Adaptive Multi-Hypergraph Convolutional Networks for 3D Object Classification

Liping Nong, Jie Peng, Wenhui Zhang, Jiming Lin, Hongbing Qiu, and Junyi Wang, *Member, IEEE*

***Abstract*—3D object classification is an important task in computer vision. In order to explore the high-order and multi-modal correlations among 3D data, we propose an adaptive multi-hypergraph convolutional networks (AMHCN) framework to enhance 3D object classification performance. The proposed network improves the current hypergraph neural networks in two aspects. Firstly, existing networks rely on hyperedge constrained neighborhoods for feature aggregation, which may introduce noise or ignore positive information outside the hyperedges. To this end, we develop the partially absorbing random walks (PARW) to hypergraph for capturing optimal vertex neighborhoods from hypergraph globally. Then, based on the PARW on hypergraph, we design a new hypergraph convolution operator to learn deep embeddings from the optimized high-order correlation, which enables effective information propagation among the most relevant vertices. Secondly, concerning the multi-modal representations in practice, the current multi-modal hypergraph learning models either treat all modalities equally or introduce abundant parameters to learn weights of different modalities. To overcome these shortcomings, we propose a simple but effective dynamic weighting strategy for combining multi-modal representations, in which the importance of each modality can be adjusted adaptively by the loss function. We apply the proposed model to 3D object classification, and the experimental results on two 3D benchmark datasets demonstrate that our method outperforms the state-of-the-art methods, testifying to the effectiveness of both our convolution method and multi-modality fusion strategy.**

***Index Terms*—Hypergraph learning, Hypergraph convolution operator, Partially absorbing random walks, Multi-modal fusion, 3D object classification**

I. Introduction

3D object classification, as the basis of 3D scene understanding and analysis, has a wide application prospect in automatic driving, medical diagnosis, digital entertainment, architecture design and many other fields. With the flourish of deep learning, various deep networks have been investigated for this task [1]–[8]. In recent years, to explore the correlations among 3D objects and potential complementarity among different 3D feature representations (i.e., multi-modal features), hypergraph neural networks have been developed for object recognition [9]–[13]. In these networks, 3D data correlations are formulated into one or multiple hypergraphs and deep learning is conducted on the hypergraph structure. Hypergraph is a powerful tool for modeling complex, high-order and multi-modal data relationships with its flexible hyperedges [9], [14]. The application of hypergraph neural networks in learning 3D data can promote feature fusion among multiple objects and multi-modality, thus bringing a breakthrough in the recognition performance. However, the investigation of deep learning on hypergraph is still in its infancy [15]. How to design effective hypergraph neural networks, especially multi-modal data oriented networks, to improve object recognition performance remains a challenging topic.

Hypergraph neural network is one of the most prominent hypergraph learning models, which has been applied to emotion recognition [16], recommendation system [17], stock analysis [18], visual classification [9], [10], [12], [19], etc. The current approaches can be divided into two categories. One is to transform the hypergraph into a normal graph[1], and then the mature graph neural networks technology [20] is applied to learn the normal graph [21]–[23]. Such methods require complicated transformation processes and are prone to lose high-order information. The other is to transform and propagate vertex (node) feature information directly on hypergraph [9]–[11], [13], [15], [19], [24]–[26]. Vertex-hyperedge-vertex transformation is the most common feature transformation and propagation mode in this kind of networks. Specifically, vertex features are first gathered to hyperedges to form hyperedge features, then the new embedding of each vertex is obtained by aggregating its associated hyperedge features. In hypergraph neural networks (HGNN) [9], a hyperedge convolution operator based on spectral convolution is first proposed to implement this transformation. This convolution operator is subsequently used in [10], [13], [15], [24]. In addition, dynamic Hypergraph Neural Networks (DHGNN) [27] also adopts this transformation. Different from HGNN, it uses multi-layer perception (MLP) based vertex convolution and hyperedge convolution to propagate features. By introducing attention mechanism [11], [25], [26], this transformation is further improved that can automatically learn the

This work was supported by the National Natural Science Foundation of China (61966007); the Guangxi Key Laboratory of Wireless Wideband Communication and Signal Processing, Guilin University of Electronic Technology (GXKL06190204, GXKL06200116,GXKL06190117); the Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education (Guilin University of Electronic Technology) (CRKL180201,CRKL180106) and the Guangxi Natural Science Foundation(2020GXNSFAA159105). *(Corresponding author: Junyi Wang.)*

Liping Nong and Jie Peng are with the School of Telecommunication Engineering, Xidian University, Xi'an 710071, China, and also with the School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China (e-mail: nongliping@stu.xidian.edu.cn; jiepengxidian@qq.com).

Junyi Wang, Jiming Lin and Hongbing Qiu are with the School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China (e-mail: wangjy@guet.edu.cn; linjm@guet.edu.cn; qiuhb@guet.edu.cn).

Wenhui Zhang is with the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China (e-mail:zhangwh@guet.edu.cn).

[1]In this paper, a normal graph refers to the usual graph in which each edge only connects two vertices.

importance of vertices and hyperedges. The vertex-hyperedge-vertex transformation is essentially weighted aggregation of neighborhood information for each node, whose neighborhood size is determined by its associated hyperedges, as shown in Fig. 1(a). However, this kind of neighborhood selection can be suboptimal. Because too large a hyperedge may introduce noise, while too small one may separate data samples from the same cluster and lead to performance degradation [12], [28]. Though some hypergraph construction methods [28], [29] may mitigate this problem, the high computational cost is non-negligible. And in fact constructing an accurate hypergraph is an NP-Hard problem [30]. Therefore, how to capture a better neighborhood for each vertex based on the existing hypergraph, such as Fig. 1(b) or Fig. 1(c), by reducing noise information and utilizing more positive information outside the hyperedges to improve learning performance is a problem worthy of in-depth study.

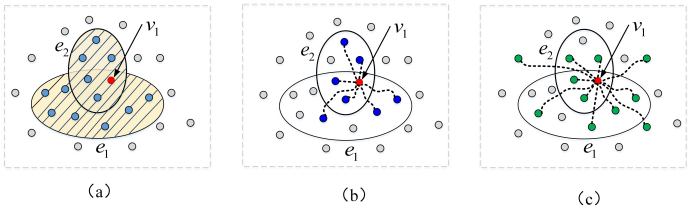


Fig. 1: Examples of different neighborhoods of a vertex v_1, in which v_1 belongs to hyperedges e_1 and e_2. (a) The neighborhood of vertex v_1 is the '/' shaded part. (b) The neighborhood of vertex v_1 is the blue dots connected by dotted lines. (c) The neighborhood of vertex v_1 is the green dots connected by dotted lines.

In addition, concerning the multi-modal representations in practice, how to effectively combine the multi-modal data is still a challenge. In the single hypergraph neural network models like [9]–[11], a hypergraph is constructed by concatenating hyperedges from features belonging to different modalities. In these methods, all modalities are treated equally, which is not able to reflect the importance of different modalities related to the task. In order to avoid this problem, recent studies have attempted to construct multi-hypergraph network structure for processing multi-modal data, in which different modalities are formulated in multi-hypergraph and the combination weights are optimized to fuse all modalities. Then, how to determine the weight values becomes an issue to be solved. Zhu et al. [31] and Shao et al. [16] adopt fully connected (FC) layer to fuse multi-modal representations. In [32], a bilinear attention map is used to integrate two multi-channel inputs. In [13], a learned weighting matrix is used to deal with different dynamic hypergraphs. In addition, Cui et al. [33] introduce a hypergraph attention module to combine representations of different hypergraphs. Although these methods can automatically learn weight values, they introduce a large number of parameters, resulting in increased training complexity. To overcome these shortcomings, it is necessary to develop a simple multi-hypergraph fusion scheme with low training complexity to learn multi-modal data.

In this paper, we propose an adaptive multi-hypergraph convolutional networks (AMHCN) architecture for 3D object classification. First of all, to enable hypergraph convolution to learn discriminative embeddings from optimal vertex neighborhoods, we develop partially absorbing random walks (PARW) [34] for hypergraph to capture clusters on hypergraph structure. PARW is a stochastic process that can reveal cluster structure under the cluster assumption [34]. It can keep the good property of random walk and well capture the global semantic information. We combine hypergraph random walk [35] and PARW theory to explore clustering on hypergraph. Compared with the methods [36]–[38] that require complex optimization processes to implement clustering or community detection on hypergraph, this method is simple and easy to implement in neural network. It is non-trivial to develop PARW on hypergraph because of the complex structure of hypergraph. We provide a detailed theoretical derivation and probabilistic interpretation for the PARW on hypergraph. Secondly, with the PARW on hypergraph to obtain the optimized higher-order relationship of each vertex, we design a new hypergraph convolution operator to learn the vertex deep embeddings. Finally, we present a dynamic weighting scheme to adaptively adjust the weights of embeddings corresponding to different modalities, so as to fuse different information adequately. It is worth noting that our combining scheme is guided by the loss functions of different modalities and no additional training parameters are introduced. The proposed network is capable of jointly considering complex data correlations and different modal data. Overall, the contributions of this work are:

1) We formulate the problem of optimizing vertex neighborhoods in hypergraph neural networks as capturing clusters on hypergraph by PARW. As far as we know, we are the first to develop PARW to hypergraph and apply it to convolution operation.
2) We present a new hypergraph convolution operator based on the PARW on hypergraph, which enables efficient feature extraction by fully exploiting the high-order relationship and clustering structure therein.
3) We propose a simple but powerful multi-modal data fusion strategy, in which the importance of different modalities can be adjusted automatically with respect to the loss function.
4) We propose a multi-hypergraph convolutional networks framework to improve 3D object classification performance. The experimental results on the National Taiwan University (NTU) and ModelNet40 datasets demonstrate the effectiveness of our network.

II. Related works

In this section, we briefly review existing works on both deep learning methods for 3D object classification and hypergraph neural networks.

