HGNN$^+$: General Hypergraph Neural Networks

Yue Gao, *Senior Member, IEEE*, Yifan Feng, Shuyi Ji, and Rongrong Ji, *Senior Member, IEEE*

Abstract—Graph Neural Networks have attracted increasing attention in recent years. However, existing GNN frameworks are deployed based upon simple graphs, which limits their applications in dealing with complex data correlation of multi-modal/multi-type data in practice. A few hypergraph-based methods have recently been proposed to address the problem of multi-modal/multi-type data correlation by directly concatenating the hypergraphs constructed from each single individual modality/type, which is difficult to learn an adaptive weight for each modality/type. In this paper, we extend the original conference version HGNN, and introduce a general high-order multi-modal/multi-type data correlation modeling framework called HGNN$^+$ to learn an optimal representation in a single hypergraph based framework. It is achieved by bridging multi-modal/multi-type data and hyperedge with hyperedge groups. Specifically, in our method, hyperedge groups are first constructed to represent latent high-order correlations in each specific modality/type with explicit or implicit graph structures. An adaptive hyperedge group fusion strategy is then used to effectively fuse the correlations from different modalities/types in a unified hypergraph. After that a new hypergraph convolution scheme performed in spatial domain is used to learn a general data representation for various tasks. We have evaluated this framework on several popular datasets and compared it with recent state-of-the-art methods. The comprehensive evaluations indicate that the proposed HGNN$^+$ framework can consistently outperform existing methods with a significant margin, especially when modeling implicit data correlations. We also release a toolbox called THU-DeepHypergraph for the proposed framework, which can be used for various of applications, such as data classification, retrieval and recommendation.

Index Terms—Hypergraph, classification, hypergraph convolution, representation learning

1 Introduction

Graph Neural Networks (GNN) [1], [2], [3], [4], [5] have attracted ever-increasing attention in recent years, which is effective in dealing with irregular correlation-based data structures such as social networks, functional networks, and recommendation systems. Despite the exciting progress made by the literature, effectively dealing with multi-modal/multi-type data and capturing high-order data correlation are still the critical problems for GNN networks. On one hand, the data correlation in the real world is far beyond pairwise correlation, which cannot be well modeled with a plain graph. For example, users in social networks may have different properties, and the correlation among those users could be in the manner of group, *e.g.*, several users may share the same hobbies or are involved in the same event, as shown in Fig. 1. Under such circumstances, it is hard to formulate such a correlation in a graph structure. A traditional graph structure is capable of formulating pairwise (i.e., first-order relationship) between two subjects, which would not be powerful enough to deal with high-order correlations and thus limit the capability on high-order correlation modeling.

Another limitation of simple graph is its weak capability in modeling multi-modal/multi-type real-world data. For example, the microblog data in social network may contain time, image, emoticon or even videos, with social connections among them, as shown in Fig. 1. Given such correlations from multi-modal/multi-type data representations, traditional GNN-based methods need to integrate multiple graphs in the learning stage, and the exploration of the correlations among multi-modal/multi-type data becomes a challenging task.

In this paper, we propose a general hypergraph neural network framework to handle the challenges on representation learning using high-order correlations. This work was first published in AAAI 2019 [6] and this journal version is an extension of HGNN. In our work, we propose hypergraph-convolution-based representation learning to handle high-order data correlation during representation learning and shows decent performance. This work is an extension of the HGNN, called HGNN+, and the framework is shown in Fig. 3. The proposed HGNN$^+$ can model high-order data correlation and is easy to incorporate with multi-modal/multi-type data. First, to better capture the underlying high-order correlations among data, the hypergraph structure is used for data correlation modeling. Hypergraph, which encodes high-order data correlation with degree-free hyperedges, is a more general structure compared with the simple graph. The illustrative comparison between graph and hypergraph is shown in Fig. 2. Each edge in a graph can only connect two vertices, while in a hypergraph, each hyperedge can connect more than two vertices and thus is more flexible. When handling multi-modal/multi-type data, the hypergraph can

• *Yue Gao, Yifan Feng, and Shuyi Ji are with BNRist, KLISS, School of Software, BLBCI, THUIBCS, Tsinghua University, Beijing 100084, China. E-mail: {kevin.gaoy, evanfeng97}@gmail.com, jisy19@mails.tsinghua.edu.cn.*
• *Rongrong Ji is with the Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Institute of Artificial Intelligence, Fujian Engineering Research Center of Trusted Artificial Intelligence Analysis and Application, Xiamen University, Xiamen, Fujian 361005, China. E-mail: rrji@xmu.edu.cn.*

Manuscript received 1 June 2021; revised 21 April 2022; accepted 24 May 2022. Date of publication 13 June 2022; date of current version 3 February 2023.
This work was supported in part by the National Natural Science Funds of China under Grants 62088102 and 62021002, in part by the Open Research Projects of Zhejiang Lab under Grant 2021KG0AB05, and in part by the National Science Fund for Distinguished Young Scholars under Grant 62025603.
(Corresponding author: Rongrong Ji.)
Recommended for acceptance by B. Kingsbury.
This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2022.3182052, provided by the authors.
Digital Object Identifier no. 10.1109/TPAMI.2022.3182052

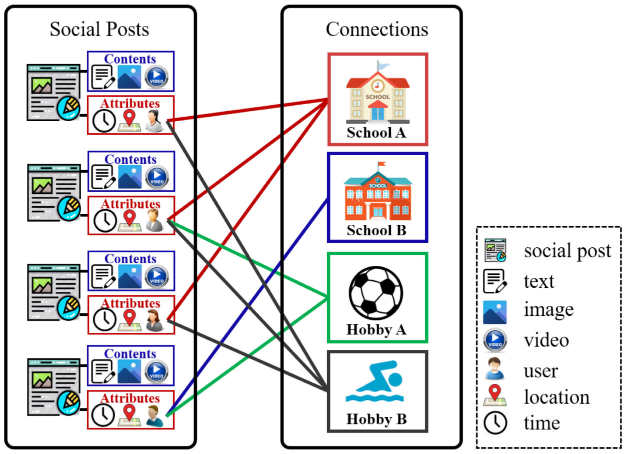


Fig. 1. Examples of high-order correlation and multi-modal data in real world. In social media data, the high-order correlation among microblogs can contain connections from the geo-locations, tags and relationships among users. For each social post, it may contain different types of data, such as images, videos, texts, and emoticons.

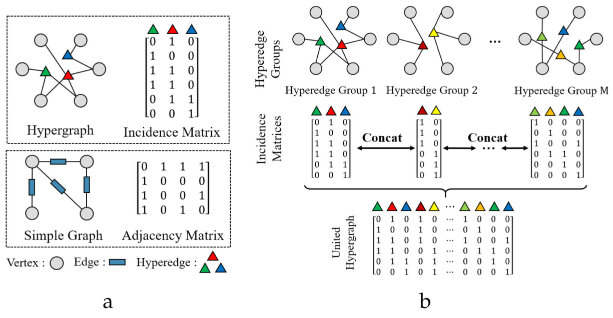


Fig. 2. The comparison between graph and hypergraph. (a) The example and representation of graph and hypergraph. (b) The general strategy of the hypergraph for multi-modal/multi-type data.

generate different types of hyperedges using the multi-modal/multi-type data and then directly concatenates these hyperedges into one hypergraph.

Compared with the conference version, we further generalize the "hyperedge group" conception. Mathematically speaking, a hyperedge group can be defined as a set family upon the vertex set. On the vertex set, there may exist multi-type relationships among vertices. One hyperedge group can be constructed based on one type of relationship, indicating different characteristics of multifarious information. We then systematically divide hypergraph modeling into two steps: hyperedge groups construction and fusion. Given the data with/without graph structure, four types of hyperedge groups are introduced to generate the hypergraph, i.e., using pairwise edge, attribute, k-Hop and neighbors in the feature space, respectively.

After generating multiple hyperedge groups, compared with the conference version, we further introduce two solutions for constructing the overall hypergraph structures. The first strategy, as adopted in the conference version [6], directly concatenates different hyperedge group incidence matrices (namely *Coequal Fusion*). Here, we want to highlight the second fusion strategy: *Adaptive Fusion*. It associates each hyperedge group with a learnable parameter, thus adaptively identifying the importance of different hyperedge groups to the overall hypergraph representation learning and better exploiting the complementary representation of multi-modal/multi-type information.

Drawing inspiration from the information flow path (vertex sequence) [10] in spatial graph convolution, we take a step forward on the basis of HGNN and develop a general two-stage hypergraph convolution in the spatial domain. By specifying four task-dependent message aggregation and updating functions, it can be flexibly applied to different scenarios. We further provide a specific spatial-domain hypergraph convolution operator that employs simple average aggregation in both two stages. An example is also discussed to show that HGNN$^+$ can be naturally extended to the directed hypergraph, which cannot be realized by HGNN as it is defined in the spectral domain.

To evaluate the performance of our HGNN$^+$ framework, comprehensive experiments on five public benchmarks with graph structure for the vertex classification task are conducted. We further conduct experiments on two datasets that contain view-based 3D objects to investigate the effectiveness of our proposed method on data without explicit graph structure. Besides, two natural hypergraph datasets are employed to demonstrate the superiority of the proposed HGNN and HGNN$^+$ in learning high-order data correlations compared with graph-based neural networks. The results have demonstrated the effectiveness of the proposed method compared with the state-of-the-art methods. In this paper, we also release a toolbox called THU-DeepHypergraph for the HGNN$^+$ framework.

In the conference version [6], we introduce the hypergraph neural networks, i.e., HGNN, for representation learning using hypergraph structure. HGNN is able to formulate complex and high-order data correlation through its hypergraph structure and can also be efficient using hyperedge convolution operations. This work was firstly published on AAAI 2019. Compared with the conference version, we have the following extensions in this journal version:

1) We complete the modeling framework and systematically introduce a general hypergraph neural network framework HGNN$^+$, which includes two main procedures, i.e., hyperedge modeling and hypergraph convolution. In hypergraph modeling, we conceptually introduce "hyperedge group" and further define four ways to generate hyperedge groups. An adaptive hyperedge groups fusion strategy is also proposed to optimally generate the hypergraph and better exploit the complementary information of multifarious correlations.
2) The original convolution strategy HGNNConv is extended to a general two-stage hypergraph convolution operation from the spatial domain. A specific variant of it is also defined (namely HGNNConv$^+$), which is much more scalable compared with the hypergraph convolution in HGNN.
3) Extensive experiments on the data with/without graph structure as well as that with hypergraph structure are conducted to demonstrate the effectiveness of the proposed method. Besides, detailed mathematical discussions are provided to offer a deeper understanding of the hypergraph structure and the proposed

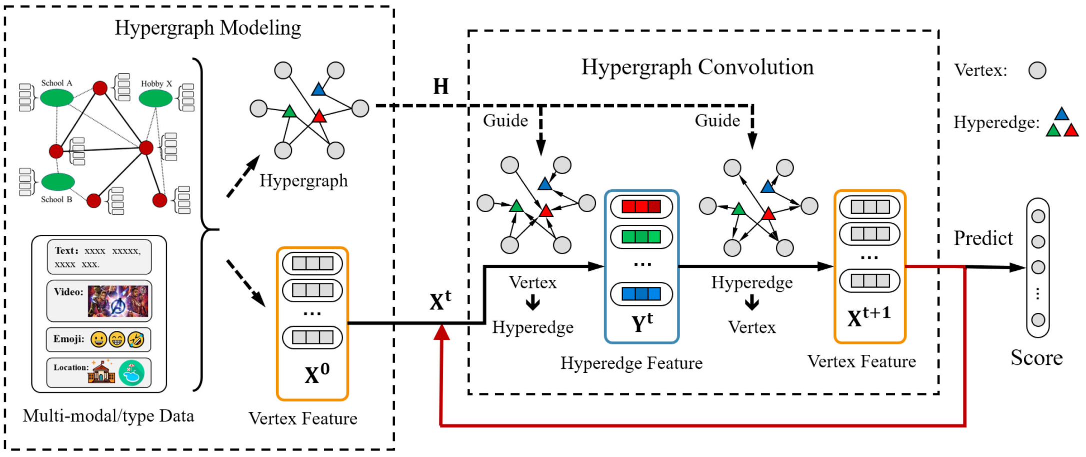


Fig. 3. The proposed hypergraph neural network framework (HGNN$^+$). In the left part, a hypergraph is employed to model raw multi-modal/type data. $\mathbf{H}$ is generated to indicate hypergraph structure and $\mathbf{X}$ denotes the vertex feature extracted from raw data. In the right part, Hypergraph Convolution Module shows one hypergraph convolution process and the red line from the module allows stacking multiple hypergraph convolution layers for deeper vertex embedding extraction.

HGNN$^+$. A toolbox for the HGNN$^+$ framework, called THU-DeepHypergraph[1], is released for public use.

The remainder of the paper is organized as follows. First, we briefly review related work on GNN and hypergraph learning in Section 2. Section 3 presents the preliminaries of hypergraphs. In Section 4, we introduce the proposed general hypergraph neural network HGNN$^+$, including hypergraph modeling and hypergraph convolution. In Section 5, we first mathematically compare the hypergraph structure with the graph structure, and then discuss the relationships among GCN, HGNN, and HGNN$^+$. Experimental results and discussions are provided in Section 6. The THU-DeepHypergraph toolbox is introduced in Section 7. We conclude this paper in Section 8.

2 Related Work

2.1 Graph Neural Networks

Recent years have witnessed the rapid progress of GNN [1], [2], [3], [4], [5]. GNN is designed for fixing the gap between deep convolution and irregular data processing, which can be directly applied on arbitrary graph structures. Generally, GNN can be mainly divided into two categories: spectral-based and spatial-based methods.

