Neural Processing Letters (2024) 56:231
https://doi.org/10.1007/s11063-024-11645-6

Distance Enhanced Hypergraph Learning for Dynamic Node Classification

Dengfeng Liu[1] · Zhiqiang Pan[1] · Shengze Hu[2] · Fei Cai[1]

Accepted: 8 May 2024

Abstract
Dynamic node classification aims to predict the labels of nodes in the dynamic networks. Existing methods primarily utilize the graph neural networks to acquire the node features and original graph structure features. However, these approaches ignore the high-order relationships between nodes and may lead to the over-smoothing issue. To address these issues, we propose a distance enhanced hypergraph learning (DEHL) method for dynamic node classification. Specifically, we first propose a time-adaptive pre-training component to generate the time-aware representations of each node. Then we utilize a dual-channel convolution module to construct the local and global hypergraphs which contain the corresponding local and global high-order relationships. Moreover, we adopt the K-nearest neighbor algorithm to construct the global hypergraph in the embedding space. After that, we adopt the node convolution and hyperedge convolution to aggregate the features of neighbors on the hypergraphs to the target node. Finally, we combine the temporal representations and the distance enhanced representations of the target node to predict its label. In addition, we conduct extensive experiments on two public dynamic graph datasets, i.e., Wikipedia and Reddit. The experimental results show that DEHL outperforms the state-of-the-art baselines in terms of AUC.

Keywords Dynamic node classification · Hypergraph learning · Graph neural networks

✉ Shengze Hu
springsun@nudt.edu.cn

✉ Fei Cai
caifei08@nudt.edu.cn

Dengfeng Liu
liudengfeng21@nudt.edu.cn

Zhiqiang Pan
panzhiqiang@nudt.edu.cn

[1] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 417000, China

[2] Laboratory for Big Data and Decision, National University of Defense Technology, Changsha 417000, China

Published online: 09 September 2024

1 Introduction

Graph network is an effective way for representing the elements and their corresponding interactions [1, 2]. Besides the static networks, the time-varying dynamic graphs exist in many realistic fields, where the nodes and edges are continuously added or deleted [3, 4]. For instance, as shown in Fig. 1, taking T_{15} and T_{16} into comparison, new nodes like v_{16} and edges like $v_{3,10}$ may appear, and the existing nodes such as v_{15} and edges like $v_{0,13}$ may disappear. Dynamic node classification aims to assign a label to each node in the evolving temporal networks [5], where the node label may constantly change due to the graph evolution. For example in Fig. 1, with the network evolving from T_{15} to T_{16}, the label of node v_0 changes.

Existing graph neural networks (GNNs) based methods for dynamic node classification mainly focus on sampling the recent neighbors [6, 7] or randomly sampling the neighbors [8, 9] from the existing graph structures. Then, the node/edge attribute features and time features are combined for information aggregation, finally generating the representation of the target node, which is then applied to the downstream node classification task [7, 10]. However, the current methods mainly focus on mining the local node features and graph structure information, which is difficult to capture the high-order relationships between nodes [11, 12]. Moreover, the deep neural networks may lead to the over-smoothing problem [13–15], making the learnt representations of the adjacent nodes highly similar, which cannot be accurately distinguished for making predictions.

To tackle the above issues, we propose a distance enhanced hypergraph learning (DEHL) approach for dynamic node classification. Specifically, we first design a time-adaptive pre-training module, in which we employ a temporal graph attention layer to pre-train the network for acquiring the dynamic temporal representations of each node. Subsequently, we propose a dual-channel convolution module: (1) On the one hand, the neighboring nodes connected to the target node are first sampled in the given graph structure for constructing the local hypergraph structure. (2) On the other hand, we employ the K-nearest neighbor (KNN) algorithm to measure the distance between the target node and its neighbors in the pre-training embedding space for constructing the global hypergraph structure. Then, the features of each node in the local and global hypergraphs are aggregated through the node convolution to obtain the local and global hyperedge representations, respectively, where the global ones can help avoid the over-smoothing problem by decreasing the similarity between nodes in the local graph structure. After that, We adopt the hyperedge convolution and the attention mechanism to aggregate the above local and global hyperedge representations to the target node for obtaining the distance enhanced representations. Finally, the time-adaptive representations and the distance enhanced representations are concatenated and inputted into a multi-layer perceptron to output the final predictive score, i.e., the probability of the target node label. To evaluate the proposed DEHL approach, we carry out comprehensive experiments on two public temporal network datasets, i.e., Wikipedia and Reddit. The experimental results demonstrate that our proposal outperforms the state-of-the-art baselines in terms of AUC. Generally, our contributions in this paper can be summarized as follows:

1. We propose a distance enhanced hypergraph learning (DEHL) method for dynamic node classification on temporal networks, which considers both the local and global high-order relationships between the pre-trained dynamic node features.
2. We exploit the high-order node relationships in the global hypergraph structures constructed by the KNN algorithm, which mitigates the over-smoothing issue by introducing the neighboring nodes out of the local graph structure.

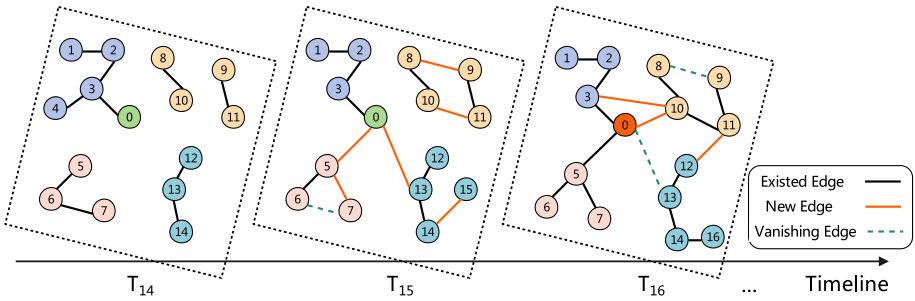


Fig. 1 A temporal network snapshot example, where the number in each circle represents the ID of the node, and the color of the circle represents the category of the node

3. We conduct experiments on two public temporal network datasets and observe the performance improvements of DEHL against the state-of-the-art baselines in terms of AUC.

2 Related Work

In this section, we briefly review the previous works from two angles, i.e., the graph-based deep learning and the hypergraph learning.

2.1 Graph-Based Deep Learning

Graph neural network is an emerging key technology to deal with the graph structure data. For example, the graph convolutional network (GCN) [16] implements the first graph convolutional neural network by mapping the time-domain feature matrix of a graph to the frequency domain and introducing the graph convolution. Moreover, GraphSAGE [17] is a spatially based graph convolution approach that samples the neighboring nodes in a multi-hop manner for aggregation to generate the representations of the target nodes. Furthermore, Velickovic et al. [18] introduce an attention mechanism to determine the importance of each neighbor to the target node by assigning different attention weights. In addition, the gated graph neural network (GGNN) incorporates the gated recurrent units (GRUs) to accumulate and propagate information across multiple neighborhood hops [19]. Furthermore, the dimensionwise separable 2D graph convolutional (DSGC) approach concurrently captures both the inter-node relationships and the relationships between the node attribute dimensions [20].

