A Survey on Hypergraph Neural Networks: An In-Depth and Step-By-Step Guide

Sunwoo Kim*
KAIST
kswoo97@kaist.ac.kr

Soo Yong Lee*
KAIST
syleetolow@kaist.ac.kr

Yue Gao
Tsinghua University
gaoyue@tsinghua.edu.cn

Alessia Antelmi
University of Turin
alessia.antelmi@unito.it

Mirko Polato
University of Turin
mirko.polato@unito.it

Kijung Shin†
KAIST
kijungs@kaist.ac.kr

ABSTRACT

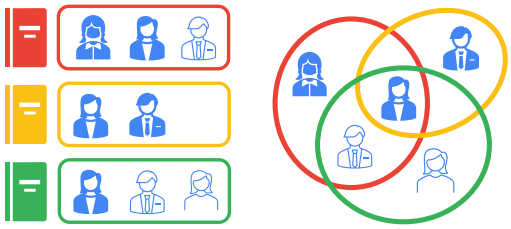
Higher-order interactions (HOIs) are ubiquitous in real-world complex systems and applications, and thus investigation of deep learning for HOIs has become a valuable agenda for the data mining and machine learning communities. As networks of HOIs are expressed mathematically as hypergraphs, hypergraph neural networks (HNNs) have emerged as a powerful tool for representation learning on hypergraphs. Given the emerging trend, we present the first survey dedicated to HNNs, with an in-depth and step-by-step guide. Broadly, the present survey overviews HNN architectures, training strategies, and applications. First, we break existing HNNs down into four design components: (*i*) input features, (*ii*) input structures, (*iii*) message-passing schemes, and (*iv*) training strategies. Second, we examine how HNNs address and learn HOIs with each of their components. Third, we overview the recent applications of HNNs in recommendation, biological and medical science, time series analysis, and computer vision. Lastly, we conclude with a discussion on limitations and future directions.

1 INTRODUCTION

Higher-order interactions (HOIs) are pervasive in real-world complex systems and applications. These relations describe multi-way or group-wise interactions, occurring from physical systems [8–10], microbial communities [87, 89], brain functions [29, 146], and social networks [2, 23, 48], to name a few. HOIs reveal structural patterns unobserved in their pair-wise counterparts and inform network dynamics. They have been shown to affect or correlate with synchronization in physical systems [9], bacteria invasion inhibition in microbial communities [87], cortical dynamics in brains [146], and contagion in social networks [23], just to name a few.

Hypergraphs mathematically express higher-order networks, or networks of HOIs [13], where nodes and hyperedges respectively represent entities and their HOIs. In contrast to an edge connecting only two nodes in pair-wise graphs, a hyperedge can connect any number of nodes, offering hypergraphs advantages in their descriptive power. For instance, as shown in Fig. 1, the co-authorship relations among researchers can be represented as a hypergraph. With their expressiveness and flexibility, hypergraphs have been routinely used to model higher-order networks in various domains [8, 23, 30, 38] to uncover their structural patterns [24, 57, 58, 66–68].

*Equal contribution
†Corresponding author



(a) Co-authors of publications (b) Hypergraph

Figure 1: An example hypergraph modeling the co-authorship relationship among five authors across three publications. Each node represents an author, while each hyperedge includes all co-authors of a publication.

As hypergraphs are extensively used, the demand grew to make predictions on them, such as node property estimation or missing hyperedge identification. Hypergraph neural networks (HNNs) have shown strong promise in solving such problems. For example, they have shown state-of-the-art performances in industrial and scientific applications, including missing metabolic reaction prediction [18], brain classification [50], traffic forecast [154], product recommendation [51], and more [34, 37, 84, 104, 134].

As such, the research on HNNs has been exponentially growing. Data from Scopus demonstrate this trend: the number of peer-reviewed publications on the topic was 5 in 2019, 15 in 2020, 38 in 2021, 83 in 2022, and 150 in 2023.[1] Simultaneously, further research on deep learning for higher-order networks is an imminent agenda for the data mining and machine learning communities. A recent position paper [92], written by authors from 23 different affiliations, underscores the need for research in learning HOIs with neural networks. Specifically, in 7 out of 11 research directions they delineate, the authors call for research in datasets, benchmarks, scalability, theory, and neural network algorithms *specifically designed for higher-order networks.*

Therefore, we provide a timely survey on HNNs that addresses the following questions:

- **Encoding (Sec. 3).** *How do HNNs effectively capture HOIs?*
- **Training (Sec. 4).** *How to encode HOIs with training objectives, especially when train labels are scarce or absent?*
- **Application (Sec. 5).** *What are notable applications of HNNs?*

[1]The research articles are counted if they include "hypergraph neural network," "hypergraph convolution," or "hypergraph attention" in their titles, abstracts, or keywords.

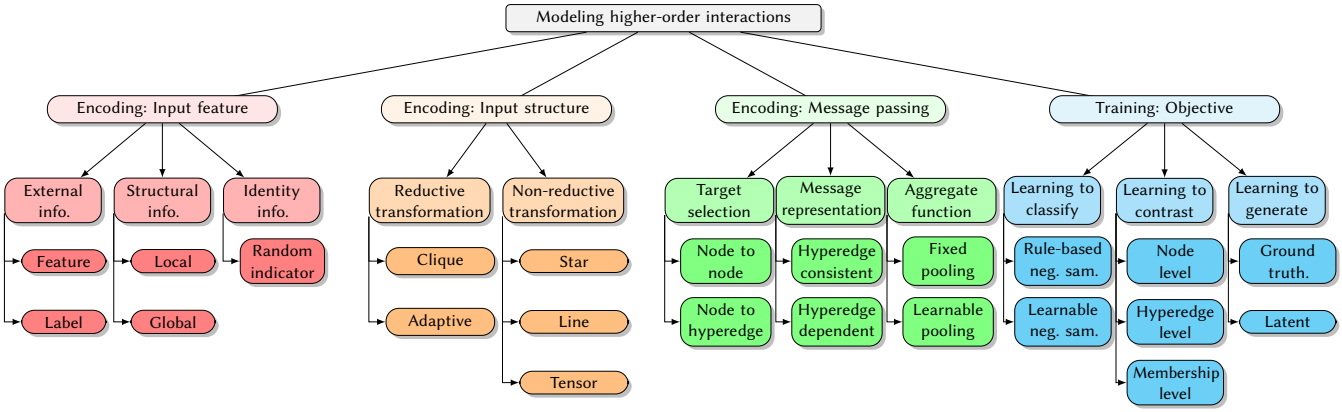


Figure 2: Taxonomy on modeling higher-order interactions. The term neg. sam. denotes negative sampling.

Our scope is largely confined to HNNs for undirected, static, and homogeneous hypergraphs, with node classification or hyperedge prediction as their downstream tasks, primarily because they comprise the majority of the publications on this topic. Compared to the prior surveys on graph and hypergraph learning [3, 36, 93, 109], the present work is marked by two important differences:

- **Step-by-step.** We break HNNs down into four design components: (*i*) input features, (*ii*) input structures, (*iii*) message-passing schemes, and (*iv*) training strategies (see Fig. 2), providing an intuitive framework to understand HNNs.
- **In-depth.** We provide an in-depth analysis of how each component of HNNs is designed to effectively model HOIs (see Table 2).