Spectral-based approaches [2], [5], [11], [12], [13] define graph convolution from the perspective of graph signal processing [11], where the graph convolution operation is interpreted as removing noise from graph signals, or smoothing the information among the linked vertices via graph Laplacian normalization. In [12], Bruna *et al.* introduced the first GNN, which adopted the graph Laplacian eigen basis as an analogy of the Fourier transform. In [14], the spectral filters can be parameterized with smooth coefficients to make them spatial-localized. In [2], the Chebyshev polynomials were simplified into 1-order polynomials to form an efficient layer-wise propagation.

Spatial-based approaches [1], [3], [15], [16] are formulated as aggregating messages from neighbor vertices, and the convolution operation is defined in groups of spatially closed vertices. In [16], the powers of a transition matrix were employed to define the neighborhood of vertices. In [3], the attention mechanisms were introduced into the graph to build attention-based architecture to perform node classification on the graph. To achieve a more powerful expression in Weisfeiler-Lehman (WL) graph isomorphism test, Keyulu *et al.* [17] leveraged a sum aggregator in neighbor message aggregation that can discriminate more types of local graph structure.

Both spectral-based and spatial-based GNNs perform representation learning on the graph structure and show decent performance. GNN has been used in various applications, such as drug-target interaction (DTI) prediction, collaborative filtering, community detection and brain functional network. For DTI prediction task, in [18] Manoochehri *et al.* adopted WL-GNN to learn the latent pattern of DTIs. For collaborative filtering task, in [19] Wang *et al.* proposed Neural Graph Collaborative Filtering (NGCF) to learn the latent collaborative signal in user-item interaction graph.

2.2 Hypergraph Learning

In recent years, hypergraph has shown strong capability of modeling and learning complex correlations.

Hypergraph learning was first introduced in [20], which conducts transductive learning and can be seen as a propagation process on the hypergraph structure. The transductive inference on hypergraph aims to minimize the label difference among vertices with stronger connections on the hypergraph. For the past few years, hypergraph learning has been extensively developed and applied in many areas. Wang *et al.* [21] constructed a complex hypergraph including global, local visual feature and tag information to learn the relevance of image in the task of tag-based image retrieval. To model brain functional connectivity networks (FCN), Xiao *et al.* [22] proposed weighted hypergraph learning, which is capable of capturing the relations among brain regions than the traditional graph based methods and the existing unweighted hypergraph based method.

Inspired by the recent immense success of deep learning, some researchers have developed deep hypergraph learning methods. For example, in our conference version Feng *et al.* [6] proposed hypergraph neural networks (HGNN)

1. https://github.com/iMoonLab/THU-DeepHypergraph

TABLE 1
Notations and Definitions

Notation	Definition
$\mathcal{G}$	$\mathcal{G}=(\mathcal{V},\mathcal{E},\mathbf{W})$ indicates a hypergraph, and $\mathcal{V}$, $\mathcal{E}$ and $\mathbf{W}$ indicate the set of vertices, the set of edges, and the weights of the edges, respectively.
$\mathcal{V}$	The set of vertices of $\mathcal{G}$.
$\mathcal{E}$	The set of hyperedges of $\mathcal{G}$.
N	The number of vertices on $\mathcal{G}$, i.e., $\lvert\mathcal{V}\rvert$.
M	The number of edges on $\mathcal{G}$, i.e., $\lvert\mathcal{E}\rvert$.
x_i^0	The initial feature for the i-th vertex on $\mathcal{G}$.
$\mathbf{X}^0$	The initial feature for all the vertices on $\mathcal{G}$.
$\mathbf{X}^t$	The input feature of convolution layer t.
X_i^t	The embedding for vertex i in layer t.
$\mathbf{W}$	The diagonal matrix of the hyperedge weights.
$d(v)$	The degree of vertex v.
$\delta(e)$	The degree of hyperedge e.
$\mathbf{D}_v$	The diagonal matrix of vertex degrees. $\mathbf{D}_v \in \mathbb{R}^{N\times N}$.
$\mathbf{D}_e$	The diagonal matrix of hyperedge degrees. $\mathbf{D}_v \in \mathbb{R}^{M\times M}$.

towards modeling and learning beyond-pairwise complex correlations. Unlike GNN, HGNN designs a vertex-hyperedge-vertex information propagating pattern to iteratively learn the data representation. Besides, in this literature, the hypergraph Laplacian is first approximated and introduced into the deep hypergraph learning method to accelerate learning. Built on our conference version [6], Bai *et al.* [7] further focused on attention module on hypergraph (Hyper-Atten), which follows the hypergraph convolution patterns defined in HGNN. Enlightened by [3], Hyper-Atten introduces a hyperedge-vertex attention learning module to adaptively identify the importance of different vertices in the same hyperedge and thus revealing the intrinsic correlation between vertices. Besides, Yadati *et al.* [8] proposed a method of training a GCN on hypergraphs (HyperGCN) for semi-supervised learning. HyperGCN is designed based on the spectral theory of hypergraphs. Given a hypergraph, HyperGCN first translates it into a simple weighted graph through a specific strategy and then performs standard GCN on the graph to learn data representations. All the above-mentioned methods are conducted on the fixed hypergraph structure. However, the initial hypergraph constructed from raw data may not be optimal.

As discussed above, most existing deep hypergraph learning methods are derived from the spectral theory of hypergraphs. Therefore, these methods are usually defined on undirected hypergraphs, which limits their applications. Besides, some methods also suffer from structural over-simplification or over-parametrization. For example, HyperGCN only considers one simple edge for each hyperedge. Even though it introduces mediators, such irreversible simplification will certainly lose key information. Hyper-Atten leverages a hyperedge-vertex attention module, which introduces a large number of parameters and may lead to overfitting problem.

3 Preliminaries of Hypergraphs

The hypergraph is a generalization of the graph. Different from the graph, an edge in the hypergraph, called hyperedge, is a subset of all vertices in the hypergraph. For clarification, we first summarize important notations and definitions throughout this paper in Table 1. A hypergraph is defined as $\mathcal{G}=(\mathcal{V},\mathcal{E},\mathbf{W})$, which includes a vertex set $\mathcal{V}$, a hyperedge set $\mathcal{E}$, and a hyperedge weight matrix $\mathbf{W}$. A hypergraph $\mathcal{G}$ can be described by an $\lvert\mathcal{V}\rvert \times \lvert\mathcal{E}\rvert$ incidence matrix $\mathbf{H}$, whose entries are defined as $\mathbf{H}(v,e)=\begin{cases}1, & \text{if } v\in e\\ 0, & \text{otherwise}\end{cases}$. For a vertex $v\in\mathcal{V}$, its vertex degree is defined as $d(v)=\sum_{e\in\mathcal{E}}\omega(e)\mathbf{H}(v,e)$. For a hyperedge $e\in\mathcal{E}$, its edge degree is defined as $\delta(e)=\sum_{v\in\mathcal{V}}\mathbf{H}(v,e)$. $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the diagonal matrices of vertex degrees and edge degrees, respectively. The initial feature set for each vertex is denoted as $\mathbf{X}^0=\{x_1^0,x_2^0,\cdots,x_N^0\}$, $x_i^0\in\mathbb{R}^{C_0}$, where C_0 is the dimension of the feature.

Given a hypergraph, the classification task turns to classify the vertices on the hypergraph, where the labels on the hypergraph are required to be smoothed through the hypergraph structure. The task can be formulated as a regularization as introduced in [20]:

$$\arg\min_f \{\mathcal{R}_{emp}(f)+\Omega(f)\}, \tag{1}$$

where $\Omega(f)$ is a regularizer on hypergraph, $\mathcal{R}_{emp}(f)$ denotes the supervised empirical loss and $f(\cdot)$ is a classification function. The regularizer $\Omega(f)$ can be defined as:

$$\Omega(f)=\frac{1}{2}\sum_{e\in\mathcal{E}}\sum_{\{u,v\}\in\mathcal{V}}\frac{w(e)\mathbf{H}(u,e)\mathbf{H}(v,e)}{\delta(e)}\left(\frac{f(u)}{\sqrt{d(u)}}-\frac{f(v)}{\sqrt{d(v)}}\right)^2, \tag{2}$$

Here we let $\mathbf{\Theta}=\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$ and $\mathbf{\Delta}=\mathbf{I}-\mathbf{\Theta}$, and the normalized $\Omega(f)$ can be rewritten as $\Omega(f)=f^\top\mathbf{\Delta}f$, where Δ is positive semi-definite and usually called the hypergraph Laplacian.

4 The Framework of Hypergraph Neural Network HGNN$^+$

In this section, we briefly introduce the framework of hypergraph neural network (HGNN$^+$), which aims to provide a general framework for representation learning on a given raw data. It is composed of two procedures, as shown in Fig. 3, i.e., hypergraph modeling and hypergraph convolution. In the step of hypergraph modeling, the available data are used to generate high-order correlations which are represented by a hypergraph. There are three types of hyperedge groups, using pairwise edge, k-Hop and neighbors in the feature space, respectively. In this procedure, all these types of hyperedge groups (if available) are generated and concatenated in a hypergraph for data correlation modeling. In the step of hypergraph convolution, a set of hypergraph convolution family, i.e., spectral hypergraph convolution and spatial hypergraph convolution, is conducted for representation learning. These convolution procedures can utilize the information from high-order correlation and multi-modal data to generate better representations.

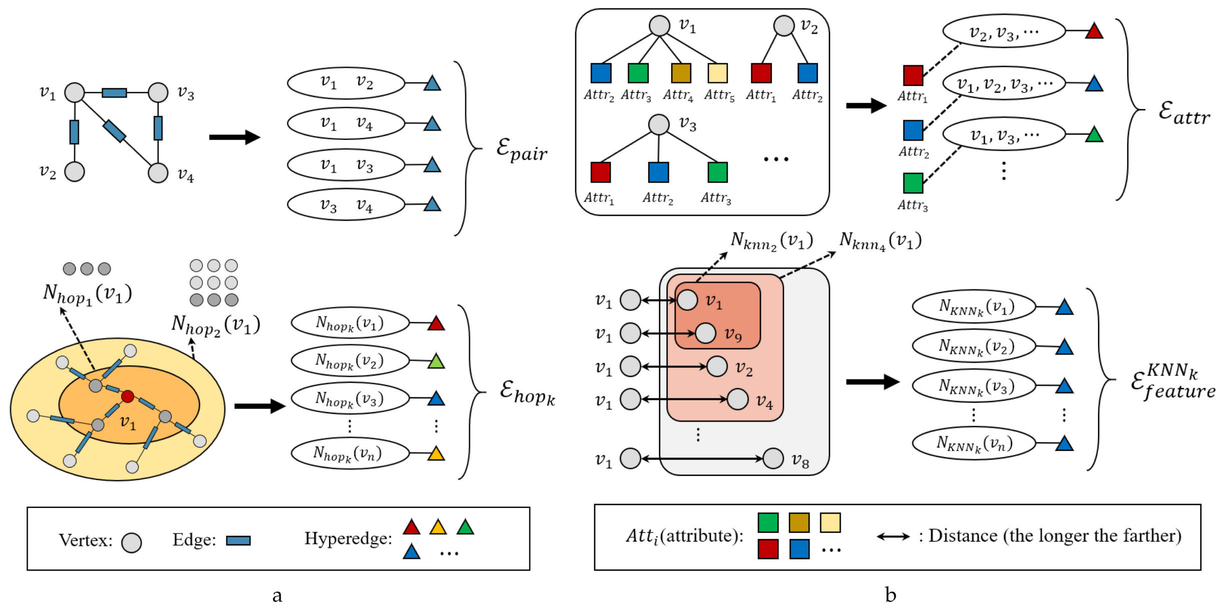


Fig. 4. Illustration for hyperedge group generation.

4.1 Hypergraph Modeling

In this section, we introduce how to construct a flexible hypergraph from the raw data if there is no natural hypergraph structure. The hypergraph structure is used to model data correlation. To better exploit the high-order correlation among the data, how to generate a good hypergraph structure is important. It is noted that there is no explicit hypergraph structure in most cases. Therefore, we need to generate the hypergraph using different strategies. Usually, the situation for hypergraph generation from scratch can be divided into three scenarios, i.e., the data with graph structure, the data without graph structure, and the data with multi-modal/multi-type representations. Given the data, three hyperedge generation strategies, which employ pairwise edge, k-Hop and neighbors in the feature space respectively, are introduced here. The strategies of using pairwise edge and k-Hop are utilized for hyperedge group generation from the data with graph structure, and that of using neighbors in feature space is employed for hyperedge group generation from the data without graph structure. Finally, all the hyperedge groups will be further concatenated to generate the overall hypergraph. We will introduce the hyperedge group generation first, and then introduce the combination of these hyperedges groups and the method to deal with multi-modal/multi-type data. Compared with the conference version, we have extended from two aspects in this journal version: 1) we systematically introduce four typical hyperedge group construction strategies toward complex applications including graph-based representation, feature-based representation, and attribute-based representation. 2) We further propose an adaptive hyperedge group fusion strategy to balance the contribution of hyperedge groups that constructed from different aspects.