A. Deep learning methods for 3D object classification

In recent years, deep learning methods have been widely investigated in 3D object recognition [1]–[8], [39]–[44], which have shown superior performance compared with traditional

hand-crafted feature methods. Given a set of 3D objects, the key issues for classification are how to extract 3D representation (3D shape descriptors) and how to train a classifier. With volumetric data as input, Wu et al. [45] propose 3D ShapeNets to learn 3D shape feature with a convolutional deep belief network, and train a linear support vector machine (SVM) to classify meshes. Maturana et al. [6] propose VoxNet to extract the features of volumetric data with a 3D convolutional neural network, and use FC layers for classification. In addition, multi-orientation volumetric CNN (MO-CNN) [46] extracts 3D shape features from volumetric data of different orientations through shared 3D CNN and a pooling operation, then uses another 3D CNN to make a prediction. With point cloud data as input, Qi et al. [1] introduce PointNet, a deep neural network that directly processes point clouds. The network captures the global feature of object through feature transformation, MLP and max pooling, then the global feature is fed into the MLP layer for classification. To capture local structures in point clouds, Qi et al. further propose PointNet++ [8], in which point sets are divided into overlapping local regions and PointNet is used to extract local features. These local features are grouped as global feature and sent into the FC layers for classification. Instead of acting on individual points like PointNet and PointNet++, Wang et al. [47] propose EdgeConv to learn and aggregate global features from local neighborhood graph, which are sent to MLP for classification. Taking multiple views of 3D objects as input, Su et al. [4] propose a multi-view convolutional neural network (MVCNN), in which the view features are generated by CNN. Then these features are fused by a pooling procedure to generate a compact 3D shape descriptor, which is then sent to another CNN for classification. Considering that MVCNN does not pay attention to the intrinsic hierarchical correlation and discriminability among views, Feng et al. [5] propose group-view convolutional neural network (GVCNN), in which a hierarchical view-group-shape architecture is used to get a better 3D shape descriptor, which is then fed into FC layer for classification. Similar deep learning networks with multiple views as input include [2], [3], [42]–[44].

All the methods mentioned above focus on how to capture better 3D object representation (i.e., 3D shape descriptor). For classifiers at the end of the network, however, they commonly adopt traditional deep learning models such as CNN [4], [46], FC layer [5], [6], [8], [47] and MLP [1], [39], [41] to directly project their own shape descriptors into the label space. This processing method has some shortcomings. The first one is that they do not consider the correlations between objects, that is, lose the global view of the data distribution in the whole dataset. The other is that they classify only by the feature representation of an object itself without effective fusion of different representations. Current deep 3D frameworks have generated a wide variety of feature representations, and some potential complementarity between these features may exist. To make use of the high-order correlations among objects and the potential relationships among different features/modality, some researchers [9]–[13] formulate 3D objects into hypergraph structure, and develop hypergraph neural networks to integrate the features of multiple objects and multi-modality, which made a great breakthrough in the performance of object classification. Inspired by these works, we develop hypergraph convolutional networks for object classification.

B. Hypergraph Neural Networks

Compared with graph neural networks [20], the investigation of hypergraph neural networks is in a nascent stage. Driven by the success of graph neural networks, some researchers propose transforming hypergraph into a normal graph, and then treating the hypergraph learning problem as a graph problem on the approximation. For example, [21]–[23] adopt different rules to transform a hypergraph to a normal graph, and then use graph convolutional networks (GCNs) [48] for learning. The transformation process of these methods is complicated and easy to lose high-order information. In order to keep the high-order relationship well, more work is to design convolution operators directly based on hypergraph theory and propagate features on hypergraph structure. HGNN [9] is the pioneer of this kind of method. In HGNN, a hyperedge convolution operator is designed based on the hypergraph Laplacian, then the output vertex feature is extracted by aggregating their related hyperedge features. This hyperedge convolution operator is subsequently used in [10], [13], [15], [24]. As an extension of HGNN, DHGNN [27] introduces a dynamic hypergraph construction module to update the hypergraph structure on each layer, and proposes vertex convolution and hyperedge convolution to aggregate vertices and hyperedges features, respectively. In addition, Bai et al. [15] utilize an attention mechanism to design a hypergraph attention operator which learns a dynamic connection of hyperedges. In [11], [25], [26], attention mechanisms are introduced into the process of vertex convolution and hyperedge convolution, which automatically learn different weights for vertices and hyperedges. These methods all adopt vertex-hyperedge-vertex feature transformation and propagation mode, i.e., the connected vertex features are first aggregated to generate hyperedge features, then the hyperedge features are aggregated as new vertex embeddings. In these methods, the convolution range (receptive field) depends on the hyperedge size, which makes it difficult to obtain effective information from hypergraph globally. Shi et al. [12] propose a hypergraph convolution operator based on the high-order correlation optimized by a hypergraph regularization framework. In form, the convolution operator bears some resemblance to ours in that they are related to the inverse of the hypergraph Laplacian. However, our operator is derived from the theory of random walk, which is more explanatory.

For multi-modal data, it is not trivial to find the optimal multi-modal feature combination. In HGNN [9], the hyperedges generated from features of different modalities are concatenated as a hypergraph. The same concatenation method is used in single hypergraph models [10], [11], [19]. These methods concatenate different modalities with equal weight, which fails to reflect the significance of different modalities related to the task. To address this problem, many researchers propose multi-hypergraph neural networks and assign different weights to combine the deep embeddings of

different hypergraphs. In [31] and [16], multiple hypergraphs are generated according to different modalities, and the multi-modal representations are automatically combined by FC layer. In [32], a bilinear attention map is used to integrate the multi-modal inputs for getting the final joint representation. Besides, Liu et al. [13] adopt an adaptive, trainable weighting matrix to group together different hypergraph representations. Cui et al. [33] propose temporal-relational hypergraph tri-attention networks, in which an attention module is used to combine the representations of different hypergraphs. Although these methods can automatically combine different hypergraph representations, they increase a large number of parameters, resulting in increased computational complexity. In this work, we try to propose an adaptive multi-hypergraph learning model, where the weights of different modalities are obtained as simple as possible without increasing the training parameters.

III. Methodology

A. Problem Statement

We consider the general problem of semi-supervised learning (SSL) with the goal of learning a mapping from a set of data points $\mathcal{X}$ to the corresponding labels in $\mathcal{Y} = \{y_1, y_2, \ldots, y_c\}$, where c is the number of categories. The input data points are divided into two parts: the labeled data $\mathcal{X}_l$ and the unlabeled data $\mathcal{X}_u$. The labels associated with the labeled data are provided. The goal of this paper is to develop a deep convolution architecture based on hypergraphs that, using data features as input, maps each vertex (object) to a corresponding label and hence predicts the labels associated with the unlabeled data $\mathcal{X}_u$. Fig. 2 shows the illustration of semi-supervised classification based on hypergraph learning with 3D objects as input.

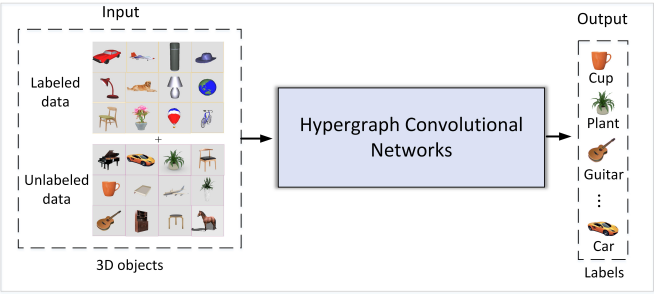


Fig. 2: An example of semi-supervised classification based on hypergraph learning with 3D objects as input.

B. Hypergraph Construction

Given a set of data points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]^T \in \mathbb{R}^{N \times d}$ with one type of d-dim feature, a hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ is constructed to formulate the correlations among objects, where $\mathcal{V}$ denotes a vertex set, $\mathcal{E}$ is a hyperedge set and $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ is a diagonal hyperedge weight matrix. Let each hyperedge $e \in \mathcal{E}$ be assigned a positive weight $w(e)$. Hyperedges can be generated by many approaches, such as kNN, ϵ-ball, and sparse representation [49]. In this work, we select the commonly used kNN strategy to construct a connected hypergraph. Specifically, for N given objects $\mathbf{X}$, let each vertex denote one object, then there are N vertices in total. The hyperedges are generated as follows: each time, one vertex is selected as the centroid, and then the distance (e.g., the Euclidean distance) between the centroid and other vertices is calculated, and finally a corresponding hyperedge e is generated by connecting the centroid and its nearest k neighbors.

A hypergraph with N vertices can be represented by an $N \times |\mathcal{E}|$ incidence matrix $\mathbf{H}$, with entries defined as:

$$h(v, e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e \end{cases} \tag{1}$$

Then, the degree of a vertex $v \in \mathcal{V}$ and the degree of a hyperedge $e \in \mathcal{E}$ are defined as follows:

$$d(v) = \sum_{e \in \mathcal{E}} w(e) h(v, e) \tag{2}$$

$$\delta(e) = \sum_{v \in \mathcal{V}} h(v, e) \tag{3}$$

We further denote $\mathbf{D}_v$ and $\mathbf{D}_e$ as the diagonal matrices of the vertex degrees and the hyperedge degrees, respectively. In this work, the weight of each hyperedge is initialized as 1.

C. Partially Absorbing Random Walks on Hypergraph

Random walk is a simple but powerful tool for extracting information on the relational structure of interacting systems [50]. At present, there are some methods that use random walk to realize node clustering or community detection on hypergraph [36]–[38]. However, they usually require complex optimization processes and are not suitable for neural networks. The recently proposed PARW [34] is a stochastic process that can implement cluster assumption in terms of random walks. It avoids the problem that a random walk converges to a stationary distribution determined only by the degree of each node without capturing the global semantic structure of the topology. It is easy to implement clustering. Therefore, we generalize it to hypergraph to reveal clustering in hypergraph structure. Specifically, PARW is a second-order Markovchain with partial absorption at each state. Given a stochastic process $X = \{X_t : t \geq 0\}$ on the state space $N = \{1, 2, \ldots, n\}$. Let the initial state be $X_0 = i$, the next state X_1 is determined by the transition probability $\mathbb{P}(X_1 = j | X_0 = i) = p_{ij}$, and the subsequent states are determined by the following transition probabilities:

$$\begin{aligned} &\mathbb{P}(X_{t+2} = j | X_{t+1} = i, X_t = k) \\ &= \begin{cases} 1, & \text{if } i = j, i = k \\ 0, & \text{if } i \neq j, i = k \\ \mathbb{P}(X_{t+2} = j | X_{t+1} = i) = p_{ij}, & \text{if } \quad i \neq k \end{cases} \end{aligned} \tag{4}$$

The transition probabilities in Eq. (4) are independent of t, hence the stochastic process X is time homogeneous. If a random walker starts from state i, it would stay in the current state with probability p_{ii}, or move to another state in the next step behaving like a usual random walk. Once it stays, it will get absorbed into the current state and remain forever. The

above process is called a partially absorbing random walk (PARW), where p_{ii} stands for the absorption rate of state i. When $0 < p_{ii} < 1$, the state i is called a partially absorbed state. It is a fully absorbing state when $p_{ii} = 1$, and transient state when $p_{ii} = 0$.