2 PRELIMINARIES

In this section, we present definitions of basic concepts related to hypergraphs and HNNs. See Table 1 for frequently-used symbols.

A *hypergraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as a set of nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$ and a set of hyperedges $\mathcal{E} = \{e_1, e_2, \cdots, e_{|\mathcal{E}|}\}$. Each hyperedge e_j is a non-empty subset of nodes (i.e., $\emptyset \neq e_j \subseteq \mathcal{V}$). Alternatively, $\mathcal{E}$ can be represented with an incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$ where $\mathbf{H}_{i,j} = 1$ if $v_i \in e_j$ and 0 otherwise. The incident hyperedges of a node v_i, denoted as $\mathcal{N}_{\mathcal{E}}(v_i)$, is the set of hyperedges that contain v_i (i.e., $\mathcal{N}_{\mathcal{E}}(v_i) = \{e_k \in \mathcal{E} : v_i \in e_k\}$). We assume that each node v_i and hyperedge e_j are equipped with (input) node features $\boldsymbol{x}_i \in \mathbb{R}^d$ and hyperedge features $\boldsymbol{y}_j \in \mathbb{R}^{d'}$, respectively.[2] Similarly, we denote node and hyperedge feature matrices as $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and $\mathbf{Y} \in \mathbb{R}^{|\mathcal{E}|\times d'}$, respectively, where the i-th row $\mathbf{X}_i$ corresponds to $\boldsymbol{x}_i$ and j-th row $\mathbf{Y}_j$ corresponds to $\boldsymbol{y}_i$. In Sec. 3.1, we detail approaches to obtain the features.

Hypergraph neural networks (HNNs) are neural functions that transform given nodes, hyperedges, and their features into vector representations (i.e., embeddings) of nodes (and/or hyperedges). Typically, their input is represented as either $(\mathbf{X}, \mathcal{E})$ or $(\mathbf{X}, \mathbf{Y}, \mathcal{E})$. HNNs first transform the input hypergraph structure to a GPU-friendly expression (Sec. 3.2). Then, HNNs perform message passing between nodes (and hyperedges) to update their embeddings (Sec. 3.3). A node (or hyperedge) message roughly refers to its vector representation for some other nodes (and hyperedges) to aggregate.

[2] Sometimes, (external) node and hyperedge features may not be given. In such cases, one may utilize structural or identity features, as described in Sec. 3.1.

Table 1: Frequently-used symbols

Notation	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Hypergraph with nodes set $\mathcal{V}$ and hyperedges set $\mathcal{E}$
$\mathbf{H} \in \{0,1\}^{\lvert\mathcal{V}\rvert\times\lvert\mathcal{E}\rvert}$	Incidence matrix
$\mathbf{X} \in \mathbb{R}^{\lvert\mathcal{V}\rvert\times d}$, $\mathbf{Y} \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times d'}$	Node features ($\mathbf{X}$) and hyperedge features ($\mathbf{Y}$)
$\mathbf{P}^{(\ell)} \in \mathbb{R}^{\lvert\mathcal{V}\rvert\times k}$, $\mathbf{Q}^{(\ell)} \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times k'}$	ℓ-th layer embeddings of nodes ($\mathbf{P}^{(\ell)}$) and hyperedges ($\mathbf{Q}^{(\ell)}$)
$\mathcal{N}_{\mathcal{E}}(v_i)$	Incident hyperedges of node v_i
$\mathbf{I}_n$	n-by-n identity matrix
$\mathbb{I}[\text{cond}]$	Indicator function that returns 1 if cond is True, 0 otherwise
$\Vert$	Vector concatenation
$\odot$	Elementwise product
$\sigma(\cdot)$	Non-linear activation function
$\lvert\mathcal{A}\rvert$	Cardinality of set $\mathcal{A}$ (i.e., the number of elements in $\mathcal{A}$)
$[a]$	Set of natural numbers that are smaller than or equal to a.
$\mathbf{M}_{i,:} \coloneqq \boldsymbol{m}_i$	i-th row of matrix $\mathbf{M}$
$\mathbf{M}_{i,j} \coloneqq m_{ij}$	(i,j)-entry of matrix $\mathbf{M}$

The message-passing operation is repeated L times, where each iteration corresponds to one HNN layer. Here, we denote the ℓ-th layer embedding matrix of nodes and hyperedges as $\mathbf{P}^{(\ell)} \in \mathbb{R}^{|\mathcal{V}|\times k}$ and $\mathbf{Q}^{(\ell)} \in \mathbb{R}^{|\mathcal{E}|\times k'}$, respectively. Unless otherwise stated, we assume $\mathbf{P}^{(0)} = \mathbf{X}$ and $\mathbf{Q}^{(0)} = \mathbf{Y}$. We use $\mathbf{I}_n$, $\Vert$, $\odot$, and $\sigma(\cdot)$ to denote the n-by-n identity matrix, vector concatenation, elementwise product, and a non-linear activation function, respectively.

3 ENCODER DESIGN GUIDANCE

In this section, we provide a step-by-step description of how HNNs encode higher-order interactions (HOIs) to effectively capture them.

3.1 Step 1: Design features to reflect HOIs

First, HNNs require a careful choice of input node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and hyperedge features $\mathbf{Y} \in \mathbb{R}^{|\mathcal{E}|\times d'}$. The quality of input features can be vital for a successful application of HNNs [69, 151]. Thus, studies have strategically crafted input features to enhance the encoding of HOIs within HNNs. Three primary approaches include the use of (*i*) external features or labels, (*ii*) structural features, and (*iii*) identity features.

3.1.1 ***External features or labels.*** External features or labels broadly refer to information that is not directly obtained from the hypergraph structure. Using external features allows HNNs to capture information that may not be transparent in hypergraph structure alone. When available, using external node features $\mathbf{X}$ and hyperedge features $\mathbf{Y}$ as HNN input is the standard practice.

Some examples of node features from widely-used benchmark datasets are bag-of-words vectors [137], TF-IDFs [27], visual object

embeddings [32], or noised label vectors [19]. Interestingly, as in label propagation, HyperND [96] constructs input node features $\mathbf{X}$ by concatenating external node features with label vectors. Specifically, one-hot-encoded label vectors and zero vectors are concatenated for nodes with known and unknown labels, respectively. Since the benchmark datasets typically lack external hyperedge features, in practice, input features of e_j have often been obtained by averaging its constituent nodes (i.e., $\boldsymbol{y}_j = \sum_{v_k \in e_j} \boldsymbol{x}_k / |e_j|$) [139].

3.1.2 ***Structural features.*** On top of external features, studies have also utilized structural features as HNN input features. Structural features are typically derived from the input hypergraph structure $\mathcal{E}$ to capture structural proximity or similarity between nodes. While leveraging them in addition to the structure $\mathcal{E}$ may seem redundant, several studies have highlighted their theoretical and empirical advantages, particularly for hyperedge prediction [115] and when transformer-based models are employed [21, 82, 101].