4.1.1 *Hyperedge Group Generation*

When the data correlation is with graph structure. In some scenarios, there are available pairwise data correlations, such as existing graph structure for the data. Here we let $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ indicate the graph structure, where $v_i \in \mathcal{V}_s$ is a vertex and $e_{s_{ij}} \in \mathcal{E}_s$ is an edge in the graph connecting v_i and v_j. Let $\mathbf{A}$ denotes the adjacency matrix of $\mathcal{G}_s$. Given such graph structure, two types of hyperedge groups can be generated as follows:

- *Hyperedge group using pairwise edge ($\mathcal{E}_{\text{pair}}$).* $\mathcal{E}_{\text{pair}}$ targets on directly transforming the graph structure to a group of 2-uniform hyperedges, as shown in the top of Fig. 4a, in which each hyperedge e_{ij} in this group just connects the two vertices v_i and v_j in the corresponding edge in the graph $\mathcal{G}_f$:

$$\mathcal{E}_{\text{pair}} = \{\{v_i, v_j\} \mid (v_i, v_j) \in \mathcal{E}_s\ \}. \tag{3}$$

  $\mathcal{E}_{\text{pair}}$ is able to fully cover the low-order (pairwise) correlation in the graph structure, which is the basic information needed in the high-order correlation modeling.
- *Hyperedge group using k-Hop neighbors ($\mathcal{E}_{\text{hop}}$).*
  $\mathcal{E}_{\text{hop}}$ aims to find the related vertices for a central one through the k-Hop reachable positions in the graph structure, as shown in the bottom of Fig. 4a. The k-Hop neighborhoods of a vertex v in graph $\mathcal{G}_s$ is defined as: $N_{\text{hop}_k}(v) = \{u \mid \mathbf{A}^k_{uv} \neq 0, u \in \mathcal{V}_s\}$. Here k can be vary from $[2, n_v]$, where n_v is the number of vertices in $\mathcal{G}_s$. The hyperedge group $\mathcal{E}_{\text{hop}}$ with k-Hop can be written as:

$$\mathcal{E}_{\text{hop}_k} = \{N_{\text{hop}_k}(v) \mid v \in \mathcal{V}\}. \tag{4}$$

  $\mathcal{E}_{\text{hop}}$ is able to exploit external correlated vertices for the central one by extending the search radius in the graph structure, which also leads to groups of vertices, instead of two vertices, for the hyperedge. It can provide richer correlation information compared with just the pairwise one in $\mathcal{E}_{\text{pair}}$.

When the data correlation is without graph structure. When there is no available graph structure for the data, we need to build it following different methods. Usually, there could be two types of data for each subject: one is the attribute-like data, and the other one is the feature(s) associated with each vertex.

- *Hyperedge group using attributes* $(\mathcal{E}_{\text{attribute}})$. Given the attribute-like data, such as geo-locations, time and other specific information shared by different subjects, a group of hyperedges using neighbors in the attribute space can be generated, as shown in the top of Fig. 4b, where each hyperedge represents one attribute a (or one subtype of the attribute if available) and connects all the subjects sharing the same attribute. Vertex subset that shares the attribute a can be denoted as $N_{\text{att}}(a)$. $\mathcal{A}$ is a set that contains all attributes or subtypes of the attribute. This group of hyperedges from the attribute can be written as:

$$\mathcal{E}_{\text{attribute}} = \{N_{\text{att}}(a) \mid a \in \mathcal{A}\}. \tag{5}$$

  Here, $\mathcal{E}_{\text{attribute}}$ can model the correlation in the attribute space from the group level.

- *Hyperedge group using features* $(\mathcal{E}_{\text{feature}})$. Given the feature for each vertex, the second type of $\mathcal{E}_{\text{feature}}$ can be generated by finding the neighbors of each vertex in the feature space. Here different strategies can be employed. Given a vertex as the centroid, its k-nearest neighbors in the feature space can be connected by a hyperedge, or all the neighbors within a distance d to the centroid (including the centroid) can be selected, as shown in the bottom of Fig. 4b.

$$\begin{cases} \mathcal{E}_{\text{feature}}^{\text{KNN}_k} = \{N_{\text{KNN}_k}(v) \mid v \in \mathcal{V}\} \\ \mathcal{E}_{\text{feature}}^{\text{distance}_d} = \{N_{\text{dis}_d}(v) \mid v \in \mathcal{V}\} \end{cases}. \tag{6}$$

  This type of hyperedges aims to find the relationship behind the feature of the vertices. It can be set in multi-scales, such as different k or d values in the neighbor finding procedure.

4.1.2 *Combination of Hyperedge Groups*

Here, several hyperedge groups can be generated using above strategies. Given generated hyperedge groups or natural hyperedge groups, we need to further combine them to generate the final hypergraph. Supposing there are K hyperedge groups $\{\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_K\}$, we can have K incidence matrices $\mathbf{H}_k \in \{0,1\}^{N\times M_k}$ respectively. The simplest fusion way to construct the incidence matrix for the hypergraph $\mathcal{G}$ is directly concatenating all the hyperedge groups as: $\mathbf{H} = \mathbf{H}_1||\mathbf{H}_2||\cdots||\mathbf{H}_K$, where $\cdot||\cdot$ is matrix concatenation operation. The hyperedge weight matrix of the hypergraph can be assigned with value 1 for treating it equally. We call the simplest fusion way as **Coequal Fusion**.

However, considering that the information richness of different hyperedge groups varies a lot, such a simple *Coequal Fusion* cannot make full use of the multi-modal hybrid high-order correlations. Therefore, in this paper, we propose an adaptive strategy for the fusion of hyperedge groups, namely *Adaptive Fusion*. More specifically, each hyperedge group is associated with a trainable parameter, which can adaptively adjust the effect of multiple hyperedge groups on the final vertex embeddings. It is defined as:

$$\begin{cases} \mathbf{w}_k = \text{copy}(\text{sigmoid}(w_k), M_k) \\ \mathbf{W} = \text{diag}(\mathbf{w}_1^1, \cdots, \mathbf{w}_1^{M_1}, \cdots, \mathbf{w}_K^1, \cdots, \mathbf{w}_K^{M_K}), \\ \mathbf{H} = \mathbf{H}_1||\mathbf{H}_2||\cdots||\mathbf{H}_K \end{cases} \tag{7}$$

where $\mathbf{w}_k \in \mathbb{R}$ is a trainable parameter shared by all hyperedges inside a specified hyperedge group k. $\text{sigmoid}(\cdot)$ is an element-wise normalization function. Vector $\mathbf{w}_k = (\mathbf{w}_k^1, \cdots, \mathbf{w}_k^{M_k}) \in \mathbb{R}^{M_k}$ denotes the generated weight vector for hyperedge group k. $\text{copy}(a, b)$ function returns a vector of size b, and the value of which is padded by copying a b times. Let $M = M_1 + M_2 + \cdots + M_K$ denotes the summation of the hyperedges in all hyperedge groups. $\mathbf{W} \in \mathbb{R}^{M\times M}$ is a diagonal matrix that indicates the weight matrix of hypergraph, with each entry $\mathbf{W}^{ii}$ denoting the weight of the corresponding hyperedge e_i. $\mathbf{H} \in \{0,1\}^{N\times M}$ indicates the incidence matrix of the hypergraph generated by concatenating $(\cdot||\cdot)$ the incidence matrices of multiple hyperedge groups.

Given the multi-model/multi-type data, multiple hyperedge groups can be generated accordingly. Hypergraph incidence matrix $\mathbf{H}$ and hyperedge weight matrix $\mathbf{W}$ will be generated from the constructed hyperedge groups, which can then be fed into Hypergraph Convolution Layer for further computation.

4.2 Hypergraph Convolution

In this subsection, we define two hypergraph convolution HGNNConv and HGNNConv$^+$ from the spectral aspect and the spatial aspect, respectively. The former is proposed in the conference version, and the later is proposed in this journal version. As for the spatial convolution on hypergraph, we first define a general spatial hypergraph convolution layer, a two stage message passing framework. In the following, HGNNConv$^+$ is proposed by specifying aggregation function in the two stages. Compared with the conference version, the extended HGNNConv$^+$ exhibits more salability, which can be easily generalized into various applications such as the directed hypergraph in future work.

4.2.1 *Spectral Convolution on Hypergraph*

Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Delta)$ with N vertices, since the hypergraph Laplacian $\mathbf{\Delta}$ is a $N \times N$ positive semi-definite matrix, the eigen decomposition $\Delta = \Phi\Lambda\Phi^\top$ can be employed to get the orthonormal eigen vectors $\mathbf{\Phi} = \text{diag}(\phi_1, \ldots, \phi_N)$ and a diagonal matrix $\Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_\mathbf{N})$ containing corresponding non-negative eigenvalues. Then, the Fourier transform for a signal $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_N})$ in hypergraph is defined as $\hat{\mathbf{x}} = \mathbf{\Phi}^\top\mathbf{x}$, where the eigen vectors are regarded as the Fourier bases and the eigenvalues are interpreted as frequencies. The spectral convolution of signal $\mathbf{x}$ and filter $\mathbf{g}$ can be denoted as

$$\mathbf{g} \star \mathbf{x} = \Phi((\Phi^\top\mathbf{g}) \odot (\Phi^\top\mathbf{x})) = \mathbf{\Phi} g(\mathbf{\Lambda})\mathbf{\Phi}^\top\mathbf{x}, \tag{8}$$

where $\odot$ denotes the element-wise Hadamard product and $g(\Lambda) = \mathbf{diag}(\mathbf{g}(\lambda_1), \ldots, \mathbf{g}(\lambda_n))$ is a function of the Fourier coefficients. However, the computation cost in forward and inverse Fourier transform is $\mathcal{O}(n^2)$. To solve the problem, we can follow [49] to parametrize $g(\Lambda)$ with K order polynomials. Furthermore, we use the truncated Chebyshev expansion as one such polynomial. Chebyshv polynomials $T_k(x)$ is recursively computed by $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. Thus, the $g(\mathbf{\Lambda})$ can be parametried as

$$\mathbf{g} \star \mathbf{x} \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{\Delta}})\mathbf{x}, \tag{9}$$

where $T_k(\tilde{\mathbf{\Delta}})$ is the Chebyshev polynomial of order k with scaled Laplacian $\tilde{\mathbf{\Delta}} = \frac{2}{\lambda_{max}}\mathbf{\Delta} - \mathbf{I}$. In Eq. (9), the expansive computation of Laplacian Eigen vectors is excluded and only matrix powers, additions and multiplications are included, which brings further improvement in computation complexity. We can further let $K = 1$ to limit the order of convolution operation due to that the Laplacian in hypergraph can already well represent the high-order correlation between nodes. It is also suggested in [2] that $\lambda_{max} \approx 2$ because of the scale adaptability of neural networks. Then, the convolution operation can be further simplified to

$$\mathbf{g} \star \mathbf{x} \approx \theta_0 \mathbf{x} - \theta_1 \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\mathbf{x}, \tag{10}$$

where θ_0 and θ_1 is parameters of filters over all nodes. We further use a single parameter θ to avoid the overfitting problem, which is defined as

$$\begin{cases} \theta_1 = -\frac{1}{2}\theta \\ \theta_0 = \frac{1}{2}\theta \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}, \end{cases} \tag{11}$$

Then, the convolution operation can be simplified to the following expression

$$\begin{aligned} \mathbf{g} \star \mathbf{x} \quad &\approx \tfrac{1}{2}\theta \mathbf{D}_v^{-1/2}\mathbf{H}(\mathbf{W}+\mathbf{I})\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\mathbf{x} \\ &\approx \theta \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\mathbf{x}, \end{aligned} \tag{12}$$

where $(\mathbf{W}+\mathbf{I})$ can be regarded as the weight of the hyperedges. $\mathbf{W}$ is initialized as an identity matrix, which means equal weights for all hyperedges.

When we have a hypergraph signal $\mathbf{X}^t$ for t-th layer, our hyperedge convolution layer **HGNNConv** can be formulated by

$$\mathbf{X}^{t+1} = \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\mathbf{X}^t\mathbf{\Theta}), \tag{13}$$

where $\mathbf{\Theta}$ is the parameter to be learned during the training process. The filter $\mathbf{\Theta}$ is applied over the nodes in hypergraph to extract features. After convolution, we can obtain $\mathbf{X}^{t+1}$, which can be used for further learning.

4.2.2 General Spatial Convolution on Hypergraph

In this subsection, we introduce the hypergraph convolution from spatial domain. First, let's briefly review the definition of a typical spatial-based graph convolution. An image can be considered as a grid graph where each pixel represents a vertex and each vertex only connects its surrounding neighborhood vertices. Each vertex (pixel) in image owns a C-channel feature. Filtering on image can be regarded as a process that central vertex aggregates its neighbors' feature with average aggregation after transforming their feature. Similarly, for a simple graph, spatial-based graph convolution takes the aggregation of its neighbor vertices to get a new representation of the central vertex. Messages in spatial graph convolution run from neighbor vertices to center vertex, which is following the definition of "Path" in simple graph. A path in graph is defined as $P(v_1, v_k) = (v_1, v_2, \cdots, v_k)$. It is a sequence of vertices with the property that each vertex in the sequence is adjacent to the vertex next to it, which means that all the vertex pairs of i and $i+1$ $(1 \leq i \leq k-1)$ have *Neighbor Relation*.

Here, we can define the spatial convolution on hypergraph. For each vertex in hypergraph, we aggregate its neighbor vertex messages to update itself according to the "path" between the central vertex and each vertex in its neighborhood. The path in hypergraph, named hyperpath [10], between two distinct vertices v_1 and v_k is defined as a sequence:

$$P(v_1, v_k) = (v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_k, v_k), \tag{14}$$

in which v_j and v_{j+1} belong to the same vertex subset that indicated by a hyperedge e_j. Obviously, each two neighbor vertices in a hyperpath is separated by a hyperedge. Message in hypergraph between two vertices is propagated through related hyperedges, which can take the advantage of high-order relationship through hyperedge compared with that in graph. For the message propagation from vertex to hyperedge and that from hyperedge to vertex using hyperpath, we first extend the *Neighbor Relation* definition among vertices to the *Inter-neighbor Relation* N over vertex set $\mathcal{V}$ and hyperedge set $\mathcal{E}$.

Definition 1. *The* Inter-Neighbor Relation $N \subseteq \mathcal{V} \times \mathcal{E}$ *on a hypergraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ *with incidence matrix* $\mathbf{H} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{E}|}$ *is defined as:*

$$N = \{\, (v, e) \,|\, \mathbf{H}(v, e) = 1, \ v \in \mathcal{V} \text{ and } e \in \mathcal{E} \,\} \tag{15}$$

Then, we define the vertex inter-neighbor set $N_v(e)$ of hyperedge e and the hyperedge inter-neighbor set $N_e(v)$ of vertex v based on the *Inter-Neighbor Relation*.