PARW theory has been applied to graph and has shown to be well suited for implementing the cluster assumption for graph-based learning [34]. Hypergraph is an extension of graph. Borrowing from the idea of the PARW on graph, we develop PARW to hypergraph for exploring latent clustering on hypergraph structure. First of all, in order to obtain the absorption probability matrix on the hypergraph, we need to find a symmetric non-negative hypergraph adjacency matrix with zeros on the diagonal entries. The hypergraph adjacency matrix defined in [51] is used in this work, i.e.,

$$\mathbf{S}_{i,j} = \begin{cases} 0, & \text{if } i = j \\ \sum_{e \in \mathcal{E}: v_i, v_j \in e} \frac{1}{\delta(e)-1}, & \text{if } i \neq j \end{cases} \tag{5}$$

We rewrite Eq. (5) in matrix form as :

$$\mathbf{S} = \mathbf{H}(\mathbf{D}_e - \mathbf{I})^{-1}\mathbf{H}^T - \mathrm{Diag}\left(\mathbf{H}(\mathbf{D}_e - \mathbf{I})^{-1}\mathbf{H}^T\right) \tag{6}$$

where $\mathrm{Diag}(\cdot)$ denotes a diagonal matrix obtained by setting to zero the off-diagonal elements of a matrix $\cdot$. In [35], a typical random walk on a hypergraph is given as $\mathbf{P} = \mathbf{D}_v^{-1}\mathbf{S}$. It means that a random walk at vertex v_i will transfer to another vertex v_j belonging to the same hyperedge e with probability $\mathbf{P}_{i,j}$ in the next step. The corresponding Laplacian is $\mathbf{L} = \mathbf{D}_v - \mathbf{S}$. With these definitions, we introduce a regularization matrix $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ with $\lambda_i \geq 0$ to define the first order transition probabilities of a PARW on hypergraph as:

$$p_{i,j} = \begin{cases} \frac{\lambda_i}{\lambda_i + d_i}, & \text{if } i = j \\ \frac{\mathbf{S}_{ij}}{\lambda_i + d_i}, & \text{if } i \neq j \end{cases} \tag{7}$$

The state i is an absorbing state (either partially or fully) when $\lambda_i > 0$, and it is a transient state when $\lambda_i = 0$. It is obvious that $p_{i,j} \geq 0$. The transition probability matrix satisfies the condition that the row sum is 1, which is proved as follows:

Proof. Let $\mathbf{1}_N \in \mathbb{R}^N$ denote the all-one vector, $\mathbf{0}_N \in \mathbb{R}^N$ denote the zero vector and $\mathbf{d}_v \in \mathbb{R}^N$ be the vector containing the degrees of vertices of hypergraph $\mathcal{G}$. Since

$$\begin{aligned} \sum_{j \in \mathcal{V}} \mathbf{S}_{ij} &= \sum_{j=i} \mathbf{S}_{ij} + \sum_{j \in \mathcal{V}, j \neq i} \mathbf{S}_{ij} \\ &= 0 + \sum_{j \in \mathcal{V}, j \neq i} \sum_{e \in \mathcal{E}: v_i, v_j \in e} \frac{1}{\delta(e) - 1} \\ &= \sum_{j \in \mathcal{V}, j \neq i} \sum_{e \in \mathcal{E}} \frac{h(i,e)h(j,e)}{\delta(e) - 1} \\ &= \sum_{e \in \mathcal{E}} \frac{h(i,e)}{\delta(e) - 1} \sum_{j \in \mathcal{V}, j \neq i} h(j,e) \\ &= \sum_{e \in \mathcal{E}} \frac{h(i,e)}{\delta(e) - 1} (\delta(e) - 1) = d_i \end{aligned}$$

$$\text{Thus, } \sum_{j \in \mathcal{V}} p_{i,j} = \frac{\lambda_i}{\lambda_i + d_i} + \sum_{j \neq i, j \in \mathcal{V}} \frac{\mathbf{S}_{ij}}{\lambda_i + d_i} = 1$$

Next we calculate the absorption probabilities on hypergraph. Let $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$ represent the absorption probability matrix on hypergraph, where a_{ij} stands for the probability that a random walk starting from state i is absorbed by state j in any finite number of steps. Referring to [34], the absorbing probabilities $\{a_{ij}\}$ satisfy the following equations:

$$a_{ii} = \frac{\lambda_i}{\lambda_i + d_i} \times 1 + \sum_{j \neq i} \frac{\mathbf{S}_{ij}}{\lambda_i + d_i} a_{ji} \tag{8a}$$

$$a_{ij} = \sum_{k \neq i} \frac{\mathbf{S}_{ik}}{\lambda_i + d_i} a_{kj}, \ i \neq j \tag{8b}$$

In fact, $\mathbf{A}$ has a closed-form on hypergraph, i.e., $\mathbf{A} = (\mathbf{\Lambda} + \mathbf{L})^{-1}\mathbf{\Lambda}$.

Proof. $\mathbf{\Lambda}$ is positive as $\lambda_i > 0$ for some i. Then $\mathbf{\Lambda} + \mathbf{L}$ is positive definite and non-singular. Moreover, since $\mathbf{D}_v$ is non-singular, $\mathbf{\Lambda} + \mathbf{D}_v$ is non-singular. Therefore, $\mathbf{I} - (\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{S} = (\mathbf{\Lambda} + \mathbf{D}_v)^{-1}(\mathbf{\Lambda} + \mathbf{L})$ is non-singular. By rewriting Eq. (8a) and Eq. (8b) in matrix form, we have:

$$(\mathbf{I} - (\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{S})\mathbf{A} = (\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{\Lambda}$$

Therefore, we get the following equation:

$$\begin{aligned} \mathbf{A} &= (\mathbf{I} - (\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{S})^{-1}(\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{\Lambda} \\ &= (\mathbf{\Lambda} + \mathbf{D}_v - \mathbf{S})^{-1}\mathbf{\Lambda} = (\mathbf{\Lambda} + \mathbf{L})^{-1}\mathbf{\Lambda} \end{aligned}$$

Below we prove that $\mathbf{A}$ is a probability matrix, i.e., $\mathbf{A}$ is a non-negative matrix with each row summing up to 1.

Proof. Suppose $\lambda_i > 0$ for some i. Then,

$$\begin{aligned} (\mathbf{\Lambda} + \mathbf{L})^{-1} &= (\mathbf{\Lambda} + \mathbf{D}_v - \mathbf{S})^{-1} \\ &= (\mathbf{I} - (\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{S})^{-1}(\mathbf{\Lambda} + \mathbf{D}_v)^{-1} \\ &= \sum_{k \geq 0} ((\mathbf{\Lambda} + \mathbf{D}_v)^{-1}\mathbf{S})^k (\mathbf{\Lambda} + \mathbf{D}_v)^{-1} \geq \mathbf{0} \end{aligned}$$

Therefore, $\mathbf{A} \geq \mathbf{0}$, where $\mathbf{0}$ denotes the zero matrix. We have proved that $\sum_{j \in \mathcal{V}} \mathbf{S}_{ij} = d_i$, so it is easy to get $\mathbf{L}\mathbf{1}_N = (\mathbf{D}_v - \mathbf{S})\mathbf{1}_N = \mathbf{0}_N$. Then we have $(\mathbf{\Lambda} + \mathbf{L})(\mathbf{A}\mathbf{1}_N - \mathbf{1}_N) = (\mathbf{\Lambda} + \mathbf{L})((\mathbf{\Lambda} + \mathbf{L})^{-1}\mathbf{\Lambda}\mathbf{1}_N - \mathbf{1}_N) = \mathbf{0}_N$. Accordingly, $\mathbf{A}\mathbf{1}_N = \mathbf{1}_N$.

According to the study in [34], by setting $\mathbf{\Lambda} = \alpha\mathbf{I}$ $(\alpha > 0)$, the absorption probabilities vary slowly within the cluster while dropping sharply outside. This nice property of the PARW with $\mathbf{\Lambda} = \alpha\mathbf{I}$ $(\alpha > 0)$ can be used to identify the clusters and has been proven to achieve superior performance on classification/semi-supervised learning. Hence, we set $\mathbf{\Lambda} = \alpha\mathbf{I}$ to capture the clustering structure on the hypergraph, which is expressed as:

$$\mathbf{A} = (\alpha\mathbf{I} + \mathbf{L})^{-1}\alpha \tag{9}$$

Eq. (9) can be understood as an optimized higher-order relationship on a hypergraph. In essence, hypergraph is a clustering hypothesis, but this original clustering is likely to be noisy and not optimal. The combination of PARW and hypergraph can obtain optimized clustering on the basis of rough clustering, thereby providing a faithful affinity map/similarity metric.

With the PARW on hypergraph, the most relevant vertices for a vertex from the entire hypergraph structure can be automatically found. That is, vertices with similar properties on the hypergraph will form a homogeneous region. Therefore,

we can design a PARW based convolution operator to compute the new features of a vertex as weighted aggregation of itself and its related region', thereby promoting effective information transmission between vertices.

