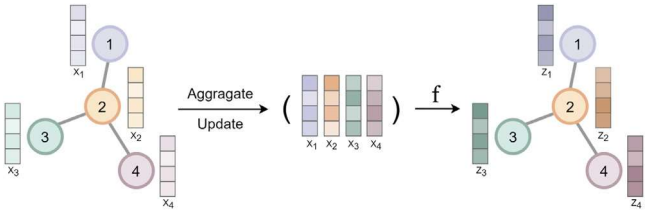


Fig. 1. Message-passing-based GNN framework. Each node in the graph aggregates and updates its representation, and f represents a multilayer perceptron.

graphs, such as SGC [27], which simplifies the graph convolution operation by removing the nonlinear activation function. FastGCN [28] introduces a layer-wise sparse approximation technique that accelerates the training process, and the ASGCN [29] adapts the sampling strategy based on the local graph structure to optimize computational efficiency. ClusterGCN [30] leverages clustering techniques to partition the graph and parallelize the convolutional operations. GraphSAINT [31]. These models simplify benchmark models or adopt node samplings to improve computational efficiency. However, based on our experimental statistics, these GNNs lead to inconspicuous improvements in general graphs, which highlights the need to find a balance between enhancing performance on large graphs and ensuring the universality of GNNs.

To address the above problems, we propose a novel **S**calable **G**raph **C**onvolution **N**etwork (SGCN). The SGCN consists of multiple scalable convolution layers (SCLs), which employ stacked convolution kernels with different graph shapes. The parameters of a kernel are shared within a whole graph. During convolution operations, the representations of nodes covered by a kernel are multiplied by corresponding weights, which enables the kernel to learn local structure patterns from substructures within the graph. To reduce the computational cost, we introduce a sampler based on degree centrality [32] to sample graph substructures.

In Fig. 2, the graph-shaped kernels in the SCL exhibit different shapes. To generalize these kernels with neighbors, we use W as a representation. In this paper, we only consider a simple graph-shaped kernel where a single neighboring node surrounds the central node. W^2 refers to $W^2(1)$, and W^3 refers to $W^3(1)$. By convolving this kernel with subgraphs consisting of corresponding nodes in the graph, we can capture a wide range of structural features present in the neighborhood. The parameters of the kernel are shared within a channel, and different kernels capture distinct patterns. This flexible kernel design allows the SGCN to adapt to different scale graphs. Extensive experiments demonstrate that the SGCN achieves state-of-the-art performance in node classification tasks across nine real-world graph datasets.

Our contributions are as follows:

- We propose a novel scalable graph convolution network (SGCN) that addresses limitations in existing graph neural networks. The SGCN overcomes the challenges of capturing diverse patterns in neighborhoods and achieving balanced performance across graphs of varying scales. By utilizing flexible graph convolution kernels of different types and sizes, the SGCN enables the learning of shared structural patterns in graphs, making it applicable to graphs of any scale.
- Extensive experiments demonstrate that the SGCN outperforms existing GNNs in node classification tasks on nine real-world graph datasets, achieving an improvement of up to 5.38%, which offers a promising solution to graph-based learning tasks.

The subsequent structure of this paper is organized as follows: In Section 2, we review related works about graph neural networks, kernel methods on graphs, and the design of graph convolutional kernels. Section 3 defines the problem of node classification and explains the message-passing mechanism, graph convolutional network, and over-smoothing problems. In Section 4, details of the proposed model are introduced, and Section 5 presents the experimental setup. Extensive experiments and analysis are presented in Section 6, and we summarize this study in Section 7.

2. Related works

This section provides recent works related to our research, focusing on graph neural networks, kernel methods on graphs, and convolutional neural kernels.

2.1. Graph neural networks

Graph neural networks (GNNs) represent a form of deep neural network that is specifically tailored for graph-structured data processing. Several noteworthy examples of GNNs include ChebNet [33], which relies on neighbor message passing for information aggregation, the GCN [16], which employs a similar neighbor-to-neighbor passage of information, and the GAT [17], which innovatively integrates a self-attention mechanism to aggregate node features using adjustable weights. GraphSAGE [18] utilizes localized neighborhood sampling and aggregation to generate novel data embeddings, while the DHGNN [34] leverages both shared and specific hypergraph convolution layers, coupled with attention mechanisms, to effectively synthesize diverse information sources and combine dual node embeddings. Further examples include FusedGAT [21], which optimizes the GAT model by trimming computation and memory requirements. The Anti-Symmetric DGN [22], which provides a stable and non-dissipative framework for designing deep graph neural networks, inspired by ordinary differential equations. And VCLANC [35], which performs self-supervised learning by reconstructing the network structure and node attributes to explore deeper information. Finally, RGDAL [36] adopts an information-theoretic principle to sieve out noisy factors for cross-network node classification. Despite the commendable contributions made by these primarily message-passing [15] methods, they lack the desired flexibility and adaptability to local graph structures across multiple channels.

In the large-scale graph context, several GNNs have been engineered specifically for node classification tasks. FastGCN [28] diminishes computational costs through the deployment of sampling techniques, and the ASGCN [29] addresses the over-smoothing issue [37] by incorporating a sparsity regularization term. This term prevents node representations from becoming increasingly similar with layer increments, thus preserving localized discriminative information. ClusterGCN [30] addresses scalability issues by capitalizing on graph clustering, while GraphSAINT [31] boosts generalization ability by amalgamating subgraph sampling with GNNs. Furthermore, PaSca [38] introduces a novel scalable graph neural architecture paradigm (SGAP) to comprehensively explore the architecture space. However, our experimental findings suggest that compared to general GNNs, these methods exhibit marginal performance improvements on smaller graphs.

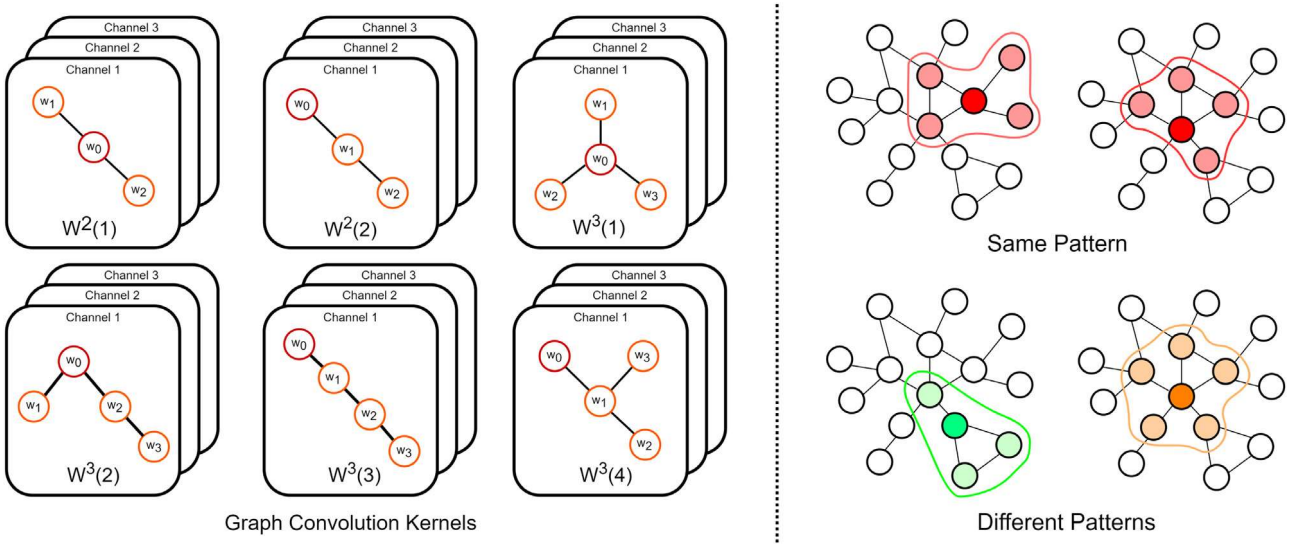


Fig. 2. Graph-Shaped kernels in different channels and patterns. The left half illustrates different shapes of W^2 and W^3 kernels, where the same kernel can be applied to different channels. The right half of the figure displays potential patterns.

2.2. Kernel methods on graphs

Graph kernels provide a foundational approach to evaluating similarities between graphs, enabling graph classification and regression [39]. Predominantly, graph kernels adhere to the R-convolution paradigm, which involves contrasting varying substructures of input graphs, such as random walks [40–42], shortest paths [43], subtrees [44], and graphlets [33].