Definition 2. *The vertex inter-neighbor set of hyperedge $e \in \mathcal{E}$ is defined as:*

$$\mathcal{N}_v(e) = \{\, v \,|\, vNe, \ v \in \mathcal{V} \text{ and } e \in \mathcal{E} \,\} \tag{16}$$

Definition 3. *The hyperedge inter-neighbor set of vertex $v \in \mathcal{V}$ is defined as:*

$$\mathcal{N}_e(v) = \{\, e \,|\, vNe, \ v \in \mathcal{V} \text{ and } e \in \mathcal{E} \,\} \tag{17}$$

Following *Definition 1, 2, and 3*, we introduce the message passing of neighbor vertex message aggregation via hyperpath for one spatial hypergraph convolution layer. Given a vertex $\alpha \in \mathcal{V}$ of hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, we aim to aggregate messages from its hyperedge inter-neighbor set $N_e(\alpha)$. To obtain those hyperedge messages for each hyperedge β in the hyperedge inter-neighbor set $\mathcal{N}_e(\alpha)$, we aggregate messages from its vertex inter-neighbor set $\mathcal{N}_v(\beta)$. Then, the two steps of hypergraph convolution make a closed message passing loop from vertex feature set X^t to X^{t+1}. A general spatial hypergraph convolution in the t-th layer can be defined as:

$$\begin{cases} \left.\begin{array}{ll} m_\beta^t & = \sum_{\alpha \in \mathcal{N}_v(\beta)} M_v^t(x_\alpha^t) \\ y_\beta^t & = U_e^t(w_\beta, m_\beta^t) \end{array}\right\} \text{Stage 1} \\ \left.\begin{array}{ll} m_\alpha^{t+1} & = \sum_{\beta \in \mathcal{N}_e(\alpha)} M_e^t(x_\alpha^t, y_\beta^t) \\ x_\alpha^{t+1} & = U_v^t(x_\alpha^t, m_\alpha^{t+1}) \end{array}\right\} \text{Stage 2} \end{cases}, \tag{18}$$

where $x_\alpha^t \in \mathbf{X}^t$ is the input feature vector of vertex $\alpha \in \mathcal{V}$ in layer $t = 1, 2, \cdots, T$, and x_α^{t+1} is the updated feature of vertex α. m_β^t is the message of hyperedge $\beta \in \mathcal{E}$, and w_β is a weight associated to hyperedge β. m_α^{t+1} denotes the message of vertex α. y_β^t is the hyperedge feature of hyperedge β which is a element of hyperedge feature set $Y^t = \{y_1^t, y_2^t, \cdots, y_M^t\}$, $y_i^t \in \mathbb{R}^{C_t}$ in layer t. $M_v^t(\cdot), U_e^t(\cdot), M_e^t(\cdot), U_v^t(\cdot)$ are the vertex message function, hyperedge update functions, hyperedge message function and vertex update function in t_{th} layer, respectively, which can be flexibility defined for specified applications.

The spatial hypergraph convolution layer is designed for high-level representation learning via the high-order relationship in hypergraph structure. Compared with the one-stage message passing in graph convolution, the two-stage spatial hypergraph convolution is composed of four flexible operations with learned differentiable functions. Similar to the neighbor relation defined in graph, a vertex's hyperedge inter-neighbors and a hyperedge's vertex inter-neighbors have no natural ordering. Therefore, a summation operation is used to aggregate vertex/hyperedge messages from $M_v^t(\cdot)/M_e^t(\cdot)$ operation.

4.2.3 HGNN$^+$ Convolution Layer Configurations

A simple spatial hypergraph convolution layer (named **HGNNConv**$^+$) via specifying the message-update functions (vertex message function $M_v^t(\cdot)$, hyperedge update function $U_e^t(\cdot)$, hyperedge message function $M_e^t(\cdot)$ and vertex update function $U_v^t(\cdot)$) are introduced as:

$$\begin{cases} M_v^t(x_\alpha^t) & = \frac{x_\alpha^t}{|\mathcal{N}_v(\beta)|} \\ U_e^t(w_\beta, m_\beta^t) & = w_\beta \cdot m_\beta^t \\ M_e^t(x_\alpha^t, y_\beta^t) & = \frac{y_\beta^t}{|\mathcal{N}_e(\alpha)|} \\ U_v^t(x_\alpha^t, m_\alpha^{t+1}) & = \sigma(m_\beta^{t+1} \cdot \mathbf{\Theta}^t) \end{cases}, \tag{19}$$

where $\mathbf{\Theta}^t \in \mathbb{R}^{C^t \times C^{t+1}}$ is a trainable parameter of layer t, which can be learned in training phase. $\sigma(\cdot)$ is an arbitrary non-linear activation function like $ReLU(\cdot)$ *etc.* Note that in Eq. (19), $x_\alpha^t/|\mathcal{N}_v(\beta)|$ and $y_\beta^t/|\mathcal{N}_e(\alpha)|$ denote the normalized vertex/hyperedge feature, which is adopted to accumulate convergence and prevent jittering in some degree.

For faster forward propagation of **HGNNConv**$^+$ in GPU/CPU devices, we rewrite it in the matrix format. Considering X^t is the input vertex feature set of layer t. From **Definition 1 and 2**, $\mathbf{H}^\top \in \{0,1\}^{M \times N}$ can control the hyperedge inter-neighbor of each vertex feature in X^t. Hence, we use it to guide each vertex to aggregate and generate the hyperedge feature set Y^t, which can be formulated as $\mathbf{Y}^t = \mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{X}^t$. In a similar way, the process that updating vertex feature set $\mathbf{X}^{t+1}$ from hyperedge feature set $\mathbf{Y}^t$ can be formulated as $\mathbf{X}^{t+1} = \sigma(\mathbf{D}_v^{-1}\mathbf{H}\mathbf{Y}^t\mathbf{\Theta}^t)$. Thus, the matrix format of **HGNNConv**$^+$ can be written as:

$$\mathbf{X}^{t+1} = \sigma(\mathbf{D}_v^{-1}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{X}^t\mathbf{\Theta}^t). \tag{20}$$

5 Discussions

In this section, we provide comprehensive analyses and comparisons of the hypergraph structure and the proposed methods for a deeper understanding.

5.1 Hypergraph vs. Graph

Here, we provide a mathematical comparison of hypergraphs and graphs via the random walks [36] and the Markov chain [35]. The proof concludes that: *from the random walks' aspect, a hypergraph with edge-independent vertex weights is equivalent to a weighted graph, and a hypergraph with edge-dependent vertex weights cannot be reduced to a weighted graph.*

In general, to accurate describe the real-world correlations, two types of hypergraphs can be constructed, i.e., hypergraphs with edge-independent vertex weights and hypergraphs with edge-dependent vertex weights. The hypergraph with edge-independent vertex weights ($\mathcal{G}_{\text{in}} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$) can model the beyond pair-wise correlations, which can be denoted by the binary hypergraph incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{E}|}$, in which vertices in each hyperedge share the same weight. In contrast, the hypergraph with edge-dependent vertex weights ($\mathcal{G}_{\text{de}} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}, \gamma\}$) can further model the variable correlation intensity in each hyperedge, which can be denoted by the weighted hypergraph incidence matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$. Assuming that the hyperedge e includes the vertex v, then we use $\gamma_e(v)$ to denote the connection intensity between e and v and $w(e)$ to denote the weight of hyperedge e. The definition of binary hypergraph incidence matrix $\mathbf{H}$, vertex degree $d(v)$ and the hyperedge degree $\delta(e)$ in hypergraphs with edge-independent vertex weights is similar to Section 3, In hypergraphs with edge-dependent vertex weights, the $d(v)$, and $\delta(e)$ can be defined as follows:

$$\begin{cases} d(v) & = \sum_{\beta \in \mathcal{N}_e(v)} w(\beta) \\ \delta(e) & = \sum_{\alpha \in \mathcal{N}_v(e)} \gamma_e(\alpha) \end{cases}, \tag{21}$$

where $\mathcal{N}_v(\cdot)$ and $\mathcal{N}_e(\cdot)$ are defined in Eq. (17) and (16), respectively.

Random walks and the Markov chain in hypergraphs. We first define the random walk in a hypergraph following [20], [34], [35], [36]. At time t, a random walker at vertex v_t will do the following:

- Pick an edge e containing vertex $v_t = v$, with probability $p_{v \to e}$.
- Pick vertex u from e, with probability $p_{e \to u}$.
- Move to vertex $v_{t+1} = u$, at time $t+1$.

Then, the transition probability $p_{v,u}$ of the corresponding Markov chain on $\mathcal{V}$ can be defined as $p_{v,u} = \sum_{e \in \mathcal{N}_e(v,u)} p_{v \to e} p_{e \to u}$, where $\mathcal{N}_e(v,u) = \mathcal{N}_e(v) \cap \mathcal{N}_e(u)$ denotes the hyperedge $\beta \in \mathcal{N}_e(v,u)$ containing vertices v and u, simultaneously. In hypergraphs with edge-independent vertex weights, we have $p_{v \to e} = w(e)/d(v)$ and $p_{e \to u} = 1/\delta(e)$. Then, the transition probability $p_{v,u}$ can be defined as: $p_{v,u} =$

$\sum_{\beta\in\mathcal{N}_e(v,u)}\frac{w(\beta)}{d(v)}\cdot\frac{1}{\delta(\beta)}$. In hypergraphs with edge-dependent vertex weights, we have $p_{v\to e} = w(e)/d(v)$ and $p_{e\to u} = \gamma_e(u)/\delta(e)$, Then, the transition probability $p_{v,u}$ can be defined as $p_{v,u} = \sum_{\beta\in\mathcal{N}_e(v,u)}\frac{w(\beta)}{d(v)}\cdot\frac{\gamma_\beta(u)}{\delta(\beta)}$.

Following [35] we provide the following Definitions and Lemmas to compare the graph and two types of hypergraphs.

Definition 4. *Let M be a Markov chain with state space X and transition probability $p_{x,y}$, for $x, y \in S$. We say M is reversible if there exists a probability distribution π over S such that $\pi_x p_{x,y} = \pi_y p_{y,x}$.*

Lemma 5. *Let M be an irreducible Markov chain with finite state space S and transition probability $p_{x,y}$ for $x, y \in S$. M is reversible if and only if there exists a weighted, undirected graph G with vertex set S such that a random walk on G and M are equivalent.*

Definition 6. *A Markov chain is reversible if and only if its transition probability satisfies*

$$p_{v_1,v_2}p_{v_2,v_3}\cdots p_{v_n,v_1} = p_{v_1,v_n}p_{v_n,v_{n-1}}\cdots p_{v_2,v_1}, \tag{22}$$

for any finite sequence of states $v_1, v_2, \cdots v_n \in S$. This definition is also known as Kolmogorov's criterion. The proof can be found in [39].

Theorem 1. *Let $\mathcal{G}_{\text{in}} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ be a hypergraph with edge-independent weights. Then, there exists a weighted, undirected graph G such that a random walk on G is equivalent to a random walk on $\mathcal{G}_{\text{in}}$.*

The proof of Theorem 1 can follow the process as:

1) *A random walk on $\mathcal{G}_{\text{in}}$ is equivalent to a random walk on a reversible Markov chain. (According to **Definition 6.**)*
2) *A random walk on a reversible Markov chain is equivalent to a random walk on a weighted, undirected graph G. (According to **Lemma 5.**) Detailed proofs of **Lemma 5** and **Theorem 1** can be found in Appendix. A, (available online).*

Theorem 2. *Let $\mathcal{G}_{\text{de}} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}, \gamma\}$ be a hypergraph with edge-dependent weights. Then, there does not exist a weighted, undirected graph G such that a random walk on G is equivalent to a random walk on $\mathcal{G}_{\text{de}}$.*

Proof of Theorem 2. *Fig. 5 provides an example that a random walk on $\mathcal{G}_{\text{de}}$ is not equivalent to a random walk on a reversible Markov chain. Thus, according to the second step of **Theorem 1**'s proof, **Theorem 2** holds.*

Then, we provide an illustrative example for an easier understanding as shown in Fig. 5. The two hypergraphs have the same connection structure but different connection intensities. At first, the transition probability $p_{v,u}$ can be computed accordingly for two types of hypergraphs. Then, we start two random walks from vertex v_0: "$v_0 \to v_1 \to v_2 \to v_0$" and "$v_0 \to v_2 \to v_1 \to v_0$". The accumulated transition probability from the two paths can be computed by $p_{v_0,v_1}\cdot p_{v_1,v_2}\cdot p_{v_2,v_0}$ and $p_{v_0,v_2}\cdot p_{v_2,v_1}\cdot p_{v_1,v_0}$, respectively. According to **Theorem 1** and **Lemma 5**, random walks on this hypergraph are reversible. Thus, we can get the same accumulated transition probability from the two reversible paths. In contrast, we will get two different accumulated transition probabilities from two reversible paths in the hypergraph with edge-independent vertex weights.

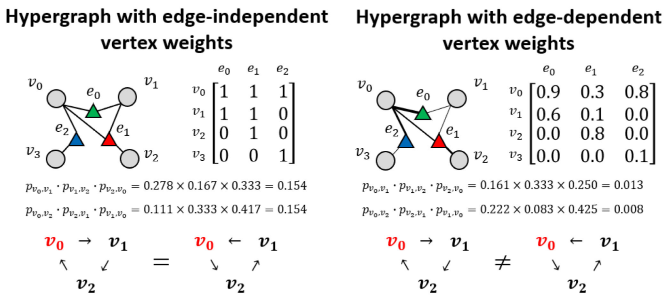

Hypergraph with edge-independent vertex weights
Hypergraph with edge-dependent vertex weights
v_0 v_1 v_2 v_3 e_0 e_1 e_2
$\begin{array}{c|ccc} & e_0 & e_1 & e_2 \\ v_0 & 1 & 1 & 1 \\ v_1 & 1 & 1 & 0 \\ v_2 & 0 & 1 & 0 \\ v_3 & 0 & 0 & 1 \end{array}$
$\begin{array}{c|ccc} & e_0 & e_1 & e_2 \\ v_0 & 0.9 & 0.3 & 0.8 \\ v_1 & 0.6 & 0.1 & 0.0 \\ v_2 & 0.0 & 0.8 & 0.0 \\ v_3 & 0.0 & 0.0 & 0.1 \end{array}$
$p_{v_0,v_1}\cdot p_{v_1,v_2}\cdot p_{v_2,v_0} = 0.278\times0.167\times0.333 = 0.154$
$p_{v_0,v_2}\cdot p_{v_2,v_1}\cdot p_{v_1,v_0} = 0.111\times0.333\times0.417 = 0.154$
$p_{v_0,v_1}\cdot p_{v_1,v_2}\cdot p_{v_2,v_0} = 0.161\times0.333\times0.250 = 0.013$
$p_{v_0,v_2}\cdot p_{v_2,v_1}\cdot p_{v_1,v_0} = 0.222\times0.083\times0.425 = 0.008$
=
≠

Fig. 5. Examples of two types of hypergraphs.