D. Hypergraph Convolution

Based on the PARW on hypergraph above to measure the transition probability between two vertices, we now design the hypergraph convolution operator to propagate the embeddings (or features) of each vertex in a hypergraph neural network. Specifically, for each hypergraph, a similarity measure $\mathbf{A} \in \mathbb{R}^{N\times N}$ is obtained to encode high-order correlations. Then combined with input data $\mathbf{X} \in \mathbb{R}^{N\times d}$, a filter function $f(\cdot)$ is designed to extract the end-to-end description of the data for classification, which can be denoted as $\mathbf{Z} = f(\mathbf{X}, \mathbf{A})$, where $\mathbf{Z} \in \mathbb{R}^{N\times c}$ denotes the matrix of convolved feature. Multiple convolutional layers are employed in our network, and the hypergraph convolution operator is defined as:

$$\mathbf{Z}^{(l+1)} = \sigma(\mathbf{A}\mathbf{Z}^{(l)}\mathbf{\Theta}^{(l)}) \tag{10}$$

where $\mathbf{Z}^{(l)}$ is the matrix of activations in the l-th layer, the initial activation is $\mathbf{Z}^{(0)} = \mathbf{X}$. $\mathbf{\Theta}^{(l)}$ is the parameter to be learned during training in layer l and σ is the activation function, e.g., $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. For example, for a two-layer convolutional network, the output of the final convolution layer is:

$$\mathbf{Z} = \mathbf{A}\ \mathrm{Relu}(\mathbf{A}\mathbf{X}\mathbf{\Theta}^{(0)})\mathbf{\Theta}^{(1)} \tag{11}$$

where $\mathbf{\Theta}^{(0)} \in \mathbb{R}^{d\times h}$ is an input-to-hidden weight matrix for a hidden layer with h feature maps. $\mathbf{\Theta}^{(1)} \in \mathbb{R}^{h\times c}$ is a hidden-to-output weight matrix.

In order to improve the computational efficiency of convolution operation, we set the elements of $\mathbf{A}$ smaller than a threshold η to 0, since most elements in $\mathbf{A}$ are actually very close to 0. The setting of η will be further discussed in the experiment.

E. Adaptive Multi-Hypergraph Convolutional Networks

In many applications, the data representation tends to be multi-modal and how to combine different modalities is still a challenging issue. To this end, we propose an adaptive convolutional networks framework based on multi-hypergraph.

Assuming there are m modalities in total, then m hypergraphs are generated accordingly. For the i-th modality data, a hypergraph $\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i, \mathbf{W}_i\}$ is generated to formulate the data relationship based on the aforementioned kNN strategy. With the hypergraph $\mathcal{G}_i$, a corresponding similarity matrix $\mathbf{A}_i$ is obtained based on Eq. (9). For each modality, we build a hypergraph neural network model to learn the intermediate representation. The layerwise propagation rule of the i-th hypergraph neural network with L layers is written as:

$$\mathbf{Z}_i = \sigma(\mathbf{A}_i(...\sigma(\mathbf{A}_i\mathbf{X}_i\mathbf{\Theta}_i^{(0)})...)\mathbf{\Theta_i}^{(L-1)}) \tag{12}$$

We consider a simple strategy to combine the m output intermediate representations and obtain the output of the whole model:

$$\mathbf{Z}_C = \sum_{i=1}^{m} \beta_i \mathbf{Z}_i \tag{13}$$

where β_i is the combination weight for the i-th modality subjecting to $\sum_{i=1}^{m} \beta_i = 1$ and $\beta_i \geq 0$.

For the value of β_i, a natural idea is to obtain through learning, but this scheme does not work well in our context. Here we present a method based on the loss function. For each modality, we use the cross-entropy error over all labeled examples to express its loss value, i.e.,

$$\mathcal{L}(\mathbf{Z}_\mathrm{i}) = -\sum_{p\in\mathcal{X}_l}\sum_{q=1}^{c} \mathbf{Y}_{p,q} \ln\ (\mathrm{softmax}(\mathbf{Z}_\mathrm{i}))_{p,q} \tag{14}$$

where $\mathcal{X}_l$ is the set of node indices that have labels. The softmax function is defined as $\mathrm{softmax}(x_i) = \exp(x_i)/\sum_j \exp(x_j)$. $\mathcal{L}(\mathbf{Z}_\mathrm{i})$ reflects the error between the estimated label and the true label for the i-th modality data. The smaller the $\mathcal{L}(\mathbf{Z}_\mathrm{i})$, the more discriminative features are extracted by the network under this mode, and the greater weight should be given. A dynamic weight function is designed to realize the above idea. That is, we calculate the minimum value of the loss function for each modality in the training process, and then obtain the weight value based on this minimum value. More specifically, for the i-th modality, we first calculate its minimum loss value of the previous t training epochs:

$$\mathrm{MIN\_}\mathcal{L}(i,t) = \min\{\mathcal{L}(\mathbf{Z}_\mathrm{i})|_1, \mathcal{L}(\mathbf{Z}_\mathrm{i})|_2, \ldots, \mathcal{L}(\mathbf{Z}_\mathrm{i})|_t\} \tag{15}$$

where $\mathcal{L}(\mathbf{Z}_\mathrm{i})|_t$ represents the loss value of the i-th modality data in the t-th training epoch. Then we use the following attention mechanism to calculate the weighted values in the t-th epoch:

$$\begin{aligned}(\beta_1(t), \beta_2(t), \ldots, \beta_m(t)) = \mathrm{softmax}(-\mathrm{MIN\_}\mathcal{L}(1,t),\\ -\mathrm{MIN\_}\mathcal{L}(2,t), \ldots, -\mathrm{MIN\_}\mathcal{L}(m,t))\end{aligned} \tag{16}$$

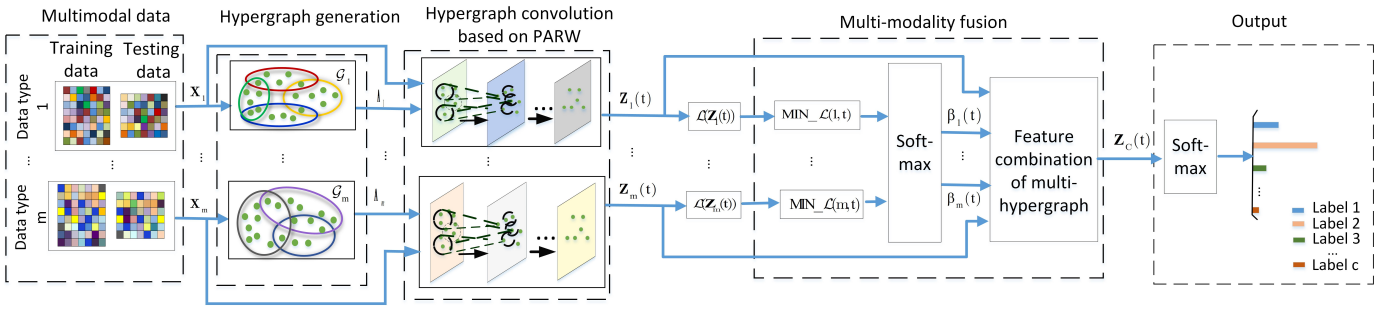


Fig. 3: Overview of the proposed adaptive multi-hypergraph convolutional networks.

In Eq. (16), we utilize the monotonically decreasing property of $\exp(-x)$ to realize the idea that the smaller the loss value, the larger the weight value. This attention mechanism makes the weight values of different modalities adaptive to the minimum loss values and does not require additional training parameters, which offers huge advantages over the current methods of introducing a large number of learning parameters, such as hypergraph attention in [33] and FC layer in [31] and [16].

Since dynamic weight values are employed, we then rewrite the Eq. (13) as:

$$\mathbf{Z}_C(t) = \sum_{i=1}^{m} \beta_i(t)\mathbf{Z}_i(t) \tag{17}$$

The overall architecture of the proposed network is illustrated in Fig. 3. First, different modalities are formulated as different hypergraph structures. Then the similarity measure matrix $\mathbf{A}_i$ concerned with hypergraph structure is pre-computed. Next, $\mathbf{A}_i$ and the corresponding feature $\mathbf{X}_i$ are sent to the i-th convolutional network for training. In this way, different modality features with various scales and dimensions are trained individually in different networks. And then the weights of all modalities are obtained according to the attention mechanism guided by their minimum loss values. Finally, the linear weighted sum of the deep embeddings of all modalities is taken as the object descriptor which is fed into the softmax layer for classification.

Computational complexity. For the calculation of the PARW probability matrix $\mathbf{A}$, it involves matrix inversion, which is of great computational complexity. But fortunately, there are many mature fast inversion methods, such as Jacobi numerical method, QR decomposition, LU decomposition and Cholesky decomposition. In our implementation, we use the LU decomposition method for inversion, which has moderate complexity (i.e., $\mathcal{O}(N^3)$) and easy to parallel computing [52]. For convolution operation, assuming that the number of non-zero elements in $\mathbf{A}$ is $|\varrho|$ and the number of convolutional layers is 2, then the computational complexity of Eq. (11) is $\mathcal{O}(|\varrho|dhc)$. If there are m modalities, then the complexity of convolution operation of the entire network is $\mathcal{O}(m|\varrho|dhc)$.

IV. Experiments

The proposed AMHCN framework is applied to 3D object classification to evaluate performance. In this section, we first introduce datasets and experimental settings (Section IV-A). Next, we provide experimental results and comparisons of AMHCN on 3D shape classification task (Section IV-B). Then, the ablation study experiments are provided (Section IV-C). After that, we validate our network design (Section IV-D) and discuss the time and space complexity of our network (Section IV-E). Finally, the visualization of features is demonstrated IV-F.

A. Datasets and Implementation Details

Two public 3D datasets, i.e., the ModelNet40 dataset [45] and the National Taiwan University (NTU) 3D model dataset [53] are selected as testing benchmarks. The ModelNet40 dataset contains 12311 objects from 40 popular categories. We follow [45] to use the official split to conduct the training/testing split, where 9843 objects are used for training and 2468 objects are used for testing. The NTU dataset is composed of 2012 3D shapes from 67 categories, such as boat, bomb, book, car, chair, guitar, gun, hat and helicopter. We employ two approaches to split this dataset. The first is to follow [9]–[11] using 1639 objects for training and 373 objects for testing. The second is to follow [54] with 50% data for training and 50% for testing. Fig. 4 demonstrates the example objects from the ModelNet40 and NTU datasets, and Table 1 summarizes the data.