The Weisfeiler–Lehman framework operates in concert with existing kernels. It deploys a relabeling process predicated on the Weisfeiler–Lehman isomorphism test [45]. Gilmer et al. [15] suggested a convolutional operation model based on continuous kernels, intending to learn feature representations of graph structures through internode information swapping. Nikolentzos et al. [46] excavated neighborhoods of various sizes from the graph via community detection and accordingly utilized graph kernels for normalization. SplineCNN [47] extends the traditional CNN convolution operators by adopting continuous kernel functions parameterized by a predetermined count of trainable weights.

Although graph kernel-based approaches have demonstrated considerable success on graph classification tasks, their direct application to node classification tasks is largely non-feasible. A typical strategy involves using graph kernels to map node features onto high-dimensional vectors, which are then processed through neural networks for node classification [48].

2.3. Convolutional neural networks

Groundbreaking research by LeCun et al. [24] introduced the concept of convolutional neural networks and convolution kernels. These techniques have proven instrumental in extracting structure patterns from images. However, due to the distinctive configuration of graph data, conventional CNNs are not directly applicable to graph data. Conversely, Tixier et al. [49] represented the graph as a conglomerate of bivariate histograms, which were processed as input for a traditional 2D CNN. Niepert et al. [50] proposed a more universal methodology for extracting locally connected regions from graphs, which are subsequently processed by a 1D CNN. These prior studies have attempted to employ convolutional operations on graph data, essentially through the reformulation of graph data. Nonetheless, the convolution kernels in these methods remain static, thereby imposing limitations on their capacity for node feature representation.

Table 1
Notations in the sgcn.

Symbols	Definition and description
G	A general undirected graph
V	Node set
X	Node feature
$G_{v_i}^{\mathcal{K}}$	$\mathcal{K}$-hop neighborhood subgraph of node v_i
$G^{\mathcal{K}}$	Subgraph sampling list for $\mathcal{K}$-hop neighborhood
H	Raw input to convolutional layer
H^k	Result of convolution with W^k
W_c^k	Kernel with size k and channel cth
Z	The output embedding of nodes
Y	The label of nodes

3. Preliminaries

3.1. Problem description

A graph is represented as $G = (V, X, A)$, where $V = \{v_1, v_2, \ldots, v_N\}$ denotes the node set. $X \in \mathcal{R}^{N\times F}$ denotes the node features, where N is the number of nodes and F is the number of feature dimensions. $A \in \mathcal{R}^{N\times N}$ denotes the adjacency matrix, and the node labels are denoted as Y with M categories. The symbols in this paper are summarized in Table 1.

3.2. Message passing

Typical graph neural networks employ message passing to aggregate node features by gathering information from neighboring nodes. This process allows central nodes to capture the features of their local subgraphs. The computation is described as follows:

$$h_v^{(k)} = MLP^{(k)}\{COMB(h_v^{(k-1)}, \sum_{u\in N(v)} h_u^{(k-1)})\} \quad (1)$$

where $h_v^{(k)}$ denotes the feature of v in the kth iteration and $N(v)$ denotes the neighbors of v. *COMB* and *MLP* represent the combination function and multilayer perceptron, respectively.

Message-passing-based graph neural networks (GNNs) aggregate messages by recognizing information from neighboring nodes at each step, thereby determining the convolution shape according to the structure of these neighboring nodes. Limitations arise with this model, however, as it inherently struggles to

capture the complex nature of local graph structures. Moreover, the weights of the multilayer perceptron *MLP* directly interact with the node feature dimension, limiting the model's potential to thoroughly learn shared structural patterns. Additionally, when handling large graphs, the corresponding adjacency matrix is typically large, necessitating substantial storage and computational resources for the application of most message-passing-based GNNs.

3.3. Graph convolutional networks

Joan et al. [51] first proposed graph convolutional neural networks based on spectral-domain and spatial-domain representations. One of the most classic GNNs, the GCN [16], is defined by the following formula:

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)}) \tag{2}$$

where $\hat{A}$ signifies the normalized adjacency matrix of the graph, H denotes the feature matrix of the nodes, and W represents the GCN's weight matrix. The formula $\hat{A}HW$ encapsulates the aggregation and transformation of neighboring node features, effectively integrating the graph structure into the learning process. The resulting matrix is subsequently passed through an activation function, introducing nonlinearity to capture intricate relationships.

3.4. Over-smoothing

The phenomenon of over-smoothing [37] can occur within the feature representations of the local graph structure and individual nodes, stemming from the proliferation of encodings [52]. This convergence of node characterization towards a limited set of values results in a diminished ability to discern nodes from disparate clusters. Specifically, nodes belonging to distinct classes but possessing proximate topological proximity tend to exhibit reduced discriminative capacity, leading to potential misclassifications.

A straightforward and effective approach to mitigate the issue of graph over-smoothing is to adopt residual connection structures like ResNet [53] architecture. This involves the fusion of the original node features with the iteratively updated embeddings, preserving the intrinsic characteristics of the nodes while allowing for the incorporation of refined information. Integrating the raw features of the network helps counteract the tendency of nodes to coalesce into a uniform feature space due to excessive smoothing.

4. Scalable graph convolution network

The architecture of the scalable graph convolution network (SGCN) is illustrated in Fig. 5. A graph G is input into a sampler to create subgraphs, and this process acquires the sampled embedding, which is denoted by H. This embedded H is input into the SGCN as the initial input. The output Z from the SGCN is then harnessed for node classification. Furthermore, a residual connection is used to mitigate the issue of over-smoothing [37].

4.1. Scalable convolution layer

An SGCN comprises several scalable convolutional layers (SCLs), where the node embedding, which is represented by H, serves as the input to the SCL. The procedure for calculating the SCL is depicted in Fig. 3. A kernel nested within the SCL is designed as a weight matrix encompassing neighboring nodes, and this convolution kernel is denoted by W^k, where k signifies the number of covered nodes in the kernel list K. Each kernel corresponds proportionately to a channel. By convolving the input data with these particular kernels, the SGCN manages to encapsulate the local structural particulars of the graph.

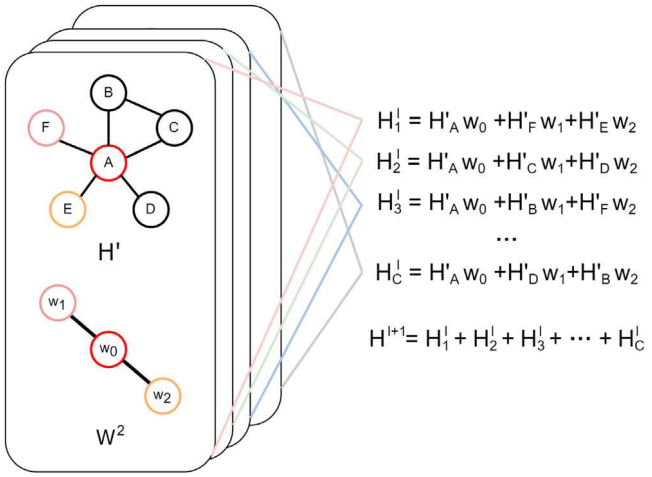


Fig. 3. An example of convolutional calculations in W^2. The graph-shaped kernel is multiplied with the central node and the two selected neighboring nodes it covers in each channel.

The SCL equation is computed as follows:

$$H^{l+1} = \sum_{c=1}^{C} W_c^k(H^l) + B \tag{3}$$

where $H^{l+1} \in \mathcal{R}^{N\times C\times L}$ is the output of the lth layer. C denotes the number of channels, $W^k \in \mathcal{R}^{C\times C}$ and $B \in \mathcal{R}^{N\times C\times L}$ denote the weight matrix and bias, respectively, and L denotes the embedding length of H.

The output embeddings generated by the last SCL using W^k are denoted by $\mathcal{H}^k$. We concatenate each embedding as $\mathcal{H}$, and average pooling is applied to aggregate features by the last dimension. The output of pooling Z_p is calculated as follows:

$$\begin{aligned} \mathcal{H} &= Concat(\mathcal{H}^1, \mathcal{H}^2, \ldots, \mathcal{H}^k), \\ Z_p &= \frac{1}{L}\sum_{i=1}^{L} \mathcal{H}_i \end{aligned} \tag{4}$$

where $\mathcal{H} \in \mathcal{R}^{N\times C\times k\times L}$ and $Z_p \in \mathcal{R}^{N\times(C\times k)}$.

4.2. Sampler

To sample nodes that contain rich structural information while reducing computational costs, a sampler based on degree centrality [32] is applied.

An example is shown in Fig. 4. When the sampling rate is 80%, the computational costs of using the W^4, W^3, and W^2 kernels are reduced by 80%, 60%, and 40%, respectively.