5.2 HGNN/HGNN$^+$ vs. GNN

GNN represents the classical convolution operator designed on graphs, such as [1], [2], [3], [4], [5]. In this subsection, we compare the HGNN and HGNN$^+$ with GNN from the spectral perspective and spatial perspective, respectively.

The comparison of HGNN and GNN from the spectral perspective. We prove that the GNN can be a special case of the HGNN mathematically. Assuming that each hyperedge only connects two nodes and has the same weight as others, the simple hypergraph (2-uniform hypergraph) can also be denoted by a graph with graph adjacency matrix $\mathbf{A}$ and vertex degree matrix $\mathbf{D}$, which can refer to the $\mathcal{E}_{\text{pair}}$ construction in Eq. (3). The corresponding hypergraph can be denoted by hypergraph incidence matrix $\mathbf{H}$, vertex degree matrix $\mathbf{D}_v$, hyperedge degree matrix $\mathbf{D}_e$, and hyperedge weight matrix $\mathbf{W}$. Under this circumstance, we have the following formulations to reduce the simple hypergraph:

$$\begin{cases} \mathbf{H}\mathbf{H}^\top & = \mathbf{A} + \mathbf{D} \\ \mathbf{D}_e^{-1} & = 1/2\mathbf{I} \\ \mathbf{W} & = \mathbf{I} \end{cases}. \tag{23}$$

Then, the hypergraph convolution defined in the conference version can be reduced as follows:

$$\begin{aligned} \mathbf{X}^{t+1} &= \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^t\mathbf{\Theta}^t) \\ &= \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\left(\frac{1}{2}\mathbf{I}\right)\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^t\mathbf{\Theta}^t\right) \\ &= \sigma\left(\frac{1}{2}\mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{D})\mathbf{D}^{-1/2}\mathbf{X}^t\mathbf{\Theta}^t\right) \\ &= \sigma\left(\frac{1}{2}(\mathbf{I}+\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{X}^t\mathbf{\Theta}^t\right) \\ &= \sigma(\mathbf{D}^{-1/2}\hat{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{X}^t\hat{\mathbf{\Theta}}^t), \end{aligned} \tag{24}$$

where $\hat{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\hat{\mathbf{\Theta}}^t = \frac{1}{2}\mathbf{\Theta}^t$. The extra $\frac{1}{2}$ can be absorbed by the learnable parameter $\mathbf{\Theta}$. We find that in modeling the simple graph, the spectral-based hypergraph convolution in the conference version has the same formation as the graph convolution in GCN [2]. Therefore, the hypergraph convolution not only inherits the powerful

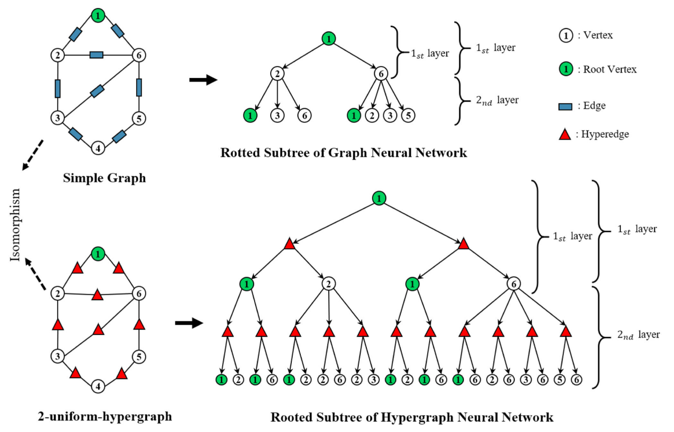


Fig. 6. Rooted subtree of graph and 2-uniform hypergraph. The correlation among six vertices is shown in left, which is represented by graph and two-uniform hypergraph, respectively. Note that for fair comparison the selected graph and two-uniform hypergraph own the same connection structure. In the right part shows the rooted subtree of graph and two-uniform hypergraph from vertex v_1, which reveals the message passing path in multi-layer GNN/HGNN$^+$.

expressive ability from GCN in handling the simple graph but also has the ability to model and learn the high-order correlation in the hypergraph.

The comparison of HGNN$^+$ and GNN from the spatial perspective. A powerful GNN model can be viewed as learning to embed the rooted subtree to low-dimensional space [17]. Rooted subtree [24] can describe not only the connection of local vertices, but also the message passing path in a graph. Therefore, we use the rooted subtree to compare HGNN$^+$ with GNN. Note that to satisfy the definition of path (also called the message passing path) in hypergraph, the node in hypergraph's rooted subtree can either be vertex or hyperedge.

Isomorphic graph structure can make a more clear comparison. Hence, we select 2-uniform hypergraph (each hyperedge only connects two vertices) for comparison. Fig. 6 depicts the rooted subtree of HGNN$^+$ and GNN for a specified vertex, which can also be expressed as the message path in graph and hyperpath in hypergraph. Obviously, the vertices in graph convolution only consider their neighbors' features, and then aggregate them to update the central vertex feature in one stage. Different from graph convolution layer, the HGNN$^+$ can be described as a hierarchical structure, which endows more powerful expression and modeling ability. HGNN$^+$ performs a two-stage i.e., vertex-hyperedge-vertex, transformation. As formulated in Eq. (18), hyperedge feature is generated according to its vertex inter-neighbor in the first stage. Then, the updated vertices features are obtained by aggregating their hyperedge inter-neighbor's features. Besides, compared with graph convolution, multi-layer hypergraph convolution has much more message interaction process. In HGNN$^+$, the rooted vertex appears more frequency in the path of subtree (like a latent extra self-loop), which is the main reason why HGNN$^+$ performs better in Ablation Study (Section 6.2.4 comparison on different convolutional strategies). Therefore, the hypergraph convolution layer can efficiently extract both low-order and high-order correlation on hypergraph by the vertex-hyperedge-vertex transformation compared with graph convolution.

5.3 HGNN$^+$ vs. HGNN

In this journal version, we further extend the HGNN from two aspects: hypergraph modeling and hypergraph convolution. Regarding hypergraph modeling, HGNN fuses the correlations within multiple hypergraphs simply by concatenating incidence matrices of them. By contrast, HGNN$^+$ further conceptually propose "hyperedge group" to adapt for multifarious information. Besides, HGNN$^+$ presents an adaptive strategy for the fusion of different hyperedge groups when generating the overall hypergraph representations to better utilize the complementarities among multifarious information features. To sum up, HGNN$^+$ promotes the modeling of complex relationships from hyperedge-by-hyperedge concatenation to group level adaptive fusion, improving not only the extensibility but also the representation capability.

With respect to the convolution, HGNN is derived from the spectral theory of hypergraphs, which leads to its monotonous form of expression and limitation in extensibility. Specifically, convolution in the spectral domain does not apply to the directed graph (hypergraph) as the graph Fourier Basis is the eigenvectors of the graph (hypergraph) Laplacian matrix and the Laplacian matrix it uses is defined on the undirected graph. In contrast, HGNN$^+$ defines hypergraph convolution in a more flexible way: designing a discrete and two-stage hypergraph convolution based on message passing from spatial domain. In this way, the convolution and aggregation operation in each stage can be defined flexibly and extended to the directed hypergraph naturally.

Definition 7. *Let $\mathcal{F}: \mathbf{X} \to \mathbf{X}'$ be a graph/hypergraph message passing layer with trainable parameter $\mathbf{\Theta} \in \mathbb{R}^{C \times C'}$, where $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$, $x_i \in \mathbb{R}^C$ and $\mathbf{X}' = \{x'_1, x'_2, \cdots, x'_N\}$, $x'_i \in \mathbb{R}^{C'}$ denotes the input/output vertex feature, respectively. $\mathcal{F}$ is said to be a symmetric message passing layer $\Leftrightarrow$ there exists a symmetric matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ such that $\mathcal{F} = \mathbf{AX\Theta}$.*

$$\begin{cases} \mathbf{X^{t+1}} \xleftarrow{\text{HGNNConv}} \sigma(\underline{\mathbf{D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2}}} \mathbf{X^t \Theta^t}) \\ \mathbf{X}^{t+1} \xleftarrow{\text{HGNNConv}^+} \sigma(\underline{\mathbf{D}_v^{-1} \mathbf{HWD}_e^{-1} \mathbf{H}^\top} \mathbf{X}^t \mathbf{\Theta}^t) \end{cases} \tag{25}$$

$\mathbf{D}_v^{-1/2}\mathbf{HWD}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$ and $\mathbf{D}_v^{-1}\mathbf{HWD}_e^{-1}\mathbf{H}^\top$ in Eq. (28) of HGNNConv/HGNNConv$^+$ play a role of "dummy adjacency matrix" to guide the vertex feature self-updating just like the adjacency matrix in graph convolution. However, in HGNNConv, the "dummy adjacency matrix" is symmetric normalized by $\mathbf{D}_v^{-\frac{1}{2}}$, which leads to a symmetric format of matrix $\mathbf{D}_v^{-\frac{1}{2}}\mathbf{HWD}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}}$ with **Definition 7**. Then, the symmetric "dummy adjacency matrix" of a hypergraph guides a symmetric message passing of vertex feature distribution and aggregation. In contrast, HGNNConv$^+$ allows both symmetric and asymmetric processes of passing messages from vertex to hyperedge (**Stage 1**) and from hyperedge to vertex (**Stage 2**). That is to say, HGNNConv is the symmetric message passing, while HGNNConv$^+$ is the asymmetric message passing.

On Directed Hypergraph. As analyzed above, our proposed HGNN$^+$ can be applied to the directed hypergraph, while HGNN is limited to the undirected case. Here we show, using a simple analytical example, how asymmetric message passing is extended to directed hypergraphs. We first give a trivial definition [23] for directed hypergraph $\hat{\mathbf{H}}$ with entries being defined as:

$$\hat{\mathbf{H}}(v,e) = \begin{cases} -1 & \text{if } v \in T(e) \\ 1 & \text{if } v \in S(e) \\ 0 & \text{otherwise} \end{cases}, \tag{26}$$

in which $T(e)$ and $S(e)$ are the set of target and source vertices for hyperedge e, respectively. Before performing HGNNConv$^+$, we split the incidence matrix $\hat{\mathbf{H}}$ into two matrices, $\hat{\mathbf{H}}_{\mathrm{s}}$ and $\hat{\mathbf{H}}_{\mathrm{t}}$, describing the source and target vertices for all hyperedges, respectively. These two incidence matrices play a crucial role in keeping the directional information during the message passing. Unlike that in the undirected hypergraph, the message passing in the directed hypergraph is guided by different incidence matrix, i.e., $\hat{\mathbf{H}}_{\mathrm{s}}$ and $\hat{\mathbf{H}}_{\mathrm{t}}$, in two stages of HGNNConv$^+$. In this case, an intuitive paradigm to perform HGNNConv$^+$ on the directed hypergraph can be simply denoted as:

$$\mathbf{X}^{t+1} = \sigma(\underline{\mathbf{D}_t^{-1}\hat{\mathbf{H}}_t\mathbf{W}\mathbf{D}_s^{-1}\hat{\mathbf{H}}_s^{\top}}\mathbf{X}^t\mathbf{\Theta}^t). \tag{27}$$

$\mathbf{D}_{\mathrm{s}}$ and $\mathbf{D}_{\mathrm{t}}$ are two matrices that conduct average aggregation in the message passing of stage 1 and 2, denoted as

$$\begin{cases} \mathbf{D}_{\mathrm{s}} = \mathbf{diag}(\mathbf{col\_sum}(\hat{\mathbf{H}}_{\mathrm{s}})) \\ \mathbf{D}_{\mathrm{t}} = \mathbf{diag}(\mathbf{col\_sum}(\hat{\mathbf{H}}_{\mathrm{t}})) \end{cases}, \tag{28}$$

in which $\mathrm{diag}(v)$ converts a vector v to a diagonal matrix and $\mathrm{col\_sum}(\mathbf{A})$ returns a vector with each element being the sum of the corresponding column in matrix $\mathbf{A}$. HGNNConv$^+$ can be applied to directed hypergraphs without losing directional information, while the HGNNConv that uses symmetric normalization will obviously ignore such key information regarding directed/asymmetric edges and thus limit its learning capacity compared with HGNNConv$^+$.

6 Experiments and Discussions

To evaluate the performance of the proposed hypergraph neural network framework, three types of experiments are conducted. The first two experiments are designed for the data with and without the graph structure. The last one is designed for the data with hypergraph structure. In the following, we introduce experimental settings, results, comparisons, and discussions, respectively.

6.1 Experimental Settings

6.1.1 Compared Methods

Seven typical state-of-the-art methods, including spectral-based (GCN [2], GraphConv [28]), non-spectral-based (GraphSAGE [1], GAT [3], and GIN [17]), and hypergraph-based methods (Hyper-Atten[7] and HyperGCN[8]), are selected for comparison. HGNN[6] and HGNN$^+$ are the proposed methods.

