International Journal of Machine Learning and Cybernetics (2024) 15:863–877
https://doi.org/10.1007/s13042-023-01944-0

ORIGINAL ARTICLE

A double-layer attentive graph convolution networks based on transfer learning for dynamic graph classification

Lei Yao[1] · Da Guo[1,2] · Xing Wang[3] · Lin Zhu[3] · Junlan Feng[3] · Yong Zhang[1,2]

Received: 1 November 2022 / Accepted: 3 August 2023 / Published online: 18 August 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract
In practical scenarios, many graphs dynamically evolve over time. The new node classification without labels and historical information is challenging. To address this challenge, we design a double-layer attentive graph convolutional network (DLA-GCN) based on the transfer learning, which mainly includes three deep learning components: the double-layer graph convolutional network (DLGCN), node multi-parameter learning (NMPL) algorithm, and domain-adversarial transfer learning (DATL) method. In terms of dynamic spatial correlation, DLGCN jointly exploits the pre-defined and adaptive adjacency matrix to capture local and global feature aggregation. An inter-graph attention mechanism is further used to produce a unified representation for each node in graphs by automatically merging different spatial correlations. To reduce the complexity and improve accuracy, the matrix decomposition method is designed to learn the node-specific patterns of nodes in the NMPL component. In terms of dynamic time correlation, DATL is proposed to learns and transfers similar features as historical information of new nodes by optimizing three different loss functions, namely source classifier loss, domain classifier loss, and target classifier loss as a whole. The experimental results on two real-world graph classification datasets show that the proposed approach can improve the accuracy by 18% and 10%, respectively, compared with the state-of-art baselines.

Keywords Double-layer graph convolutional network · Node multi-parameter learning · Transfer learning · Dynamic graph classification task

1 Introduction

Learning node embeddings in graphs is an important yet challenging task due to its wide application in various fields such as social media and medicine [1, 2]. The existing approaches of graph representation learning mainly focus on static graphs, which can be divided into two categories: solutions based on the pre-defined adjacent matrix [3–5] and solutions based on the adaptive adjacency matrix [6–8]. The former solutions employ a spatial-based or spectrum-based graph convolution neural network [1, 2] to capture local spatial correlation. However, these approaches do not extract global information and are difficult to achieve global optimization. The latter solutions employ a well-designed learning parameter matrix and one-hop graph convolution neural network to capture the hidden inter-dependencies between distant nodes adaptively. However, these approaches do not consider local and global consistency for feature aggregation.

Many methods [1, 2] based on graph convolution neural network (GCN) employ sharing parameters as a common training method to capture the prominent patterns across all nodes. However, owing to complex spatio-temporal factors and external factors (such as weather), adjacent nodes may have diverse patterns [6]. It is not reasonable to only capture the significant patterns and specific patterns of each node must be considered.

Generally, many graphs in the real world are dynamic, which indicates that their structure evolves over time. Owing to new nodes do not have labels and historical information, new node classification faces challenges in the dynamic graph. Recurrent approaches [9, 10] employ the Recurrent

✉ Yong Zhang
yongzhang@bupt.edu.cn

[1] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

[2] Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing 100876, China

[3] Center of AI and Intelligent Operation R &D, China Mobile Research Institute, Beijing, China

Neural Network (RNN) to learn historical information from graph snapshots and use the classical GCN model to extract spatial correlation. These approaches perform well on node representation learning, but they cannot perform reliably for unsupervised classification [9]. Self-attentional approaches [11, 12] can dynamically capture long-term temporal correlation and have a good link prediction performance. However, the performance is still weak in the classification task of new nodes without historical information.

Transfer learning is a method of knowledge transfer between domains. It aims to transfer the information of source domains and improve the performance of target domains. Nowadays, with the success of graph convolution, many approaches [13, 14] is transfer knowledge from the graph of an existing domain where nodes are labeled to classify nodes in the target domain to solve the problem of unlabeled nodes in the target domain. Inspired by these methods, we employ transfer learning on the time steps and transfer the features of known nodes to make the new nodes lean rich historical information so as to enhance the classification performance of unknown nodes.

To resolve these problems, we propose a novel deep learning model based on transfer learning named DLA-GCN, which employs multiple components along spatial and temporal dimensions. Specifically, we design the DLGCN component to fulfill local and global feature aggregation and propose NMPL algorithm to capture different patterns of nodes. Then, we propose the DATL method to learn and transfer dynamic temporal correlation. On two real-world dynamic graph datasets, we evaluate DLA-GCN for the unsupervised dynamic graph node classification task. The experimental results show that DLA-GCN performs consistently over time and outperforms several state-of-the-art baselines. The main contributions of this paper are summarized as follows:

- To capture adaptively global and local spatial correlation, we propose the DLGCN component. Specifically, we employ two different adjacency matrices and an inter-graph attention mechanism to integrate local and global consistency automatically and learn effective node feature embedding in the network.
- By modeling the node parameters, we design the NMPL algorithm. Specifically, the matrix decomposition method is designed to learn the two smaller parameter matrices instead of learning the unique parameter space of each node. The approach can enhance the perception of spatial-temporal features and improve the classification performance.
- We propose the DATL method to exploit source information and target information with different loss functions, so that domain-invariant and feature representations can be effectively learned to reduce the domain discrepancy for new node classification. DATL can significantly accelerate the training speed and boost the classification accuracy.

The remainder of this paper is organized as follows. A brief review of related work is provided in Sect. 2. In Sect. 3, we detail the DLGCN, NMPL, and DATL components to capture spatial-temporal evolutionary features of dynamic graphs. The experiment settings and results are shown in Sect. 4. Section 5 comes to concluding remarks.

2 Related work

Our work involves static graph representation learning, dynamic graph classification task, and transfers learning, and this section will cover the most recent developments.

Static graph embeddings. Recently, many approaches based on graph neural network architectures have achieved great success. Amit Roy et al. [5] proposes the unified spatio-temporal graph convolution network (USTGCN) framework. Based on STSGCN, the framework employs the temporal mask to improve the ability to learn spatiotemporal correlations and models different historical data components (such as hourly, daily, and weekly components) to improve the fine-grained representation. Lei Bai et al. [6] designs the adaptive adjacency matrix to learn spatial correlation automatically and captures hidden correlation between distant nodes easily. However, these approaches only focus on an aspect of spatial correlation and do not both consider local and global consistency [15]. Meanwhile, These methods only employ the shared parameter to capture the prominent patterns. Therefore, we propose the double-layer graph convolution network and NMPL to learn the spatial structural properties and diversified patterns of nodes.

Dynamic graph The dynamic graphs include two common ways: snapshot sequence [11], which consists of a series of graph snapshots at different time steps, and timestamped graph [16], which is a dynamic graph with evolves over continuous time. Although snapshot-based approaches can be employed to timestamped graphs by reducing the number of time steps, these approaches may not perform reliably due to the lack of a fine-grained snapshot sequence [17].

Dynamic graph classification task: Dynamic graph classifications mainly fall into two categories. One class of these approaches employs the recurrent convolutional network, such as evolving graph convolutional networks (EvolveGCN), proposed by Aldo Pareja et al. [9]. The model captured spatio-temporal dynamics by employing long short-term memory (LSTM) or gated recurrent units (GRUs) to evolve the GCN parameters. It is worth mentioning that GCN is not trained, and its parameters are only updated using RNN. Although the performance of

the model is good in short-term dynamic graph node classification, the performance is degraded as the number of time steps increases. The other class uses a self-attention mechanism to capture historical information. Aravind Sankar et al. [11] proposed dynamic self-attention network (DYSAT), which employed the encoder of transformer [18] to learn the dynamic temporal evolution. However, it could not learn the historical information of new nodes and model them [19].

These methods focus on discussing and evaluating the classification performance of nodes with labels and do not extract the historical information of new unseen nodes. We seek to employ the transfer learning method to model new nodes.