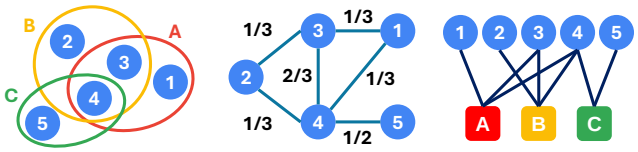
Broadly speaking, studies have leveraged either local or global structural features. To capture local structures around each node, some HNNs use the incidence matrix $\mathbf{H}$ as part of the input features [82, 115, 151]. Notably, HyperGT [82] parameterizes its structural node features $\mathbf{X}' \in \mathbb{R}^{|\mathcal{V}| \times k}$ and hyperedge features $\mathbf{Y}' \in \mathbb{R}^{|\mathcal{E}| \times k}$ as follows: $\mathbf{X}' = \mathbf{H}\boldsymbol{\Theta}$ and $\mathbf{Y}' = \mathbf{H}^T\boldsymbol{\Phi}$, where $\boldsymbol{\Theta} \in \mathbb{R}^{|\mathcal{E}| \times k}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{|\mathcal{V}| \times k}$ are learnable weight matrices. Some HNNs leverage structural patterns within each hyperedge. Intuitively, the importance or role of each node may vary depending on hyperedges. For instance, WHATsNet [21] uses within-order positional encoding, where node centrality order within each hyperedge serves as edge-dependent node features (detailed in Sec. 3.3.2). Also, a study [88] utilizes the occurrence of each hypergraphlet (i.e., a predefined pattern of local structures describing the overlaps of hyperedges within a few hops) around each node or hyperedge as input features. Global features based on roles and proximity in the entire hypergraph context have also been adopted. For example, Hyper-SAGNN [151] uses a Hyper2Vec [44] variant to incorporate structural features preserving node proximity, and VilLain [69] employs, as input node features, distributions over node labels inferred from the structure. THTN [101] integrates learnable node centrality, uniqueness, and positional encodings.

3.1.3 ***Identity features.*** Some HNNs use identity features, especially for recommendation applications. Generally, identity features refer to features uniquely assigned to each node (and hyperedge), enabling HNNs to learn distinct embeddings for each node (and hyperedge) [145, 160]. Prior studies have typically used randomly generated features or separately learnable ones [51, 129–131].

3.2 Step 2: Express hypergraphs to reflect HOIs

Some HNNs transform the input hypergraph structure to better capture the underlying HOIs. They employ either (*i*) reductive or (*ii*) non-reductive expressions of hypergraph structures (See Fig. 3).

3.2.1 ***Reductive transformation.*** One way to represent hypergraph structure is through *reductive transformation*. In this approach, each node from the original hypergraph is preserved as a node in the graph, while hyperedges are transformed into pairwise edges (see Fig. 3(b)). Reductive transformation enables the direct



(a) Hypergraph (b) Clique-expanded graph with edge weights of Eq. (1) (c) Star-expanded graph

Figure 3: An example hypergraph (a), its clique-expanded graph (b), and its star-expanded graph (c).

application of methods developed for graphs, such as spectral filters [32], to hypergraphs. However, it may result in information loss, and the original hypergraph structure may not be precisely recovered after transformation. Reductive transformation includes two approaches: clique and adaptive expansion. Each expansion is represented as $\tau : (\mathcal{E}, \mathbf{X}, \mathbf{Y}) \mapsto \mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. We elaborate on the definition of each entry a_{ij} of $\mathbf{A}$ for both expansions.

Clique expansion. Clique expansion converts each hyperedge $e_j \in \mathcal{E}$ into a clique (i.e., complete subgraph) formed by the set e_j of nodes (see Fig. 3(b)). Consider two distinct hypergraphs: ($e_1 = \{v_1, v_2, v_3\}$) and ($e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_1, v_3\}$, and $e_3 = \{v_2, v_3\}$). Despite their changes, both result in identical clique-expanded graph ($e_1 = \{v_1, v_2\}$, $e_2 = \{v_1, v_3\}$, and $e_3 = \{v_2, v_3\}$) if edges are unweighted. This example illustrates that, in clique expansion, assigning proper edge weights is crucial for capturing HOIs. Edge weighting approaches [32, 99, 106] have employed (*i*) the co-occurrence of node pairs so that pairs appearing together more frequently in hyperedges should be assigned larger weights, and (*ii*) hyperedge sizes so that node pairs in larger hyperedges are assigned smaller weights than those in smaller hyperedges. An example [106] is

$$a_{ij} = \sum\nolimits_{e_k \in \mathcal{E}} \frac{\delta(v_i, v_j, e_k)}{|e_k|}, \tag{1}$$

where $\delta(v_i, v_j, e_k) = \mathbb{I}[(\{v_i, v_j\} \subseteq e_k) \wedge (i \neq j)]$, and $\mathbb{I}[\texttt{cond}]$ is an indicator function that returns 1 if cond is `True` and 0 otherwise.

Adaptive expansion. Within each transformed clique, some edges may be redundant or even unhelpful. Adaptive expansion selectively adds edges within each clique, often tailored to a specific downstream task [97, 137]. For example, AdE [97] employs a feature-distance-based weighting strategy. AdE first obtains projected node features $\mathbf{X}' \in \mathbb{R}^{|\mathcal{V}| \times d}$ by $\mathbf{X}' = \mathbf{X} \odot \mathbf{W}$ where every row vector of $\mathbf{W}$ is the same as $\boldsymbol{w} \in \mathbb{R}^d$; and $\boldsymbol{w} = \texttt{sigmoid}(\sigma((\sum_{v_k \in \mathcal{V}} \boldsymbol{x}_k)\boldsymbol{\Theta}^{(1)})\boldsymbol{\Theta}^{(2)})$, where $\boldsymbol{\Theta}^{(1)}$ and $\boldsymbol{\Theta}^{(2)}$ are learnable weight matrices. Then, AdE selects two distant nodes $v_{i,j}$ and $v_{k,j}$ within each hyperedge e_j, i.e., $\{v_{i,j}, v_{k,j}\} = \arg\max_{\{v_i, v_k\} \in \binom{e_j}{2}} |\sum_{t=1}^{d} (\mathbf{X}'_{i,t} - \mathbf{X}'_{k,t})|$. After that, it connects the all nodes in e_j with $v_{i,j}$ and $v_{k,j}$, essentially adding $\mathcal{E}'_j = \{\{v_{i,j}, v_t\} : v_t \in e_j \setminus \{v_{i,j}\}\} \cup \{\{v_{k,j}, v_t\} : v_t \in e_j \setminus \{v_{k,j}\}\}$. Lastly, AdE assign weights to each edge in $\mathcal{E}'_j$ as follows:

$$a_{ik} = \sum_{e_j \in \mathcal{E}} \frac{\mathbb{I}[\{v_i, v_k\} \in \mathcal{E}'_j] \xi(i,k)}{\sum_{\{v_s, v_t\} \in \binom{e_j}{2}} \xi(s,t)}, \tag{2}$$

where $\xi(i,k) = \exp\left(\|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2 \sum_{t=1}^{d} (\mathbf{X}'_{i,t} - \mathbf{X}'_{k,t})^2 / \theta_t^2\right)$, and θ_t, $\forall t \in [d]$ are learnable scalars.