The SGCN constructs a $\mathcal{K}$-hop subgraph based on the central node and samples a subgraph sequence according to the neighbor's degree centrality. The sampler is shown in Algorithm 1, where **K_hop_subgraph** is used to obtain a $\mathcal{K}$-hop subgraph around node v_i by A and $\mathcal{K}$, and **Choose** returns the subgraph composed of s_i nodes with the highest *deg_c* in the *subset*.

For each node v_i, $G_{v_i}^{\mathcal{K}}$ denotes the subgraph with a $\mathcal{K}$-hop neighborhood ($\mathcal{K}$ can be 1, 2, $\cdots$, etc.), where $G^{\mathcal{K}}$ is a subgraph list composed of $G_{v_i}^{\mathcal{K}}$, as shown in Eq. (5). Each subgraph element $G_{v_i}^{\mathcal{K}}$ contains a node set V_{v_i} and adjacency matrix A_{v_i}.

$$G^{\mathcal{K}} = [G_{v_1}^{\mathcal{K}}, \ldots, G_{v_N}^{\mathcal{K}}] = [(V_{v_1}, A_{v_1}), \ldots, (V_{v_N}, A_{v_N})] \tag{5}$$

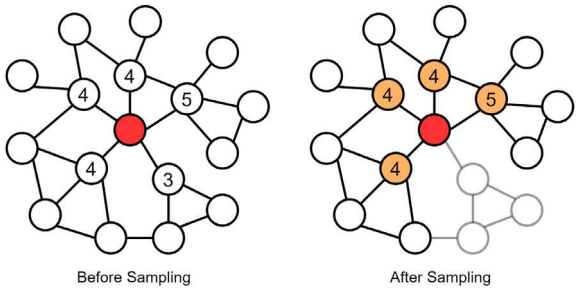


Fig. 4. Comparison of the computational cost under the implemented sampling strategy. Note that, for the kernels W^4, W^3, and W^2, each convolution necessitates the selection of 4, 3, and 2 neighbors, respectively, from the central node's 5 neighbors prior to sampling. Upon 80% sampling of the neighboring nodes, each convolution then requires the selection of 4, 3, and 2 neighbors from the 4 neighbors of the central node.

Algorithm 1 Sampler in SGCN.

Input: General graph $G = (V, A)$, sample ratio r, sample hop $\mathcal{K}$.
Output: Sampled subgraphs sequence $G^{\mathcal{K}}$.
Initialize subgraph list $G^{\mathcal{K}}$;
for v_i **in** V **do**
  $subset$ = **K_hop_subgraph**$(v_i, A, \mathcal{K})$;
  $s_i = r \times$ **len**$(subset)$;
  Initialize list deg_c;
  for v_i **in** $subset$ **do**
    $w_{v_i} = \sum_{j=1}^{v_i} A_{ij}(i \neq j)$;
    $deg\_c.append(w_{v_i})$;
  end for
  $G_{v_i}^{\mathcal{K}}$ = **Choose**$(subset, degrees, s_i)$;
  $G^{\mathcal{K}}.append(G_{v_i}^{\mathcal{K}})$;
end for
return Sampled subgraphs sequence $G^{\mathcal{K}}$;

The features of subgraph $G_{v_i}^{\mathcal{K}}$ are $X_{v_i} \in \mathcal{R}^{s_i \times \mathcal{K} \times F}$, which are composed of the corresponding node features. s_i denotes the number of nodes in $G_{v_i}^{\mathcal{K}}$, and a sum function acting on the first dimension is used to align features. Each node v_i's feature h_i is updated as follows:

$$h_i = \sum_{j=1}^{s_i} x_{j,k,F} \quad (x_{j,k,F} \in X_{v_i}) \tag{6}$$

where $h_i \in \mathcal{R}^{\mathcal{K} \times F}$. Each h_i is concatenated to compose the sampled embeddings that combine original features and structure information. The sampled embeddings H are as follows:

$$H = Concat(h_1, \ldots, h_N) \tag{7}$$

where $H \in \mathcal{R}^{N \times F \times L}$.

4.3. Residual connection

Inspired by ResNet [53], to alleviate the over-smoothing problem [37] in GNNs, we downsample H as follows:

$$Z_d = W_d H + B_d \tag{8}$$

where $Z_d \in \mathcal{R}^{N \times (C \times k)}$ is the output of downsampling and W_d and B_d are the weight and bias in the residual connection, respectively. Z_d is added to the output embedding of the last SCL, which is formulated as follows:

$$Z = Z_p + Z_d \tag{9}$$

4.4. Classification

Z is used to predict the label after a fully connected layer, and the model's prediction $\hat{Y} \in \mathcal{R}^{N \times M}$ is obtained after applying *softmax* as follows:

$$\hat{Y} = softmax(W_c Z + B_c) \tag{10}$$

M is the number of classes.

The loss function for the node classification is expressed as follows:

$$\mathcal{L}_r = -\sum_{v \in V} \sum_{i=1}^{M} Y \ln \hat{Y} + \lambda \|\theta\|^2 \tag{11}$$

where λ denotes a regularization parameter and θ denotes the parameters of the model.

Algorithm 2 Framework of the SGCN.

Input: General graph $G = (V, X, A)$, sample ratio r, kernel list K, channel size C.
Output: The final predicted label Y.
Sample subgraph list G^k for each node (Algorithm 1);
Construct node embeddings H from G^k (Eqs. (6) (7));
while not converged **do**
  for k **in** K **do**
    Compute H^k using convolutional kernels W^k (Eq. (3));
  end for
  Obtain Z_p by concatenating the convolution result $\mathcal{H}$ and pooling (Eq. (4));
  Obtain the final embedding Z using the residual network (Eqs. (8) (9));
  Produce the final predicted label Y using classification (Eq. (10));
  Update θ using cross-entropy loss Eq. (11);
end while
return The predicted label Y;

5. Experimental setup

5.1. Data description

To evaluate the performance of the SGCN on the node classification task, we considered the following ten real-world datasets:

Cora [54]: A citation network of scientific publications in machine learning that consists of 2,708 papers represented by a bag-of-words feature, where edges represent citation links.

CiteSeer [54]: A citation network with 3,327 papers. Each paper is represented by a feature vector based on word occurrences, and edges represent citation links.

Photo [55]: A social network of photos with associated tags. This dataset contains 7,650 photos and their corresponding tag information. Nodes represent goods, and edges represent that two goods are frequently bought together.

CS [55]: A citation network of computer science publications that consists of 18,333 scientific papers. Nodes represent authors as connected by an edge if they coauthored a paper.

PubMed [54]: A citation network of biomedical literature containing 19,717 papers. Each paper is represented by a word vector, and edges represent citation links.

CoraFull [54]: A larger version of the Cora dataset containing 19,793 papers. Nodes represent documents, and edges represent citation links.

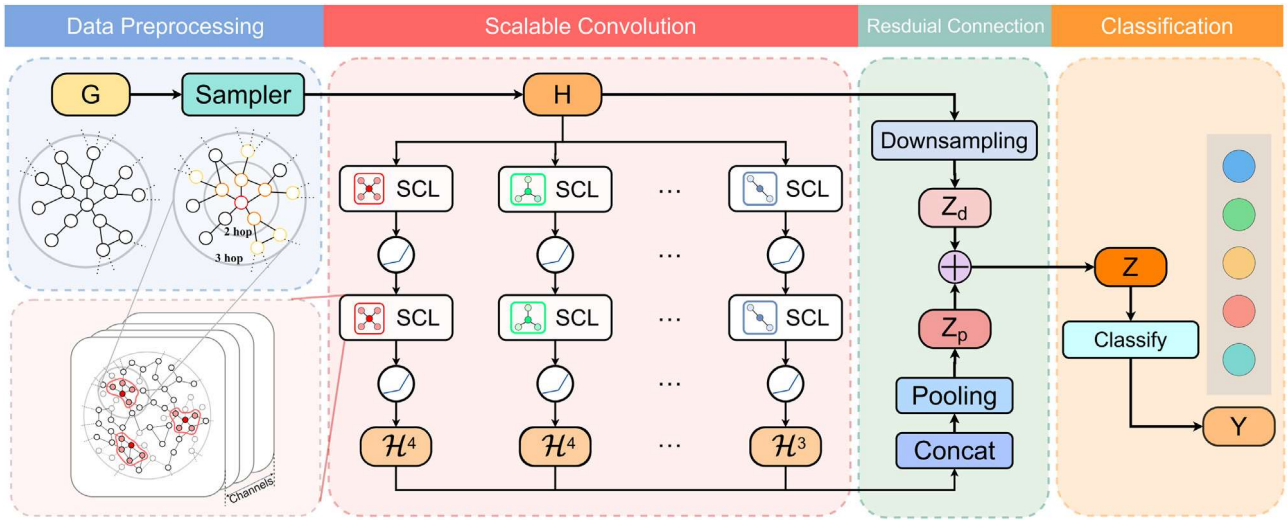


Fig. 5. Framework of SGCN. The data preprocessing involves sampling important subgraphs for each node to get original node embeddings H. In scalable convolution, convolution is performed on H using different graph-shaped kernels. The residual connectivity incorporates Z_d after down-sampling H and scalable convolution result after concat and pooling operation Z_p through addition. The result of residual connectivity Z is employed in classification to predict the label Y.