In addition to the above graph-based deep learning approaches designed for the static graphs, the methods considering both the graph structure and the time dynamics on temporal networks are also investigated. For example, Zhu et al. [21] propose to extend the static graph models on the dynamic graphs by adding the regularization, and learn the temporal latent space via the non-negative matrix factorization. Moreover, Dyngraph2vec [22] adopt the recurrent layers to capture the temporal information and utilize the nonlinear layers to consider the graph structure information, respectively. Furthermore, EvolveGCN [23] combines the recurrent neural network and GCN to evolve the GCN parameters so as to extract the temporal information effectively.

In addition to the aforementioned methods obtaining the node representations at each timestamp, focusing on the node representation connections between different timestamps

provides another idea of the temporal information learning [24–26]. For instance, Trivedi et al. [27] design a representation learning framework for dynamic graphs (DyREP) which contains a dual time scale deep temporal point process model to capture the interleaved dynamics underlying the observed processes. In addition, Xu et al. [28] design a temporal encoding function based on the Bochner's theorem [29] and capture the graph structure information by the temporal graph attention layer.

Although the graph-based deep learning techniques have achieved impressive ability in acquiring the node representations, they usually merely focus on the existing graph structure, which limits the ability of accurately classifying nodes. This is due to that the GNNs fail to consider both the higher-order local and global relationships, leading to incorrect classifications. Furthermore, the GNN based approaches usually suffer from the over-smoothing problem, where the high similarities between the node representations may result in inaccurate node classifications.

2.2 Hypergraph Learning

Traditional graph-based deep learning methods are often applied to model the pair-wise relationships in graph-structured data [30]. In addition, to uncover the high-order node relationships, hypergraphs are utilized to extract the comprehensive feature and achieve enhanced learning of graph representations, which have been extensively applied in various research domains such as recommender systems [31, 32], computer vision tasks [33–35], etc.

Hypergraph learning is first introduced by Zhou et al. [36] to extend the spectral clustering methods to the hypergraphs. Then, Huang et al. [37] propose a hypergraph construction approach based on KNN and the search radius method. Moreover, Zhang et al. [38] propose to use the data correlation in the label space and the feature space to optimize the hypergraph structure. Furthermore, the hypergraph networks with hyperedge neurons (HNHN) [39] utilizes the association matrix for the convolution operation to update the hypernode and hyperedge representations. After that, Zhang et al. [40] learn the label propagation by a dual optimization process based on the hypergraph to optimize the hypergraph structure. The hyperedge-based embedding (HEBE) approach [41] learns the representation for each entity involved in an event through the hypergraphs. And Feng et al. [42] propose the hypergraph neural network (HGNN), utilizing the hypergraph Laplacian to represent the hypergraphs. In addition, the dynamic hypergraph neural networks (DHGNN) [30] combines KNN and the K-means clustering algorithm to dynamically construct the hypergraph structure, which accurately obtains the inherent higher-order correlation information hidden in the intrinsic graph structure.

While the aforementioned hypergraph learning techniques have achieved notable progress in the node representation learning, they are mainly applicable to the static networks and cannot be directly applied to the temporal graphs.

3 Approach

In this section, we first formalize the task of dynamic node classification. A dynamic network can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the node set and the edge set, respectively, which are both time-varying. Each edge $(v_i, v_j, t_m) \in \mathcal{E}$ indicates that node v_i and node v_j interact at the timestamp t_m. Furthermore, the label matrix is represented by $Y \in \mathbb{R}^{N \times U}$, where N denotes the number of nodes in the node set $\mathcal{V}$, U corresponds to the

Table 1 Major notations applied in this paper

Variable	Description
$\mathcal{G}$	The dynamic network
$\mathcal{V}$	The node set of the dynamic network
$\mathcal{E}$	The edge set of the dynamic network
v_i	The i-th node of the node set $\mathcal{V}$
v_0	The target node of the node set $\mathcal{V}$
t_m	The m-th timestamp
Y	The label matrix
N	The number of nodes in the node set $\mathcal{V}$
U	The number of label classes
y_i	The label of node v_i
$\mathcal{N}_{v_0}(t_m)$	The temporal neighboring nodes of the target node at t_m
$\mathbf{M(t)}$	The representation of the target node as well as its neighbors
$\|\|$	The concatenation operation
$\Phi(t)$	The time encoding functional mapping
$\alpha^l_{v_0,i}$	The attention weight of the neighboring node v_i
$\mathbf{x}_{v_0}$	The target node features
$\mathbf{h}^l_{v_i}$	The features of the neighbor v_i in the l-th layer
$\mathbf{h}^l_{v_0}(t_m)$	The neighbor representation of target node at the l-th GNN layer at t_m
$\tilde{\mathbf{h}}^L_{v_0}(t_m)$	The final time-aware representation of the target node at t_m
$\mathbf{f}^l_{v_0}$	The aggregated local hyperedge features
$\mathbf{f}^g_{v_0}$	The aggregated global hyperedge features
$\mathbf{z}_{v_0}$	The final concatenated representation of the target node

number of classes, and each row $y_i \in \mathbb{R}^U$, denotes the one-hot label vector for node v_i. The aim of dynamic node classification task is to accurately predict the labels of nodes. Then, we will give a brief definition of hypergraph. In a simple graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, edges in $\mathcal{E}_s$ are 2-element subsets of the node set $\mathcal{V}_s$, thereby expressing pair-wise relationships between two nodes. By contrast, hyperedges in a hypergraph $\mathcal{G}_h = (\mathcal{V}_h, \mathcal{E}_h)$ are non-empty arbitrary subsets of $\mathcal{V}_h$ [43]. Therefore, hyperedges are able to naturally represent complex multi-way relationships occurring between any number of nodes [44, 45]. It should be noted that simple graph can be seen as a special case of hypergraph where all hyperedges have a degree of 2 [30]. For clarity, we summarize the primary notations used in this paper in Table 1.

Subsequently, we introduce the proposed Enhanced Hypergraph Learning (DEHL) method in detail, which mainly consists of three components, i.e., the time-adaptive pre-training (see Sect. 3.1), the dual-channel convolution (see Sect. 3.2) and the prediction (see Sect. 3.3). Figure 2 presents an overview of DEHL. Specifically, given the temporal network $\mathcal{G}$, we first design a time-adaptive pre-training module to learn the dynamic features of each node through the temporal graph attention layer, thus constructing the pre-training embedding space. Then, we propose a dual-channel convolution method to acquire both the local and global node relationships for enhancing the node representation learning. Finally, the time-aware node embeddings and the high-order relationships enhanced node embeddings are stacked and inputted into the classifier for predicting the label of the target node.