简化的；减少的

3.2.2 ***Non-reductive transformation.*** Non-reductive transformation of hypergraph structure includes star expansion [19, 21, 102, 122], line expansion [142], and tensor representation [55, 116, 121]. They express hypergraph structure without information loss; that is, the hyperedges $\mathcal{E}$ can be exactly recovered after transformation.

Star expansion. A *star-expanded graph* of a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has two new groups of nodes: the *node group*, which is the same as the node set $\mathcal{V}$ of $\mathcal{G}$, and the *hyperedge group*, consisting of nodes corresponding to the hyperedges $\mathcal{E}$ (refer to Fig. 3(c)). Star expansion captures HOIs by connecting each node (i.e., a node from the node group) with the hyperedges (i.e., nodes from the hyperedge group) it belongs to, resulting a bipartite graph between the two groups. Star expansion is expressed as $\tau : (\mathcal{E}, \mathbf{X}, \mathbf{Y}) \mapsto \mathbf{A}$, where each entry of $\mathbf{A} \in \mathbb{R}^{(|\mathcal{V}|+|\mathcal{E}|)\times(|\mathcal{V}|+|\mathcal{E}|)}$ is defined as

$$a_{ij} = \begin{cases} \mathbb{I}[v_i \in e_{j-|\mathcal{V}|}], & \text{if } 1 \le i \le |\mathcal{V}| < j \le |\mathcal{V}| + |\mathcal{E}|, \\ \mathbb{I}[v_j \in e_{i-|\mathcal{V}|}], & \text{if } 1 \le j \le |\mathcal{V}| < i \le |\mathcal{V}| + |\mathcal{E}|, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here, we assume WLOG that the corresponding index of $v_i \in \mathcal{V}$ in $\mathbf{A}$ is i, and the corresponding index of $e_j \in \mathcal{E}$ in $\mathbf{A}$ is $|\mathcal{V}| + j$.

Line expansion. In a line-expanded graph [142] of a hypergraph of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each *pair* of a node and a hyperedge containing it is represented as a distinct node. That is, its node set is $\{(v_i, e_j) : v_i \in e_j, e_j \in \mathcal{E}\}$. Edges are established between these nodes to connect every pair of distinct nodes (v_i, e_j) and (v_k, e_l) where $i = k$ or $j = l$.

Tensor expression. Several recent HNNs represent hypergraphs as tensors [55, 116, 121]. For example, T-HyperGNNs [116] expresses a $k-$uniform (i.e., $|e_j| = k, \forall e_j \in \mathcal{E}$) hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a $k-$order tensor $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|^k}$. That is, if $k = 3$, $\mathcal{A}_{i,j,k} = 1$ if $\{v_i, v_j, v_k\} \in \mathcal{E}$, and $\mathcal{A}_{i,j,k} = 0$ otherwise.

3.3 Step 3: Pass messages to reflect HOIs

With input features (Sec. 3.1) and structure (Sec. 3.2), HNNs learn node (and hyperedge) embeddings. They use *neural message passing functions* for each node (and hyperedge) to aggregate information (messages) from other nodes (and hyperedges). Three questions arise: (*i*) *whose* messages should be aggregated? (*ii*) *what* messages should be aggregated? (*iii*) *how* should they be aggregated?

3.3.1 ***Whose messages to aggregate (target selection).*** For message passing, we should decide whose message to aggregate, typically based on the structural expression of the input hypergraph (Sec. 3.2). We provide three representative examples: one clique-expansion-based approach and two star-expansion-based ones.[3]

On clique-expanded graphs ($\mathcal{V} \to \mathcal{V}$). Similar to typical graph neural networks, clique-expansion-based HNNs perform message passing between neighboring nodes [7, 11, 32, 39, 96, 106, 137]. They also often incorporate techniques that are effective in applying graph neural networks. This is unsurprising since clique expansion transforms a hypergraph into a homogeneous graph. A notable instance is WCE-GNN [106], which constructs a propagation matrix $\mathbf{W}$ from $\mathbf{A}$ (see Eq. (1)) using a re-normalization trick [62] as follows:

$$\mathbf{W} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{4}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_{|\mathcal{V}|}$, and $\tilde{\mathbf{D}}$ is the diagonal degree matrix, i.e., $\tilde{\mathbf{D}}_{i,i} = \sum_{k=1}^{|\mathcal{V}|} \tilde{\mathbf{A}}_{i,k}$. Then, node embeddings at each ℓ-th layer are updated using $\mathbf{W}$ as follows:

$$\mathbf{P}^{(\ell)} = \sigma\left(((1-\alpha_\ell)\mathbf{W}\mathbf{P}^{(\ell-1)} + \alpha_\ell \mathbf{P}^{(0)})((1-\beta_\ell)\mathbf{I}_k + \beta_\ell \mathbf{\Theta}^{(\ell)})\right), \tag{5}$$

where $\alpha_\ell, \beta_\ell \in [0, 1]$ are hyperparameters, $\mathbf{\Theta}^{(\ell)} \in \mathbb{R}^{k\times k}$ is a learnable weight matrix, and $\mathbf{P}^{(0)} = \mathtt{MLP}(\mathbf{X})$.

On star-expanded graphs ($\mathcal{V} \to \mathcal{E}$ and $\mathcal{E} \to \mathcal{V}$). In HNNs based on star expansion, message passing occurs from the node group to the hyperedge group ($\mathcal{V} \to \mathcal{E}$) and vice versa ($\mathcal{E} \to \mathcal{V}$) [19, 21, 25, 122, 139], either sequentially or simultaneously.

First, we illustrate sequential message passing using ED-HNN [122]. Its message passing at each ℓ-th layer for each node $v_i \in \mathcal{V}$ is formalized as follows:

$$\boldsymbol{q}_j^{(\ell)} = \sum_{v_k \in e_j} \mathtt{MLP}_1\left(\boldsymbol{p}_k^{(\ell-1)}\right), \tag{6}$$

$$\boldsymbol{r}_i^{(\ell)} = \sum_{e_k \in \mathcal{N}_\mathcal{E}(v_i)} \mathtt{MLP}_2\left(\left[\boldsymbol{p}_i^{(\ell)} \| \boldsymbol{q}_k^{(\ell)}\right]\right), \tag{7}$$

$$\boldsymbol{p}_i^{(\ell)} = \mathtt{MLP}_3\left(\left[\boldsymbol{p}_i^{(\ell-1)} \| \boldsymbol{r}_i^{(\ell)} \| \boldsymbol{x}_i \oplus |\mathcal{N}_\mathcal{E}(v_i)|\right]\right), \tag{8}$$

where $\boldsymbol{x} \oplus c$ denotes the concatenation of vector $\boldsymbol{x}$ and scalar c; and $\mathtt{MLP}_1$, $\mathtt{MLP}_2$, and $\mathtt{MLP}_3$ are MLPs shared across all layers. Note that, in Eq. (6), hyperedge embeddings are updated by aggregating the embeddings of their constituent nodes. Subsequently, in Eq. (7) and Eq. (8), node embeddings are updated by aggregating transformed embeddings of incident hyperedges. Note that message passing in distinct directions (Eq. (6) and Eq. (7)) occurs sequentially.