Physics [55]: A co-authorship network of 34,493 high-energy physics theory papers. Nodes represent authors, and an edge connects authors if they coauthored a paper.

Flickr [31]: A social network of photos with 89,250 photos and their associated tag information. Nodes represent users or accounts, and edges represent relationships between these users.

Reddit [18]: A social network of posts and comments on the website Reddit, comprising 232,965 posts and their associated comments. Nodes represent individual posts or submissions made by users on the Reddit platform, and edges represent replies, comments, or other forms of interactions between the posts.

Amazon [31]: A social network dataset that contains product reviews and metadata from Amazon for classifying product categories based on buyer reviews and interactions. The nodes and edges represent entities and relationships between entities, respectively.

These benchmark datasets have different scales and sparsity, and they were used to evaluate the performance and generalizability of the SGCN.

5.2. Baselines

For node classification, the following strong baselines achieved notable performance:

GCN [16]: The GCN is a widely used graph convolution network that updates the features of a node by averaging its neighboring nodes' features.

GAT [17]: The GAT aggregates neighbor features using multi-head attention, achieving SOTA performance on various datasets.

GraphSAGE [18]: GraphSAGE is the first inductive graph neural network that aggregates node features by sampling neighboring nodes. It predicts graph context and labels using the aggregated information.

MPNN [15]: The MPNN utilizes convolutional operations based on continuous kernels to learn feature representations of graph structures by facilitating information exchange between nodes.

SplineCNN [47]: SplineCNN is a type of graph convolutional neural network that employs continuous B-spline kernels on irregularly structured data such as graphs.

FastGCN [28]: FastGCN interprets graph convolutions as integral transforms of embedding functions under probability measures. It provides a sampling method that depends on the importance of nodes.

ASGCN [29]: The ASGCN constructs the network layer by layer in a top-down manner and samples the lower layer conditioned on the top layer. This approach allows shared sampled neighborhoods and avoids overexpansion due to fixed-size sampling.

ClusterGCN [30]: ClusterGCN samples a block of nodes associated with a dense subgraph, which is identified by exploiting the graph clustering structure. It restricts the neighborhood search within this subgraph.

GraphSAINT [31]: GraphSAINT introduces a graph neural network model based on a sampled graph, where calculations on each minibatch are performed on the sampled graph without the neighbor explosion phenomenon.

PaSca [38]: PaSca presents a scalable graph neural architecture paradigm that can search for well-performing and scalable GNN architectures using multi-objective optimization to balance trade-offs between different criteria.

FusedGAT [21]: An optimized version of the GAT based on the DGNN [56] that fuses message passing computation for accelerated execution and a lower memory footprint.

Anti-Symmetric DGN [22]: A framework for stable and non-dissipative deep graph network design inspired by ordinary differential equations.

5.3. Parameter settings

In the case of the scalable graph convolution network (SGCN), the datasets were randomly divided in the following manner: 60% were allocated for training, 20% were set aside for validation, and the remaining 20% were reserved for testing. The sample ratio was established at either 0.2 or 0.5. Utilizing the Adam optimizer, the learning rate was established at 0.003, while the weight decay was set to $5e^{-4}$. Alterations within a specified limit may enhance performance under particular circumstances. For the baseline methodologies, a similar data division of 6:2:2 was employed. Corresponding to the SGCN, the learning rate was established at 0.003, the hidden layer size was set to 128, and the dropout rate was 0.4. All other parameters were initialized based on the specifications suggested in their associated research papers. All of the baseline models were implemented by PyG [57]. Due to the current lack of support for complex graph computations in mainstream frameworks such as PyTorch and TensorFlow, we have

Table 2
Node classification performance.

Datasets	Cora	CiteSeer	Photo	CS	PubMed	CoraFull	Physics	Flickr	Reddit	Amazon
$\|V\|$	2,708	3,327	7,650	18,333	19,717	19,793	34,493	89,250	232,965	1,569,960
$\|E\|$	5,429	4,732	238,162	163,788	44,338	126,842	495,924	899,756	114,615,892	264,339,468
# classes	7	6	8	15	3	70	5	7	41	107
GCN	88.38 ± 0.44	75.35 ± 0.57	89.98 ± 2.08	90.08 ± 0.11	85.35 ± 0.20	50.11 ± 2.06	95.13 ± 2.11	41.66 ± 0.01	–	–
GAT	86.04 ± 0.61	71.90 ± 0.36	91.78 ± 2.98	91.72 ± 0.44	86.78 ± 0.08	64.72 ± 0.56	95.83 ± 1.78	46.10 ± 1.58	–	–
GraphSAGE	88.23 ± 0.32	75.54 ± 0.46	90.34 ± 2.12	93.77 ± 0.79	87.59 ± 0.27	53.42 ± 1.32	96.27 ± 1.09	44.05 ± 3.69	–	–
FusedGAT	85.05 ± 0.78	73.69 ± 0.86	92.71 ± 0.32	91.35 ± 0.05	85.57 ± 0.40	65.65 ± 0.48	95.84 ± 0.10	47.69 ± 0.63	–	–
Anti-Symmetric	86.39 ± 0.24	75.72 ± 0.30	93.90 ± 0.15	93.9 ± 0.12	87.32 ± 0.06	67.06 ± 0.51	95.82 ± 0.37	47.32 ± 0.40	–	–
MPNN	87.29 ± 0.41	73.03 ± 0.81	65.37 ± 0.29	85.47 ± 0.33	84.78 ± 0.23	–	–	–	–	–
SplineCNN	88.32 ± 0.37	74.98 ± 0.39	93.97 ± 0.42	94.31 ± 0.23	86.74 ± 0.79	70.44 ± 0.27	96.01 ± 0.22	47.79 ± 2.24	–	–
FastGCN	87.25 ± 0.27	72.88 ± 0.36	91.35 ± 1.81	94.18 ± 0.08	87.51 ± 0.15	69.52 ± 1.05	95.55 ± 2.17	46.60 ± 1.32	92,40 ± 0.16	–
ASGCN	87.93 ± 0.41	72.64 ± 0.45	92.81 ± 1.91	91.71 ± 1.82	88.59 ± 0.22	69.66 ± 0.87	96.31 ± 0.05	47.75 ± 1.62	93.35 ± 0.46	–
ClusterGCN	86.35 ± 0.98	72.97 ± 1.30	93.14 ± 0.34	93.74 ± 0.22	88.79 ± 0.28	69.09 ± 0.37	96.37 ± 0.15	47.32 ± 0.38	91.72 ± 3.19	75.90 ± 0.39
GraphSAINT	86.81 ± 0.30	75.29 ± 0.31	90.39 ± 0.45	93.93 ± 0.27	88.59 ± 0.25	69.76 ± 0.22	96.39 ± 0.07	47.11 ± 0.74	93.31 ± 0.43	78.70 ± 0.17
PaSca	87.13 ± 3.18	75.09 ± 0.69	90.28 ± 0.17	92.94 ± 0.38	86.32 ± 0.13	–	96.30 ± 0.70	44.38 ± 1.40	93.23 ± 0.10	–
SGCN	**88.66 ± 0.33**	**81.10 ± 0.14**	**94.98 ± 0.35**	**94.85 ± 0.30**	**89.10 ± 0.34**	**73.43 ± 0.27**	**96.70 ± 0.20**	**51.19 ± 0.40**	**94.63 ± 0.14**	**79.91 ± 0.23**
Least Improvement	0.28	5.38	1.01	0.54	0.31	2.99	0.31	3.40	1.28	1.54
Avg. Improvement	1.56	7.01	5.42	2.59	2.11	8.48	0.71	5.03	1.82	3.38

Table 3
Time cost (in minutes) of models on the large graph.

Datasets	Physics	Flickr	Reddit
FastGCN	39.15	90.16	270.35
ASGCN	45.25	108.39	310.47
ClusterGCN	43.89	100.33	298.81
GraphSAINT	46.39	115.91	330.12
PaSca	44.37	101.72	346.31
SGCN	42.65	105.79	315.19

implemented an equivalent representation of graph convolution using one-dimensional convolutions.