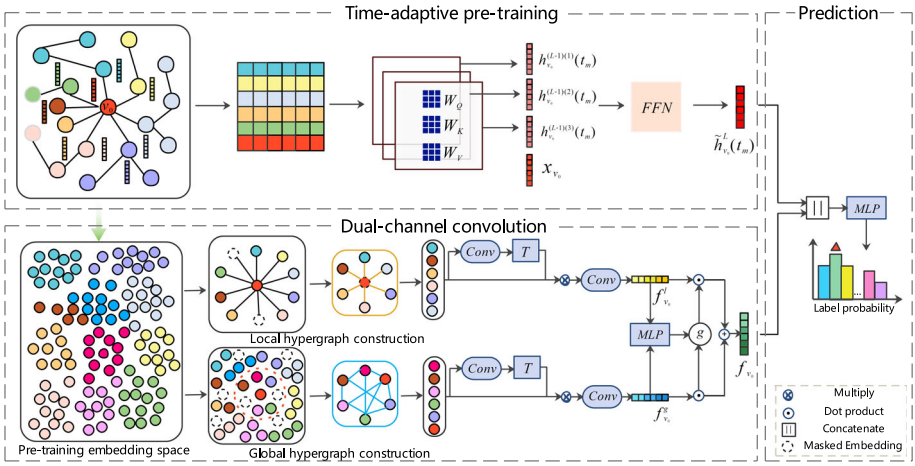


Fig. 2 The framework of our proposed Distance Enhanced Hypergraph Learning (DEHL)

3.1 Time-Adaptive Pre-training

In this subsection, we design a time-adaptive pre-training component to pre-train the nodes by learning their embeddings in the constructed pre-training embedding space, where the node/edge attributes and time features are integrated for representing each node. In detail, supposing the target node is denoted as v_0, the representation of v_0 is generated by taking its temporal neighboring nodes $\mathcal{N}_{v_0}(t_m) = \{v_1, ..., v_s\}$ of the target node v_0 at the timestamp t_m into account. First, the representation of the target node as well as its neighbors are obtaining by concatenating the node features as well as the time features:

$$\mathbf{M(t)} = [\mathbf{h}_{v_0}^{l-1}(t_m) \| \Phi(0), \mathbf{h}_{v_1}^{l-1}(t_1) \| \Phi(t - t_1), \ldots, \mathbf{h}_{v_s}^{l-1}(t_s) \Phi(t - t_s)]^\top, \tag{1}$$

where $||$ is the concatenation operation, $\mathbf{h}_{v_i}^{l-1} \in \mathbb{R}^d$ denotes the features of the neighbor v_i in the $l-1$ layer, and $\Phi(t - t_i) \in \mathbb{R}^{d_T}$ is the corresponding time features. Besides considering the node and time features, it is also convenient to extend our proposal for considering the edge features by concatenating them in Eq. (1).

As for the time encoding, the continuous functional mapping $\Phi(t)$ is defined by adopting the concepts from the Bochner's Theorem [29], which can be formulated as follows:

$$\Phi(t) = \sqrt{\frac{1}{d}}[\cos(\omega_1 t), \sin(\omega_1 t), \cos(\omega_2 t), \sin(\omega_2 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)], \tag{2}$$

where $\{\omega_1, \ldots, \omega_d\}$ are the learnable parameters.

Then, the representations of the target node and the neighbors obtained in Eq. (1) are transformed through different linear projections:

$$\begin{aligned} \mathbf{q}(t) &= \mathbf{W}_Q[M(t)]_0, \\ \mathbf{K}(t) &= \mathbf{W}_K[M(t)]_{1:s}, \\ \mathbf{V}(t) &= \mathbf{W}_V[M(t)]_{1:s}, \end{aligned} \tag{3}$$

where the "query" $\mathbf{q}(t) \in \mathbb{R}^{d_h}$ is transformed from the target node, the "key" and "value" $\mathbf{K}(t), \mathbf{V}(t) \in \mathbb{R}^{d_h \times s}$ are transformed from the neighbors, and $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_h \times (d+d_T)}$ are the weight matrices which are trainable.

After that, we perform the self-attention mechanism to determine the importance of different neighboring nodes, which are generated as follows:

$$\alpha^{l}_{v_0,i} = \text{Softmax}(\mathbf{q}^{\top}\mathbf{K}_i) = \frac{\exp(\mathbf{q}^{\top}\mathbf{K}_i)}{\sum_{j\in\mathcal{N}_{v_0}(t_m)}\exp(\mathbf{q}^{\top}\mathbf{K}_j)}, \tag{4}$$

where the attention weight $\alpha^{l}_{v_0,i}$ indicates the importance of the neighboring node v_i in the temporal neighboring structure $\mathcal{N}_{v_0}(t_m)$.

Next, we produce the neighboring representation of the target node v_0 the at timestamp t_m by summing the neighboring embeddings using the generated attention scores,

$$\mathbf{h}^{l}_{v_0}(t_m) = \text{Softmax}\left(\frac{\mathbf{q}(t)\mathbf{K}(t)^{\top}}{\sqrt{d}}\right)\mathbf{V}(t), \tag{5}$$

where $\mathbf{h}^{l}_{v_0}(t_m) \in \mathbb{R}^{d_h}$ is the neighbor representation at the l-th GNN layer.

Finally, in order to jointly consider the target node features $\mathbf{x}_{v_0} \in \mathbb{R}^{d}$ and the neighboring features, we concatenate these two characteristics as follows:

$$\tilde{\mathbf{h}}^{L}_{v_0}(t_m) = \text{MLP}(\mathbf{h}^{l}_{v_0}(t_m)\|\mathbf{x}_{v_0}) = \mathbf{W}_1(\text{ReLU}(\mathbf{W}_0[\mathbf{h}^{l}_{v_0}(t_m)\|\mathbf{x}_{v_0}] + \mathbf{b}_0)) + \mathbf{b}_1, \tag{6}$$

where $\mathbf{W}_0 \in \mathbb{R}^{d_h\times(d_h+d)}$,$\mathbf{W}_1 \in \mathbb{R}^{d\times d_h}$,$\mathbf{b}_0 \in \mathbb{R}^{d_h}$,$\mathbf{b}_1 \in \mathbb{R}^{d}$ are the trainable parameters. Finally, the output of the last GNN layer, i.e., $\tilde{\mathbf{h}}^{L}_{v_0}(t_m) \in \mathbb{R}^{d}$ where L is the GNN layer number, is adopted as the final representation of the target node v_0 at the timestamp t_m.

In addition, multi-head attention can be applied to improve the model performance and robustness, where we can concatenate the output of the g heads to generate the final node representation as follows:

$$\tilde{\mathbf{h}}^{L}_{v_0}(t_m) = \text{FFN}(\mathbf{h}^{l(1)}_{v_0}(t_m)\| \ldots \|\mathbf{h}^{l(g)}_{v_0}(t_m)\|\mathbf{x}_{v_0}), \tag{7}$$

where $\mathbf{h}^{l(p)}_{v_0}(t_m)$ is the output vector of target node v_0 in the p-th attention, and p ranges from 1 to g.

3.2 Dual-Channel Convolution

Current graph neural networks mianly focus on mining the local relationships between nodes, ignoring the global high-order node relationships. To deal with this issue, we propose a dual-channel convolution method, which not only constructs the local hypergraph from the original graph structure, but also builds a global hypergraph by the K-nearest neighbor algorithm in the pre-train embedding space by considering the global features. Furthermore, incorporating the hypergraph representation does not simply capture the connected node features but can incorporate the global embedding space features in a distance-enhanced way, thereby alleviating the oversmoothing issue to some degree.