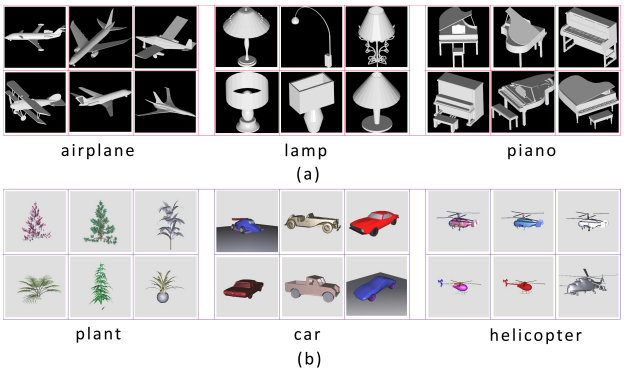


Fig. 4: Example 3D objects from ModelNet40 and NTU datasets. (a) ModelNet40. (b) NTU.

We follow [9], [54] to employ two recent state-of-the-art multi-view based 3D shape descriptors to represent each object, including multi-view convolutional neural network (MVCNN) [4] feature and group-view convolutional neural network (GVCNN) [5] feature. These two features are chosen because they contain rich information about 3D shapes. The process of extracting these two features is as follows: First, a set of 12 virtual cameras are used to capture views of an object with 30 degree interval, so that each object contains 12 views. Then, the 4096-d MVCNN feature and 2048-d GVCNN feature are extracted to describe objects according to [4] and [5], respectively.

We use Pytorch to implement our model and deploy it on a NVIDIA RTX TITAN GPU with 24G memory. For each modality data, the kNN strategy is employed to generate a corresponding hypergraph. In this way, each dataset generates two hypergraphs based on the MVCNN and GVCNN representations. We set $k = 10$ for both the ModelNet40 and NTU datasets. Each modality is learned with a two-layer hypergraph neural network, and the dimension of each hidden layer is set to 128. To avoid overfitting, we randomly dropout the features at 0.1 probability for the hidden layers. The tanh is chosen as the nonlinear activation function. The Adam optimizer [55] is employed to minimize the cross-entropy loss function with a learning rate of 0.001. We train the network model for a maximum of 600 epochs for the NTU and 1000 epochs for the ModelNet40. The regularization of PARW is set to $\alpha = 35$ for the NTU and $\alpha = 60$ for the ModelNet40.

TABLE I: Data Statistics of ModelNet40 and NTU datasets.

Dataset		Objects	Training objects	Testing objects	Categories	Feature	
						MVCNN	GVCNN
ModelNet40		12311	9843	2468	40	4096	2048
NTU	The first split	2012	1639	373	67	4096	2048
	The second split	2012	1006	1006	67	4096	2048

B. 3D Shape Classification Results

The classification accuracy is used as the evaluation metric. On the ModelNet40 dataset, to evaluate the performance of the proposed method, the following deep learning methods are selected for comparison, including methods with volumetric data as input (VoxNet [6] and PC-GAN [7]), methods that take point cloud data as input (PointNet++ [8], AGCN [39], LFT-Net [40] and FFPointNet [41]) and methods with multi-view as input (MVCNN [4], GVCNN [5], MLVCNN [2], DAN [42], MVSG-DNN [43], HMVCM [44], DRCNN [3], tMHL [56], CDMH [57], HGNN [9], iMHL [54], HGAT [11] and MHGNN [10]). Among the multi-view methods, tMHL [56], CDMH [57], HGNN [9], iMHL [54], HGAT [11] and MHGNN [10] all use hypergraph learning to classify 3D objects, which are similar to ours. For comparison purposes, we denote the input of this kind of methods as multi-view (+Hypergraph). The 10-times average accuracy of the proposed AMHCN for the ModelNet40 and comparisons among these methods are demonstrated in Table II. It can be observed that the proposed AMHCN achieves best classification accuracy compared with the other methods. For example, compared with the voxel-based classification methods VoxNet and PC-GAN, the proposed AMHCN achieves gains of 14.86% and 5.16%. As for the point cloud based methods such as PointNet++, AGCN and LFT-Net, our network outperforms them with the improvement of 7.16%, 5.26% and 4.66%, respectively. Compared with multi-view 3D recognition methods such as MVCNN, GVCNN and HMVCM, the proposed AMHCN obtains gains of 7.76%, 4.76 % and 3.29%, respectively. Compared with HGNN, HGAT and MHGNN, which also use hypergraph neural network to classify objects, our network improves 1.16%, 0.76% and 0.36%, respectively. It is worth mentioning that the multi-scale hypergraph neural network (MHGNN) achieves comparable results to ours. However, its training complexity is higher than ours and its performance on NTU dataset is worse than ours. These will be discussed further below.

On the NTU dataset, we used two different training/test splits. The experiments are repeatedly run over 10 times and the average accuracies are reported in Table III. In the first split (i.e., Train/test: 1639/373), we select HGNN [9], CDMH [57], MHGNN [10] and HGAT [11] for comparison. They are all hypergraph learning methods. In the second split (i.e., Train/test: 1006/1006), MVCNN [4], GVCNN [5], MVCNN+SVM [54], GVCNN+SVM [54], tMHL [56] and iMHL [54] are used for comparison. It can be seen that when the first split is adopted, the proposed AMHCN is 2.33% higher than HGNN and 1.03% higher than both the recently proposed MHGNN and HGAT. For the second split, AMHCN achieves gains of 17.13%, 17.68%, 14.50% and 13.18 % compared with MVCNN, GVCNN, MVCNN+SVM and GVCNN+SVM, respectively. When compared with multi-hypergraph learning methods tMHL and iMHL, our network outperforms them by 5.82% and 1.75%.

It is worth noting that the classification accuracy of the proposed AMHCN in the second split is 5.55% higher than that in the first split. It means that the performance gets worse as more training data is used. This is also the case with HGNN. Some articles such as [58] also reported similar phenomena. This may be caused by overfitting. Specifically, when the training data is very large and the testing data is

TABLE II: Experimental comparison of different methods in terms of classification accuracy on ModelNet40 dataset.

Method	Input	Accuracy
VoxNet [6]	Voxel	83.00%
PC-GAN [7]	Voxel	92.70%
PointNet++ [8]	Point Cloud	90.70%
AGCN [39]	Point Cloud	92.60%
LFT-Net [40]	Point Cloud	93.20%
FFPointNet [41]	Point Cloud	93.50%
MVCNN [4]	Multi-View	90.10%
GVCNN [5]	Multi-View	93.10%
MLVCNN [2]	Multi-View	94.16%
DAN [42]	Multi-View	93.50%
MVSG-DNN [43]	Multi-View	92.30%
HMVCM [44]	Multi-View	94.57%
DRCNN [3]	Multi-View	96.84%
tMHL [56]	Multi-View (+Hypergraph)	96.19% (#)
CDMH [57]	Multi-View (+Hypergraph)	96.76%
HGNN [9]	Multi-View (+Hypergraph)	96.70%
iMHL [54]	Multi-View (+Hypergraph)	97.16%
HGAT [11]	Multi-View (+Hypergraph)	97.10%
MHGNN [10]	Multi-View (+Hypergraph)	97.50%
AMHCN	Multi-View (+Hypergraph)	**97.86**%

[1] The mark "#" means the result is quoted from [54].

TABLE III: Experimental comparison of different methods in terms of classification accuracy on NTU dataset.

Dataset split	Method	Input	Accuracy
Train/test: 1639/373	HGNN [9]	Multi-View(+Hypergraph)	84.20%
	CDMH [57]	Multi-View(+Hypergraph)	84.45%
	MHGNN [10]	Multi-View (+Hypergraph)	85.50%
	HGAT [11]	Multi-View (+Hypergraph)	85.50%
	AMHCN	Multi-View (+Hypergraph)	**86.53**%
Train/test: 1006/1006	MVCNN [4]	Multi-View	74.95% (#)
	GVCNN [5]	Multi-View	74.40% (#)
	MVCNN+SVM [54]	Multi-View	77.58%
	GVCNN+SVM [54]	Multi-View	78.90%
	tMHL [56]	Multi-View (+Hypergraph)	86.26% (#)
	HGNN [9]	Multi-View (+Hypergraph)	89.76% (*)
	iMHL [54]	Multi-View (+Hypergraph)	90.33%
	AMHCN	Multi-View (+Hypergraph)	**92.08**%

[1] The mark"*" indicates the data is reproduced from code provided by the author of the paper.
[2] The mark "#" means the result is quoted from [54].

TABLE IV: Comparison of classification performance of AMHCN and its variants.

Dataset	Generated Hypergraph	Method	Feature		Accuracy
			MVCNN	GVCNN	
NTU	Single hypergraph	SHCN(hyperedge convolution)	✓	✗	85.00%
			✗	✓	90.70%
			✓	✓	90.51%
		SHCN(PARW-based convolution)	✓	✗	88.73%
			✗	✓	91.52%
			✓	✓	91.07%
	Multiple hypergraph	**AMHCN**	✓	✓	**92.08**%
ModelNet40	Single hypergraph	SHCN(hyperedge convolution)	✓	✗	91.04%
			✗	✓	92.67%
			✓	✓	96.72%
		SHCN(PARW-based convolution)	✗	✓	91.93%
			✓	✗	92.90%
			✓	✓	97.64%
	Multiple hypergraph	**AMHCN**	✓	✓	**97.86**%

relatively small, the model may take the characteristics of a large number of training samples as the general characteristics of all potential samples, resulting in poor generalization ability of the model.

Overall, the proposed method can achieve superior performance on both ModelNet40 and NTU datasets. The better performance of our network can be attributed to the following three reasons. Firstly, the use of hypergraph structure to capture high-order relationships among 3D shapes makes our method achieve better performance than those without hypergraph such as MVCNN and GVCNN. In principle, the high-order correlation is more informative, thus being helpful to improve classification performance. Secondly, by optimizing vertex neighborhoods with PARW, the proposed convolution operator can learn more discriminative embeddings. Finally, the proposed multi-hypergraph fusion strategy can better adjust the weights of different modalities related to classification task, thereby promoting the full integration of different information.