Transfer learning: The existing approaches are currently divided into three categories. (1) Instance-based transfer learning. For example, Dai Wenyuan et al. [20] proposes the TrAdaBoost algorithm, which calculates task similarity to make the feature distribution of the target domain close to that of the source domain. This approach is easy to implement. However, the algorithm performs poorly when the source and target domains come from different domains [21]. (2) Feature-based transfer learning. For example, Yaroslav Ganin et al. [22] proposes the DANN framework. The model learns similar features between the source and target domains and transferred them using the DAN method. (3) Parameter-based transfer learning, such as pre-learning models. For example, Google [23] proposes bidirectional encoder representation from transformers (BERT), which tackles downstream tasks by pre-training and fine-tuning model parameters.

Instance-based transfer learning requires the source and target domains to come from the same domain. Parameter-based transfer learning needs a large amount of data and the performance is not outstanding for a single task such as the dynamic graph classification task [24–26]. Therefore, we propose the DATL method to learn and transfer similar features at adjacent time steps with three loss functions.

3 Methodology

In this section, we will present the problem definition and the neural network structure design. The main parameters used in DLA-GCN are summarized in Table 1.

3.1 System model

In this section, the definition of the dynamic graph can be defined on a series of observed snapshots $G = \{\zeta^1, \zeta^2, \cdots, \zeta^t, \cdots, \zeta^T\}$ as the research subject, where T is the number of time steps. At each time step t, each graph snapshot $\zeta^t = (V, \varepsilon^t)$ is a weighted undirected graph with a node-set V, a link set ε^t, an adjacency matrix A^t, and a node feature matrix X^t. Dynamic graph representation learning is to learn the embeddings $e_v^t \in R^d$ for each node $v \in V$ in graphs at the time step $t = \{1, 2, \cdots, T\}$. And the embeddings $e_v^t \in R^d$ represent the aggregation of the spatial features and temporal evolution such as link connection and node addition at time step t. New node classification in dynamic graphs aims to identify the labels of new node by employing the original label set l_o in the source domain and the embeddings $e_v^t \in R^d$.

We propose a new neural network structure named DLA-GCN. As shown in Fig. 1, the model consists of the DLGCN, DATL, and NMPL components. These approaches learn the spatial structural properties by combining the DLGCN and NMPL and employs the DATL algorithm to learn and transfer similar features of dynamic graphs at adjacent time steps. Note that nodes in this section dynamically evolve. In other words, the number of nodes in each time step is different. Therefore, after feature extraction at t, we add 0 to match the

Table 1 Key notation and definition

Symbol	Meaning	Symbol	Meaning
G	The series of observed snapshots	b_1, b_2, b_3	The first, second and third learnable bias
$\zeta^t = (V, \varepsilon^t)$	The graph snapshot and its nodes, links at the time step t	$A_{uv}, e_{uv}, \alpha_{uv}$	The pre-defined adjacent matrix, attention weight and attention value of two adjacent nodes v and u
T, N	The total number of time steps and nodes	$\sigma(\cdot)$	The activation function
v, u	The target node and its neighbor	S_u, x_u	The feature embedding and matrix of the neighbor node u
A^t, X^t	The node adjacency and feature matrix at the time step t	Z_{local}, Z_{global}	The local and global feature representation
e_v^t	The embeddings of the node v at the time step t	E_A, d_c	The learnable node embedding and its dimension
l_o, l_p	The original and predicted labels	I_N, D	The identity and degree matrix
W^s	The learnable parameter shared by all nodes	∂_l, ∂_g	The importance of local feature and global feature
x_v, S_v	The feature matrix and embedding of the target node	Z	The spatial feature representation of nodes
d, λ, u_r	The binary variable, adaptive parameter and learning rate	E_ψ, W_ψ	The decomposed shared and specific parameter
L_y, L_d, L	The label, domain and total loss	b_{ψ_1}, b_{ψ_2}	The first and second decomposed learnable bias

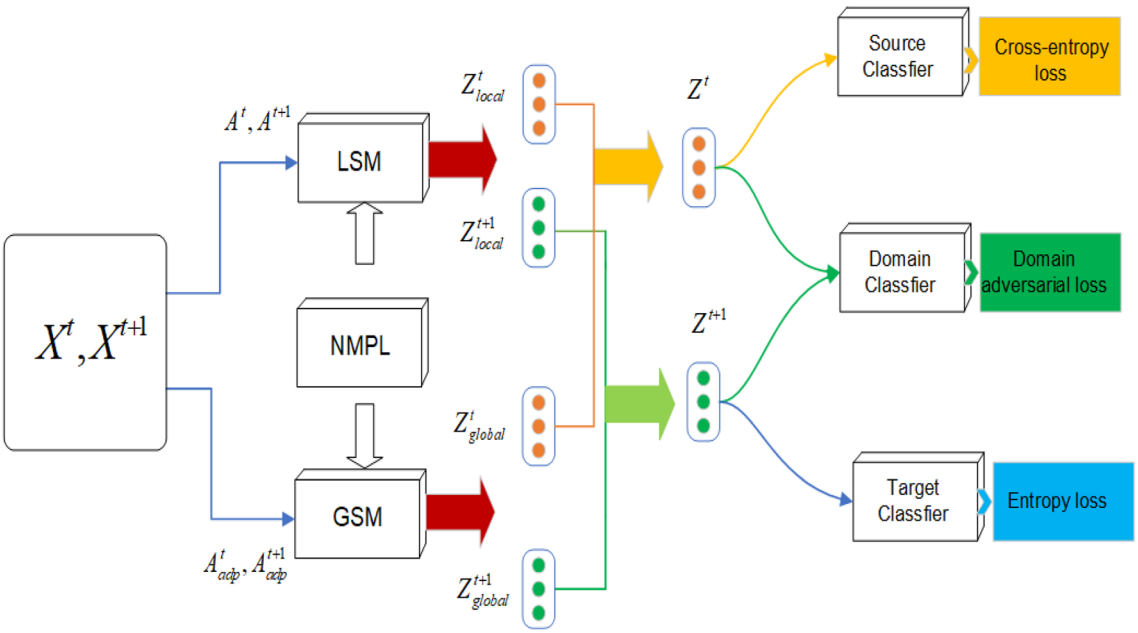

Fig. 1 Network architecture of DLA-GCN

dimension in adjacent time steps t and $t+1$. This approach avoids dimension mismatches and prevents the occurrence of the multi-zero matrix.

3.2 Spatial module

We proposes DLGCN to capture local and global features of spatial structure properties and learn more hidden information. Figure 2 illustrates the overall architecture of DLGCN. The network consists of a local spatial module, a global spatial module, and an inter-graph attention mechanism. Essentially, the spatial feature representation is the aggregation of spatial information at each time step. In other words, all operations of DLGCN should be completed in a time step and the total number of nodes in a time step remain unchanged.

3.2.1 Local spatial module

The one-hop neighbors of the each node v are aggregated by an attention mechanism to capture the local features.

First, the feature embedding of each node is defined as:

$$S_v = W^s x_v + b_1 \tag{1}$$

where W^s denotes a learnable parameter shared by all nodes, x_v is the feature matrix of each node v as input of the layer, and b_1 is a learnable bias.

The pre-defined adjacent matrix is obtained by the graph snapshot ζ at the time step t, and the one-hop neighbors of the each node v are selected as node topology information through the pre-defined adjacent matrix. Furthermore, the attention weight of the node v and its neighbors can be calculated by:

$$\begin{aligned} S_u &= W^s x_u + b_2 \\ e_{uv} &= \sigma(A_{uv}[S_u + S_v])\forall(u,v) \in \varepsilon \end{aligned} \tag{2}$$

where A_{uv} denotes the pre-defined adjacency matrix of two adjacent nodes u and v and $\sigma(\cdot)$ is activation functions, such as ReLu, Sigmoid, and so on. Note that the attention weight is obtained by adding the embedding matrix. Other techniques, such as splicing, can also be used to calculate the embedding matrix.