3.2.1 Local Relationships Acquisition

Local hypergraph construction To take the local node relationships into consideration, we aggregate the information from the local neighboring nodes in the original graph structure. Specifically, given a target node v_0, the full neighbors of v_0 in the original graph structure at the timestamp t_m can be denoted as $\mathcal{N}_{v_0}(t_m)$. Then, we first sample a subset of $\mathcal{N}_{v_0}(t_m)$ as $\mathcal{N}^{l}_{v_0}(t_m) = \{v_{u_1}, \ldots, v_{u_o}\}$, where u_o is the number of the sampled neighbors. In addition,

it's worth noting that the sampling procedure should obey the chronological order, i.e., the sampled neighbors must interact with v_0 before the timestamp t_m. After sampling the subset $\mathcal{N}^l_{v_0}(t_m)$, the contained neighboring nodes can be considered as a local hyperedge for acquiring the local node relationships. **Local node convolution.** After obtaining the local hyperedge where the contained neighbor set can denoted as $\mathcal{N}^l_{v_0}(t_m)$, we propagate information from $\mathcal{N}^l_{v_0}(t_m)$ to v_0 through the local node convolution operation. However, the existed methods such as maximum pooling, average pooling, and pre-computed transform matrix approaches generated from the extant graph structure are incompetent in effectively distinguishing the feature information between the neighboring nodes [30]. Consequently, we adopt an MLP to learn the neighboring node features for acquiring the transform matrix and then utilize 1-dimensional convolution to compact the extracted feature information for generating the hyperedge representation. This can be formulated as follows:

$$\begin{aligned} \mathbf{T}^l &= \text{MLP}(\mathbf{E}^l_{v_0}), \\ \mathbf{f}^l_{v_0} &= \text{conv}(\mathbf{T}^l \cdot \text{MLP}(\mathbf{E}^l_{v_0})), \end{aligned} \tag{8}$$

where $\mathbf{E}^l_{v_0}$ is the embedding of the sampled neighboring nodes $\mathcal{N}^l_{v_0}(t_m)$ generated by the time-adaptive pre-training component, conv denotes the one-dimensional convolution, and $\mathbf{f}^l_{v_0} \in \mathbb{R}^d$ denotes the aggregated local hyperedge features.

3.2.2 Global Relationships Acquisition

Global hypergraph construction In order to capture the high-order global node relationships, we adopt a K-nearest neighbor algorithm to dynamically construct the global hypergraph. Specifically, we first select the $K-1$ nearest nodes to the target node v_0 in the pre-traning embedding space $\mathbf{E}$. Similar to sampling in the original graph structure, the neighboring nodes arising subsequent to the target node v_0 required to be masked, conforming to the temporal constraints. These sampled $K-1$ nearest neighboring nodes, in conjunction with the target node v_0, constitute the global hyperedge that can be denoted as $\mathcal{N}^g_{v_0}(t_m) = \{v_0, v_{u_1}, v_{u_2}, \ldots, v_{u_{K-1}}\}$. **Global node convolution.** After that, we conduct the global node convolution in a similar way of aggregating the features of each sampled neighbor in the original graph structure to the local hyperedge, which can be formulated as follows:

$$\begin{aligned} \mathbf{T}^g &= \text{MLP}(\mathbf{E}^g_{v_0}), \\ \mathbf{f}^g_{v_0} &= \text{conv}(\mathbf{T}^g \cdot \text{MLP}(\mathbf{E}^g_{v_0})) \end{aligned} \tag{9}$$

where $\mathbf{E}^g_{v_0}$ is the embedding of the neighbors in the global hyperedge, i.e., $\mathcal{N}^l_{v_0}(t_m)$, and $\mathbf{f}^g_{v_0} \in \mathbb{R}^d$ denotes the aggregated global hyperedge features.

By processing each layer of feature embedding in the above manner, the hypergraph structure can be dynamically adjusted with the evolution of the feature embedding space and the network development to a larger size, thus obtaining an enhanced hypergraph structure for modeling the high-order node relationships in temporal graphs.

3.2.3 Hyperedge Convolution

After generating the local and global hyperedge representations as $\mathbf{f}^l_{v_0}$ and $\mathbf{f}^g_{v_0}$ through the above local and global relationships acquisition modules, we adopt an attention mechanism

to combine them together as the final target node representation, which can be formulated as follows:

$$\mathbf{g} = \text{Softmax}(\mathbf{w}_e[\mathbf{f}^l_{v_0}, \mathbf{f}^g_{v_0}] + \mathbf{b}_e), \tag{10}$$

where $\mathbf{g} \in \mathbb{R}^2$ is the weight score between the local and global hyperedges generated by the attention mechanism, $\mathbf{w}_e \in \mathbb{R}^d$ and $\mathbf{b}_e \in \mathbb{R}$ are the learnable parameters.

Finally, the final target node representation can be generated by summing the local and global hyperedge representations in a weighted way:

$$\mathbf{f}_{v_0} = \mathbf{g}[\mathbf{f}^l_{v_0}, \mathbf{f}^g_{v_0}], \tag{11}$$

where $\mathbf{f}_{v_0} \in \mathbb{R}^d$ is the final representation of the target node yielded through the dual-channel convolution component.

3.3 Prediction and Algorithm

Given the target node v_0, the time-aware embedding can be obtained as $\tilde{\mathbf{h}}^L_{v_0}$ by the time-adaptive pre-training component. In addition, the local and global node relationship enhanced embeddings can be generated as $\mathbf{f}_{v_0}$ based on the dual-channel convolution method. After that, we concatenate $\tilde{\mathbf{h}}^L_{v_0}$ and $\mathbf{f}_{v_0}$, and then input them into the multi-layer perceptron (MLP) to generate the final scores for predicting the target node label, which can be formulated as follows:

$$\mathbf{z}_{v_0} = \text{MLP}([\tilde{\mathbf{h}}^L_{v_0} || \mathbf{f}_{v_0}]) = \mathbf{W}_3(\text{ReLU}(\mathbf{W}_2[\tilde{\mathbf{h}}^L_{v_0} || \mathbf{f}_{v_0}])), \tag{12}$$

where $\mathbf{W}_2 \in \mathbb{R}^{d_r \times (2d)}$ and $\mathbf{W}_3 \in \mathbb{R}^{d_m \times d_r}$ are the trainable parameters, and $\mathbf{z}_{v_0} \in \mathbb{R}^{d_m}$ is the final concatenated representation of the target node v_0. To train the proposed DEHL model, we adopt the binary cross-entropy function as the optimization objective to learn the trainable parameters:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\sigma(\mathbf{z}_{v_i})) + (1 - y_i)\log(1 - \sigma(\mathbf{z}_{v_i}))], \tag{13}$$

where N is the total training sample size, σ indicates the Sigmoid activation function, and y_i denotes the label of node v_i. Finally, we apply the back propagation in time (BPTT) algorithm to train the proposed DEHL model.

The primary procedures for DEHL training are presented in Algorithm 1. For the whole nodes in the dynamic network, we treat the original features of the nodes as the representations of the 0-th layer (line 1). Then, we generate the neighbor representation (line 4) and the target node representation (line 5) through the time-adaptive pre-training component. Next, we construct the local hypergraph of the target node, and adopt local node convolution to generate the aggregated local hyperedge features (line 6). We adopt a K-Nearest Neighbor algorithm to construct the global hypergraph (line 7–11). Similar to the local hyperedge features, we use global node convolution to generate the aggregated global hyperedge features (line 12). We obtain the distance-aware representation through attention mechanism (line 13). Then, we calculate the prediction score (line 14) and the binary cross-entropy loss (line 15). Finally, we employ the BPTT to optimize the DEHL parameters (line 17). Denoting the average neighboring size of nodes as $\bar{N}$, the per-batch time complexity of DEHL can be expressed as $O((\bar{N})^l)$.

Algorithm 1 Training process of DEHL.