C. Ablation Study

To verify the effectiveness of the proposed convolution and the multi-hypergraph fusion strategy, we conduct ablation tests to compare the proposed AMHCN with its variants. In this experiment, the NTU dataset is split into 50% for training and 50% for testing.

1) Effectiveness of PARW-based convolution: We construct three single hypergraph models based on MVCNN feature, GVCNN feature, and MVCNN+GVCNN features (i.e., concatenating these two features corresponding incidence matrices), respectively, and apply the proposed PARW-based convolution on each hypergraph to learn deep embeddings. In addition, to compare the performance of the proposed convolution, we also use the hyperedge convolution in HGNN [9] to learn the three hypergraphs (all experimental settings are consistent with this paper). Hyperedge convolution is the most commonly used convolution operator [15], [31], and its expression is:

$$\mathbf{Y} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta} = \mathbf{G}\mathbf{X}\mathbf{\Theta} \tag{18}$$

where $\mathbf{G} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}$.

We denote the single hypergraph convolutional network using the proposed PARW-based convolution as SHCN(PARW-based convolution), and the one using hyperedge convolution as SHCN(hyperedge convolution). The corresponding experimental results are reported in Table IV. It can be seen that in single hypergraph learning, the proposed PARW-based convolution achieves better performance than hyperedge convolution on both NTU and ModelNet40 datasets. For example, on the NTU dataset, for the hypergraph constructed only by MVCNN feature, the proposed PARW-based convolution outperforms the hyperedge convolution with the improvement of 3.73%. For the hypergraph constructed by GVCNN feature, the proposed convolution operator is improved by 0.82%. On the ModelNet40 dataset, when learning the hypergraph constructed by MVCNN+GVCNN features, the proposed PARW-based convolution obtains gains 0.92% compared with the hyperedge convolution.

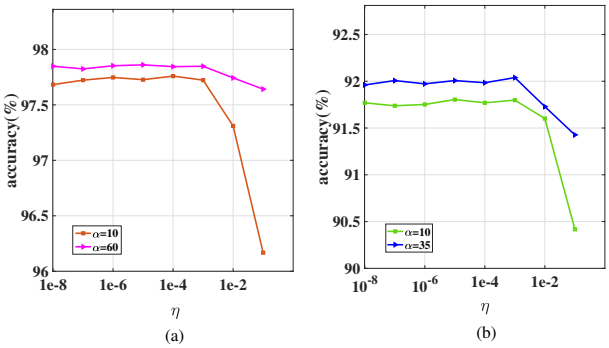


Fig. 5: The influence of η on the classification accuracy on (a) ModelNet40, (b) NTU.

The main reason why the proposed PARW-based convolution works is that PARW can optimize vertex neighborhood (or receptive field). To verify this, we conducted more experiments. We observe that most of the elements in the absorption probability matrix $\mathbf{A}$ of PARW are 0 or very close to 0. We set the elements of $\mathbf{A}$ smaller than a threshold η to 0, to explore the relationship between the vertex neighborhood size and network performance. The impact of η on network performance under different regularization parameter α is shown in Fig. 5. It can be observed that when $\eta \leq 10^{-3}$, the network can achieve relatively stable performance. This means that for a vertex, only vertices with similarity (confidence) above 10^{-3} are used for feature aggregation. We make statistics of the non-zero elements (i.e., the active nodes) in $\mathbf{A}$ at $\eta = 10^{-3}$ and compare with those in $\mathbf{G}$ of the hyperedge convolution. Fig. 6 reports the statistical results for the first ten nodes on the hypergraph generated by GVCNN+MVCNN features. It can be seen that the neighborhood sizes corresponding to the same vertex in the two matrices are mostly different. This means that, compared with the hyperedge convolution, the PARW-based convolution may use more vertices with high correlation outside the hyperedges for feature aggregation, or remove noisy vertices and only utilize a small number of relevant ones. Learning vertex embeddings in the optimized neighborhood is the most direct reason why PARW-based convolution works. Another possible reason is that it changes the similarity measurement between vertices in the neighborhood and gives higher weights to vertices with higher correlation, thus promoting better propagation of features. In summary, compared with convolution operators (e.g., hyperedge convolution) whose vertex receptive field depends on the hyperedge, the proposed PARW-based method can capture better receptive field for a vertex from the whole hypergraph structure, thereby obtaining more discriminative embeddings.

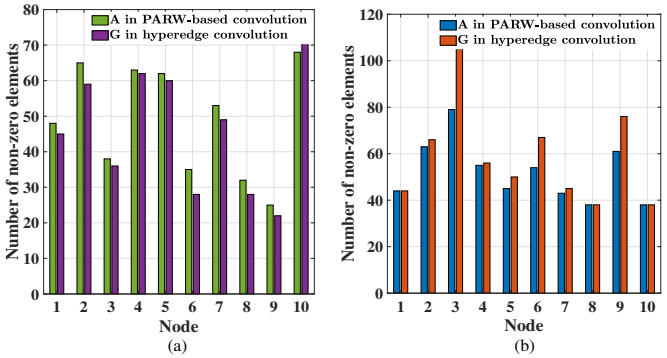


Fig. 6: Neighborhood sizes of the first 10 vertices in different convolution operators on (a) NTU. (b) ModelNet40.

2) Effectiveness of multi-modal fusion strategy: As for the effectiveness of the proposed multi-modal fusion strategy, it can be seen from Table IV that the performance of the proposed AMHCN is better than those of single hypergraph networks, both on NTU and ModelNet40. For example, AMHCN improves 3.35% on the NTU and 5.93% on the ModelNet40 compared with SHCN(PARW-based convolution) which only uses MVCNN feature. Compared with SHCN(PARW-based convolution) using multi-modality, AMHCN has a relatively obvious improvement on the NTU dataset (i.e., a gain of 1.01%), and a slight improvement on the ModelNet40 dataset. Fig. 7 provides the testing process of AMHCN, SHCN(PARW-based convolution) and SHCN(hyperedge convolution) methods using multi-modal features (i.e., both MVCNN and GVCNN features). These results once again demonstrate the effectiveness of the proposed convolution operator and multi-modal combination scheme.

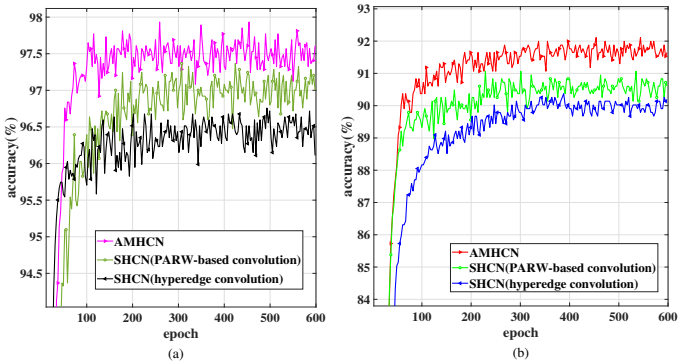


Fig. 7: Testing process of AMHCN and its variants on (a) ModelNet40, (b) NTU.

D. On Parameters

In this section, we discuss the effect of hyperparameters on network performance. For the NTU dataset, we use the second split, i.e., 50% for training, 50% for testing.

1) Influence of α: The parameter α is a regularizer of PARW that controls the absorption rate of each state. Under different sparse threshold values η, the effect of α on network performance is shown in Fig. 8. It can be observed that the ModelNet40 dataset can achieve better classification accuracy when α is about 50 to 80, while for NTU it is about 30 to 60. When α is too large or too small, the network can not get the best performance. These results can be explained that when α is too small, the probability mass distributes evenly within each cluster or even on the whole hypergraph, which causes the vertices of different clusters to be mixed into the neighborhood of a vertex, making vertices indistinguishable. When α is too large, most probability mass is concentrated on a vertex itself, and the convolution almost only uses its own features, resulting in low discrimination. When α is set appropriately, the new embeddings of a vertice can utilize not only its own feature, but also those of the most relevant vertices, so as to extract the most discriminating features and improve the classification performance.

2) Influence of k: The parameter k determines the number of vertices connected by hyperedges in the hypergraph construction. We vary k values from 5 to 30 at intervals of 5. As shown in Fig. 9, our network is sensitive to hyperedge size because our convolution is based on the hypergraph structure. Both the ModelNet40 and NTU datasets achieve their best classification accuracy when k is 10. As k gradually increases from 10, the performance becomes worse. The possible reason is that too large the hyperedge will introduce noise, resulting in a loss of discrimination.

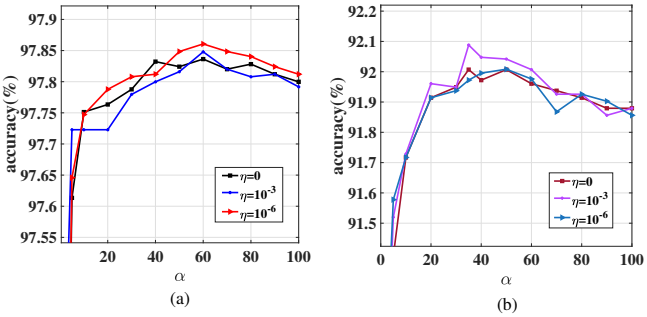


Fig. 8: The influence of α on the classification accuracy on (a) ModelNet40, (b) NTU.

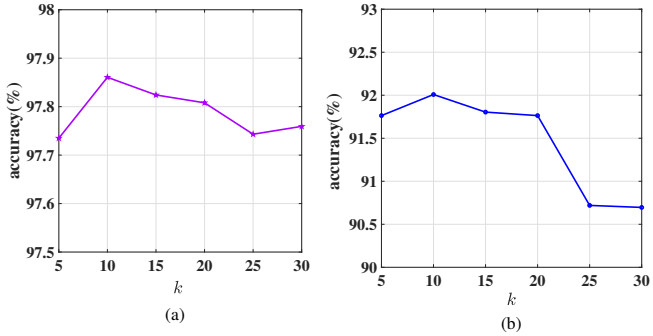


Fig. 9: Classification performance comparison with different values of k on the two datasets. (a) ModelNet40. (b) NTU.