Then the calculated attention is normalized by the following equation:

$$\alpha_{uv} = \frac{exp(e_{uv})}{\sum\limits_{w \in N_v}^{n} exp(e_{uv})} \tag{3}$$

Finally, the local feature representation of each node v is calculated by aggregating the one-hop neighbors according to the previously required attention weight method:

$$Z_{local} = z_v = \sigma(\sum_{w \in N_v}^{n} \alpha_{uv}(W^s x_u + b_2))\forall v \in V \tag{4}$$

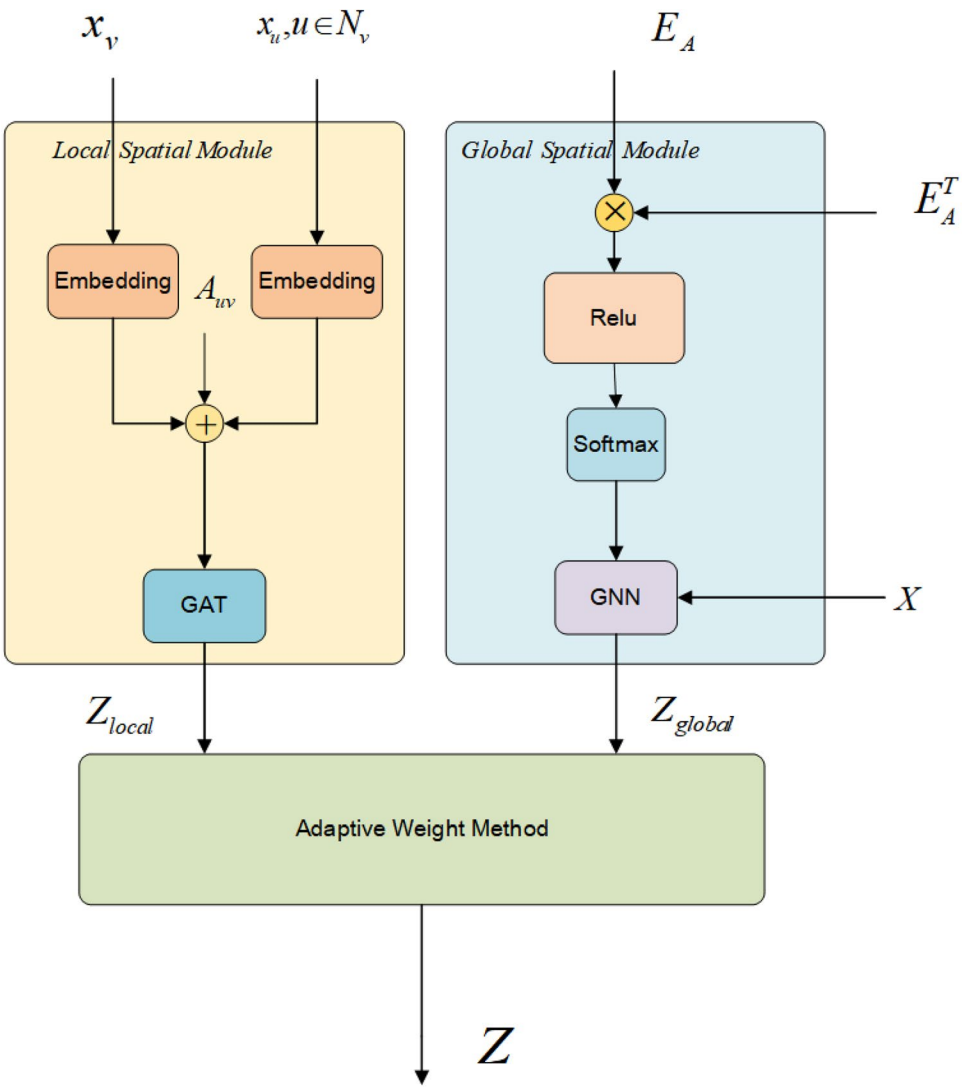

Fig. 2 Framework of the DLGCN method

3.2.2 Global spatial module

In addition to local feature learning which calculated by distance or similarity, we employ an adaptive adjacency matrix to model the global information.

Specifically, a learnable node embedding $E_A \in R^{N \times d}$ is randomly initialized, where N denotes the total number of nodes and d_c is the each node embedding dimension (generally $d_c \ll N$). The global features of nodes are calculated by multiplying E_A and E_A^T:

$$A_{adp} = I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = Softmax(ReLU(E_A \cdot E_A^T)) \quad (5)$$

The learned adjacency matrix is normalized by softmax. The adaptive adjacency matrix captures global spatial information using E_A, which is more simple than [35]. Finally, the global feature embedding is represented as:

$$Z_{global} = (Softmax(Relu(E_A \cdot E_A^T)))XW^s + b_3 \quad (6)$$

where X denotes the node feature matrix, W^s is the learnable parameter shared by all nodes, and b_3 is the learnable node bias.

3.2.3 Inter-graph attention

After performing the DLGCN component, we obtain two feature embeddings Z_{local}, Z_{global}. We need to aggregate different embeddings to produce a unified representation. As embeddings from the local and global spatial module contribute differently to learning the representation, we propose an inter-graph attention scheme to capture the significance of each feature embedding.

The attention mechanism between the local and global feature representation is calculated by the equation:

$$\begin{aligned} \partial_l &= \frac{exp(Z_{local})}{exp(Z_{local}) + exp(Z_{global})} \\ \partial_g &= \frac{exp(Z_{global})}{exp(Z_{local}) + exp(Z_{global})} \end{aligned} \tag{7}$$

where ∂_l and ∂_g denote the importance of local feature and global feature, respectively.

Furthermore, the spatial feature representation of nodes is denoted as:

$$Z = \partial_l Z_{local} + \partial_g Z_{global} \tag{8}$$

3.3 Node multi-parameter learning

In deep learning models, parameter sharing is a widely used technique that can significantly reduce the number of model parameters during training, thus accelerating training and improving the model's generalization ability. However, in dynamic graphs, each node may have unique features and attributes, so the use of shared parameters may not accurately capture the unique characteristics of each node. Moreover, if only one shared parameter space is learned, it may not be able to handle the complexity and dynamic variability of node parameters. Therefore, to fully exploit node parameter information while maintaining node uniqueness, we propose the NMPL algorithm, which can learn individual parameters for each node and improve the model's performance. As a result, each node should maintain its own parameter space to learn node-specific patterns.

By considering the matrix decomposition method, this NMPL method learns two smaller parameter matrices instead of learning the unique parameter space of each node. The parameter W^s can be represented by $W^s = E_\psi \cdot W_\psi$: (1) a shared parameter $E_\psi \in R^{N \times d_c}$, where d_c denotes the embedding dimension, and $d_c \ll N$. (2) a specific parameter $W_\psi \in R^{d \times N}$. The method can be interpreted as capturing the node-specific patterns from a set of candidate patterns discovered from all nodes in graphs. As a result, NMPL not only reduces the number of parameters but also learns the more fine-grained patterns of the nodes.

By employing E_A as a shared parameter between local and global features, Eqs. 4 and 6 can be re-expressed as:

$$\begin{aligned} Z_{local} &= z_v = \sigma(\sum_{w \in N_v}^{n} \alpha_{uv}(E_A W_\psi x_u + E_A b_{\psi_1})) \forall v \in V \\ Z_{global} &= (Softmax(Relu(E_A \cdot E_A^T)))XE_A\theta + E_A b_{\psi_2} \end{aligned} \tag{9}$$

3.4 Domain-adversarial transfer learning

To better learn a knowledge transfer across different time step to assist in the dynamic node classification task, our model consists of a adversarial module, a source classifier as well as a target classifier working together to to learn both domain node representations, thus enabling classifying nodes in the next time step. The overall objective is as follows:

$$L(Z^t, Y^t, Z^{t+1}) = L_S(Z^t, Y^t) - \lambda * L_{DA}(Z^t, Z^{t+1}) - \gamma * L_T(Z^{t+1}) \tag{10}$$

where λ and γ denote the hyper-parameters that tune the trade-off between domain loss, target loss function and total loss function during the learning process. The L_S, L_{DA}, L_T are the source classifier loss, the domain classifier loss and the target classifier loss, respectively.