Require: dynamic network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; epochs: the count of training iterations
Ensure: **S**: the node embeddings in $\mathcal{V}$; Ω: the trainable parameters of DEHL
$\mathbf{h}_{v_0}^0 = \mathbf{x}_{v_0}, \forall v_0 \in \mathcal{V}$;
for epoch in range(epochs) **do**
  for v_0 in $\mathcal{V}$ **do**
    $\mathbf{h}_{v_0}^l(t_m)$ = the neighbor representation based on Eqs. (1) - (5);
    $\tilde{\mathbf{h}}_{v_0}^L(t_m)$ = the time-aware representation based on Eqs. (6) and (7);
    $\mathbf{f}_{v_0}^l$ = the aggregated local hyperedge features based on Eq. (8);
    for v_j in $\mathcal{N}_{v_0}(t_m)$ **do**
      $d = \text{distance}\left(\tilde{\boldsymbol{h}}_{v_0}^L(t_m), \tilde{\boldsymbol{h}}_{v_j}^L(t_m)\right)$;
      $d = \text{sort}(d)$;
      $\text{KNN} = \text{topK}(d, K)$;
    end for
    $\mathbf{f}_{v_0}^g$ = the aggregated global hyperedge features based on Eq. (9);
    $\mathbf{f}_{v_0}$ = the distance-aware representation based on Eqs. (10) and (11);
    $\mathbf{z}_{v_0}$ = prediction based on Eq. (12);
    Model optimization loss: L based on Eq. (13);
  end for
  utilize BPTT for the optimization of Ω;
end for
return S and Ω

4 Experiments

4.1 Research Questions

We conduct extensive experiments by addressing the following four research questions:

1. Can our DEHL model beat the state-of-the-art baselines for the dynamic node classification task on real-world temporal network datasets?
2. How is the contribution of the time-adaptive pre-training component and the dual-channel convolution module to the performance of DEHL?
3. How is the performance of DEHL with different sampling size of neighbors selected for constructing the global hypergraph?
4. How is the impact of the critical hyper-parameters, i.e., the node embedding dimension and the time embedding dimension, on the performance of DEHL?

4.2 Datasets

To verify the effectiveness of our proposed DEHL model, we conduct comprehensive experiments on two real-world temporal network datasets, i.e., Wikipedia[1] and Reddit.[2] Wikipedia contains the interactions between users and pages for one month, which treats users and pages as nodes, and the act of a user editing the page represents an interaction. The dynamic binary labels indicate whether a Wikipedia user is banned from editing a page. Similar to the Wikipedia dataset, Reddit gathers users and subreddits as nodes, and treats users' posting to the subreddits which last for a month as the interactions. The dynamic binary labels represent

[1] http://snap.stanford.edu/jodie/wikipedia.csv.

[2] http://snap.stanford.edu/jodie/reddit.csv.

Table 2 Statistics of the datasets used in the experiments, where the density is computed as $E/N(N-1)$, with E denoting the number of edges in the dataset and N representing the number of nodes in the dataset

Dataset	Wikipedia			Reddit		
	10 days	20 days	30 days	10 days	20 days	30 days
# Nodes	4289	5210	5693	9025	9790	10,175
# Edges	40,000	40,000	40,000	40,000	40,000	40,000
# Edge features	172	172	172	172	172	172
# Edge features type	LIWC	LIWC	LIWC	LIWC	LIWC	LIWC
# Nodes with dynamic labels	62	57	55	6	15	22
# Density (%)	0.2175	0.1474	0.1234	0.0491	0.0417	0.0386
# Dynamic labels proportion (%)	0.1550	0.1425	0.1375	0.0150	0.0375	0.0550
Label type	Binary	Binary	Binary	Binary	Binary	Binary

if a Reddit user is banned from posting under a subreddit. Moreover, both the interaction features on Wikipedia and Reddit are converted into the text features.

In addition, for both Wikipedia and Reddit, we first select three subsets of the whole dataset with the timespan of 10, 20 and 30 days, respectively. Then, we further randomly sample 40,000 edges from each selected subset according to the corresponding contained positive/negative label ratios. Table 2 reports the detailed statistics of six preprocessed datasets adopted for our experiments. Furthermore, we partition each preprocessed dataset into the training, validation, and test sets with the split ratios of 70%, 15%, and 15%, respectively.

4.3 Baselines

We compare the performance of DEHL with the following competitive baselines:

- *GAT* [18] aggregates the features from the neighboring nodes to the target node using an attention mechanism.
- *GraphSAGE* [17] uses the neighbor sampling to encode the graph structure, where the features of the sampled neighboring nodes are aggregated to obtain the target node representations.
- *GAT+T* incorporates the temporal features of nodes by concatenating the node embeddings obtained from GAT with the time embeddings encoded via Eq. (2).
- *GraphSAGE+T* concatenates the node embeddings generated by GraphSAGE and the time embeddings generated by Eq. (2) for representing the target node.
- *TGAT* [28] utilizes a temporal attention mechanism to encode the temporal dynamics as well as the graph structure for generating the node representations.
- *TGN* [46] maintains for each node an evolving memory through the message function, message aggregator, and memory updater components, and utilizes the leveraged embedding module to produce the representation of each node.
- *NeurTWs* [47] introduces spatiotemporal-biased random walks for representative motif extraction coupled with a neural ordinary differential equation component for irregularly-sampled temporal nodes to be embedded explicitly.

Table 3 Model performance of different dynamic node classification approaches. The optimal result in each column is denoted in boldface

Dataset	Wikipedia			Reddit		
	10 days	20 days	30 days	10 days	20 days	30 days
GAT	80.12	86.27	80.59	78.60	60.27	62.63
GAT+T	81.87	87.13	81.07	79.93	60.96	63.99
GraphSAGE	81.99	85.68	80.65	78.46	60.24	61.66
GraphSAGE+T	82.76	86.58	81.67	81.90	60.69	63.28
TGAT	83.07	87.44	82.04	83.76	61.48	65.57
TGN	86.56	90.05	82.95	84.66	62.76	65.44
NeurTWs	87.29	90.67	83.09	84.32	63.03	66.02
DEHL	**88.01**	**92.26**	**83.50**	**85.20**	**66.51**	**67.47**

4.4 Experimental Setup

The hyper-parameter tuning of DEHL is performed on the validation data to acquire the optimal configurations specific to each dataset. In particular, the node embedding dimension and the time embedding dimension are both tuned in {60, 80, 100, 120, 140}. The number of attention heads is ranged in {1, 2, 3, 4, 5}. At last, we set both the node and time embedding dimensions to 100 and adopt 2 attention heads for the overall experiments. Moreover, the Adam optimizer with a learning rate of $1e^{-4}$ is employed to train the parameters of DEHL using a batch size of 100. Finally, AUC is adopted as the metric to evaluate the dynamic node classification performance of our proposed DEHL model as well as the baseline methods. In addition, we train DEHL for 50 epochs, and the early stopping strategy is adopted as a regularization to prevent overfitting by stopping training if the performance does not improve for 5 epochs.