Second, we present an example of simultaneous message passing with HDS^{ode} [139]. Its message passing at each ℓ-th layer for each node $v_i \in \mathcal{V}$ and each hyperedge $e_j \in \mathcal{E}$ is formalized as follows:

$$\boldsymbol{r}_i^{(\ell)} = \boldsymbol{p}_i^{(\ell-1)} + \sigma(\boldsymbol{p}_i^{(\ell-1)}\mathbf{\Theta}_{(v)} + \boldsymbol{b}_{(v)}), \tag{9}$$

$$\boldsymbol{q}_j^{(\ell)} = \boldsymbol{q}_j^{(\ell-1)} + \sigma(\boldsymbol{q}_j^{(\ell-1)}\mathbf{\Theta}_{(e)} + \boldsymbol{b}_{(e)}), \tag{10}$$

$$\boldsymbol{p}_i^{(\ell)} = (1-\alpha_{(v)})\boldsymbol{r}_i^{(\ell)} + \frac{\alpha_{(v)}}{|\mathcal{N}_\mathcal{E}(v_i)|} \sum\nolimits_{e_l \in \mathcal{N}_\mathcal{E}(v_i)} \boldsymbol{q}_l^{(\ell)}, \tag{11}$$

$$\boldsymbol{q}_j^{(\ell)} = (1-\alpha_{(e)})\boldsymbol{q}_i^{(\ell)} + \frac{\alpha_{(e)}}{|e_j|} \sum\nolimits_{v_l \in e_j} \boldsymbol{r}_l^{(\ell)}, \tag{12}$$

where $\alpha_{(v)}, \alpha_{(e)} \in [0, 1]$ are hyperparameters, $\mathbf{\Theta}_{(v)}, \mathbf{\Theta}_{(e)} \in \mathbb{R}^{k\times k}$ are learnable weight matrices, and $\boldsymbol{b}_{(v)}, \boldsymbol{b}_{(e)} \in \mathbb{R}^k$ are learnable biases. After projecting node and hyperedge embeddings (Eq. (9) and Eq. (10)), each node embedding is updated by aggregating the projected embeddings of its incident hyperedge (Eq. (11)), and each hyperedge embedding is updated by aggregating the projected embeddings of its constituent nodes (Eq. (12)). Note that message passing in both directions (Eq. (11) and Eq. (12)) happens simultaneously; and updates (Eq. (11) and Eq. (12) follow after both ends.

3.3.2 ***What messages to aggregate (message representation).*** After choosing message targets, the next step is determining *message representations*. HNNs typically use embeddings from the previous layer as messages, which we term *hyperedge-consistent messages* [25, 45]. In contrast, several recent studies propose adaptive

[3] Regarding target selection, adaptive-expansion- [98], line-expansion- [141] and tensor-representation-based [121] are similar to clique-expanded ones ($\mathcal{V} \to \mathcal{V}$).

message transformation based on its target, which we refer to as *hyperedge-dependent messages* [4, 21, 108].

Hyperedge-consistent messages. In this widely-used approach [19, 45, 139], embeddings from the previous layer are directly treated as vector messages. A notable example is UniGNN [45], a family of HNNs that obtain node (and hyperedge) embeddings by aggregating the embeddings from its incident hyperedges (or constituent nodes). UniGIN, a special case of UniGNN, is formalized as follows:

$$\boldsymbol{q}_j^{(\ell)} = \sum_{v_l \in e_j} \boldsymbol{p}_k^{(\ell-1)} \; ; \; \boldsymbol{p}_i^{(\ell)} = \left((1+\epsilon)\boldsymbol{p}_i^{(\ell-1)} + \sum_{e_l \in \mathcal{N}_{\mathcal{E}}(v_i)} \boldsymbol{q}_l^{(\ell)} \right) \boldsymbol{\Theta}^{(\ell)},$$

where $\epsilon \in \mathbb{R}$ can either be a learnable or fixed scalar, and $\boldsymbol{\Theta}^{(\ell)} \in \mathbb{R}^{k \times k'}$ is a learnable weight matrix.

Hyperedge-dependent messages. The role or importance of a node may vary across the hyperedges it belongs to [20, 21]. Several studies [4, 21, 108] have devised hyperedge-dependent node messages, enabling a node to send tailored messages to each hyperedge it belongs to. For example, MultiSetMixer [108] learns different node messages for each incident hyperedge to aggregate, with the following specific message passing function:

$$\boldsymbol{q}_j^{(\ell)} = \frac{1}{|e_j|} \sum_{v_k \in e_j} \boldsymbol{p}_{k,j}^{(\ell-1)} + \mathtt{MLP}_1^{(\ell)} \left(\mathtt{LN} \left(\frac{1}{|e_j|} \sum_{v_k \in e_j} \boldsymbol{p}_{k,j}^{(\ell-1)} \right) \right), \quad (13)$$

$$\boldsymbol{p}_{i,j}^{(\ell)} = \boldsymbol{p}_{i,j}^{(\ell-1)} + \mathtt{MLP}_2^{(\ell)} \left(\mathtt{LN} \left(\boldsymbol{p}_{i,j}^{(\ell-1)} \right) \right) + \boldsymbol{q}_j^{(\ell)}, \quad (14)$$

where $\boldsymbol{p}_{i,j}^{(\ell)}$ is the ℓ-th layer message of v_i that is dependent on e_j, $\mathtt{MLP}_1^{(\ell)}$ and $\mathtt{MLP}_2^{(\ell)}$ are MLPs, and $\mathtt{LN}$ is layer normalization [5].

Alternatively, some HNNs update messages based on hyperedge-depdendent node features. WHATsNet [21] introduces within-order positional encoding ($\mathtt{wope}$) to adapt node messages for each target. Within each hyperedge, it ranks constituent nodes according to their centralities for positional encoding. Formally, let $\mathbf{F} \in \mathbb{R}^{|\mathcal{V}| \times T}$ be a node centrality matrix, where T denotes the number of centrality measures (e.g., node degree), and $\mathbf{F}_{i,t}$ denotes the t-th centrality measure score of node v_i. The order of an element c in a set C is defined as $\mathtt{Order}(c, C) = \sum_{c' \in C} \mathbb{I}[c' \leq c]$. Then, $\mathtt{wope}$ of a node v_i at a hyperedge e_j is defined as follows:

$$\mathtt{wope}(v_i, e_j) = \Big\|_{t=1}^{T} \frac{1}{|e_j|} \mathtt{Order}(\mathbf{F}_{i,t}, \{\mathbf{F}_{i,t} : v_i \in e_j\}). \quad (15)$$

Finally, hyperedge-dependent node messages are defined as follows:

$$\boldsymbol{p}_{i,j}^{(\ell)} = \boldsymbol{p}_i^{(\ell)} + \mathtt{wope}(v_i, e_j)\boldsymbol{\Psi}^{(\ell)}, \quad (16)$$

where $\boldsymbol{\Psi}^{(\ell)} \in \mathbb{R}^{T \times k}$ is a learnable projection matrix.[4]

3.3.3 *How to aggregate messages (aggregation function).*

The last step is to decide how to aggregate the received messages for each node (and hyperedge). We can use fixed or learnable pooling.