Source classifier loss: For the source classifier, the negative log-probability method is employed as the loss function and is written as:

$$L(Z^t, Y^t) = -\frac{1}{N_t^K} \sum_{i=1}^{N_t^K} log(\hat{y}_t) \tag{11}$$

where y_i denotes the label of the i-th node in the source domain, $\hat{y}_t$ is the classification prediction for the i-th source labeled node, respectively. The loss of the source classifier is minimized to improve the prediction accuracy.

Domain classifier loss: Different from minimizing the loss of the label classifier, the domain classifier can make the features of the source and target domains indistinguishable by maximizing the domain discriminator loss. To achieve this, we learn a domain classifier with an adversarial transfer training. On the one hand, the source classifier can classify each node into the correct class via minimizing the source classifier loss. On the other hand, node representations from different domains can be similar, so that the domain classifier cannot differentiate if the node comes from source domain or target domain. In this paper, Gradient Reversal Layer (GRL) [27] is used for adversarial training. Learning a GRL is adversarial in such a way that: the reversal gradient enforces to be maximized, at the same time, the cross-entropy domain classifier loss is be minimized:

$$L_{DA}(Z^t, Z^{t+1}) = -\frac{1}{N_{sum}} \sum_{i=1}^{N_{sum}} dlog(\hat{m}_i^{t+1}) + (1-d)log(1-\hat{m}_i^t) \tag{12}$$

where N_{sum} denotes the total number of nodes in the source and target domain, d is the binary variable, which indicates whether the feature comes from the source or target domain. The $\hat{m}_i^{t+1}$ and $\hat{m}_i^t$ are the domain prediction for the i-th document in the target domain and source domain, respectively.

Target classifier loss: An entropy loss is placed on the target classifier. Unlike the source classifier, we do not use cross-entropy as the label loss, because we do not have the label information for the new node in the target domain. In order to utilize the data in the target domain, we employ an entropy loss for the target classifier:

$$L_T(Z^{t+1}) = -\frac{1}{N_{t+1}^U}\sum_{i=1}^{N_{t+1}^U}\hat{y}_i log(\hat{y}_i) \tag{13}$$

Where $\hat{y}_i$ denotes the the classification prediction for the i-th node the target domain.

$L_T(Z^t, Y^t)$, $L_{DA}(Z^t, Z^{t+1})$ and $Z_T(Z^{t+1})$ are jointly optimized via our objective function in Eq. 10, and all parameters are optimized using the standard backpropagation algorithms.

Algorithm 1 summarizes the unsupervised dynamic graph classification learning model.

Algorithm 1 Unsupervised dynamic graph node classification

Input: Dynamic graph snapshot datasets $G = \{\zeta^1, \zeta^2, \cdots, \zeta^t, \cdots, \zeta^T\}$
Output: The classification results $\{l_p^1, l_p^2, \cdots, l_p^t, \cdots, l_p^T\}$

Acquire the feature matrix X^t, adjacency matrix A_{adp}^t and original label l_o
At the spatial dimension:
for t in T **do**
    Calculate local spatial representation Z_{local}^t by(1)-(4)
    Use NMPL algorithm, Calculate Z_{local}^t replace (4) by (9)
    Determine global spatial representation Z_{global}^t by (5)-(6)
    Use NMPL algorithm, Calculate Z_{global}^t replace (6) by (9)
    In order to fuse global and local features, employ adaptive weights ∂_l^t and ∂_g^t by (7), then Calculate Z^t by (8)
end for
At the time dimension:
for t in T **do**
    Calculate the source classifier loss L_S of t by (11)
    Determine the domain loss L_{DA} of t and $t+1$ by (12)
    Determine the target loss L_T of $t+1$ by (13)
    Calculate the total loss L of t and $t+1$ by (10), transfer the similar feature
end for

4 Experiments and analysis

4.1 Dataset

This paper uses two real-world dynamic graph datasets: Cora dataset [28] and the BlogCatalog dataset [29].

The Cora dataset consists of 2708 nodes classified into one of seven classes, including Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning, and Theory. Meanwhile, the citation network consists of 5429 edges. Each node represents a scientific paper, and the edges denote reference flows of scientific papers. The Cora dataset is widely used in the field of dynamic graph classification task [30, 31]. And our purpose is to more accurately study the impact of research papers. Specifically, as time progresses, the influence of papers tends to evolve. By learning the dynamic graph, we can investigate the impact and its variations more accurately. We use temporal directed edges that represent citations from one paper to another, with timestamps of the citing paper's publication date as in [32].The method of transforming the Cora static graph dataset into a dynamic graph dataset is as follows: Firstly, select these papers between 1900 and 1988 as known nodes, and use their connection relationships, feature attributes, and corresponding classification labels to construct the first static graph of the initial time step. Secondly, for each additional time stamp, papers' feature attributes and connection relationships until 1999 are added to incorporate new nodes without labels, and

complete the construction of the topologies of graphs at different time stamps. Finally, a dynamic graph dataset with 12 time steps is formed.

The given adjacency matrix of Cora is simply formulated based on the practical citation outcome between the two papers. In terms of the technical components, papers *i* and *j* are on completely different topics, for example computer sciences and chemistry, while the citation between them is barely because the algorithm/model developed in paper *i* is used in *j* for special application. It means, apart from the given adjacency matrix, there are other possible viewpoints to better represent the relationship between papers *i* and *j* [33, 34]. To achieve this, we utilize the similarity of node features and an attention mechanism when computing the representations of each node in the graph. Specifically, during the representation process, we assign different weights to neighboring node features based on their dissimilarity. This allows us to capture more diverse and informative information. By incorporating the node feature similarity and attention mechanism, we enhance the learning of richer and more relevant information for each node in the computational graph [35, 36].

The BlogCatalog dataset is a network of social relationships in the BlogCatalog website, which consists of 5196 nodes and 171743 edges. And the 8189 nodes' attributes are constructed by keywords, which are generated by users as a short description of their blogs. The social network has 6 labels which represents the topic categories provided by the authors. The BlogCatalog dataset is a network dataset that contains social relationships between blog users. This dataset is widely used in the field of social network analysis and dynamic graph classification task. Moreover, the BlogCatalog dataset is used as a dataset in many important work [37, 38].

4.2 Experimental setting

We implement DLA-GCN in Python3.6 with Tensorflow [39] and employ mini-batch gradient descent with Adam optimizer for training. On the Cora data set, for structural multi-head self-attention of graph attention network (GAT), the number of layer and attention heads are 1and 16, respectively. Each attention calculates 8 features(for a total of 128 dimensions). For DATL, the adaptive parameter λ is chosen among 9 values between 10^{-3} and 1 on a logarithmic scale. $\lambda \approx 0.31$ has the best performance and the learning rate is set as $u_r = 10^{-3}$. Two MLP layers and softmax are used to perform multi-classification training of the target and source domains. The balance parameters set to 0.8. On the BlogCatalog data set, self-attention GAT layers and attention heads are 2 and 16, respectively. Each attention calculates 16 features. For DATL, the adaptive parameter λ is set as 0.64, and fix the learning rate u_r is fixed as 10^{-3}. The balance parameters set to 0.8. Finally, we select 60% of all data sets as training sets, 15% as validation sets, and 25% as test sets.