6. Experimental analysis

In this section, we analyze the experimental results of the SGCN, including the node classification, kernel shape, ablation study, parameter sensitivity, and visualization.

6.1. Node classification

To affirm the effectiveness of the SGCN and the baselines, they were subjected to a node classification task. Each method was executed across various datasets, and the accuracy was calculated along with the standard deviation. The respective results are presented in Table 2. The GCN, GAT, GraphSage, FusedGAT, and Anti-Symmetric methodologies were applied to general graphs, the MPNN and SplineCNN constituted kernel methods, while FastGCN, the ASGCN, ClusterGCN, GraphSAINT, and PaSca were employed on large graphs. Various methods are differentiated via lines for clarity.

As can be observed from Table 2, the SGCN achieves peerless accuracy across all benchmark datasets. Furthermore, the SGCN outperforms the baseline by over 1% on the CiteSeer, Photo, CoraFull, Flickr, Reddit, and Amazon datasets. In comparison to each individual baseline, the SGCN records an average improvement exceeding 1.5%, except for on the Physics dataset. This result could potentially be ascribed to the relatively straightforward nature of classification in the realm of physics, as all models exhibit commendable performance on this dataset. Significantly, the SGCN vastly outperforms other methods on CiteSeer, reaching an accuracy of 81.10%. Conversely, the highest performance exhibited by the Anti-Symmetric model only achieved 75.72%, which is 5.38% below that of the SGCN. On the Physics dataset, all models displayed competitive performance with scores of approximately 96%. The SGCN also exemplifies robust performance on large-scale graphs, superseding the optimum baseline on the Flickr dataset by 3.4%.

While ClusterGCN and SplineCNN exhibit notable performance on several datasets, SplineCNN records considerable performance gaps in comparison to the SGCN on CiteSeer, PubMed, and Flickr. Notably, the SGCN exceeds SplineCNN by 4.8% on CiteSeer. Due to their high computational power consumption and memory usage, these kernel methods are impractical for large-scale graphs. The experimental statistics suggest that the SGCN boasts a potent expressiveness and consistently retains a low standard deviation across different scales of graphs.

We performed a comparative analysis of time consumption among large-scale graph models and the SGCN on the Physics, Flickr, and Reddit datasets, and the results are summarized in Table 3. Overall, FastGCN has the lowest time consumption among the models. On the Physics dataset, the time consumption of each model is relatively close, with the SGCN exhibiting a time consumption of 42.65 min, which is second only to FastGCN. On the Flickr dataset, ClusterGCN and PaSca perform similarly to the SGCN but with lower time consumption compared to the ASGCN and GraphSAINT. On the Reddit dataset, FastGCN and ClusterGCN have the best time performance, while the ASGCN and SGCN have similar time consumption. These experiments demonstrate that SGCN achieves state-of-the-art performance without increasing time consumption.

6.2. Kernel shape

In this section, we examine the influence of different kernel shapes on the node classification task. We employed three types of convolutional kernels: W^2, W^3, and a combination of W^2 and W^3, which is denoted as $W^2 + W^3$. As depicted in Fig. 6, we observe that the combination kernel, W^2+W^3, consistently exhibits the best performance across all datasets except for Cora, suggesting that the cooperation between kernels enables the SGCN to learn the neighborhood more comprehensively. For the Cora citation network, W^3 performs the best. This result implies that larger kernels are advantageous when handling citation datasets within similar fields. On multiple datasets, including Cora, Citeseer, CS, PubMed, Physics, and Flickr, the performance of W^3 is superior to that of W^2, which can be attributed to the fact that W^3 learns more general patterns in the data. On the Photo dataset, the performance of W^2 is higher than that of W^3. We speculate that this result could be due to the dense nature of the edges in the Photo dataset. W^2 tends to perform well on dense edges, and a similar phenomenon is observed on other dense graphs such as CS, CoraFull, and Reddit, where the performance of W^2 is comparable to that of W^3.

Table 4
Ablation study on node classification.

Datasets	Cora	CiteSeer	Photo	CS	PubMed	CoraFull	Physics	Flickr	Reddit	Amazon
SGCN	**88.66 ± 0.33**	**81.10 ± 0.14**	**94.98 ± 0.35**	**94.85 ± 0.30**	**89.10 ± 0.34**	**73.43 ± 0.27**	**96.70 ± 0.20**	**51.19 ± 0.40**	**94.63 ± 0.14**	**79.91 ± 0.23**
SGCN - {Res}	87.22 ± 0.43	77.41 ± 0.15	93.05 ± 0.74	94.22 ± 0.45	88.23 ± 0.65	70.05 ± 0.20	95.47 ± 0.44	48.39 ± 0.68	93.42 ± 0.43	78.79 ± 0.61
SGCN - {Deg}	86.79 ± 0.34	75.67 ± 0.39	92.22 ± 0.37	92.10 ± 0.28	87.31 ± 0.45	68.53 ± 0.22	96.04 ± 0.27	48.63 ± 0.29	93.23 ± 0.31	79.01 ± 0.35

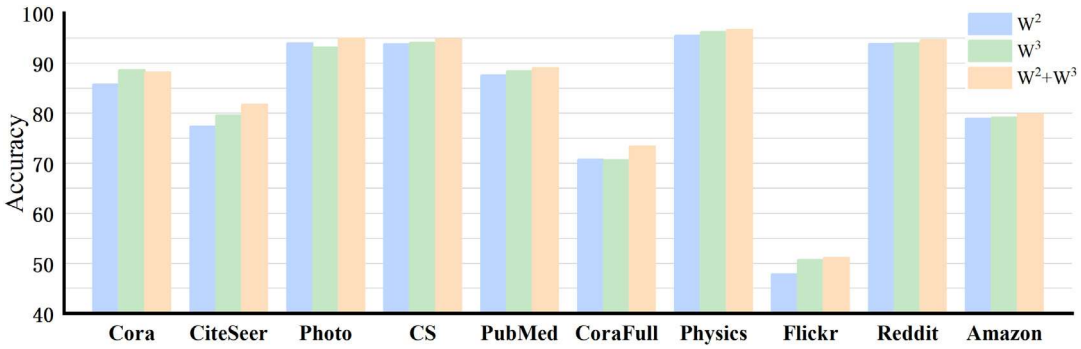


Fig. 6. Impact of kernel shape on performance.

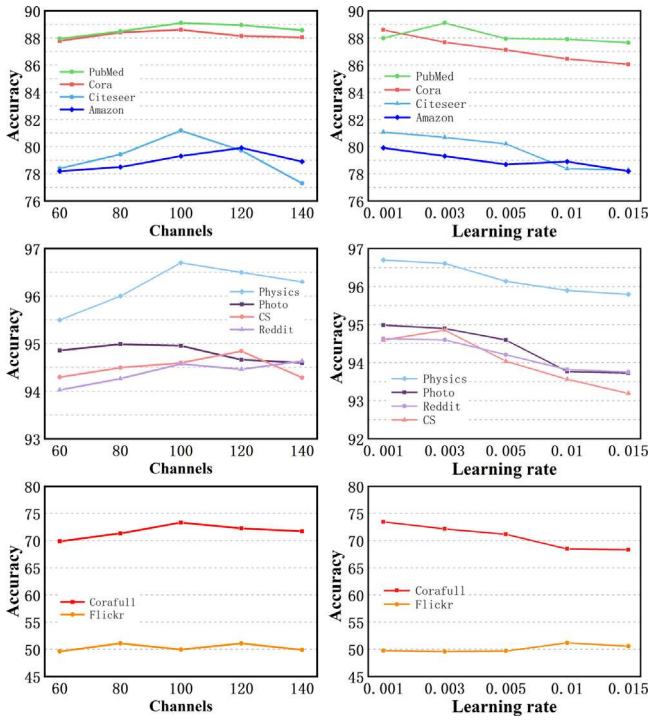


Fig. 7. Impact of the number of channels and learning rate on the performance.

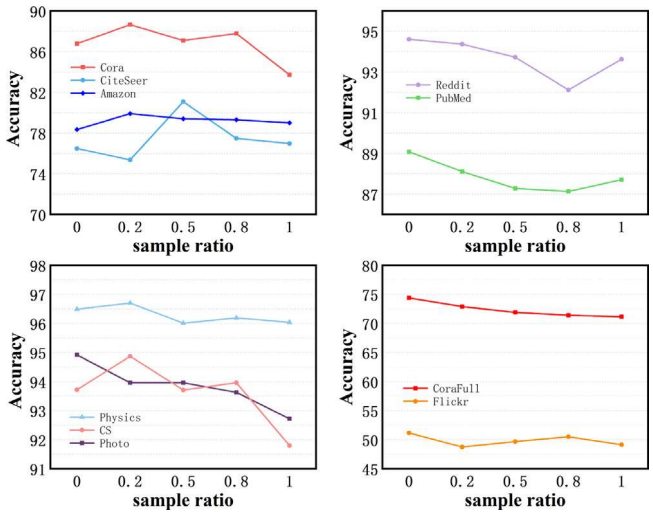


Fig. 8. Impact of the sample ratio on performance.