[4] Similarly, each hyperedge e_j's message to each node v_i at the ℓ-th layer is defined as $\boldsymbol{q}_{j,i}^{(\ell)} = \boldsymbol{q}_j^{(\ell)} + \mathtt{wope}(v_i, e_j)\boldsymbol{\Psi}^{(\ell)}$. WHATsNet aggregates $\{q_{k,i}^{(\ell)} : e_k \in \mathcal{N}_{(\mathcal{E})}(v_i)\}$ to obtain $\mathbf{p}_i^{(\ell)}$ via set attention proposed by Lee et al. [71]. We omit the detailed message passing function since we focus on describing how dependent messages are obtained.

Fixed pooling. Many HNNs use fixed pooling functions, including summation [45, 122] or average [35, 123]. For example, ED-HNN [122] uses summation to aggregate the embeddings of constituent nodes (or incident hyperedges), as described in Eq. (6) and Eq. (7). Clique-expansion-based HNNs without adaptive edge weights also fall into this category [99, 106]. For example, WCE-GNN [106] uses a fixed propagation matrix $\mathbf{W}$ (see Eq. (5)) to aggregate node embeddings, as follows: $\boldsymbol{p}_i^{(\ell)} = \sum_{v_k \in \mathcal{V}} \mathbf{W}_{i,j} \boldsymbol{p}_k^{(\ell-1)}$.

Learnable pooling. Several recent HNNs enhance their pooling functions through attention mechanisms, allowing for weighting messages during aggregation. Two prominent styles are *target-agnostic attention* [17, 19] and *target-aware attention* [21, 102].

Target-agnostic attention functions consider the relations among messages themselves. AllSetTransformer [19] is an example. Denote the embeddings of the incident hyperedges of v_i at each ℓ-th layer as $\mathcal{S}^{(\ell)}(v_i) \coloneqq \{\boldsymbol{q}_k^{(\ell)} : e_k \in \mathcal{N}_{\mathcal{E}}(v_i)\}$ and its matrix expression as $\mathrm{S}^{(\ell,i)} \in \mathbb{R}^{|\mathcal{S}^{(\ell)}(v_i)| \times k}$. Then, $\boldsymbol{p}_i^{(\ell)}$ is derived from $\mathrm{S}^{(\ell,i)}$ as follows:

$$\mathtt{MH}(\boldsymbol{\theta}, \mathrm{S}) = \Big\|_{t=1}^{h} \left(\omega \left(\theta_t \left(\mathtt{MLP}_{t,1}^{(\ell)}(\mathrm{S}) \right)^T \right) \mathtt{MLP}_{t,2}^{(\ell)}(\mathrm{S}) \right), \quad (17)$$

$$\boldsymbol{p}_i^{(\ell)} = \mathtt{LN}\left(\boldsymbol{\mathcal{H}}_i^{(\ell)} + \mathtt{MLP}_3^{(\ell)} \left(\boldsymbol{\mathcal{H}}_i^{(\ell)} \right) \right); \boldsymbol{\mathcal{H}}_i^{(\ell)} = \mathtt{LN}\left(\boldsymbol{\theta} + \mathtt{MH}\left(\boldsymbol{\theta}, \mathrm{S}^{(\ell,i)} \right) \right),$$

where $\mathtt{LN}$ is layer normalization [5], $\omega(\cdot)$ is row-wise softmax, $\boldsymbol{\theta} = \|_{t=1}^{T} \theta_t$ is a learnable vector; and $\mathtt{MLP}_{t,1}$, $\mathtt{MLP}_{t,2}$, and $\mathtt{MLP}_3$ are MLPs. Note that Eq. (17) is a widely-used multi-head attention operation [111], where $\boldsymbol{\theta}$ serves as queries, and S serves as keys and values. This process is target agnostic since it considers only the global variables $\boldsymbol{\theta}$ and the embeddings S of incident hyperedges, without considering the embedding of the target v_i itself.

In target-aware attention approaches, target information is incorporated to compute attention weights. HyGNN [102] is an example with the following message passing function:

$$\boldsymbol{p}_i^{(\ell)} = \sigma\left(\sum_{e_k \in \mathcal{N}_{\mathcal{E}}(v_i)} \frac{\mathtt{Att}_{(\mathcal{V})}^{(\ell)}(\boldsymbol{q}_k^{(\ell-1)}, \boldsymbol{p}_i^{(\ell-1)}) \boldsymbol{q}_k^{(\ell-1)} \boldsymbol{\Theta}^{(\ell,1)}}{\sum_{e_s \in \mathcal{N}_{\mathcal{E}}(v_i)} \mathtt{Att}_{(\mathcal{V})}^{(\ell)}(\boldsymbol{q}_s^{(\ell-1)}, \boldsymbol{p}_i^{(\ell-1)})} \right), \quad (18)$$

$$\boldsymbol{q}_j^{(\ell)} = \sigma\left(\sum_{v_k \in e_j} \frac{\mathtt{Att}_{(\mathcal{E})}^{(\ell)}(\boldsymbol{p}_k^{(\ell)}, \boldsymbol{q}_j^{(\ell-1)}) \boldsymbol{p}_k^{(\ell)} \boldsymbol{\Theta}^{(\ell,2)}}{\sum_{v_s \in e_j} \mathtt{Att}_{(\mathcal{E})}^{(\ell)}(\boldsymbol{p}_s^{(\ell)}, \boldsymbol{q}_j^{(\ell-1)})} \right). \quad (19)$$

Here, $\mathtt{Att}_{(\mathcal{V})}^{(\ell)}(\boldsymbol{q}, \boldsymbol{p}) = \sigma(\boldsymbol{q}^T \boldsymbol{\psi}_1^{(\ell)} \times \boldsymbol{p}^T \boldsymbol{\psi}_2^{(\ell)}) \in \mathbb{R}$ and $\mathtt{Att}_{(\mathcal{E})}^{(\ell)}(\boldsymbol{p}, \boldsymbol{q}) = \sigma(\boldsymbol{p}^T \boldsymbol{\psi}_3^{(\ell)} \times \boldsymbol{q}^T \boldsymbol{\psi}_4^{(\ell)}) \in \mathbb{R}$ are attention weight functions, where $\{\boldsymbol{\psi}_1^{(\ell)}, \boldsymbol{\psi}_2^{(\ell)}, \boldsymbol{\psi}_3^{(\ell)}, \boldsymbol{\psi}_4^{(\ell)}\}$ and $\{\boldsymbol{\Theta}^{(\ell,1)}, \boldsymbol{\Theta}^{(\ell,2)}\}$ are sets of learnable vectors and matrices, respectively. Note that the attention weight functions consider messages from both sources and targets. Target-aware attention has also been incorporated into clique-expansion-based HNNs, with HCHA [7] as a notable example.

4 OBJECTIVE DESIGN GUIDANCE

In this section, we outline training objectives for HNNs to capture HOIs effectively, particularly when label supervision is weak or absent. Below, we review three branches: (*i*) learning to classify, (*ii*) learning to contrast, and (*iii*) learning to generate.

Table 2: Summary of hypergraph neural networks (HNNs).