4.3 Baseline

In order to demonstrate the effectiveness of our proposed model, we employ the following methods as baselines. We compare our approaches with both state-of-the-art GCN-based dynamic node classification models and models based on graph transfer learning methods. Note that to apply GCN to dynamic graphs, two inputs are required: a feature matrix composed of all the nodes in the graph, which can be obtained by considering the nodes at each time step, and an adjacency matrix representing the connections between nodes at each time step. In this paper, the adjacency matrix is obtained by calculating the similarity between node features. In summary, we apply GCN to the task of node classification on dynamic graphs using the feature matrix of nodes at each time step and the pre-defined adjacency matrix.

Table 2 Comparison between different models

	Cora			BlogCatalog		
Model	Accuracy	AUC	F1	Accuracy	AUC	F1
GCN	28.8	52.2	15.4	61.6	88.3	67.3
GAT-AE	34.0	54.3	27.2	60.1	86.4	64.8
DYSAT	41.8	70.1	30.4	70.4	90.7	69.7
DS-TAGCN	56.1	72.6	40.3	72.6	91.4	74.1
FADGC	59.2	75.3	45.1	75.8	92.8	76.9
TL-DCRNN	61.5	78.4	52.7	78.3	93.2	77.4
NodeTrans	61.5	85.3	63.2	80.2	95.8	81.1
DLA-GCN	**76.3**	**92.4**	**76.1**	**88.2**	**98.3**	**86.2**

Bold indicates the optimal metric result

State-of-the-art GCN-based dynamic node classification models:

- *GCN* [28]: It is a classic network which can efficiently capture the spatial information.
- *GAT-AE* [40]: The model is a graph convolutional network with an attention mechanism, which is an autoencoder to model the spatial correlation.
- *DySAT* [11]: DySAT is a dynamic self-attention approach, which uses GAT to capture spatial correlation, and uses the self-attentional architecture of transformer encoder to capture dynamic temporal correlation
- *DS-TAGCN* [12]: DS-TAGCN is a a dual-stream topology attentive graph convolutional network for dynamic graph node classification, which can learn the evolution pattern of node attributes and graph topology simultaneously.
- *FADGC* [41]: The model is a stable and scalable dynamic GCN method using a fine-grained attention mechanism.

Dynamic node classification models based on graph transfer learning:

- *TL-DCRNN* [42]: TL-DCRNN is a diffusion convolutional recurrent neural network based on a new transfer learning approach. The network performs dynamic node classification through downstream fine-tuning, such as the MLP and Conv operations.
- *NodeTrans* [14]: NodeTrans is a graph transfer learning approach which combines the spatial-temporal graph network and transfer learning. The model performs dynamic node classification through downstream fine-tuning, such as the MLP and Conv operations.

4.4 Experimental results

Table 2 summarizes the performances of different models on the Cora and BlogCatalog datasets. We adopt accuracy, AUC and Fl as our evaluation metrics.

Short-term classification performance analysis: As shown in Table 2, DLA-GCN is superior to the baselines for all three metrics with significant improvements. In particular, DLA-GCN achieves 18% and 10% improvements in the accuracy on the Cora and BlogCatalog dataset compared with DYSAT, which is the state-of-the-art method. Dynamic Graph-based methods (DYSAT, DS-TAGCN and FADGC) have better performances than the traditional static graph embedding methods (GCN and GAT-AE). It shows that the spatio-temporal dynamics GCN with an attention mechanism encoding both local graph structure and nodes evolution pattern have competitive advantages than traditional GCN-based models in dynamic node classification. Compared with dynamic GCN-based node classification methods (DS-TAGCN and FADGC), TL-DCRNN and NodeTrans have better performance, confirming the superiority of transfer learning in dynamic node classification.

Long-term classification performance analysis: Fig. 3 further shows the accuracy performance of different models over the 12 time steps in Cora and BlogCatalog datasets. As can be seen from the results: (1) The a dynamic GCN-based node classification methods (DS-TAGCN, FADGC) can achieve high accuracy in short-term timesteps(the first step), but the accuracy will decline rapidly with the accumulation of missing historical information for new nodes in long-term nodes classification. (2) DLA-GCN balances short-term and long-term classification well and achieves the best performance for almost all horizons. This finding shows that the robustness of the DLA-GCN is superior to

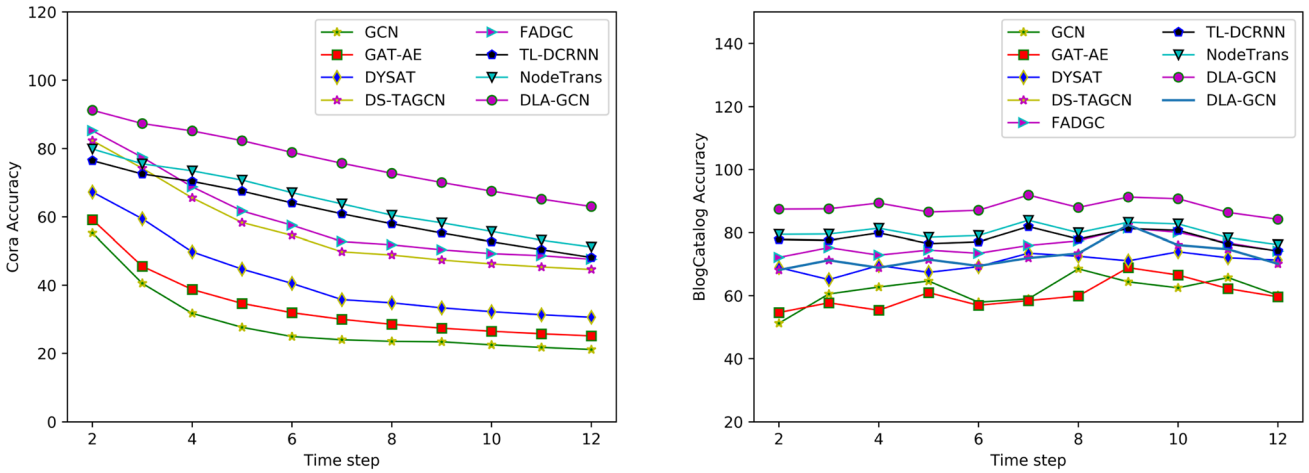

Fig. 3 Performance changes over 12 time steps of different models on the Cora and BlogCatalog datasets

Table 3 Comparisons of classification performances of the different DLA-GCN variants

Model	Cora			BlogCatalog		
	Accuracy	AUC	F1	Accuracy	AUC	F1
DLA-GCN	**76.3**	**92.4**	**76.1**	**88.2**	**98.3**	**86.2**
w/o DATL	67.5	85.2	66.4	74.7	92.1	75.6
w/o ADP	72.7	91.2	73.6	84.8	96.3	83.4
w/o SAT	74.6	91.8	74.8	86.2	97.5	84.2
w/o NMPL	71.3	90.7	69.7	81.4	96.1	81.6
r/ DATL w/ SAT	70.3	89.6	73.5	76.0	92.8	76.9
r/ SF w/ OR	74.4	91.5	74.1	85.7	97.2	84.8
r/ IG w/ CN	71.9	91.0	72.3	80.1	95.8	81.1
r/ NMPL w/ SP	74.3	91.7	74.6	83.5	96.2	82.3