6.3. Ablation study

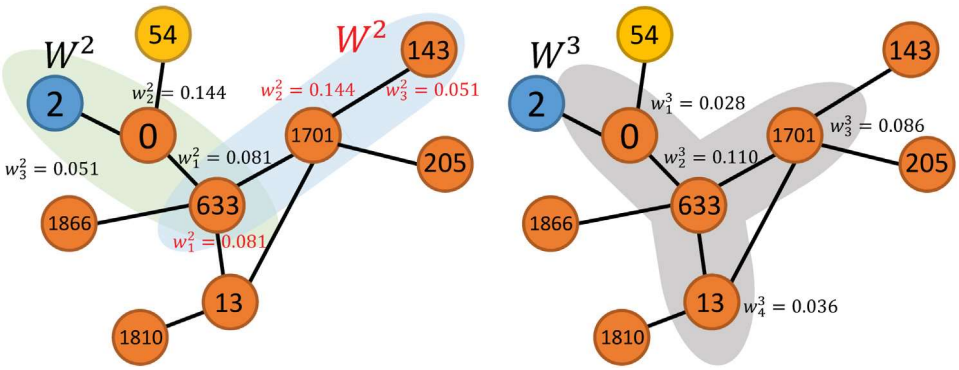
We investigated the contributions of each component constituting the scalable graph convolution network (SGCN) using ablation studies, and the accuracy of the variants is presented in Table 4. The term SGCN-Res signifies the removal of the residual connection, whereas SGCN-Deg symbolizes the elimination of degree centrality from the sampler. As per the findings from Table 4, the exclusion of either the residual connection or the node sampling mechanism results in a notable decline in accuracy.

The residual connection fulfills a crucial function by amalgamating initial representations and corresponding representations following convolutional layers. This melding bolsters the global optimization capabilities of the SGCN. The node sampling mechanism in the SGCN utilizes degree centrality to gauge the importance of nodes, which effectively captures nodes that contribute predominantly to the final node representations.

The performances resulting from reducing SGCN-Res and SGCN-Deg are closely parallel on the larger-scale datasets, including PubMed, Physics, Flickr, Reddit, and Amazon. On the smaller-scale datasets, SGCN-Deg demonstrates a lesser performance than SGCN-Res, suggesting that degree centrality has an enhanced impact on smaller-scale datasets. This phenomenon may occur because, within smaller datasets, nodes deemed important may exert a more significant influence over the representations of other nodes, leading to a more discernible improvement.

6.4. Parameter sensitivity

In this subsection, we explore the sensitivities of various parameters. The first column of Fig. 7 delineates the influence of channels on the SGCN's performance. On datasets such as Cora, CiteSeer, Photo, CS, PubMed, and CoraFull, the SGCN's accuracy curves exhibit a pronounced peak in the middle with declines on each side. The peak performance of the SGCN is achieved when



(a) Kernel W^2 (b) Kernel W^3

Fig. 9. Case Study of SGCN patterns on the Cora dataset. (a) illustrates two patterns learned by W^2, while (b) shows the patterns learned by W^3. w represents the weight of each node in the patterns, and the different node colors represent their corresponding categories.

the channel configuration is set to 100 for the Cora, CiteSeer, PubMed, and CoraFull datasets. However, on the CS and Amazon datasets, the apex is reached with 120 channels, and the photo reaches its peak with 80 channels. The SGCN's performance on the Flickr dataset is relatively stable, while on the Reddit dataset, there is an improvement as the number of channels increases.

The second column of Fig. 7 demonstrates the effects of varying learning rates on the performance of the SGCN. Across most datasets, excluding the PubMed, CS, and Flickr datasets, there is a noted decline in the SGCN's performance as the learning rate increases. This flexion suggests that the SGCN performs more efficiently with a lower learning rate. The PubMed and CS datasets achieve peak performance when the learning rate is adjusted to 0.003, while on the Flickr and Amazon datasets, the best performance is recorded with a learning rate of 0.01.

Fig. 8 describes the impact of the sample ratio within the 3-hop neighborhoods on the SGCN's performance. The sample ratio profoundly influences the SGCN's effectiveness, although trends differ across various datasets. The maximum accuracy is reached when the sample ratio is set to 0.2 for the Cora, Physics, CS, and Amazon datasets, while an exceptional accuracy improvement is observed for CiteSeer at a sampling ratio of 0.5, substantially surpassing the performance of the baseline.

6.5. Case study

A case study that employs the Cora dataset is described here, and it scrutinizes patterns utilizing different kernels within the SGCN. The intention of this study is to discern the behaviors of two kernel types, namely, W^2 and W^3, within the 2-hop neighborhood of node 633 (disregarding irrelevant nodes). Fig. 9 displays the impacts of these kernels with corresponding weights marked next to the involved nodes, nodes distinguished by different colors signify different classes.

In the study, the SGCN was applied to the neighborhood of node 633, and it utilized two W^2 kernels and one W^3 kernel. Both W^2 and W^3 kernels exhibit higher weights on the central node compared to the neighboring nodes. In Fig. 9(a), node 633 has a weight of $w_2^2 = 0.144$, which is greater than the weights of $w_1^2 = 0.051$ and $w_3^2 = 0.081$, which were assigned to its neighboring nodes. Similarly, in Fig. 9(b), node 633 has a weight of $w_2^3 = 0.110$, which is greater than the weights of $w_1^3 = 0.028$, $w_3^3 = 0.086$, and $w_4^3 = 0.036$ assigned to its neighboring nodes. These results indicate that both the W^2 and W^3 kernels learn to prioritize the central node over the neighboring nodes, especially when they belong to the same label.