GCN [2]. As an efficient spectral network, it realizes convolution of irregular data though a spectral representation of graphs, and avoids overfitting on local neighborhood structures in graphs featuring wide distributions of node degree.

GraphSAGE [1]. This is a general inductive non-spectral method. Instead of adopting full graph Laplacian to generate embeddings, GraphSAGE utilizes learnable aggregation functions to improve receptive filed expansion and reduce the computing complexity.

GAT [3]. GAT introduces a self-attention strategy in the propagation step to improve the efficiency of weight allocation of the nodes in a single neighborhood.

GIN [17]. GIN is a simple yet powerful GNN, which shows superior performance against other GNN variants. It leverages a sum aggregator and further introduces multilayer perceptrons (MLPs) to parameterize universal multiset functions, generalizing the Weisfeiler-Lehman (WL) graph isomorphism test and achieving maximum discriminative power among GNNs.

GraphConv [28]. Based on the k-dimensional Weisfeiler-Leman algorithm, GraphConv proposes k-dimensional GNNs (k-GNNs), and therefore it can capture higher-order graph structures at different scales.

HyperGCN [8]. HyperGCN trains a GCN on hypergraphs for semi-supervised learning based on the spectral theory of hypergraphs.

Hyper-Atten [7]. It follows the convolution pattern defined in [6] and further introduces a hyperedge-vertex attention mechanism to adaptively learn the importance of different vertices in the same hyperedge.

HGNN and HGNN$^+$. Our methods of the initial conference version and this extended journal version.

6.1.2 Other Common Settings

Training Details. For comparing methods, the reported model configurations are used. The learning rate on three citation network datasets is set as 0.01 and 0.001 on two social media network datasets for all methods. The dropout is set as 0.5. Adam optimizer is employed to minimize the cross-entropy loss on classification task. The weight decay is set as 0.0005.

Loss Functions. For the single-label classification task, the commonly-used cross entropy loss is used for optimization $\mathcal{L} = -\sum_{v\in V}\sum_{c=1}^{C} Y_{vc}\log(O_{vc})$, where C denotes the number of categories and V is the vertex set with size N. $Y \in \{0,1\}^{N\times C}$ is the ground truth vertex label encoded using a one-hot encoding schema. $O \in \mathbb{R}^{N\times C}$ is the output of the softmax layer. For the multi-label classification task, the binary cross entropy loss function is adopted.

Evaluation Metrics. For evaluation, four widely-used metrics, including accuracy (Acc), macro f1 score (F1_ma), exact match ratio (EMR), and example-based accuracy (EB-Acc) are calculated to comprehensively compare the performance of different methods. The former two metrics are used in single-label classification task, and the latter two are adopted for multi-label classification task.

6.2 Vertex Classification on the Data With Graph Structure

6.2.1 Datasets

Five public benchmarks are selected in this experiment, which belong to two categories, i.e., publication citation

TABLE 2
Detailed Information of Five Datasets With Graph Structure

	Citation Network			Social Media Network	
Dataset	Cora	Citeseer	PubMed	Github Web ML	Facebook Page-Page
Classes	7	6	3	2	4
Nodes	2708	3327	19717	37700	22470
Edges	5429	4732	44338	289003	171002
Features	1433	3703	500	4005	4714

network (Cora [25], Citeseer and Pubmed [26]) and social media network (Github Web ML and Facebook [27]). In citation network dataset, each paper is indicated with a vertex associated with an initial feature, which corresponds to a bag of words. The task is to predict which category each paper belongs to. Social media dataset contains relations among web sites, and vertex features are extracted from words of each site. The statistical characteristics of datasets with graph structure are shown in Table 2.

Citation Network. The three widely applied citation network datasets, including Cora [25], CiteSeer and PubMed [26], are composed of sparse bag-of-words feature vectors for each scientific publication, with citation relationships among publications represented by corresponding edges, and ground truth topics as their labels.

Social Media Network. The two selected social media network datasets are GitHub Web ML [27] and Facebook Page-Page [27]. The Github Web and Machine Learning Developers Dataset (GitHub Web ML) constructs a graph describing developers who have starred at least 10 repositories and their relationships, with each node in the graph representing a developer, and each edge between two nodes representing the follower relationships between two developers. The node features are extracted with user information including their locations, employers and the repositories stared at. Therefore, the prediction of whether the working field of a developer is web or machine learning is converted into a bi-classification task of graph nodes. The Facebook Page-Page dataset contains 22,470 verified Facebook sites collected with Facebook Graph API and a graph describing inter-site relationships, with each node representing an official Facebook page and labelled one of the four following categories: politicians, governmental organizations, television shows, and companies. The node features are extracted with page owners' generalization of page themes. One edge is established between each two nodes represented by mutual like sites.

6.2.2 Settings

Data pre-processing. The raw vertex feature is a binary matrix with dimension $N \times C$. For three citation network datasets, each non-zero entry denotes whether a specified word appears in the publication. Following the setting in [29], the row-wise normalization is applied to the feature of each vertex. For two social media network datasets, each non-zero entry denotes that a specified attribute is associated with the website/user. These attributes can be the location of developers, the same users they follow or other kinds of shared data. Different from the vertex feature in citation network (including that uses only words), the vertex feature in social network is more complex. Under such circumstances, using the row-wise normalization directly may be unfair for different attributes. Therefore, raw vertex feature without row-wise normalization is used for two social network datasets.

Train/validation/test Split. For each dataset, 5/10 samples for each category are randomly selected for training and 5 samples for each category are randomly selected for validation. The rest of the vertices are used for testing for all datasets in our experiments. For each experiment, the validation set is used to select the best model in the training stage. Different from the conference version, the training/validation/testing data split process repeats 20 times for different methods, and the average performances for each method are reported for a fair comparison.

Hypergraph Construction. For our proposed method HGNN$^+$, three types of hyperedge groups $\mathcal{E}_{\text{pair}}$ (Eq. (3)), $\mathcal{E}_{\text{hop}_1}$ (Eq. (4)), and $\mathcal{E}_{\text{hop}_2}$ (Eq. (4)) are used for hypergraph generation and **Adaptive Fusion** strategy is adopted for hyperedge groups fusion. Here, two convolutional layers are adopted for generating the refined embeddings, and then the output is fed into a softmax layer to predict the probability distribution over all categories for each vertex. The hidden dimension is fixed to 64 for all datasets.

6.2.3 Experimental Results and Discussions

Experimental results on all five datasets are provided in Tables 3 and 4. From these results we can have the following observations:

1) HGNN$^+$ achieves better or comparable performance compared with GCN, GAT and other graph-based methods. For example, HGNN$^+$ obtains gains of 6.6%, 8.49%, 8.17%, 9.81%, and 10.55% compared with GCN, GAT, GraphSAGE, GIN, and GraphConv, in terms of Acc when 5 samples are used for training, on the Facebook dataset.
2) Compared with other hypergraph-based methods, i.e., HyperGCN and Hyper-Atten, our proposed HGNN$^+$ can also yields consistent better performance. In particular, HGNN$^+$ outperforms HyperGCN and Hyper-Atten by 5.02% and 1.23% in terms of F1_ma when 5 samples are used for training, on the Github Web ML dataset.
3) With fewer training samples, the proposed method HGNN$^+$ can achieve more gains, which shows that the proposed methods can work well with limited training samples.
4) The proposed HGNN$^+$ achieves a more remarkable performance improvement against graph-based methods on the social media network data than that on the publication network data.

The better performance of the proposed framework can be attributed to the following factors. First, compared with the graph structure which can only represent pairwise correlation, the hypergraph structure is able to deeply exploit the high-order correlation among the data, even behind the simple pairwise correlation. As shown in the hypergraph generation procedure of our proposed framework, the hyperedge group using pairwise edge corresponds to the

TABLE 3
Experimental Results When Using 5 Samples Per Category for Training

Methods	Cora		Citeseer		Pubmed		Github Web ML		Facebook	
	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma
GCN	0.6906	0.6765	0.5862	0.5398	0.6938	0.6899	0.7218	0.6679	0.5771	0.5462
GAT	0.7026	0.6911	0.6021	0.5510	0.7051	0.7038	0.7296	0.6641	0.5582	0.5362
GraphSAGE	0.6794	0.6648	0.5648	0.5247	0.6840	0.6806	0.7278	0.6718	0.5614	0.5405
GIN	0.6812	0.6675	0.5540	0.5195	0.6828	0.6797	0.7201	0.6428	0.5450	0.5194
GraphConv	0.6709	0.6571	0.5539	0.5219	0.6854	0.6800	0.6744	0.6150	0.5376	0.5140
HyperGCN	0.6951	0.6819	0.5890	0.5387	0.6976	0.6949	0.7256	0.6707	0.5775	0.5564
Hyper-Atten	0.7269	0.7118	0.5959	0.5484	0.7093	0.7050	0.7552	0.7086	0.6430	0.6179
HGNN	0.7143	0.6999	0.5471	0.5193	0.6973	0.6910	0.6872	0.6405	0.5520	0.5303
HGNN$^+$	**0.7319**	**0.7141**	**0.6064**	**0.5486**	**0.7138**	**0.7053**	**0.7637**	**0.7209**	**0.6431**	**0.6204**

The best results are marked in bold type, and the second-best results are marked with underline.

graph structure, while the hyperedge group using k-Hop neighbors can further represent the second order and even higher order reachable neighbors in the graph structure for each vertex. These higher order correlations are beyond the traditional pairwise relationship and yet lead to better representation of the latent data relationship. More importantly, such hypergraph structure can be very flexible to deal with multi-modal/multi-type representations, which is effective when handling heterogeneous data.

From the aspect of correlation representation, the **graph-based methods** are described by the adjacency matrix. Although the adjacency matrix can concisely depict the connection structure of a graph, the complex higher-order information (connected component, k-order reachable neighbors, *etc.*) on the graph cannot be obtained directly. In most cases, both the information explicitly presented by the adjacency matrix (the connection structure of the graph) and the information implicitly expressed by the adjacency matrix (the complex higher-order information on the graph) is helpful for the representation learning of vertices on the graph. **Existing methods based on the GNN only leverage the information that is explicitly presented by the adjacency matrix, hoping to sideways capture the high-order correlations on the graph for further representation learning by stacking multiple layers. However, stacking multiple layers of GCN may fail into the trap of rigid k-hop neighborhood smoothing.** For example, each vertex's feature in the output layer of a **three-layer** GCN is generated by uniformly smoothing its **3-hop** neighbor's features. In fact, the neighborhood information of different hops contributes differently to the learning task in different datasets, which is also verified in the ablation study on different hyperedge groups shown in Table 5. Thus, the GNNs with fixed number of layers may produce sub-optimal performance in different datasets. By contrast, our proposed framework can explicitly describe different "modality/hop" information on the graph by defining multiple hyperedge groups as well as introducing the weights of hyperedge groups to balance the influence of different higher-order information on vertex representation learning.

As for **the hypergraph-based methods**, the poor performance can be dedicated to their structural simplification or over-parametrization. HyperGCN bears unsatisfactory performance, probably due to the fact that it simplifies the initial hypergraph structure to perform graph convolution on hypergraphs. Such simplification is irreversible and thus will certainly lose crucial information. Hyper-Atten leverages the hyperedge-vertex attention module and therefore yields better performance compared with HyprGCN. Nonetheless, such a complex attention strategy introduces a large number of parameters and makes the model easily suffer from the overfitting problem. In contrast, HGNN$^+$ proposes the hyperedge group-level attention mechanism, which can consider the relationships among different hyperedge groups and vertices as well as cut down the volume of learnable parameters, thus effectively preventing overfitting and achieving more stable improvements.

The better performance gains from the cases with less labeled data indicates that the proposed framework can be more effective to deal with few labeled sample situation.

TABLE 4
Experimental Results When Using Ten Samples Per Category for Training

Methods	Cora		Citeseer		Pubmed		Github Web ML		Facebook	
	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma
GCN	0.7632	0.7491	0.6383	0.5972	0.7305	0.7284	0.7737	0.7261	0.6577	0.6398
GAT	0.7670	0.7485	0.6472	0.6086	0.7378	0.7356	0.7739	0.7258	0.6382	0.6201
GraphSAGE	0.7325	0.7168	0.5932	0.5570	0.7230	0.7195	0.7598	0.7151	0.6413	0.6258
GIN	0.7319	0.7209	0.5964	0.5632	0.7259	0.7219	0.7459	0.6934	0.6135	0.5966
GraphConv	0.7314	0.7205	0.5931	0.5632	0.7164	0.7130	0.6862	0.6491	0.5988	0.5772
HyperGCN	0.7586	0.7451	0.6411	0.6015	0.7309	0.7282	0.7876	0.7375	0.6514	0.6377
Hyper-Atten	0.7684	0.7536	0.6398	0.5973	0.7336	0.7277	0.7865	0.7409	0.6961	0.6827
HGNN	**0.7718**	**0.7579**	0.6399	0.5945	0.7287	0.7235	0.7426	0.6957	0.6205	0.6055
HGNN$^+$	0.7671	0.7496	**0.6643**	**0.6196**	**0.7408**	**0.7327**	**0.7910**	**0.7500**	**0.6997**	**0.6853**

The best results are marked in bold type, and the second-best results are marked with underline.

TABLE 5
Ablation Study: Comparison on the Effectiveness of Different Hyperedge Groups With HGNN$^+$

	Hyperedge Groups				Citeseer		Cora		Facebook		Github Web ML		Pubmed	
	pair	hop-1	hop-2	hop-3	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma	Acc	F1_ma
Coequal	✓	✓	✓	✓	0.6009	0.5438	0.6515	0.6387	0.5956	0.5781	OOM	OOM	0.6481	0.6403
	✓	✓	✓		0.6141	0.5543	0.7216	0.7049	0.6332	0.5946	0.7009	0.6248	0.6974	0.6836
	✓	✓			0.5946	0.5523	0.7066	0.6864	**0.6441**	0.6128	**0.7887**	**0.7367**	0.7060	0.7033
	✓		✓		**0.6142**	**0.5613**	0.7182	0.6938	0.6275	0.5976	0.7040	0.6632	0.6968	0.6898
	✓				0.5876	0.5428	0.6903	0.6735	0.5767	0.5324	0.7219	0.6583	0.6928	0.6882
Adaptive	✓	✓	✓		0.6064	0.5486	**0.7319**	**0.7141**	0.6431	**0.6204**	0.7637	0.7209	**0.7138**	**0.7053**

The best results are marked in bold type, and the second-best results are marked with underlining. The OOM denotes the Out of Memory.