Bold indicates the optimal metric result

Note that w/o represents "without", w/ represents "with", "r/" represents "replace", "SAT" denotes a self-attention mechanism, "ADP" denotes the adaptive adjacency matrix, "OR" denotes original nodes with labels as the source domain input, "SF" denotes similar nodes as the source domain input, "CN" denotes similar nodes as the concatenation operation, "IG" denotes inter-graph attention method, "SP" denotes sharing parameters method

those of the other methods. Our model mainly utilizes the double-layer graph and transfer learning methods to improve the classification performance. On the 12 time steps, our model gradually decreases due to the continuous increase of new nodes. In contrast, most other models rapidly decrease in the first few time steps because the new nodes can only be classified through spatial information extraction, without any historical information or corresponding labels, making it impossible to learn better feature performance from known nodes. The accuracy of the GCN method using static graph is 0.81, however, the accuracy of the DLA-GCN method using dynamic graph is only 0.76. The main reason is that static and dynamic graphs contain different node information. Dynamic graphs have less node information such as the missing label and historical information. After the static graph is converted to dynamic graph, the accuracy of the existing method decreases because of the missing node information [31, 32, 43, 44]. Our method effectively improves the prediction accuracy of dynamic graphs. The dynamic graph consists of a series of graph snapshots at 12 different time steps, and new nodes have no labels or historical information. When converting the Cora dataset into a dynamic graph with 12 temporal steps, only the first temporal step contains nodes between 1900 and 1988 with corresponding labels, and the additional nodes until 1999 added in each temporal step do not have labels. Therefore, new node are only classified through unsupervised learning. As the number of temporal steps increases, the performance will decrease, and the overall performance is only 0.76. In contrast to the dynamic graph dataset, the Cora static graph dataset consists of 2708 labeled nodes that can be classified through supervised learning. It has the complete knowledge for node classification tasks, which does not change with the increase of temporal steps.

4.5 Model ablation analysis

To sufficiently investigate the effect of different components and source domains input in the DLA-GCN, we evaluate the prediction performance of eight variants of the DLA-GCN.

(1) Effectiveness evaluation of DATL component: Table 3 summarizes classification performances of the different DLA-GCN variants on the Cora and BlogCatalog dataset. Compared with "without (w/o) DATL" and "r/ DATL w/ SAT" variants, the DLA-GCN achieves significant improvements in dynamic node classification. The comparison results proves that the effectiveness of the DATL component. According to the left panel of Fig. 4, our model consistently outperforms the "r/ DATL w/ SAT" variant, which demonstrates that the DATL component obtains the better performance in almost all time steps compared with the self-attention mechanism. When the the "without (w/o) DATL" and "r/ DATL w/ SAT" variants are compared, the slightly better performance indicates that the self-attention mechanism improves learning efficiency [11] by capturing some historical information of other nodes.

(2) Effectiveness evaluation of different source domains input: Experimental results of models based on different source domains input on Cora and BlogCatalog datasets are shown in the right panel of Fig. 4. When the DLA-GCN is compared with the "r/ SF w/ OR" variant, the smaller performance indicates that information of similar features may be slightly less than the original

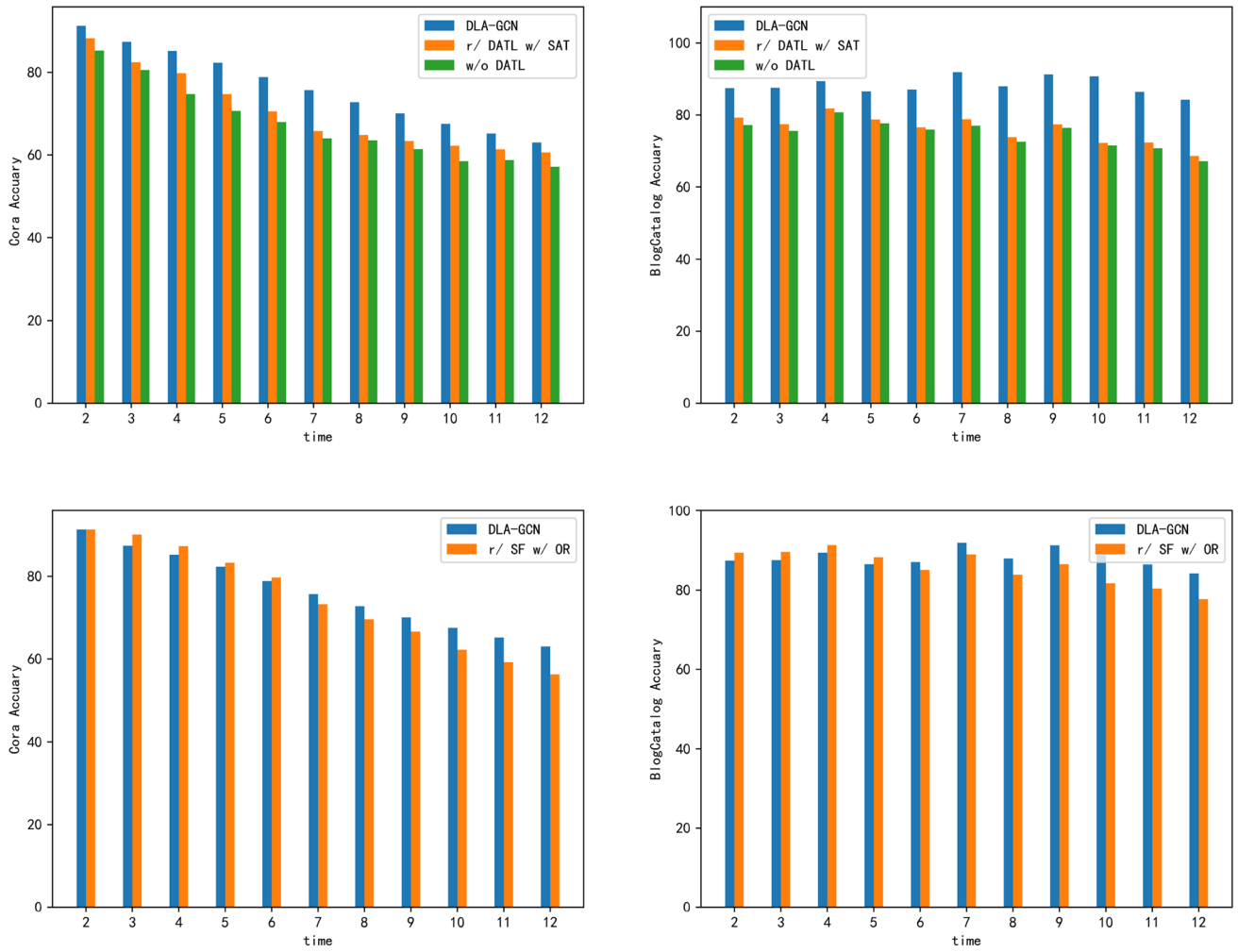

Fig. 4 Influence of transfer learning on the Cora and BlogCatalog datasets

features in the shot-term time step. However, with time steps increase, DLA-GCN achieves significant improvements compared with the "r/ SF w/ OR" variant. The results prove that the effectiveness of the similar features in the long-term node classification task. In other words, The correlation between the new nodes and the original nodes gets weaker over time to lower the performance of transferring features.

(3) Effectiveness evaluation of double-layer graph convolutional network component: From table 3, We can observe that (1) Compared with the "without (w/o) ADP", the better performance of the DLA-GCN demonstrates the importance of capturing strong local dependencies. (2) the "without (w/o) SAT" variant is superior to the "without (w/o) ADP", which shows that global feature extraction methods are more effective than local feature extraction approaches, which indicates global spatial module based on the adaptive matrix can capture the local correlation of some nodes.

(4) Effectiveness evaluation of inter-graph attention component: As shown as Fig. 5, the accuracy curves of DLA-GCN are above the curves of the "r/ IG w/ CN" variant. This result demonstrates that local and global spatial correlations are different and dynamic and prove that the inter-graph attention component enables the dynamic selection of information flows from two different spatial feature extraction modules.