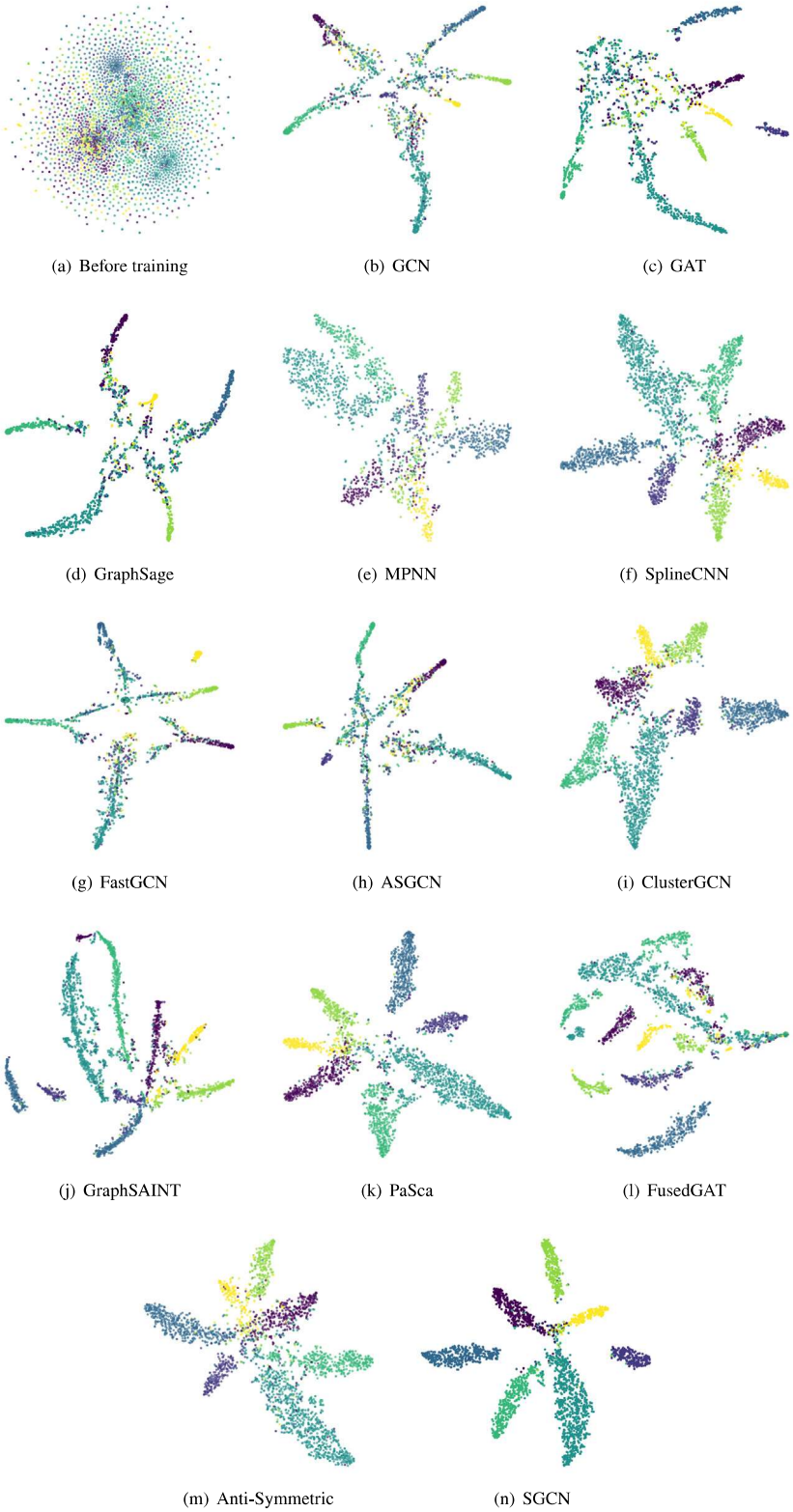
6.6. Visualization

We leveraged t-SNE [58] to visualize node representations, thus exploring the performance differentials between the SGCN and the baseline variants. Fig. 10 depicts these visualization results, mapping the node representations yielded by the SGCN into a two-dimensional subspace. Initially, prior to training, there is notable chaos in the node distributions. However, post-training, discernible clusters form in most GNNs, leading to multi-cluster formations. The GCN, GraphSage, FastGCN, and ASGCN yield six clusters, and the GAT and MPNN produce seven clusters, albeit with some nodes of different labels intermingled. GraphSAINT results in eight clusters, while the FusedGAT visualization appears disorderly. Nodes are clustered per their labels in the SGCN, SplineCNN, ClusterGCN, PaSca, and Anti-Symmetric DGN. Notably, clusters formed by nodes in the SGCN appear denser than those in the baselines.

Furthermore, we also visualized the node distribution on the CiteSeer dataset in Fig. 11. The distribution delineations of the GCN, GAT, GraphSage, MPNN, FastGCN, ASGCN, and FusedGAT are less distinct. GraphSAINT forms six cone-like clusters, although there remain unclassified nodes at the center. PaSca categorizes the nodes into seven clusters, whereas SplineCNN, ClusterGCN, the Anti-Symmetric DGN, and the SGCN precisely classify the nodes into six clusters with a relatively straightforward distribution. However, the distribution of nodes in the SGCN demonstrates a higher level of concentration compared to ClusterGCN.

7. Conclusion

Graph neural networks (GNNs) are based on message passing, which acts on the feature dimension of nodes during aggregation. The integration modes employed by these GNNs are simple, which limits their flexibility and expression capability when learning the local structural features of graphs. To address this limitation, we propose a scalable graph convolutional network (SGCN). The SGCN introduces graph-shape convolution kernels with different sizes and designs, and the kernels operate convolutions on the subgraphs and weights of the kernels. By sharing parameters within the graph, each kernel performs convolutions from different perspectives. Extensive experiments validate the generalization and stability of the SGCN for graphs of various scales, and it achieves state-of-the-art performance on node classification tasks. The SGCN offers a new perspective for the design of graph kernel convolution. As a powerful GNN model, the SGCN has been extended to a broader range of representation learning applications. For instance, the SGCN can serve as a substitute for the GCN in RGDAL [36] to achieve improved node classification performance and improved clustering performance for models based on node representations such as VCLANC [35].



(a) Before training (b) GCN (c) GAT

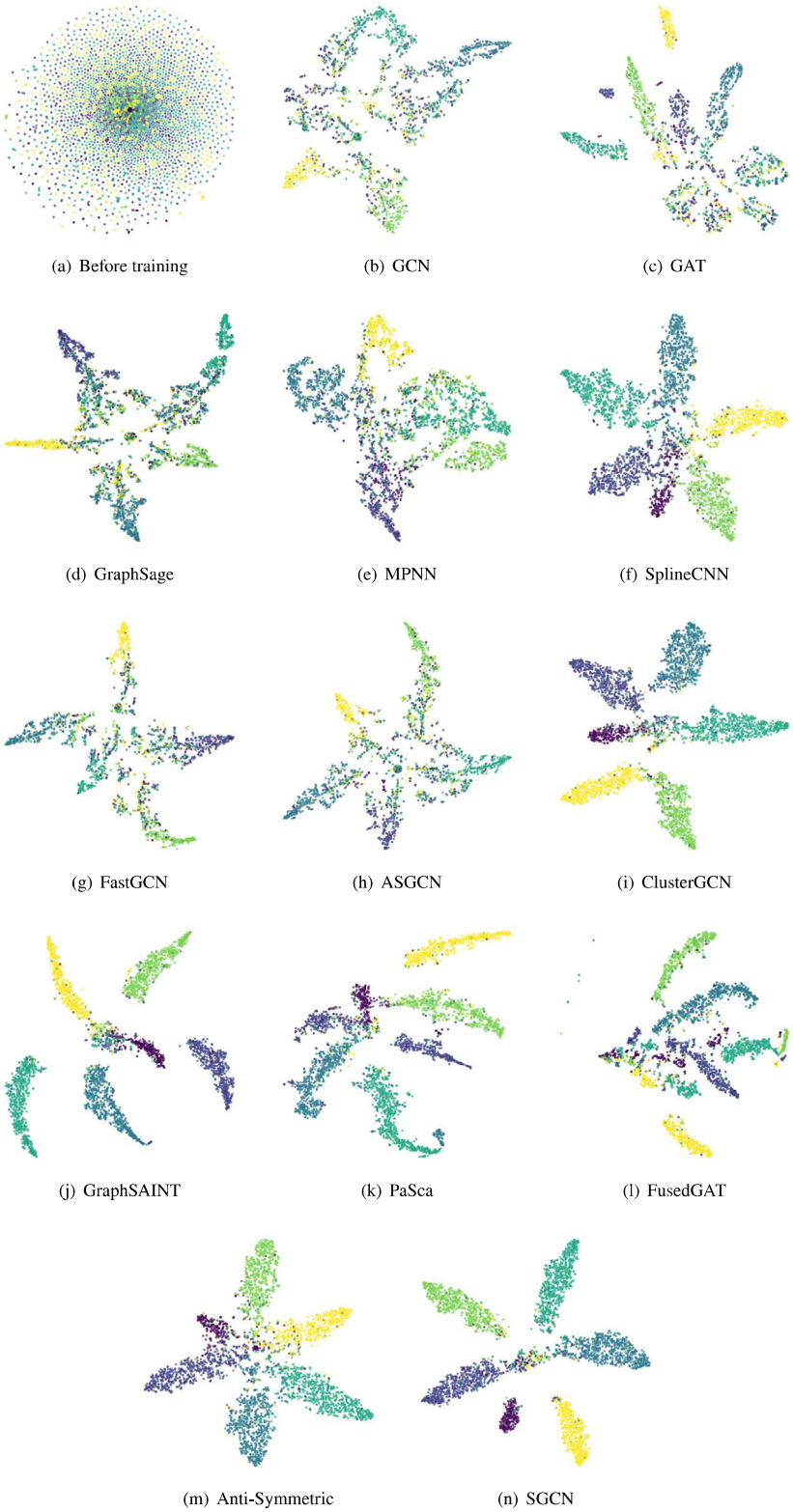
(d) GraphSage (e) MPNN (f) SplineCNN

(g) FastGCN (h) ASGCN (i) ClusterGCN

(j) GraphSAINT (k) PaSca (l) FusedGAT

(m) Anti-Symmetric (n) SGCN

Fig. 10. Visualization of node classification task on Cora.



(a) Before training (b) GCN (c) GAT

(d) GraphSage (e) MPNN (f) SplineCNN

(g) FastGCN (h) ASGCN (i) ClusterGCN

(j) GraphSAINT (k) PaSca (l) FusedGAT

(m) Anti-Symmetric (n) SGCN

Fig. 11. Visualization of node classification task on CiteSeer.

CRediT authorship contribution statement

Zhenhua Huang: Supervision, Methodology, Formal analysis, Conceptualization. **Wenhao Zhou:** Writing – original draft, Data curation. **Kunhao Li:** Writing – review & editing, Software, Methodology. **Zhaohong Jia:** Supervision.

Declaration of competing interest

We, the authors, hereby declare that we have no known competing financial interests or personal relationships that may have influenced the outcome or interpretation of our work.

Data availability

Data will be made available on request

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Number 71971002). We also thank the team of PyG [57] for their support during our experiments.