Name	Year	Venue	(Structure) Reductive? Yes	(Structure) Reductive? No	(Embedding Type) Edge Dependent? Yes	(Embedding Type) Edge Dependent? No	(Aggregation) Learnable? Yes	(Aggregation) Learnable? No
HGNN [32]	2019	AAAI	✔			✔		✔
HyperGCN [137]	2019	NeurIPS	✔			✔	✔	
HCHA [137]	2019	Pat. Rec.	✔			✔	✔	
HNHN [25]	2020	ICML		✔		✔		✔
UniGNN [45]	2021	IJCAI		✔		✔	✔	
HO Transformer [56]	2021	NeurIPS		✔		✔	✔	
AllSet [19]	2022	ICLR		✔		✔	✔	
HyperND [96]	2022	ICML	✔			✔	✔	
H-GNN [150]	2022	ICML	✔			✔		✔
EHNN [55]	2022	ECCV		✔		✔	✔	
LE$_{GCN}$ [141]	2022	CIKM		✔		✔		✔
HERALD [150]	2022	ICASSP	✔			✔	✔	
HGNN+ [35]	2022	TPAMI		✔		✔		✔
HNN [4] *	2022	arXiv		✔	✔			✔
ED-HNN [122]	2023	ICLR		✔		✔		✔
PhenomNN [125]	2023	ICML	✔			✔	✔	
WHATsNet [21]	2023	KDD		✔	✔		✔	
SheafHyperGNN [28]	2023	NeurIPS	✔			✔	✔	
MeanPooling [65]	2023	AAAI		✔		✔		✔
HENN [39]	2023	LoG	✔			✔		✔
HyGNN [102]	2023	ICDE		✔		✔	✔	
HGraphormer [99]	2023	arXiv	✔			✔	✔	
MultiSetMixer [108]	2023	arXiv		✔	✔			✔
HDE^{ode} [139]	2024	ICLR		✔		✔		✔
HyperGT [82]	2024	ICASSP		✔		✔	✔	
THNN [121]	2024	SDM		✔		✔		✔
UniG-Encoder [161]	2024	Pat. Rec.		✔		✔		✔
WCE-GNN [106]	2024	arXiv	✔			✔		✔
HelHNN [106]	2024	arXiv		✔		✔	✔	
HyperMagNet [11]	2024	arXiv	✔			✔	✔	

* The specific method proposed by Aponte et al. [4] is named HNN.

4.1 Learning to classify

HNNs can learn HOIs by classifying hyperedges [47, 63, 115, 138, 151]. To this end, positive and negative hyperedges are considered. A positive hyperedge is an observed, ground-truth hyperedge, and a negative hyperedge often refers to a generated "fake" hyperedge, considered unlikely to exist. By learning to classify them, HNNs may capture the distinguishing patterns of the ground-truth HOIs.

4.1.1 ***Heuristic negative sampling.*** We discuss popular negative sampling (NS) strategies to obtain negative hyperedges [94]:

- **Sized NS**: each negative hyperedge contains k random nodes.
- **Motif NS**: each negative hyperedge contains a randomly chosen k adjacent nodes.
- **Clique NS**: each negative hyperedge is created by replacing a random node in a positive hyperedge with another node adjacent to the remaining nodes.

Similarly, many HNNs use rule-based NS for hyperedge classification [47, 63, 115, 138, 151]. Others leverage domain knowledge to design NS strategies [18, 124].

4.1.2 ***Learnable negative sampling.*** Notably, Hwang et al. [47] show that training HNNs with the aforementioned NS strategies may cause overfitting to negative hyperedges of specific types. This may be attributed to the vast population of potential negative hyperedges, where their samples may not adequately represent this population. To mitigate the problem, they employ adversarial training of a generator that samples negative hyperedges.

4.2 Learning to contrast

Contrastive learning (CL) aims to maximize agreement between data obtained from different views. Intuitively, views refer to different versions of the same data, original or augmented. Training neural networks with CL has shown strong capacity in capturing the input data characteristics [49, 81]. For HNNs, several CL techniques have been devised to improve their learning of HOIs [60, 63, 65, 126]. Here, we describe three steps of CL for HNNs: (*i*) obtaining views, (*ii*) encoding, and (*iii*) computing contrastive loss.

4.2.1 ***View creation and encoding.*** First, we obtain views for contrast. This can be achieved by augmenting the input hypergraph, using *rule-based* [60, 65] or *learnable* [126] methods.

Rule-based augmentation. This approach stochastically corrupts node features and hyperedges. For nodes, an augmented feature matrix is obtained by either zeroing out certain entries (i.e., feature values) of $\mathbf{X}$ [63, 65] or adding Gaussian noise to them [98]. For hyperedges, augmented hyperedges are obtained by excluding some nodes from hyperedges [65] or perturbing hyperedge membership (e.g., changing $e_i = \{v_1, v_2, v_3\}$ to $e_i' = \{v_1, v_2, v_4\}$) [78].

Learnable augmentation. This approach utilizes a neural network to generate views [126]. Specifically, HyperGCL [126] generates synthetic hyperedges $\mathcal{E}'$ using HNN-based VAE [61].

Once an augmentation strategy $\tau : (\mathbf{X}, \mathcal{E}) \mapsto (\mathbf{X}', \mathcal{E}')$ is decided, a hypergraph-view pair $(\mathcal{G}^{(1)}, \mathcal{G}^{(2)})$ can be obtained in two ways:

- $\mathcal{G}^{(1)}$ is the original hypergraph with $(\mathbf{X}, \mathcal{E})$, and $\mathcal{G}^{(2)}$ is an augmented hypergraph with $(\mathbf{X}', \mathcal{E}')$, where $(\mathbf{X}', \mathcal{E}') = \tau(\mathbf{X}, \mathcal{E})$ [126].
- Both $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are augmented by applying τ to $(\mathbf{X}, \mathcal{E})$ [65]. They are likely to differ due to the stochastic nature of τ.

Then, the message passing on two views (sharing the same parameters) results in two pairs of node and hyperedge embeddings denoted by $(\mathbf{P}', \mathbf{Q}')$ and $(\mathbf{P}'', \mathbf{Q}'')$ [63, 65].

4.2.2 ***Contrastive loss.*** Then, we choose a contrastive loss. Below, we present *node-*, *hyperedge-*, and *membership*-level contrastive losses. Here, $\tau_x, \tau_e, \tau_m \in \mathbb{R}$ are hyperparameters.

Node level. A node-level contrastive loss is used to (*i*) maximize the similarity between the same node from two different views and (*ii*) minimize the similarity for different nodes [60, 63, 65, 126]:

$$\mathcal{L}^{(v)}(\mathbf{P}', \mathbf{P}'') = \frac{-1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \log \frac{\exp(\texttt{sim}(\boldsymbol{p}_i', \boldsymbol{p}_i'')/\tau_v)}{\sum_{v_k \in \mathcal{V}} \exp(\texttt{sim}(\boldsymbol{p}_i', \boldsymbol{p}_k'')/\tau_v)}, \quad (20)$$

where $\texttt{sim}(\boldsymbol{x}, \boldsymbol{y})$ denotes a metric (e.g., cosine similarity) for assessing the similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Hyperedge level. A hyperedge-level contrastive loss is implemented in a highly similar manner [63, 65]:

$$\mathcal{L}^{(e)}(\mathbf{Q}', \mathbf{Q}'') = \frac{-1}{|\mathcal{E}|} \sum_{e_j \in \mathcal{E}} \log \frac{\exp(\texttt{sim}(\boldsymbol{q}'_j, \boldsymbol{q}''_j)/\tau_e)}{\sum_{e_k \in \mathcal{E}} \exp(\texttt{sim}(\boldsymbol{q}'_i, \boldsymbol{q}''_k)/\tau_e)}. \quad (21)$$