(5) Effectiveness evaluation of NMPL component: The experimental results of models based on different parameters are shown in the right panel of Fig. 5. From the sub-figure, the performance of DLA-GCN is generally better than the "r/ NMPL w/ SP" variant at different time slots. Its good performance proves that the effectiveness of NMPL component. We can conclude from the comparison that sharing parameters only learn the prominent patterns of all nodes and do not capture the possible node-specific patterns. In brief, the NMPL algorithm not only employs the sharing parameters to learn prominent pat-

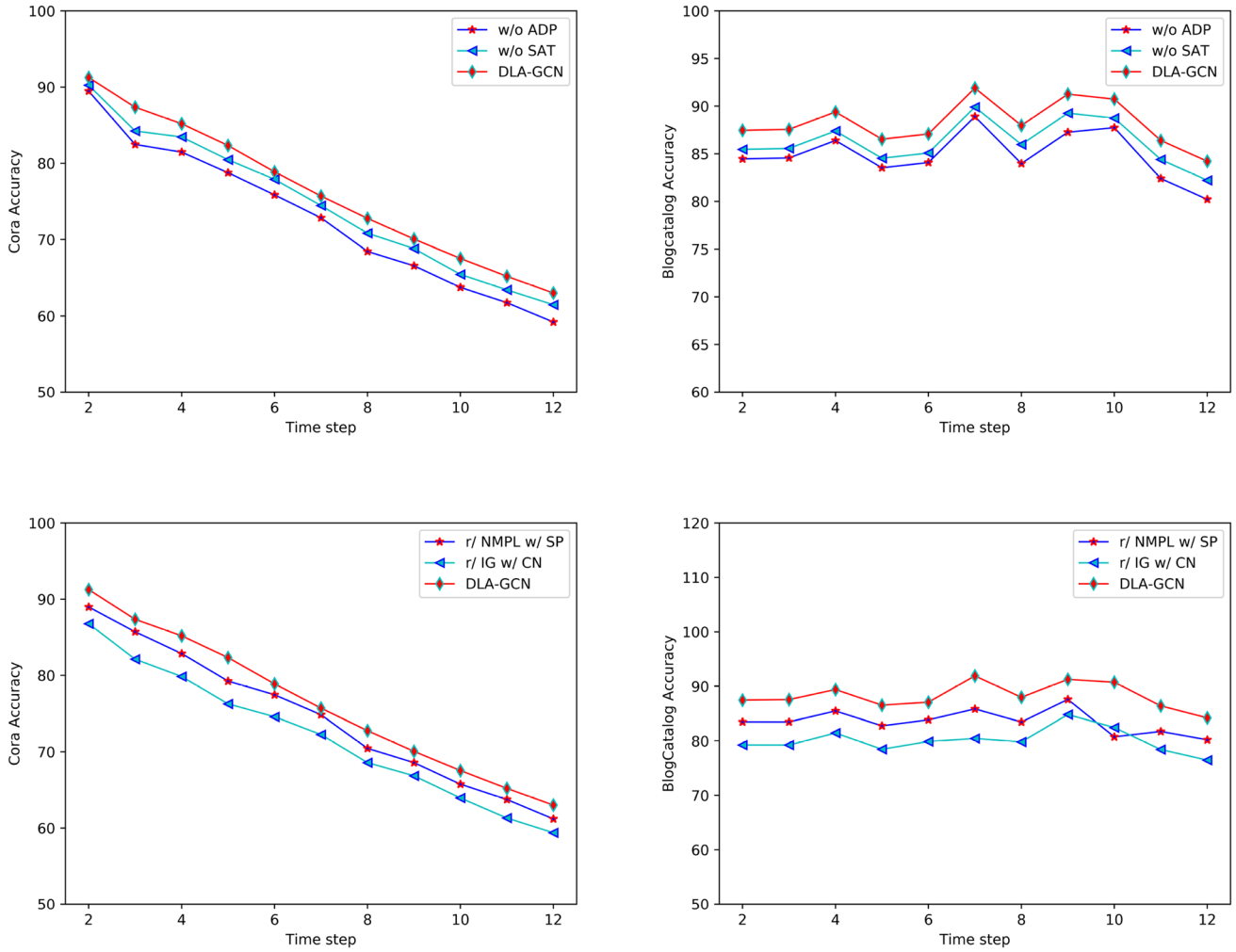

Fig. 5 Ablation experiment of the different spatial modules,attention weight and NMPL on the Cora and BlogCatalog datasets

Table 4 Time complexity comparison between different time steps on BlogCatalog datasets

Model	TrainingTime(s/espoch)			Statics
	The 12th time step	The 14th time step	The 16th time step	$O(N^2)$
DLA-GCN	45.27	68.15	103.597	

terns but independent parameters to capture more fine-grained patterns.

(6) Reasonableness evaluation of time steps: In order to demonstrate the relationship between time steps and time complexity, we conducted comparative experiments as shown in Table 4. From Table 4, it can be observed that the model complexity exhibits an exponential relationship with the number of nodes. As the number of nodes in the dynamic graph increases with each time step, we need to consider the relationship between the number of

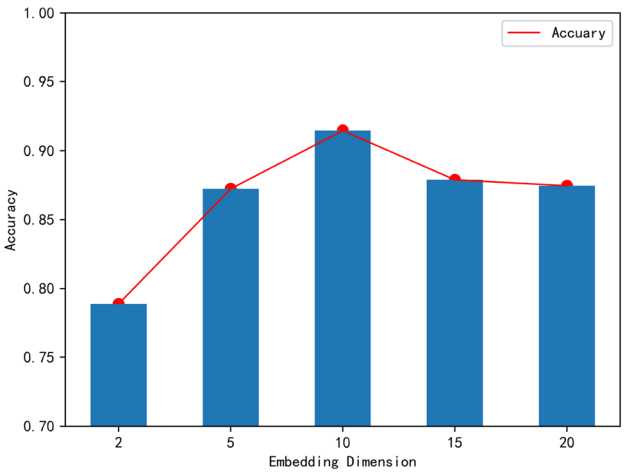

Fig. 6 Influence of embedding dimension on the Cora dataset

Table 5 Time complexity comparison and model performance on the Cora and BlogCatalog datasets. Note that "SP" denotes sharing parameters method

Model	Cora		BlogCatalog	
	Training Epoch	Accuracy	Training Epoch	Accuracy
DLA-GCN	80	**76.3**	95	**88.2**
r/ NMPL w/ SP	**75**	74.4	**87**	83.5

Bold indicates the optimal metric result

Table 6 Computation cost comparison on the Cora and BlogCatalog datasets

Model	TrainingTime(s/epoch)	
	Cora	BlogCatalog
GCN	7.58	15.43
GAT-AE	10.45	16.72
DYSAT	30.76	35.21
DS-TAGCN	146.08	218.84
FADGC	128.35	164.08
TL-DCRNN	24.66	32.54
NodeTrans	18.08	24.65
DLA-GCN	**13.78**	**18.37**

Bold indicates the optimal metric result

time steps and time complexity. By comparing the time complexity between time steps the 16th, 14th, and 12th, we found that the time complexity increases significantly after the 12th time steps, which indicates that we should balance the relationship between long-term analysis and complexity. Meanwhile inspired by the time step usage in other authoritative papers on dynamic graphs. [30, 37], we chose 12 time steps for long-term performance analysis.

4.6 Model analysis

Embedded dimension: The dimension of node embedding is a key hyper-parameter in DLA-GCN. It influences DLGCN to capture spatial correlation and decides the shared parameter diversity. The performance of different embedded dimensions on the Cora dataset is given in Fig. 6. DLA-GCN has good performance on all embedded dimensions. Besides, DLA-GCN has the best performance when the embedding dimension is 10. An excessively small or large embedded dimension makes the performance of DLA-GCN weaker. Therefore, DLA-GCN may employ the appropriate embedding dimension to balance the performance and complexity.

The NMPL algorithm time complexity: To demonstrate the appropriateness of the parameter increase in the NMPL algorithm, we present in Table 5 the time complexity and model performance of the DLA-GCN and the 'r/ NMPL w/ SP' variant. From Table 5, it can be observed that DLA-GCN exhibits significant improvement in performance compared to the 'r/ NMPL w/ SP' variant on both datasets, while the increase in time complexity is not significant. Therefore, considering the significant performance improvement, the computational cost of the NMPL algorithm is moderate.