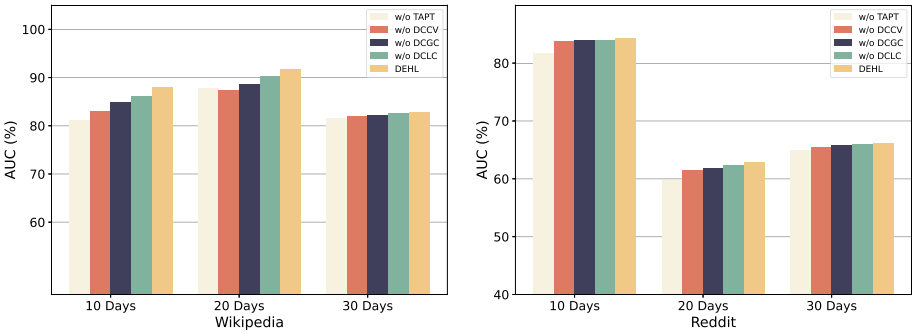
5 Results and Discussion

5.1 Overall Performance

We conduct a comparison between the proposed DEHL and the state-of-the-art dynamic node classification baselines on the Wikipedia and Reddit datasets. The experimental results are presented in Table 3.

Clearly, for the baselines, by comparing GAT and GraphSAGE against their respective temporal versions, i.e., GAT+T and GraphSAGE+T, it is obvious that the temporal version GAT+T consistently outperforms GAT in all cases on both Wikipedia and Reddit. The same phenomenon can be observed in the comparisons between GraphSAGE+T and GraphSAGE. This suggests that taking the time information into consideration on the basis of the node embeddings obviously contributes to the classification performance improvement across various scenarios. Moreover, apart from Reddit-10 days, NeurTWs achieves a superior performance than other baselines in all cases on both datasets.

Next, let us consider our proposed DEHL model. First, as shown in Table 3, DEHL consistently yields a better performance than the competitive baselines in terms of AUC on both datasets, validating the effectiveness of our proposal.



(a) Wikipedia dataset (b) Reddit dataset

Fig. 3 The results of ablation study on Wikipedia and Reddit datasets

Specifically, DEHL improves the performance over the best-performing baseline NeurTWs in terms of AUC by 0.72%, 1.59%, and 0.41% for Wikipedia-10 days, Wikipedia-20 days, and Wikipedia-30 days, respectively. As for the Reddit dataset, the performance increasing rates over NeurTWs are 0.88%, 3.48%, and 1.45% on Reddit-10 days, Reddit-20 days, and Reddit-30 days, respectively. An interesting finding is observed that the performance improvement generally increases and then decreases with the timespan of the Wikipedia and Reddit dataset increasing.

5.2 Ablation Study

In this subsection, we explore the contribution of different components in our proposal by comparing DEHL with its four variants, including: (1) w/o TAPT which removes the time-adaptive pre-training component, (2) w/o DCCV which abandons the dual-channel convolution component, (3) w/o DCGC which discards the dual-channel global convolution learning component, (4) w/o DCLC which eliminates the dual-channel local convolution learning component. The results of DEHL and the variants in terms of AUC on Wikipedia and Reddit are presented in Fig. 3.

From Fig. 3, it can be found that DEHL outperforms w/o TAPT and w/o DCCV in all cases on both datasets, indicating that both the time-adaptive pre-training module and the dual-channel convolution component contribute to the improvement of the dynamic node classification performance. Specifically, comparing w/o TAPT and DEHL, we find that after abandoning the time-adaptive pre-training module, the model performance obviously drops, indicating that incorporating the temporal attribute features of nodes is beneficial for dynamic node classification. In addition, the removal of the dual-channel convolution component leads to an obvious performance drop compared to the full DEHL model. This could be attributed to that the dual-channel convolution module takes both the local and global relationships into account, thus yielding an accurate representation of the target node.

Moreover, removing the time-adaptive pre-training module generally results in a larger degradation of the model performance than abandoning the the dual-channel convolution component on both datasets except the case of Wikipedia-20 days. This could be attributed to the fact that compared with the gain brought by incorporating the temporal attribute features, the high-order features captured by the hypergraph structure play a more important role in accurate dynamic node classification.

Table 4 The impact of the hypergraph sampling size on Wikipedia and Reddit. The optimal and suboptimal performance in each column are underlined and boldfaced, respectively

Dataset	Wikipedia			Reddit		
	10 days	20 days	30 days	10 days	20 days	30 days
$K = 2^3$	84.28	**92.26**	**83.50**	83.26	63.62	65.81
$K = 2^4$	<u>87.96</u>	<u>91.38</u>	<u>83.12</u>	**85.20**	**66.51**	64.64
$K = 2^5$	**88.01**	91.75	82.93	84.26	62.84	<u>66.12</u>
$K = 2^6$	87.52	90.14	82.45	83.90	<u>66.49</u>	65.49
$K = 2^7$	86.20	90.99	82.96	<u>84.43</u>	64.46	63.79
$K = 2^8$	85.49	89.85	82.56	81.12	62.99	**67.47**

Besides, as evidenced in Fig. 3, DEHL superior performance over w/o DCGV and w/o DCLV across all experimental conditions on both datasets. This finding suggests that the inclusion of both global and local convolution learning in the dual-channel convolution component led to measurable gains in dynamic node classification. We can further find that w/o DCGV conferred greater performance gains than w/o DCLV. This could be attributed to that time-adaptive pre-training module inherently captures some local relationships, while the global high-order relationships introduced by global convolution learning may be more beneficial for representing target node embedding.

5.3 Impact of Hypergraph Sampling Size

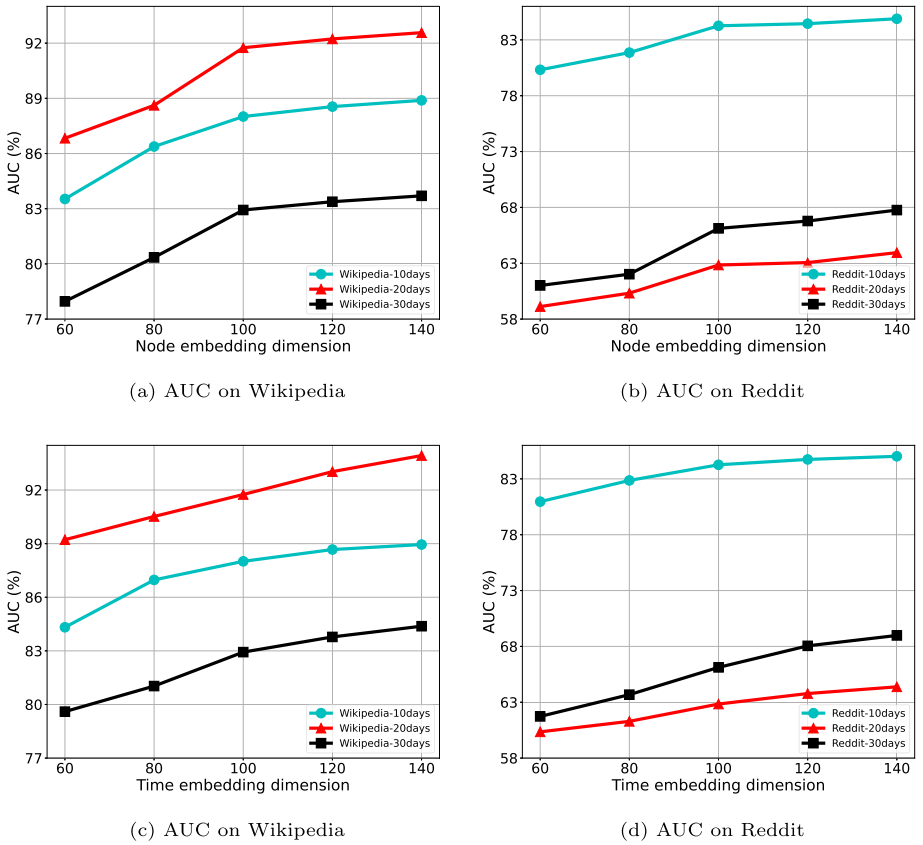
In order to analyze the impact of the sampling size K involved in the global relationships acquisition module, we range K in $\{2^3, 2^4, 2^5, 2^6, 2^7, 2^8\}$ to evaluate the performance of DEHL. The experimental results in terms of AUC are presented in Table 4.