References

[1] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 3693–3702.
[2] H. Pei, B. Wei, K.C.-C. Chang, Y. Lei, B. Yang, Geom-GCN: Geometric graph convolutional networks, in: Proceedings of the 8th International Conference on Learning Representations, ICLR, 2020.
[3] S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui, Graph neural networks in recommender systems: A survey, ACM Comput. Surv. 55 (5) (2022) 1–37.
[4] J. Huang, R. Xie, Q. Cao, H. Shen, S. Zhang, F. Xia, X. Cheng, Negative can be positive: Signed graph neural networks for recommendation, Inf. Process. Manage. 60 (4) (2023) 103403.
[5] 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, in: Proceedings of the 28th Advances in Neural Information Processing Systems, NeurIPS, 2015, pp. 2224–2232.
[6] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, P. Riley, Molecular graph convolutions: Moving beyond fingerprints, J. Comput. Aided Mol. Des. 30 (2016) 595–608.
[7] Z. Cui, K. Henrickson, R. Ke, Y. Wang, Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, IEEE Trans. Intell. Transp. Syst. 21 (11) (2019) 4883–4894.
[8] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, D.-Y. Yeung, GaAN: Gated attention networks for learning on large and spatiotemporal graphs, 2018, CoRR.
[9] J. Liu, Y. Chen, X. Huang, J. Li, G. Min, GNN-based long and short term preference modeling for next-location prediction, Inform. Sci. 629 (2023) 1–14.
[10] X. Song, J. Li, T. Cai, S. Yang, T. Yang, C. Liu, A survey on deep learning based knowledge tracing, Knowl.-Based Syst. 258 (2022) 110036.
[11] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. (tog) 38 (5) (2019) 1–12.
[12] R. Hu, A. Rohrbach, T. Darrell, K. Saenko, Language-conditioned graph networks for relational reasoning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2019, pp. 10294–10303.
[13] T. Nguyen, R. Grishman, Graph convolutional networks with argument-aware pooling for event detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2018, pp. 5900–5907.
[14] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the 33th AAAI Conference on Artificial Intelligence, AAAI, 2019, pp. 7370–7377.
[15] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, pp. 1263–1272.
[16] T.N. K., M. W., Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, ICLR, 2017.
[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al., Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, ICLR, 2018.
[18] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st Advances in Neural Information Processing Systems, NeurIPS, 2017, pp. 1025–1035.
[19] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, 2020, pp. 639–648.
[20] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun, Masked label prediction: Unified message passing model for semi-supervised classification, in: Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI, 2021, pp. 1548–1554.
[21] H. Zhang, Z. Yu, G. Dai, G. Huang, Y. Ding, Y. Xie, Y. Wang, Understanding gnn computational graph: A coordinated computation, io, and memory perspective, Proc. Mach. Learn. Syst. 4 (2022) 467–484.
[22] A. Gravina, D. Bacciu, C. Gallicchio, Anti-symmetric DGN: A stable architecture for deep graph networks, in: Proceedings of the 11th International Conference on Learning Representations, ICLR, 2023.
[23] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, J. Pei, Am-gcn: Adaptive multi-channel graph convolutional networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD, 2020, pp. 1243–1253.
[24] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[25] Q. Zhang, Y.N. Wu, S.-C. Zhu, Interpretable convolutional neural networks, in: Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2018, pp. 8827–8836.
[26] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NeurIPS, 2016, pp. 3844–3852.
[27] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: Proceedings of the 36th International Conference on Machine Learning, ICML, 2019, pp. 6861–6871.
[28] J. Chen, T. Ma, C. Xiao, Fastgcn: Fast learning with graph convolutional networks via importance sampling, in: Proceedings of the 6th International Conference on Learning Representations, ICLR, 2018.
[29] W. Huang, T. Zhang, Y. Rong, J. Huang, Adaptive sampling towards fast graph representation learning, in: Proceedings of the 32nd Advances in Neural Information Processing Systems, NeurIPS, 2018, pp. 4563–4572.
[30] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD, 2019, pp. 257–266.
[31] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, V. Prasanna, Graphsaint: Graph sampling based inductive learning method, in: Proceedings of the 8th International Conference on Learning Representations, ICLR, 2020.
[32] L.C. Freeman, et al., Centrality in social networks: Conceptual clarification, Soc. Network: Critical Concepts Sociol.. Londres: Routledge 1 (2002) 238–263.
[33] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, K. Borgwardt, Efficient graphlet kernels for large graph comparison, in: Proceedings of Artificial Intelligence and Statistics, PMLR, 2009, pp. 488–495.
[34] L. Wu, D. Wang, K. Song, S. Feng, Y. Zhang, G. Yu, Dual-view hypergraph neural networks for attributed graph learning, Knowl.-Based Syst. 227 (2021) 107185.
[35] S. Yang, S. Verma, B. Cai, J. Jiang, K. Yu, F. Chen, S. Yu, Variational co-embedding learning for attributed network clustering, Knowl.-Based Syst. 270 (2023) 110530.
[36] S. Yang, B. Cai, T. Cai, X. Song, J. Jiang, B. Li, J. Li, Robust cross-network node classification via constrained graph mutual information, Knowl.-Based Syst. 257 (2022) 109852.
[37] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI, 2018, pp. 5900–5907.
[38] W. Zhang, Y. Shen, Z. Lin, Y. Li, X. Li, W. Ouyang, Y. Tao, Z. Yang, B. Cui, Pasca: A graph neural architecture search system under the scalable paradigm, in: Proceedings of the 2022 ACM Web Conference, 2022, pp. 1817–1828.
[39] N.M. Kriege, F.D. Johansson, C. Morris, A survey on graph kernels, Appl. Netw. Sci. 5 (1) (2020) 1–42.
[40] H. Kashima, K. Tsuda, A. Inokuchi, Marginalized kernels between labeled graphs, in: Proceedings of the 12th International Conference on International Conference on Machine Learning, AAAI Press, ICML, 2003, pp. 321–328.
[41] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, J.-P. Vert, Extensions of marginalized graph kernels, in: Proceedings of the 21th International Conference on Machine Learning, ICML, 2004, p. 70.

[42] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt, Graph kernels, J. Mach. Learn. Res. 11 (2010) 1201–1242.
[43] K.M. Borgwardt, H.-P. Kriegel, Shortest-path kernels on graphs, in: Proceedings of the 5th IEEE International Conference on Data Mining, ICDM, 2005, p. 8.
[44] J. Ramon, T. Gärtner, Expressivity versus efficiency of graph kernels, in: Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences, 2003, pp. 65–74.
[45] N. Shervashidze, P. Schweitzer, E.J. Van Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeiler-Lehman graph kernels, J. Mach. Learn. Res. 12 (9) (2011) 2539–2561.
[46] G. Nikolentzos, P. Meladianos, A.J.-P. Tixier, K. Skianis, M. Vazirgiannis, Kernel graph convolutional neural networks, in: Proceedings of the 27th International Conference on Artificial Neural Networks, ICANN, 2018, pp. 22–32.
[47] M. Fey, J.E. Lenssen, F. Weichert, H. Müller, Splinecnn: Fast geometric deep learning with continuous B-spline kernels, in: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2018, pp. 869–877.
[48] S.K. Arul Prakash, C.S. Tucker, Node classification using kernel propagation in graph neural networks, Expert Syst. Appl. 174 (2021) 114655.
[49] A.J.-P. Tixier, G. Nikolentzos, P. Meladianos, M. Vazirgiannis, Graph classification with 2d convolutional neural networks, in: Proceeding of International Conference on Artificial Neural Networks, ICANN, 2019, pp. 578–593.
[50] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: International Conference on Machine Learning, ICML, 2016, pp. 2014–2023.
[51] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: Proceedings of the 2nd International Conference on Learning Representations, ICLR, 2014.
[52] G.X. Feng, Over-Smoothing Algorithm and Its Application to GCN Semi-supervised Classification No. 002, ICPCSEE, 2020, pp. 197–215.
[53] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778.
[54] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: Proceeding of the 33rd International Conference on Machine Learning, ICML, 2016, pp. 40–48.
[55] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, 2018, CoRR abs/1811.05868.
[56] K. Huang, J. Zhai, Z. Zheng, Y. Yi, X. Shen, Understanding and bridging the gaps in current GNN performance optimizations, in: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP, 2021, pp. 119–132.
[57] M. Fey, J.E. Lenssen, Fast graph representation learning with PyTorch Geometric, in: Proceedings of ICLR Workshop on Representation Learning on Graphs and Manifolds, ICLR, 2019.
[58] V.D.M. Laurens, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. (2008) 2579–2605.