Computation cost: In order to further evaluate the computation cost between DLA-GCN and baseline models, we compare training time with GCN, GAT-AE, DYSAT, DS-TAGCN, FADGC, TL-DCRNN. and NodeTrans models (as shown in Table 6). FADGC has a longer training time and DS-TAGCN has the longest training time owing to the complex spatio-temporal self-attention mechanism. DLA-GCN employs the DATL method to transfer the similar feature, resulting in a decrease in the trainingtime.

5 Conclusion

In this work, we propose DLGCN to capture more spatial correlation. Specifically, The network employs the different adjacency matrices to exploit both local and global information of the graphs. Furthermore, we propose an inter-graph attention mechanism to adaptively aggregate a unified embedding for node classification task. To both capture prominent patterns and node-specific patterns, we design the NMPL algorithm. Specifically, the algorithm employs the matrix decomposition method to learn the two smaller parameter matrices instead of learning the unique parameter space of each node. In terms of dynamic temporal correlation, we propose DATL method to learn and transfer similar features at different time steps as historical information of new nodes. The experiments verify the effectiveness of the double-layer graph and NMPL in terms of spatial feature aggregation, and also verify the effectiveness of DATL in terms of temporal dynamics.

Acknowledgements This work has been supported by the National Natural Science Foundation of China under Grant No.61971057 and MoE-CMCC "Artifical Intelligence" Project No. MCM20190701.

Funding This work has been supported by the National Natural Science Foundation of China under Grant No.61971057 and MoE-CMCC "Artifical Intelligence" Project No. MCM20190701.

Data availability The data sets supporting the results of this article are included within the references.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710
2. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864
3. Yu B, Yin H, Zhu Z (2017) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875
4. Song C, Lin Y, Guo S, Wan H (2020) Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 914–921
5. Roy A, Roy KK, Ali AA, Amin MA, Rahman AM (2021) Unified spatio-temporal modeling for traffic forecasting using graph neural network. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE
6. Bai L, Yao L, Li C, Wang X, Wang C (2020) Adaptive graph convolutional recurrent network for traffic forecasting. Adv Neural Inf Process Syst 33:17804–17815
7. Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 922–929
8. Li R, Wang S, Zhu F, Huang J (2018) Adaptive graph convolutional neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32
9. Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, Kaler T, Schardl T, Leiserson C (2020) Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5363–5370
10. Li Y, Yu R, Shahabi C, Liu Y (2017) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926
11. Sankar A, Wu Y, Gou L, Zhang W, Yang H (2020) Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 519–527
12. Ruan J, Chen H, Wang Z, Chen S (2021) Ds-tagcn: A dual-stream topology attentive gcn for node classification in dynamic graphs. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE
13. Dai Q, Wu X-M, Xiao J, Shen X, Wang D (2022) Graph transfer learning via adversarial domain adaptation with graph convolution. IEEE Transactions on Knowledge and Data Engineering
14. Yin X, Li F, Shen Y, Qi H, Yin B (2022) Nodetrans: A graph transfer learning approach for traffic prediction. arXiv preprint arXiv:2207.01301
15. Hou W, Huang H, Peng Q, Yu R, Yu L, Wang L (2022) Spatial-hierarchical graph neural network with dynamic structure learning for histological image classification. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 181–191. Springer
16. Trivedi R, Dai H, Wang Y, Song L (2017) Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In: International Conference on Machine Learning, pp. 3462–3471. PMLR
17. Micheli A, Tortorella D (2022) Discrete-time dynamic graph echo state networks. Neurocomputing 496:85–95
18. Han K, Xiao A, Wu E, Guo J, Xu C, Wang Y (2021) Transformer in transformer. Adv Neural Inform Process Syst 34
19. Qu H, Li L, Li Z, Zheng J, Tang X (2022) Robust discriminative projection with dynamic graph regularization for feature extraction and classification. Knowl-Based Syst 253:109563
20. Huang X, Rao Y, Xie H, Wong T-L, Wang FL (2017) Cross-domain sentiment classification via topic-related tradaboost. In: Thirty-First AAAI Conference on Artificial Intelligence
21. Fu, D., He, J.: Sdg: A simplified and dynamic graph neural network. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2273–2277 (2021)
22. Wu M, Pan S, Zhou C, Chang X, Zhu X (2020) Unsupervised domain adaptive graph convolutional networks. In: Proceedings of The Web Conference 2020, pp. 1457–1467
23. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
24. Kim B-H, Ye JC, Kim J-J (2021) Learning dynamic graph representation of brain connectome with spatio-temporal attention. Adv Neural Inf Process Syst 34:4314–4327
25. Liu Q, Dong Y, Zhang Y, Luo H (2022) A fast dynamic graph convolutional network and cnn parallel network for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing
26. Chen J, Jiao L, Liu X, Li L, Liu F, Yang S (2021) Automatic graph learning convolutional networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 60:1–16
27. Liao T, Chen J-C, Jeng S-K, Tai C (2022) Cross-domain knowledge transfer for skeleton-based action recognition based on graph convolutional gradient reversal layer. In: 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 387–390. IEEE
28. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
29. Khan W, Haroon M (2022) An efficient framework for anomaly detection in attributed social networks. Int J Inform Technol. https://doi.org/10.1007/s41870-022-01044-2
30. Tu E, Wang Z, Yang J, Kasabov N (2022) Deep semi-supervised learning via dynamic anchor graph embedding in latent space. Neural Netw 146:350–360
31. Galke L, Vagliano I, Franke B, Zielke T, Hoffmann M, Scherp A (2023) Lifelong learning on evolving graphs under the constraints of imbalanced classes and new classes. Neural Netw. https://doi.org/10.1016/j.neunet.2023.04.022
32. Lombardo G, Poggi A, Tomaiuolo M (2022) Continual representation learning for node classification in power-law graphs. Futur Gener Comput Syst 128:420–428
33. Yao K, Liang J, Liang J, Li M, Cao F (2022) Multi-view graph convolutional networks with attention mechanism. Artif Intell 307:103708

34. Jin T, Dai H, Cao L, Zhang B, Huang F, Gao Y, Ji R (2022) Deepwalk-aware graph convolutional networks. Sci China Inform Sci 65(5):152104
35. Huang J, Du L, Chen X, Fu Q, Han S, Zhang D (2023) Robust mid-pass filtering graph convolutional networks. In: Proceedings of the ACM Web Conference 2023, pp. 328–338
36. Sharma K, Verma S, Medya S, Bhattacharya A, Ranu S (2023) Task and model agnostic adversarial attack on graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 15091–15099
37. Zhang G, Hu Z, Wen G, Ma J, Zhu X (2023) Dynamic graph convolutional networks by semi-supervised contrastive learning. Pattern Recognit 139:109486
38. Mo X, Wan B, Tang R, Ding J, Liu G (2023) Attention-based network embedding with higher-order weights and node attributes. CAAI Transactions on Intelligence Technology
39. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, et al. (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467
40. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. Stat 1050:20
41. Wu B, Liang X, Zheng X, Guo Y, Tang H (2022) Improving dynamic graph convolutional network with fine-grained attention mechanism. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3938–3942. IEEE
42. Mallick T, Balaprakash P, Rask E, Macfarlane J (2021) Transfer learning with graph neural networks for short-term highway traffic forecasting. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 10367–10374. IEEE
43. Zhang C-Y, Yao Z-L, Yao H-Y, Huang F, Chen CP (2022) Dynamic representation learning via recurrent graph neural networks. IEEE Transactions on Systems, Man, and Cybernetics: Systems
44. Pan J, Li H, Teng J, Zhao Q, Li M (2022) Dynamic network representation learning method based on improved gru network. Comput Inform 41(6):1491–1509

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.