First, on the Wikipedia-10 days dataset, with K increasing, the performance of DEHL first exhibits an increasing trend and then decreases, where the best performance is achieved when K is 2^5. However, for Wikipedia-20 days and Wikipedia-30 days, as K increases, the performance of DEHL consistently declines. This could be due to that on the datasets of a long timespan, a small sampling size K is abundant for exploiting the high-order node relationships, while a large K may introduce much bias which hurts the performance.

Moreover, on the Reddit-10 days and Reddit-20 days, as K increases, the performance of DEHL first increases, reaching the peak performance at $K = 2^4$, and then shows a generally decreasing trend. While on Reddit-30 days, the performance of DEHL fluctuates with K increasing and achieves the best performance at $K = 2^8$. The different phenomenon of the results on the Wikipedia datasets and on the Reddit datasets may be due to that the sparsity of them are different as shown in Table 4, where the sparsity of the Reddit datasets is obviously relatively lower. That is, the global relationships acquisition module requires a large sampling size K for incorporating the high-order node relationships in the scenarios of low data sparsity.

5.4 Analysis of Hyper-Parameter Sensitivity

To test the sensitivity of DEHL to the key hyper-parameters, i.e., the node embedding dimension and the time embedding dimension. We conduct an empirical analysis by tuning both



(a) AUC on Wikipedia

(b) AUC on Reddit

(c) AUC on Wikipedia

(d) AUC on Reddit

Fig. 4 The impact of node embedding dimension and time embedding dimension on Wikipedia and Reddit datasets

the node and time embedding dimensions in {60, 80, 100, 120, 140}. The results in terms of AUC on Wikipedia and Reddit are shown in Fig. 4.

From Fig. 4, we can observe that with both the node and time embedding dimensions increasing from 60 to 100, the performance of DEHL first increases and then stabilizes. This indicates that the ability of the dimension increasing for improving the model performance reaches a threshold around th dimension 100, beyond which the performance improvement is minor. In addition, compared to the results on the Wikipedia datasets, DEHL exhibits a relatively higher improvement for the classification task on the Reddit datasets as the timespan increases from 10 to 30 days. This difference may be attributed to the unbalanced distribution of dynamic label proportions in Reddit as shown in Table 2.

6 Conclusions and Future Work

In this paper, we propose a distance enhanced hypergraph learning (DEHL) method for dynamic node classification. First, we propose a time-adaptive pre-training component to obtain the temporal features of nodes. Then, we design a dual-channel convolution module to

capture the local and global high-order relationships for generating the distance enhanced target node representations, which effectively overcomes the over-smoothing issue. At last, we concatenate the time-adaptive representations and the distance enhanced representations for the final downstream node classification. Extensive experiments conducted on two dynamic graph datasets, i.e., Wikipedia and Reddit, demonstrate the effectiveness of DEHL in terms of AUC.

As to future work, we plan to introduce additional supervision signals such as adversarial learning [48] for exploiting the node relationships, and we are also interested in adapting our proposal to the dynamic node classification in the zero-shot [49] scenarios.

Author Contributions DL designed and drafted this paper. ZP, SH and FC further modified and finalized the paper.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Ma Y, Guo Z, Ren Z, Tang J, Yin D (2020) Streaming graph neural networks. In: Proceedings of the 43rd ACM international SIGIR conference on research and development in information retrieval (SIGIR), pp 719–728. https://doi.org/10.1145/3397271.3401092
2. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24. https://doi.org/10.1109/tnnls.2020.2978386
3. Holme P, Saramäki J (2012) Temporal networks. Phys Rep 519(3):97–125. https://doi.org/10.1016/j.physrep.2012.03.001
4. Yang C, Wang C, Lu Y, Gong X, Shi C, Wang W, Zhang X (2022) Few-shot link prediction in dynamic networks. In: Proceedings of the 15th ACM international conference on web search and data mining (WSDM), pp 1245–1255. https://doi.org/10.1145/3488560.3498417
5. Wen Z, Fang Y, Liu Z (2021) Meta-inductive node classification across graphs. In: Proceedings of the 44th ACM international SIGIR conference on research and development in information retrieval (SIGIR), pp 1219–1228. https://doi.org/10.1145/3404835.3462915
6. Kazemi SM, Goel R, Jain K, Kobyzev I, Sethi A, Forsyth P, Poupart P (2020) Representation learning for dynamic graphs: a survey. J Mach Learn Res 21:2648–2720
7. Khoshraftar S, An A (2022) A survey on graph representation learning methods. CoRR https://doi.org/10.48550/arXiv.2204.01855
8. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: a review of methods and applications. AI Open 1:57–81
9. Barros CD, Mendonça MR, Vieira AB, Ziviani A (2021) A survey on embedding dynamic graphs. ACM Comput Surv 55(1):1–37. https://doi.org/10.1145/3483595
10. Cai H, Zheng VW, Chang KC-C (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637. https://doi.org/10.1109/tkde.2018.2807452
11. Çatalyürek ÜV, Devine KD, Faraj MF, Gottesbüren L, Heuer T, Meyerhenke H, Sanders P, Schlag S, Schulz C, Seemaier D, Wagner D (2023) More recent advances in (hyper) graph partitioning. ACM Comput Surv 55(12):1–38. https://doi.org/10.1145/3571808