3) Influence of Convolution Layer and Feature Map: Now we discuss the effect of the number of convolutional layers and value of the feature maps on network performance. We use the network architecture with two convolution layers. We vary the number of feature maps of the first convolution layer in [128, 256, 512, 1024]. The length of the convolved features in the second layer is equal to the number of categories. As shown in Fig. 10(a), the network can achieve relatively stable performance on the two datasets when the value of the feature maps varies in a large range. This shows that the network is not sensitive to the number of feature maps.

The number of hidden layers indicates the depth of the network. We vary it from 1 to 5. The experimental results are shown in Fig. 10(b). It can be seen that the best results are obtained with a 2-layer model for the two datasets. When the number of layers is greater than 2, the network performance does not increase as the network becomes deeper. This is because the network parameters increase with the number of hidden layers, resulting in overfitting of the network. Therefore, it is impractical to improve performance only by deepening the network.

E. Time and Space Complexity Analysis

Following PointNet [1], we use the number of parameters (#params) to measure the space complexity of the network,

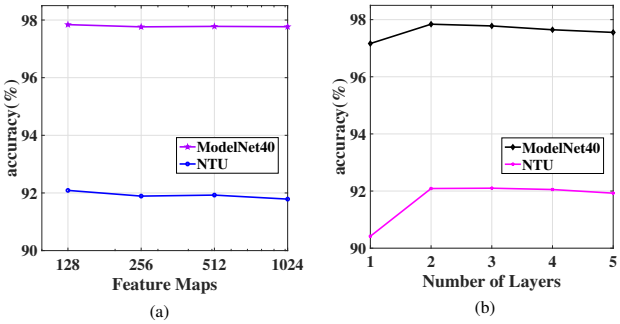


Fig. 10: Classification accuracy comparison with various feature maps and number of layers. (a) Feature maps. (b) Number of layers.

and floating-point operations/sample (FLOPs/sample) to measure the time complexity of the network. The experimental results of the proposed AMHCN and the comparison with the other methods are shown in Table V. It can be seen that the number of parameters of our network is almost equal to that of HGNN, and lower than MHGNN. In terms of computational cost, the FLOPs of AMHCN decreases with the increase of threshold η. This is because the larger the η, the sparser the matrix $\mathbf{A}$, and the smaller the amount of convolution operation of the network. On the Modelnet40 dataset, when $\eta = 10^{-2}$, the FLOPs/sample of AMHCN is even slightly smaller than that of HGNN. On the NTU, the FLOPs/sample of AMHCN is equal to that of HGNN when $\eta = 10^{-2}$, and our network achieves better classification accuracy. In general, taking consideration of the classification accuracy, the space and time complexity, our network is much more suitable for applications in practice.

TABLE V: Time and space complexity of deep architectures for 3D data classification.

Method		ModelNet40			NTU		
		#params	FLOPs/sample	Accuracy	#params	FLOPs/sample	Accuracy
HGNN [9]		0.79M	1.61M	96.70%	0.80M	1.61M	89.75%
MHGNN [10]		0.82M	-	97.50%	-	-	-
AMHCN	$\eta = 10^{-2}$	0.80M	1.60M	97.74%	0.80M	1.61M	91.46%
	$\eta = 10^{-3}$	0.80M	1.62M	97.84%	0.80M	1.63M	92.08%
	$\eta = 10^{-6}$	0.80M	2.06M	97.86%	0.80M	2.01M	92.01%

F. Visualization

To gain an intuitive understanding about the representation learning ability of AMHCN, we used T-SNE to project the learned feature representation (i.e., the feature before the last softmax layer) of our network into 2D space. Meanwhile, for comparison, we also embed the original MVCNN feature, GVCNN feature and the learned feature representation of HGNN into 2D space, in which HGNN takes both GVCNN and MVCNN features as input. Fig. 11 shows the t-SNE visualization of these features. The first column denotes the visualization of MVCNN feature. The second column denotes the visualization of GVCNN feature, and the third and fourth columns are the visualizations of HGNN and AMHCN output

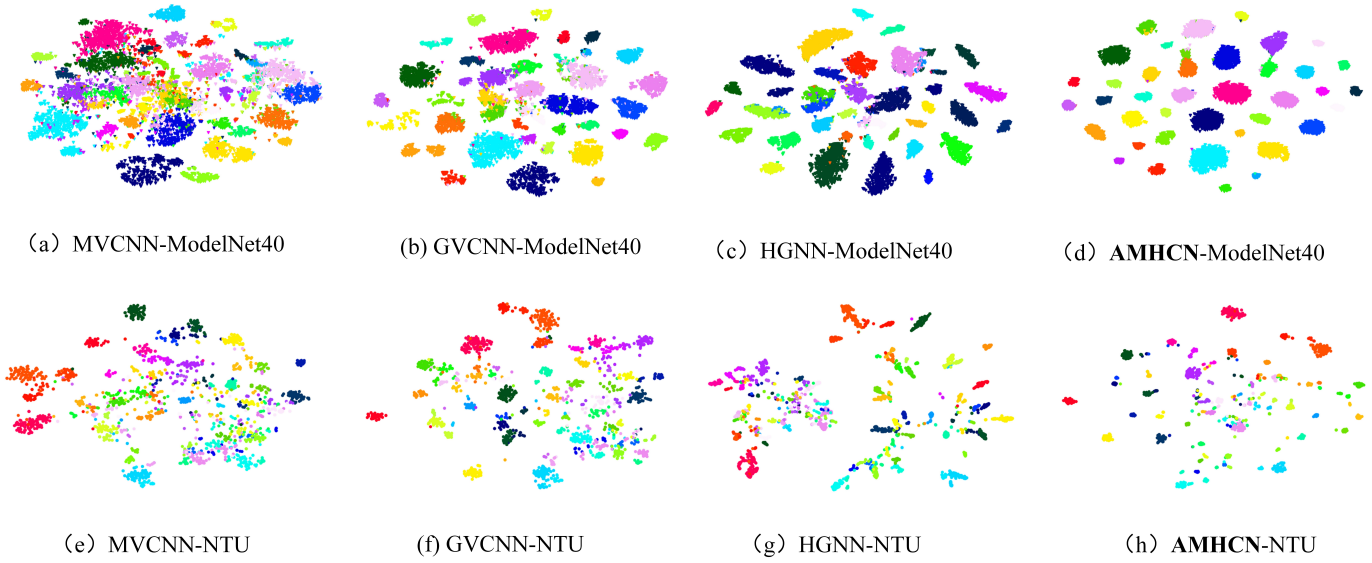


Fig. 11: The t-SNE visualization of features. The points with the same color indicate the same category.

features, respectively. The points with the same color indicate the same category. It can be observed that the extracted features from HGNN and AMHCN are much more discriminative than original MVCNN and GVCNN features. Furthermore, the proposed AMHCN demonstrates superior clustering capability, especially on the ModelNet40 dataset, which can clearly distinguish almost all categories.

V. Conclusion

We have proposed an adaptive multi-hypergraph convolutional networks framework for semi-supervised classification, which not only explores the complex relationships among different objects but also jointly combines multi-modal data. The proposed network can effectively utilize the clustering ability of the PARW on hypergraph to further optimize vertex neighborhood, thus improving the network performance. In addition, the proposed multi-modal fusion strategy guided by the loss function can effectively promote multi-modal fusion with low complexity. These contributions enable us to achieve state-of-the-art classification performance on two 3D benchmark datasets.

There are several potential improvements and extensions to our network that could be addressed as future work. One is to speed up the calculation of the absorption probability matrix of PARW. Although we have adopted LU decomposition to speed up the inversion operation, it is still a relatively large time cost for handling large-scale datasets. In the future, we will explore more strategies for fast matrix inversion. The second is to introduce attention mechanism for improving our PARW-based hypergraph convolution operator. The third is to explore more applications for the proposed network.

References

[1] C.R. Qi, H. Su, K. Mo and L.J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[2] J. Jiang, D. Bao, Z. Chen, X. Zhao, and Y. Gao, "Mlvcnn: Multi-loop-view convolutional neural network for 3d shape retrieval," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, 2019, pp. 8513–8520.

[3] K. Sun, J. Zhang, J. Liu, R. Yu and Z. Song, "Drcnn: Dynamic routing convolutional neural network for multi-view 3d object recognition," *IEEE Trans. Image Process.*, vol. 30, pp. 868–877, 2020.

[4] H. Su, S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.

[5] Y. Feng, Z. Zhang, X. Zhao, R. Ji and Y. Gao, "Gvcnn: Group-view convolutional neural networks for 3d shape recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 264–272.

[6] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2015, pp. 922–928.

[7] A. A. M. Muzahid, W. Wanggen, F. Sohel, M. Bennamoun, L. Hou, and H. Ullah, "Progressive conditional gan-based augmentation for 3d object recognition," *Neurocomputing*, vol. 460, pp. 20–30, 2021.

[8] C.R. Qi, L. Yi, H. Su and L.J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Int. Conf. Neural Inf. Process Syst.*, 2017, pp. 5099–5108.

[9] Y. Feng, H. You, Z. Zhang, R. Ji and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3558–3565.

[10] J. Bai, B. Gong, Y. Zhao, F. Lei, C. Yan and Y. Gao, "Multi-scale representation learning on hypergraph for 3d shape retrieval and recognition," *IEEE Trans. Image Process.*, vol. 30, pp. 5327–5338, 2021.

[11] C. Chen, Z. Cheng, Z. Li and M. Wang, "Hypergraph attention networks," in *Proc. IEEE 19th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, 2020, pp. 1560–1565.

[12] H. Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao, Y. Gao and J. Sun, "Hypergraph-induced convolutional networks for visual classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30(10), pp. 2963–2972, 2019.

[13] S. Liu, P. Lv, Y. Zhang, J. Fu, J. Cheng, W. Li, Z. Bing and M. Xu, "Semi-dynamic hypergraph neural network for 3d pose estimation," in *Proc. IJCAI*, 2020, pp. 782–788.

[14] M.M. Wolf, A.M. Klinvex and D.M. Dunlavy, "Advantages to modeling relational data using hypergraphs versus graphs," in *Proc. IEEE High Perform. Extreme Comput. Conf.*, 2016, pp. 1–7.