When the labeled data is limited, which is the case in most applications, the data correlation plays more important role in the representation learning process. Therefore, a good and sufficient data correlation could help a lot in the task.

We attribute the significant improvement that HGNN$^+$ achieves on the social media network data to the rich high-order correlations behind the data. The social media network data encode diverse information such as address and hobby as vertex features, while the publication network data simply uses word one-hot encoding as vertex features. In addition to the explicit "follow" associations, there are plenty of implicit complex relationships (*e.g.*, communities of interest) in the social media network data, remains to be captured and learned by HGNN$^+$. These informative vertex features and complex vertex relationships enable the HGNN$^+$ to achieve more remarkable improvements compared with graph-based methods. We must note that the current extension is just a simple implementation of using this graph structure data in our framework. Other hyperedge generation methods can be also used in applications.

6.2.4 Ablation Experiments

In this subsection, three ablation studies are conducted. First, we compare HGNN$^+$ with HGNN in Table 6. Then in Table 5 we demonstrate the effectiveness of the proposed adaptive fusion strategies. Finally, we conduct experiments to compare different convolutional strategies, i.e., GCNConv and HGNNConv$^+$, and experimental results are shown in Table 7.

Comparison with HGNN. For better comparison, we split hypergraph neural network into two parts: hypergraph structure and hypergraph convolution. In this case, HGNN is composed of hop-1 (structure) and HGNNConv (convolution), while HGNN$^+$ is composed of adaptive fusion from multiple hyperedge groups (structure) and HGNNConv$^+$ (convolution). In Table 6, **Adaptive** means using default hyperedge group configuration and adaptive fusion. As shown in Table 6, the combination of adaptive and HGNNConv$^+$ outperforms HGNNConv and hop-1 in all three datasets. When using the same hypergraph structure hop-1/adaptive, HGNNConv$^+$ yields better performance than HGNNConv. The reason lies in that when derived from spectral domain, HGNNConv uses chebyshev polynomials to reduce computation cost, which may lead to inaccurate structure representation. In contrast, HGNNConv$^+$ directly adopts inherent hypergraph for message passing in representation learning without simplification and thus showing better performance.

Overall, the HGNN framework only constructs hyperedges via linking vertex with its 1-hop neighborhoods for final hypergraph. The HGNN$^+$ framework further proposes hyperedge group concept for hypergraph modeling, which not only uses 1-hop correlation but also includes other complex correlations. Each hyperedge group can be built to describe one type of correlation. In HGNN$^+$ framework, multiple hyperedge groups $\left(\mathcal{E}_{\text{pair}}, \mathcal{E}_{\text{hop}_k}, \mathcal{E}_{\text{attribute}}, \mathcal{E}_{\text{feature}}^{\text{KNN}_k}, \mathcal{E}_{\text{feature}}^{\text{distance}_d}\right)$ can be built toward different correlation description. From the other aspect, the HGNN framework only use $\mathcal{E}_{\text{hop}_1}$ for hypergraph construction. Besides, the HGNN$^+$ framework designs an adaptive multiple hyperedge groups fusion strategy to generate a uniform hypergraph, which encodes multiple types of high-order correlation, simultaneously. In summary, the HGNN$^+$ framework takes advantage of hypergraph in multi-modal data modeling, and explicit multiple correlations

TABLE 6
Comparison of HGNN and HGNN+

		HGNNConv		HGNNConv$^+$	
		hop-1	Adaptive	hop-1	Adaptive
Cora	**Acc**	0.7143	0.6990	0.7289	**0.7319**
	F1_ma	0.6999	0.6847	0.7151	**0.7141**
Citeseer	**Acc**	0.5741	0.6061	0.5828	**0.6064**
	F1_ma	0.5179	0.5449	0.5238	**0.5486**
Pubmed	**Acc**	0.6973	0.6938	0.7033	**0.7038**
	F1_ma	0.6910	0.6859	0.6983	**0.7053**

The best results are marked in bold type.

TABLE 7
Ablation Study: Comparison on Different Convolutional Strategies When Five Samples are Used for Training

	GCN (w/o self-loop)		HGNN$^+$ (pair only)	
	Acc	F1_ma	Acc	F1_ma
Citeseer	0.5544	0.5013	**0.5876**	**0.5428**
Cora	0.6799	0.6687	**0.6903**	**0.6735**
Facebook	0.4793	0.4564	**0.5767**	**0.5324**
Github	0.6338	0.5851	**0.7219**	**0.6583**
Pubmed	0.6706	0.6675	**0.6928**	**0.6882**

The best results are marked in bold type.

modeling makes it easier for the later hypergraph convolutions to capture the hidden patterns among data.

On Hyperedge Group. Compared with graph neural network, the proposed HGNN$^+$ framework utilizes hypergraph to represent data correlations, in which the hyperedge groups are the components to carry the high-order correlations. To investigate how such hyperedge groups work in the learning procedure, we conduct experiments with respect to different combinations of hyperedge groups, including $\mathcal{E}_{pair}$, $\mathcal{E}_{hop-1}$, $\mathcal{E}_{hop-2}$ and $\mathcal{E}_{hop-3}$. In the combination of hyperedge groups, all hyperedge groups share equal weights. The experimental results with respect to different hyperedge group combinations are shown in Table 5. From these results, we can have the following observations.

First, the method using just $\mathcal{E}_{pair}$, in which only the pairwise correlation is employed, performs worst in most cases. When high-order correlations, including $\mathcal{E}_{hop-1}$, $\mathcal{E}_{hop-2}$ and $\mathcal{E}_{hop-3}$, are employed, the performance could improve in most cases, which demonstrates the effectiveness of high-order correlation on representation learning.

Second, we can also notice that although the introducing of high-order correlation can improve the performance, there is no common pattern of hyperedge group combinations which can always perform best. It indicates that different high-order correlations may have varied impact on the representation learning. With the adaptive hyperedge weight optimization in the proposed framework, the performance can outperform or at least be close to that from the best hyperedge group combination, as shown in Table 5.

On Convolution Strategy. To evaluate the effectiveness of the proposed hypergraph convolution, we conduct experiments to compare GCN without self-loop and our HGNN$^+$ using just $\mathcal{E}_{pair}$, which only utilizes the pairwise correlation in the hypergraph structure, for fair comparison. The results on five datasets are provided in Table 7.

As shown in these results, the proposed HGNN$^+$ consistently outperforms GCN in all experiments on all evaluation metrics. Compared with the single layer graph convolution, a single layer hypergraph convolution is a hierarchical message passing (a vertex-hyperedge-vertex way referring to Section 5.2) that can handle information from multiple low-order/high-order correlation groups. Therefore, the proposed hypergraph convolution strategy is efficient on modeling the low-order structure with its hierarchical message passing strategy in each layer and thus leads to better performance.

6.3 Vertex Classification on the Data Without Graph Structure

In this part of the experiments, the classification task is conducted on the data which does not contain the explicit graph structure. These experiments aim to evaluate how the proposed hypergraph neural network framework can exploit data representation when there is no explicit graph structure, which is more common in real applications.

6.3.1 Datasets

Here, two public 3D object datasets are employed, including the ModelNet40 [40] dataset and the NTU [41] dataset. The ModelNet40 dataset consists of 12,311 3D objects from 40 popular categories, and the same training/testing split is applied as introduced in [40], where 9,843 objects are used for training and 2,468 objects are used for testing. The NTU dataset is composed of 2,012 3D shapes from 67 categories, including car, chair, chess, chip, clock, cup, door, frame, pen, plant leaf and so on. In the NTU dataset, 80% of data are used for training and the other 20% of data are used for testing. In this experiment, each 3D object is represented by the extracted features. Here, two 3D shape representation methods are employed, including Multi-view Convolutional Neural Network (MVCNN [42]) and Group-View Convolutional Neural Network (GVCNN [43]). These two methods are selected due to that they have shown satisfactory performance on 3D object representation. We follow the experimental settings of MVCNN and GVCNN to generate multiple views of each 3D object. Here, 12 virtual cameras are employed to capture views with an interval angle of 30 degrees, and then both the MVCNN and the GVCNN features are extracted accordingly.

6.3.2 Settings

In this part of the experiments, we combine the two features as the vertex feature for all methods. The compared methods and the evaluation metrics are the same as that in Section 6.2.

Graph/Hypergraph Construction. In experiments without graph structure (on ModelNet40 and NTU datasets), K Nearest Neighbor (KNN) algorithm is adopted for hypergraph construction. Specifically, we build two hyperedge groups $\mathcal{E}_{\mathrm{mvcnn}}^{\mathrm{KNN}_8}$ and $\mathcal{E}_{\mathrm{gvcnn}}^{\mathrm{KNN}_8}$ for MVCNN feature and GVCNN feature, respectively. The final hypergraph is constructed by fusing the two hyperedge groups with the coequal strategy. Different from the conference version, the graph structure is constructed by weighted hypergraph expansion (more details refer to Appendix B, available in the online supplemental material). This graph construction strategy is adopted to guarantee the same original connection structure for both graph-based methods and hypergraph-based methods for a fair comparison.

6.3.3 Experimental Results and Discussions

The experimental results on two datasets are provided in Table 8. As shown in these results, the proposed hypergraph neural network framework, i.e., HGNN$^+$ and HGNN, significantly outperforms all other compared methods. For example, HGNN$^+$ achieves gains of 3.75%, 4.02%, 3.48%, 4.02% and 3.22% compared with GCN, GAT, GraphSAGE, GIN and GraphConv in terms of Acc in the NTU dataset. Similar results can be observed in the ModelNet40 dataset.

When there is no explicit graph structure, the proposed method performs much better than traditional graph neural network methods. These results indicate that the proposed framework is able to deeply exploit the underneath high-order correlation by generating a hypergraph structure. The superior performance of HGNN$^+$ and HGNN can sufficiently demonstrate the effectiveness of our proposed method on the classification task using data without graph structure.

6.4 Vertex Classification on the Data With Hypergraph Structure

In this part of the experiments, two datasets, i.e., Cooking-200 and MovieLens2k-v2, where hypergraph structure

TABLE 8
Experimental Results on the ModelNet40 and NTU Dataset

	ModelNet40		NTU	
	Acc	F1_ma	Acc	F1_ma
GCN	0.9485	0.9276	0.8043	0.7688
GAT	0.9575	0.9268	0.8016	0.7511
GraphSAGE	0.9473	0.9188	0.807	0.7699
GIN	0.9485	0.9275	0.8016	0.7590
GraphConv	0.9566	0.9370	0.8096	0.7664
HyperGCN	0.9546	0.9410	0.8177	0.7677
Hyper-Atten	0.9611	0.9419	0.8150	0.7616
HGNN	0.9680	0.9520	0.8311	0.7798
HGNN$^+$	**0.9692**	**0.9577**	**0.8418**	**0.7944**

The best results are marked in bold type.

TABLE 9
Experimental Results on Cooking-200 and MovieLens2k-v2

	Expansion	Cooking-200		MovieLens2k-v2		
		Acc	F1_ma	EMR	EB-Acc	EB-Pre
GCN	unweighted	0.3110	0.2608	0.0686	0.2081	0.3096
	weighted	0.3271	0.2849	0.0814	0.233	0.3363
HGNN	–	0.4564	0.3725	0.1043	0.2482	0.3652
HGNN$^+$	–	**0.4785**	**0.3883**	**0.1214**	**0.2808**	**0.4175**

The best results are marked in bold type.

naturally exists are selected. The objective of these experiments is to evaluate the effectiveness of the proposed hypergraph convolution operator compared with the graph convolution operator on natural hypergraph data.

6.4.1 Datasets

The Cooking dataset is collected from Yummly.com[2], in which vertex denotes the dish and hyperedge denotes the ingredient. Each dish is also associated with category information, which indicates the dish's cuisine like Chinese, Japanese, French, and Russian. To avoid vertex's feature over-smoothing, we drop the ingredients that contribute more than 200 dishes and contribute only one dish. After preprocessing, the Cooking-200 (including 7403 dishes, 2755 ingredients, and 20 cuisines) is generated. The MovieLens2k-v2 is an extension of the MovieLens10M dataset, published by the GroupLeans research group[3]. It links the movies of the MovieLens dataset with their corresponding web pages at Internet Movie Database[4] (IMDB) and Rotten Tomatoes movie review systems[5]. The MovieLens2k-v2 dataset includes 2113 users, 10197 movies, 13222 tags (47957 tag assignments, i.e., tuples [user, tag, movie]), 4060 directors, and 20 movie genres (like Adventure, Animation, Comedy, and Crime). Due to each movie may associate with more than one genre, experiments on the MovieLens dataset are indeed **multi-label classification task**. The vertices in the two datasets do not associate with vertex features.

6.4.2 Settings

Considering that the identity matrix can identify each row (vertex feature) because of the orthogonality, the identity matrix is fed into the model as the initial vertex feature.

Train/validation/test Split. In both two datasets, we randomly sample 10 vertices for training and 10 vertices for validation for each category, and the rest vertices are used for testing.