12. Gao Y, Feng Y, Ji S, Ji R (2023) Hgnn$^+$: general hypergraph neural networks. IEEE Trans Pattern Anal Mach Intell 45:3181–3199. https://doi.org/10.1109/tpami.2022.3182052
13. Li Q, Han Z, Wu X-M (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the 32th AAAI conference on artificial intelligence (AAAI), vol 32 . https://doi.org/10.1609/aaai.v32i1.11604
14. Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi K-i, Jegelka S (2018) Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th international conference on machine learning (ICML), vol. 80, pp 5453–5462
15. Chen J, Ma T, Xiao C (2018) Fastgcn: fast learning with graph convolutional networks via importance sampling. In: Proceedings of the 6th international conference on learning representations (ICLR). https://doi.org/10.48550/arXiv.1801.10247
16. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations (ICLR)
17. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of the 31th conference on neural information processing systems (NeurIPS), pp 1024–1034 . https://doi.org/10.48550/arXiv.1706.02216
18. Velickovic P, Cucurull G, Casanova A, et al (2018) Graph attention networks. In: Proceedings of the 6th international conference on learning representations (ICLR)
19. Li Y, Tarlow D, Brockschmidt M, Zemel RS (2016) Gated graph sequence neural networks. In: Proceedings of the 4th international conference on learning representations (ICLR)
20. Li Q, Zhang X, Liu H, Dai Q, Wu X-M (2021) Dimensionwise separable 2-d graph convolution for unsupervised and semi-supervised learning on graphs. In: Proceedings of the 27th International conference on knowledge discovery and data mining (KDD), pp 953–963. https://doi.org/10.1145/3447548.3467413
21. Zhu L, Guo D, Yin J, Ver Steeg G, Galstyan A (2016) Scalable temporal latent space inference for link prediction in dynamic social networks. IEEE Trans Knowl Data Eng 28(10):2765–2777. https://doi.org/10.1109/tkde.2016.2591009
22. Goyal P, Chhetri SR, Canedo A (2020) dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowl-Based Syst 187:104816. https://doi.org/10.1016/j.knosys.2019.06.024
23. 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 34th AAAI conference on artificial intelligence (AAAI), vol **34**, pp 5363–5370. https://doi.org/10.1109/tkde.2016.2591009
24. Goyal P, Kamra N, He X, Liu Y (2018) Dyngem: deep embedding method for dynamic graphs. CoRR arxiv:1805.11273
25. Zhou L, Yang Y, Ren X, Wu F, Zhuang Y (2018) Dynamic network embedding by modeling triadic closure process. In: Proceedings of the 32th AAAI conference on artificial intelligence (AAAI), vol 32. https://doi.org/10.1609/aaai.v32i1.11257
26. Zhang, Z., Cui, P., Pei, J., Wang, X., Zhu, W.: Timers: Error-bounded svd restart on dynamic networks. In: Proceedings of the 32th AAAI conference on artificial intelligence (AAAI), vol **32** (2018). https://doi.org/10.1609/aaai.v32i1.11299
27. Trivedi R, Farajtabar M, Biswal P, Zha H (2019) Dyrep: Learning representations over dynamic graphs. In: Proceedings of the 7th international conference on learning representations (ICLR)
28. Xu D, Ruan C, Korpeoglu E, Kumar S, Achan K (2020) Inductive representation learning on temporal graphs. In: Proceedings of the 8th international conference on learning representations (ICLR)
29. Xu D, Ruan C, Korpeoglu E, Kumar S, Achan K (2019) Self-attention with functional time representation learning. In: Proceedings of the 33th conference on neural information processing systems (NeurIPS), pp 15889–15899
30. Jiang J, Wei Y, Feng Y, Cao J, Gao Y (2019) Dynamic hypergraph neural networks. In: Proceedings of the 28th conference on international joint conference on artificial intelligence (IJCAI), pp 2635–2641. https://doi.org/10.24963/ijcai.2019/366
31. Zhu Y, Guan Z, Tan S, Liu H, Cai D, He X (2016) Heterogeneous hypergraph embedding for document recommendation. Neurocomputing 216:150–162. https://doi.org/10.1016/j.neucom.2016.07.030
32. Tan S, Bu J, Chen C, Xu B, Wang C, He X (2011) Using rich social media information for music recommendation via hypergraph model. ACM Trans Multimed Comput Commun Appl 7(1):1–22. https://doi.org/10.1145/2037676.2037679
33. Wang M, Liu X, Wu X (2015) Visual classification by ℓ_1-hypergraph modeling. IEEE Trans Knowl Data Eng 27(9):2564–2574. https://doi.org/10.1109/tkde.2015.2415497
34. Gao Y, Wang M, Tao D, Ji R, Dai Q (2012) 3-d object retrieval and recognition with hypergraph analysis. IEEE Trans Image Process 21(9):4290–4303. https://doi.org/10.1109/tip.2012.2199502

35. Huang Y, Liu Q, Zhang S, Metaxas DN (2010) Image retrieval via probabilistic hypergraph ranking. In: Proceedings of the 23rd conference on computer vision and pattern recognition (CVPR), pp 3376–3383 . https://doi.org/10.1109/CVPR.2010.5540012
36. Zhou D, Huang J, Schölkopf B (2006) Learning with hypergraphs: Clustering, classification, and embedding. In: Proceedings of the 20th conference on neural information processing systems (NeurIPS), pp 1601–1608
37. Huang Y, Liu Q, Metaxas D (2009) Video object segmentation by hypergraph cut. In: Proceedings of the 25th conference on computer vision and pattern recognition (CVPR), pp 1738–1745 https://doi.org/10.1109/CVPR.2009.5206795
38. Zhang Z, Lin H, Gao Y (2018) Dynamic hypergraph structure learning. In: Proceedings of the 27th conference on international joint conference on artificial intelligence (IJCAI), pp 3162–3169. https://doi.org/10.24963/ijcai.2018/439
39. Dong Y, Sawin W, Bengio Y (2020) HNHN: Hypergraph networks with hyperedge neurons. CoRR https://doi.org/10.48550/arXiv.2006.12278
40. Zhang Y, Wang N, Chen Y, Zou C, Wan H, Zhao X, Gao Y (2020) Hypergraph label propagation network. In: Proceedings of the 34th AAAI conference on artificial intelligence (AAAI), vol. **34**, pp 6885–6892. https://doi.org/10.24963/ijcai.2018/439
41. Zhang R, Zou Y, Ma J (2020) Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In: Proceedings of the 4th international conference on learning representations (ICLR)
42. Feng Y, You H, Zhang Z, Ji R, Gao Y (2019) Hypergraph neural networks. In: Proceedings of the 33th AAAI conference on artificial intelligence (AAAI), vol 33, pp 3558–3565 . https://doi.org/10.1609/aaai.v33i01.33013558
43. Yang C, Wang R, Yao S, Abdelzaher TF (2022) Semi-supervised hypergraph node classification on hypergraph line expansion. In: Proceedings of the 31st international conference on information knowledge management (CIKM), pp 2352–2361. https://doi.org/10.1145/3511808.3557447
44. Wang J, Ding K, Hong L, Liu H, Caverlee J (2020) Next-item recommendation with sequential hypergraphs. In: Proceedings of the 43rd international conference on research and development in information retrieval (SIGIR), pp 1101–1110. https://doi.org/10.1145/3397271.3401133
45. Amburg I, Veldt N, Benson AR (2020) Clustering in graphs and hypergraphs with categorical edge labels. In: Proceedings of the 20th international conference on the web conference (WWW), pp 706–717 . https://doi.org/10.1145/3366423.3380152
46. Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, Bronstein M (2020) Temporal graph networks for deep learning on dynamic graphs. In: Proceedings of the 35th international conference on machine learning (ICML)
47. Jin M, Li Y, Pan S (2022) Neural temporal walks: motif-aware representation learning on continuous-time dynamic graphs. In: Proceedings of the 33th conference on neural information processing systems (NeurIPS), pp 15889–15899
48. Chen J, Gong Z, Mo J, Wang W, Wang W, Wang C, Dong X, Liu W, Wu K (2022) Self-training enhanced: network embedding and overlapping community detection with adversarial learning. IEEE Trans Neural Netw Learn Syst 33(11):6737–6748. https://doi.org/10.1109/TNNLS.2021.3083318
49. Chen J, Wang J, Dai Z, Wu H, Wang M, Zhang Q, Wang H (2023) Zero-shot micro-video classification with neural variational inference in graph prototype network. In: Proceedings of the 31st ACM international conference on multimedia (MM). https://doi.org/10.1145/3581783.3611740