[15] S. Bai, F. Zhang and P.H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognit.*, vol. 110, 2021.

[16] J. Shao, J. Zhu, Y. Wei, Y. Feng and X. Zhao, "Emotion recognition by edge-weighted hypergraph neural network," in *Proc. IEEE ICIP*, 2019, pp. 2144–2148.

[17] Z. Xia, W. Zhang and Z. Weng, "Social recommendation system based on hypergraph attention network," *Comput. Intell. Neurosci.*, 2021.

[18] R. SSawhney, S. Agarwal, A. Wadhwa, T. Derr and R. R. Shah, "Stock

selection via spatiotemporal hypergraph attention network: A learning to rank approach," *Proc. AAAI*, pp. 497–504, 2021.

[19] L. Nong, J. Wang, J. Lin, H. Qiu, L. Zheng and W. Zhang, "Hypergraph wavelet neural networks for 3d object classification," *Neurocomputing*, vol. 463, pp. 580–595, 2021.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32(1), pp. 4–24, 2021.

[21] N. Yadati, M. Nimishakavi, P. Yadav, A. Louis and P. Talukdar, "Hypergcn: A new method for training graph convolutional networks on hypergraphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1511–1522.

[22] S. Bandyopadhyay, K. Das and M. N. Murty, "Line hypergraph convolution network: Applying graph convolution for hypergraphs," *arXiv preprint*, 2020.

[23] C. Yang, R. Wang, S. Yao and T. Abdelzaher, "Hypergraph learning with line expansion," in *arXiv preprint arXiv:2005.04843*, 2020.

[24] J. Zhu, X. Zhao, H. Hu and Y. Gao, "Emotion recognition from physiological signals using multi-hypergraph neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo*. IEEE, 2019, pp. 610–615.

[25] K. Ding, J. Wang, J. Li, D. Li and H. Liu, "Be more with less: Hypergraph attention networks for inductive text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 4927–4936.

[26] J. Wang, K. Ding, Z. Zhu and J. Caverlee, "Session-based recommendation with hypergraph attention networks," in *Proc. SDM*, 2021, pp. 82–90.

[27] J. Jiang, Y. Wei, Y. Feng, J. Cao and Y. Gao, "Dynamic hypergraph neural networks," in *Proc. Twenty-Eighth Int. Joint Conf. Artif. Intell.*, 2019, pp. 2635–2641.

[28] T. Jin, Z. Yu, Y. Gao, S. Gao, X. Sun and C. Li, "Robust $l2$-hypergraph and its applications," *Inform. Sci.*, vol. 501, pp. 708–723, 2019.

[29] Z. Zhang, H. Lin and Y. Gao, "Dynamic hypergraph structure learning," in *Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2018, pp. 3162–3169.

[30] M. M. GGarasuie, M. Shabankhah and A. Kamandi, "Improving hypergraph attention and hypergraph convolution networks," in *IKT*, 2020, pp. 67–72.

[31] J. Zhu, X. Zhao, H. Hu and Y. Gao, "Emotion recognition from physiological signals using multi-hypergraph neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, 2019, pp. 610–615.

[32] E. S. Kim, W. Y. Kang, K. W. On, Y. J. Heo and B. T. Zhang, "Hypergraph attention networks for multimodal learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 14 581–14 590.

[33] C. Cui, X. Li, J. Du, C. Zhang, X. Nie, M. Wang and Y. Yin, "Temporal-relational hypergraph tri-attention networks for stock trend prediction," *arXiv preprint arXiv:2107.14033*, 2021.

[34] X. Wu, Z. Li, A. M. So, J. Wright and S.-F. Chang, "Learning with partially absorbing random walks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 3077–3085.

[35] A. Banerjee, "On the spectrum of hypergraphs," *Linear Algebra Appl.*, vol. 614, pp. 82–110, 2021.

[36] A. Eriksson, D. Edler, A. Rojas, M. de Domenico, and M. Rosvall, "How choosing random-walk model and network representation matters for flow-based community detection in hypergraphs," *Commun. Phys.*, vol. 4, no. 1, pp. 1–12, 2021.

[37] T. Carletti, D. Fanelli, and R. Lambiotte, "Random walks and community detection in hypergraphs," *J. Phys. Complexity*, vol. 2, no. 1, pp. 1–13, 2021.

[38] K. Hayashi, S. G. Aksoy, C. H. Park, and H. Park, "Hypergraph random walks, laplacians, and clustering," in *ICS*, 2020, pp. 495–504.

[39] Z. Xie, J. Chen and B. Peng, "Point clouds learning with attention-based graph convolution networks," *Neurocomputing*, vol. 402, pp. 245–255, 2020.

[40] Y. Gao, X. Liu, J. Li, Z. Fang, X. Jiang and K. M. S. Huq, "Lft-net: Local feature transformer network for point clouds analysis," *IEEE Trans. Intell. Transport. Syst.*, 2022.

[41] S. A. Bello, C. Wang, N. M. Wambugu and J. M. Adam, "Ffpointnet: Local and global fused feature for 3d point clouds analysis," *Neurocomputing*, vol. 461, pp. 55–62, 2021.

[42] D. S. W. Nie, Y. Zhao and Y. Gao, "Dan: Deep-attention network for 3d shape recognition," *IEEE Trans. Image Process.*, vol. 30, pp. 4371–4383, 2021.

[43] H. Y. Zhou, A. A. Liu, W. Z. Nie and J. Nie, "Multi-view saliency guided deep neural network for 3-d object retrieval and classification," *IEEE Trans. on Multimedia*, vol. 22, no. 6, pp. 1496–1506, 2020.

[44] A. A. Liu, H. Zhou, W. Nie, Z. Liu, W. Liu, H. Xie, Z. Mao, X. Li and D. Song, "Hierarchical multi-view context modelling for 3d object classification and retrieval," *Inform. Sci.*, vol. 547, pp. 984–995, 2021.

[45] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1912–1920.

[46] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 5648–5656.

[47] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein and J.M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38(5), pp. 1–12, 2019.

[48] T.N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. on Learn. Represantions*, April 24–26, 2017.

[49] M. Liu, J. Zhang, P. Yap and D.Shen, "View-aligned hypergraph learning for alzheimer's disease diagnosis with incomplete multi-modality data," *Med. Image Anal.*, vol. 36, pp. 123–134, 2017.

[50] T. Carletti, F. Battiston, G. Cencetti, and D. Fanelli, "Random walks on hypergraphs," *Phys. Rev. E*, vol. 101, no. 2, p. 022308, 2020.

[51] R. Mulas, C. Kuehn, T. Böhle and J. Jost, "Random walks and laplacians on hypergraphs: When do they match?" *arXiv preprint arXiv:2106.11663*, 2021.

[52] C. Ozcan and B. Sen, "Investigation of the performance of lu decomposition method using cuda," *Procedia Technolog*, vol. 1(1), pp. 50–54, 2012.

[53] D.Y. Chen, X.P. Tian, Y.T. Shen and M. Ouhyoung, "On visual similarity based 3d model retrieval," *Comput. Graph. Forum*, vol. 22(3), pp. 223–232, 2003.

[54] Z. Zhang, H. Lin, X. Zhao, R. Ji and Y. Gao, "Inductive multi-hypergraph learning and its application on view-based 3d object classification," *IEEE Trans. Image Process.*, vol. 27(12), pp. 5957–5968, 2018.

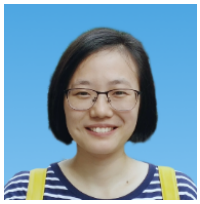
[55] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learn. Represantions*, 2014.

[56] Y. Gao, M. Wang, D. Tao, R. Ji and Q. Dai, "3-D object retrieval and recognition with hypergraph analysis," *IEEE Trans. Image Process.*, vol. 21(9), pp. 4290–4303, 2012.

[57] Z. Zhang, H. Lin, J. Zhu, X. Zhao and Y. Gao, "Cross diffusion on multi-hypergraph for multi-modal 3d object recognition," in *Proc. Pacific Rim Conf. Multimedia*, 2018, pp. 38–49.

[58] Q. Wu, S. Liu, C. Miao, Y. Liu and C. Leung, "A social curiosity inspired recommendation model to improve precision, coverage and diversity," in *IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, 2016, pp. 240–247.

Liping Nong is currently pursuing the Ph.D. degree in the School of Telecommunication Engineering, Xidian University, Xi'an, China. Her research interests include graph signal processing, hypergraph learning and temporal data mining.



Jie Peng is currently pursuing the Ph.D. degree with the School of Telecommunication Engineering, Xidian University, Xi'an, China. Her research interests include mobile and wireless network, resource allocation in mobile cloud computing and stochastic network optimization.

Wenhui Zhang received her M.S. Degree in Mechanical Engineering from Nanjing University of Science and Technology, Nanjing, China, in 2000. She is currently a professor with the school of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China. Her research interests include graph signal processing, hypergraph learning.

Jiming Lin received his Ph.D. degree in acoustics from Nanjing University, China. Subsequently, he held a two years postdoctoral fellowship at state key laboratory for novel software technology at Nanjing University. Since 2004, he has been a professor with the school of Information and Communication, Guilin University of Electronic Technology. From 2012 to 2013, he was a visiting researcher at Advanced Wireless Technologies Lab., Heriot-Watt University, Edinburgh, Scotland. His research interests include graph signal processing and ultra-wideband communications.

Hongbing Qiu received his B.S., M.S. and Ph.D. degree in Information and Communication Engineering from Xidian University, Xi'an, China, in 1984, 1991, and 2004, respectively. Currently, he is a professor of School of Communication and Information Engineering at Guilin University of Electronic Technology, Guilin, China. He is a doctoral supervisor at Xidian University and Guilin University of Electronic Technology. His areas of interest include ultra-wideband wireless communication systems and wireless sensor networks.

Junyi Wang received his M.S. degree in fundamental mathematics from Xiangtan University, and the Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications, Beijing, China, in 2003 and 2008, respectively. He is currently a professor with the Academy of Information and Communication, Guilin University of electronic Technology. His current research interests include graph signal processing, stochastic network optimization and network coding.