Hypergraph Construction. In the Cooking-200 dataset, each dish consists of several ingredients, and each ingredient can be shared with several dishes. Thus, the ingredient naturally acts as the hyperedge, which connects those related dishes (vertices). Finally, a hypergraph with 7403 vertices and 2755 hyperedges is constructed in the Cooking-200 dataset. In the MovieLens2k-v2 dataset, the movie acts as the vertex, and the hyperedge can be built from the correlations of [movie, tag, weight] and [movie, director]. Consequently, two hyperedge groups — tag-based hyperedge group and director-based hyperedge group are constructed. To compare the learning ability of GCNConv, HGNNConv, and HGNNConv$^+$, the adaptive fusion strategy is not adopted here. We directly concatenate the two hyperedge groups to generate the final hypergraph for a fair comparison in the MovieLens2k-v2 dataset.

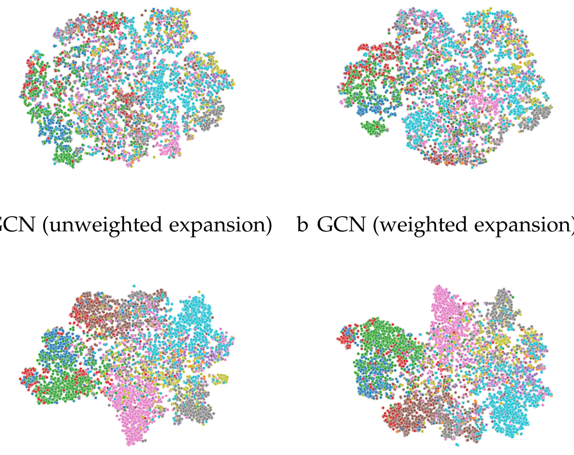
Hypergraph Expansion. Considering that the hypergraph incidence matrix cannot being directly handled by the graph neural network [2], two commonly used hypergraph expansion methods [33], [45], i.e., unweighted clique expansion and weighted clique expansion, are employed to transfer the hypergraph structure into the simple graph structure. More details are provided in Appendix B, available in the online supplemental material.

6.4.3 Experimental Results and Discussions

Experimental results are shown in Table 9. The proposed hypergraph-based methods HGNN and HGNN$^+$ exhibit significant improvement compared with the graph-based method GCN in natural hypergraph real-world datasets. Specifically, as for the single-label classification task, HGNN$^+$ achieves gains of 16.75% and 15.14% on Acc compared with unweighted and weighted expansion GCN in the Cooking-200 dataset, respectively. As for the multi-label classification task, HGNN$^+$ achieves 10.79% and 8.12% on EB-Pre compared with the unweighted and weighted expansion GCNs in the MovieLens2k-v2 dataset respectively.

From the experimental results , we can have the following observations. First, the weighted hypergraph expansion GCN performs better than the unweighted-based way. The main reason is that the unweighted hypergraph expansion only inherits the connection relationships among vertices from the original hypergraph. In contrast, the weighted expansion links vertices with different intensities according to the original hyperedges and thus reserves more high-order information. Moreover, the hypergraph-based methods consistently exhibit much better performance than graph-based methods. The main reasons are two-fold. First, although hypergraph expansion methods like clique expansion can transfer the high-order hypergraph structure into

2. https://www.yummly.com
3. https://grouplens.org/
4. https://www.imdb.com/
5. https://www.rottentomatoes.com/



a GCN (unweighted expansion) b GCN (weighted expansion)

c HGNN d HGNN^{+}

Fig. 7. A t-SNE visualization of graph-based methods and hypergraph-based methods on the Cooking-200 dataset.

the low-order graph structure so as to model the high-order correlations among data into a graph , it will still cause structure distortion and yield unsatisfactory performance, which also has been discussed in [46], [47], [48]. Besides, the HGNN and HGNN^{+} are inherently proposed to learn in hypergraph structure. Specifically, the two-stage message passing strategy $\mathcal{V} \rightarrow \mathcal{E}, \mathcal{E} \rightarrow \mathcal{V}$ of HGNN and HGNN^{+} can effectively capture the high-order information in hypergraph compared with the the one-stage message passing strategy $\mathcal{V} \rightarrow \mathcal{V}$ adopted in graph-based methods, thus leading to the better performance.

6.5 Visualization

Considering that there naturally exists hypergraph structure in the Cooking-200 dataset, indicating that it contains complex high-order correlations among data, the Cooking-200 dataset is selected for visualization to intuitively compare the learning ability of graph-based methods and hypergraph-based methods. The t-SNE method is utilized to visualize the output of the last-layer convolution and the results are shown in Fig. 7. It can be observed from the results that compared with graph-based methods, hypergraph-based methods like HGNN and HGNN^{+} yield discernible clustering, which qualitatively verifies the effectiveness of the proposed method.

7 THU-DeepHypergraph: An Open Toolbox of the HGNN^{+} Framework

To facilitate the development of hypergraph-based representation learning, we develop *THU-DeepHypergraph* (THU-DH), an open-source python toolbox for the general hypergraph neural network framework, which is built upon PyTorch. As shown in Fig. 8, THU-DH can be decoupled into three parts: Graph Learning, Hypergraph Learning, and Structure Transform. The graph learning part includes the graph sampling, graph pooling, and a general graph-based message passing framework. The Structure Transform part implements some typical hypergraph expansion algorithms like clique expansion and star expansion, which can reduce the high-order data into the low-order data for graph learning on hypergraph dataset.

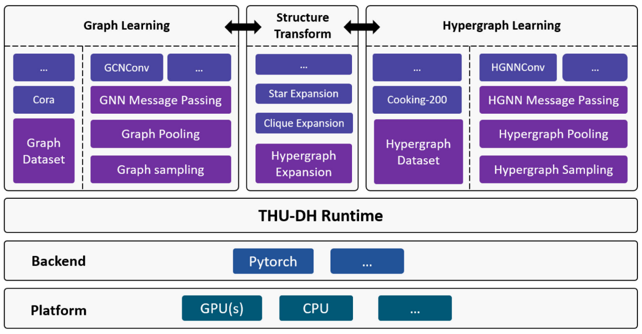


Fig. 8. A framework illustration of THU-DH toolbox.

As for the hypergraph learning, some hypergraph modeling algorithms have been integrated into the Hypergraph Dataset module. Specifically, the hypergraph modeling can be further divided into two modules, i.e., hyperedge group generation and fusion, as stated in Section 4. To simplify the construction of hypergraph, THU-DH provides common APIs for hyperedge group generation, including pairwise edge-based, k-Hop neighbors-based, attributes-based and features-based methods. THU-DH also supports two strategies for hyperedge group fusion: coequal fusion and adaptive fusion. Here we recommend the adaptive fusion strategy since it can ensure more robust performance improvements. For hypergraph convolution, the convolution operators in the spectral domain (HGNNConv) and the spatial domain (HGNNConv^{+}) have been implemented based on the HGNN Message Passing module in THU-DH. The Hypergraph Pooling module and Hypergraph Sampling module are designed for hypergraph learning on large scale hypergraph. Researchers can conveniently use this toolbox to handle various tasks, such as node classification, network classification, image segmentation, graph regression prediction, by assembling different modules according to the task characteristics. Data pre-processing algorithms included in the Hypergraph Dataset module are implemented for multiple tasks, such as pathological image sampling, magnetic resonance image enhancement, etc. We also provide some example codes and visualization tools to help researchers get started quickly and evaluate intuitively. We endeavor to make the toolbox more efficient. Specifically, sparse matrix techniques are used for the hypergraph representation and calculation, which greatly improves the running speed and decrease memory consumption.

8 Conclusion

In this journal version, we extend our previous HGNN work and introduce a general hypergraph neural network framework HGNN^{+} for representation learning. The proposed HGNN^{+} framework has advantage on modeling high-order data correlations from multi-modal/multi-type data. Four types of data correlation generation methods are introduced in this paper and an adaptive hyperedge fusion strategy is provided to generate the overall hypergraph. A hypergraph convolution in the spatial domain is introduced to learn the representation. Experiments on 9 datasets and comparisons with state-of-the-art methods demonstrate the effectiveness of our proposed methods. The results and mathematical discussions reveal that the proposed framework is able to achieve the new state-of-the-art performance, especially when there is no explicit data correlations. The proposed HGNN$^+$ framework can be used in various of applications, such as data classification, retrieval and recommendation. A tool of the proposed framework, called THU-DeepHypergraph, is released for public use.

References

[1] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[3] P. Veličković *et al.*, "Graph attention networks," in *Proc. Int. Conf. Mach. Learn.*, 2018.

[4] J. Gilmer *et al.*, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[6] Y. Feng *et al.*, "Hypergraph neural networks," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2019, pp. 3558–3565.

[7] S. Bai, F. Zhang, and P. H. S. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognit.*, vol. 110, 2021, Art. no. 107637.

[8] N. Yadati *et al.*, "HyperGCN: A new method for training graph convolutional networks on hypergraphs," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2019, pp. 1511–1522.

[9] J. Jiang *et al.*, "Dynamic hypergraph neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 2635–2641.

[10] R. Dharmarajan and K. Kannan, "Hyper paths and hyper cycles," *Int. J. Pure Appl. Math.*, vol. 98 no. 3, pp. 309–312, 2015.

[11] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2012.

[12] J. Bruna *et al.*, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014.

[13] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, 2011.

[14] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[15] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7134–7143.

[16] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.

[17] K. Xu *et al.*, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2018.

[18] H. E. Manoochehri, S. S. Kadiyala, and M. Nourani, "Predicting drug-target interactions using Weisfeiler-Lehman neural network," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Informat.*, 2019, pp. 1–4.

[19] X. Wang *et al.*, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.

[20] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1601–1608.

[21] Y. Wang, L. Zhu, X. Qian, and J. Han, "Joint hypergraph learning for tag-based image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 4437–4451, Sep. 2018.

[22] L. Xiao *et al.*, "A Hypergraph learning method for brain functional connectivity network construction from FMRI data," *Med. Imag.*, vol. 11317, 2020, Art. no. 1131710.

[23] G. Gallo *et al.*, "Directed hypergraphs and applications," *Discrete Appl. Math.*, vol. 42, no. 2/3, 1993, Art. no. 177.

[24] N. Shervashidze *et al.*, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 9, 2011, Art. no. 12.

[25] P. Sen *et al.*, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.

[26] G. Namata *et al.*, "Query-driven active surveying for collective classification," in *Proc. 10th Int. Workshop Mining Learn. Graphs*, 2012, Art. no. 1.

[27] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *J. Complex Netw.*, vol. 9, 2021, Art. no. cnab014.

[28] C. Morris *et al.*, "Weisfeiler and Leman go neural: Higher-order graph neural networks," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2019, pp. 4602–4609.

[29] O. Shchur *et al.*, "Pitfalls of graph neural network evaluation," in *Proc. Workshop RRL*, 2018.

[30] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[31] I. Amburg, N. Veldt, and A. Benson, "Clustering in graphs and hypergraphs with categorical edge labels," in *Proc. Int. World Wide Web Conf.*, 2020, pp. 706–717.

[32] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 387–388.

[33] S. Agarwal, K. Branson, and S. Belongie, "Higher order learning with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 17–24.

[34] A. Ducournau and A. Bretto, "Random walks in directed hypergraphs and application to semi-supervised image segmentation," *Comput. Vis. Image Understanding*, vol. 120, pp. 91–102, 2014.

[35] U. Chitra and B. Raphael, "Random walks on hypergraphs with edge-dependent vertex weights," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1172–1181.

[36] T. Carletti *et al.*, "Random walks on hypergraphs," *Phys. Rev. E*, vol. 101, 2020, Art. no. 022308.

[37] J. Li, J. He, and Y. Zhu, "E-tail product return prediction via hypergraph-based local graph cut," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 519–527.

[38] D. Aldous and J. Fill, "Reversible markov chains and random walks on graphs," 2002. [Online]. Available: http://statwww.berkeley.edu/users/aldous/RWG/book.html

[39] F. P. Kelly, *Reversibility and Stochastic Networks[M]*, New York, NY, USA: Cambridge Univ. Press. 2011.

[40] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.

[41] D. Y. Chen *et al.*, "On visual similarity based 3D Model retrieval," *Comput. Graph. Forum*, vol. 22, no. 3, pp. 223–232, 2003.

[42] H. Su *et al.*, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.

[43] Y. Feng *et al.*, "GVCNN: Group-view convolutional neural networks for 3D shape recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 264–272.

[44] C. Yang *et al.*, "Hypergraph learning with line expansion," 2020, *arXiv:2005.04843*.

[45] J. Y. Zien, M. D. F. Schlag, and P. Chan K., "Multilevel spectral hypergraph partitioning with arbitrary vertex sizes," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 9, pp. 1389–1399, Sep. 1999.

[46] I. E. Chien, H. Zhou, and P. Li, "HS^2: Active learning over hypergraphs with pointwise and pairwise queries," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2466–2475.

[47] P. Li and O. Milenkovic, "Submodular hypergraphs: P-Laplacians, cheeger inequalities and spectral clustering," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3014–3023.

[48] M. Hein *et al.*, "The total variation on hypergraphs-learning on hypergraphs revisited," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, Art. no. 26.

[49] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, Art. no. 29.

Yue Gao (Senior Member, IEEE) received the BS degree from the Harbin Institute of Technology, Harbin, China, and the ME and PhD degrees from Tsinghua University, Beijing, China, where he is currently an associate professor with the School of Software.

Yifan Feng received the BE degree in computer science and technology from Xidian University, Xi'an, China, in 2018, and the MS degree from Xiamen University, Xiamen, China, in 2021. He is currently working toward the PhD degree with the School of Software, Tsinghua University, Beijing, China.

Shuyi Ji received the BE degree from the School of Software, Tsinghua University, Beijing, China, in 2019, where she is currently working toward the PhD degree in machine learning.

Rongrong Ji (Senior Member, IEEE) is currently a professor and the director with the Intelligent Multimedia Technology Laboratory, School of Informatics, Xiamen University, Xiamen, China. His research interests include innovative technologies for multimedia signal processing, computer vision, and pattern recognition, with more than 100 papers published in international journals and conferences. He serves as an associate/guest editor for international journals and magazines, such as *Neurocomputing*, *Signal Processing*, and *Multimedia Systems*.