Membership level. A membership-level contrastive loss is used to make the embeddings of incident node-hyperedge pairs distinguishable from those of non-incident pairs, across two views [65]:

$$\mathcal{L}^{(m)}(\mathbf{P}', \mathbf{Q}'') = \frac{-1}{K} \sum_{e_j \in \mathcal{E}} \sum_{v_i \in \mathcal{V}} \underbrace{\mathbb{1}_{i,j} \log \frac{\exp(\mathcal{D}(\boldsymbol{p}'_i, \boldsymbol{q}''_j)/\tau_m)}{\sum_{v_k \in \mathcal{V}} \exp(\mathcal{D}(\boldsymbol{p}'_k, \boldsymbol{q}''_j)/\tau_m)}}_{\texttt{when } \boldsymbol{q}''_j \texttt{ is an anchor}}$$
$$- \frac{1}{K} \sum_{e_j \in \mathcal{E}} \sum_{v_i \in \mathcal{V}} \underbrace{\mathbb{1}_{i,j} \log \frac{\exp(\mathcal{D}(\boldsymbol{q}''_j, \boldsymbol{p}'_i)/\tau_m)}{\sum_{e_k \in \mathcal{E}} \exp(\mathcal{D}(\boldsymbol{q}''_k, \boldsymbol{p}'_i)/\tau_m),}}_{\texttt{when } \boldsymbol{p}'_i \texttt{ is an anchor}}$$

where $\mathbb{1}_{s,j} = \mathbb{I}[v_s \in v_j]$; $\mathcal{D}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}$ is a discriminator for assigning higher value to incident pairs than non-incident pairs [41, 112].

4.3 Learning to generate

HNNs can also be trained by learning to generate hyperedges. Existing HNNs aim to generate either (*i*) ground-truth hyperedges to capture their characteristics or (*ii*) latent hyperedges potentially beneficial for designated downstream tasks.

4.3.1 ***Generating ground-truth HOIs.*** Training neural networks to generate input data has shown strong efficacy in various domains [40, 91]. In two recent studies, HNNs are trained to generate ground-truth hyperedges to effectively reflect HOIs [26, 59]. Between them, HypeBoy [59] formulates hyperedge generation as a *hyperedge filling task*, where the objective is to identify the missing node for a given subset of a hyperedge. Overall, HypeBoy involves three steps: (*i*) hypergraph augmentation, (*ii*) node and hyperedge-subset encoding, and (*iii*) loss-function computation.

HypeBoy obtains the augmented node feature matrix $\mathbf{X}'$ and augmented input topology $\mathcal{E}'$, respectively by randomly masking some entries of $\mathbf{X}$ and by randomly dropping some hyperedges from $\mathcal{E}$. Hypeboy, then, feeds $\mathbf{X}'$ and $\mathcal{E}'$ into an HNN to obtain node embedding matrix $\mathbf{P}$. Subsequently, for each node $v_i \in e_j$ and subset $q_{ij} = e_j \setminus \{v_i\}$, HypeBoy obtains (final) node embedding $\boldsymbol{p}_i = \texttt{MLP}_1(\boldsymbol{p}_i)$ and subset embedding $\boldsymbol{q}_{ij} = \texttt{MLP}_2(\sum_{v_k \in q_{ij}} \boldsymbol{p}_k)$. Lastly, the HNN is trained to make embeddings of the 'true' node-subset pairs similar and of the 'false' node-subset pairs dissimilar. Specifically, it minimizes the following loss:

$$\mathcal{L} = - \sum_{e_j \in \mathcal{E}} \sum_{v_i \in e_j} \log \frac{\exp(\texttt{sim}(\boldsymbol{p}_i, \boldsymbol{q}_{ij}))}{\sum_{v_k \in \mathcal{V}} \exp(\texttt{sim}(\boldsymbol{p}_k, \boldsymbol{q}_{ij}))}, \quad (22)$$

where $\texttt{sim}(\boldsymbol{x}, \boldsymbol{y})$ is a cosine similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$.

4.3.2 ***Generating latent HOIs.*** HNNs can be trained to generate latent hyperedges, especially when (*i*) (semi-)supervised downstream tasks and (*ii*) suboptimal input hypergraph structures are assumed. Typically, the training methods let HNNs generate potential, latent hyperedges, which are used for message passing to improve downstream task performance [14, 72, 148, 152].

For example, HSL [14] adopts a learnable augmenter to replace unhelpful hyperedges with generated ones. HSL first prunes hyperedges by using a masking matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, where each $j-$th column is $m_j = \texttt{sigmoid}((\log(\frac{z_j}{1-z_j}) + (\epsilon_0 - \epsilon_1))/\tau)$: ϵ_0 and ϵ_1 are random samples from $\texttt{Gumbel(0, 1)}$; $\tau \in \mathbb{R}$ is a hyperparameter; each $z_k \in [0, 1], \forall e_k \in \mathcal{E}$ is a learnable scalar. An unhelpful e_k is expected to have small z_k and, thus, is likely to be pruned.

After performing pruning by $\hat{\mathbf{H}} = \mathbf{H} \odot \mathbf{M}$, where $\odot$ indicates Hadamard product, HSL modifies $\hat{\mathbf{H}}$ by adding generated latent hyperedges $\Delta\mathbf{H}$. Specifically, $\Delta\mathbf{H}_{i,j} = 1$ if $(\mathbf{H}_{i,j} = 0) \wedge (\mathbf{S}_{i,j} \in \texttt{top}(\mathbf{S}, N))$, and 0 otherwise. $\texttt{top}(\mathbf{S}, N)$ denotes the set of top-N entries in a learnable score matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$. Each score in $\mathbf{S}$ is $\mathbf{S}_{i,j} = \frac{1}{T} \sum_{t=1}^{T} \cos(\boldsymbol{w}_t \odot \boldsymbol{p}_i, \boldsymbol{w}_t \odot \boldsymbol{q}_i)$, where $\{\boldsymbol{w}_t\}_{t=1}^{T}$ are learnable vectors, and cos is cosine similarity. Importantly, an HNN is trained to obtain $\mathbf{P}$ and $\mathbf{Q}$. To summarize, node and hyperedge similarities learned by an HNN serve to generate latent hyperedges $\Delta\mathbf{H}$. Lastly, $\hat{\mathbf{H}} + \Delta\mathbf{H}$ is fed into another HNN for a target downstream task (i.e., node classification). All learnable components, including the HNN for augmentation, are trained in an end-to-end manner.[5]

Note that the HNNs learning to generate latent hyperedges generally implement additional loss functions to encourage the latent hyperedges to be similar to the original ones [148, 152]. Furthermore, some studies have explored generating latent HOIs when input hypergraph structures are not available [36, 53, 156].

5 APPLICATION GUIDANCE

HNNs have been adopted in various applications, including recommendation, bioinformatics and medical science, time series analysis, and computer vision. Their central concerns involve hypergraph construction and hypergraph learning task formulation.

5.1 Recommendation

5.1.1 ***Hypergraph construction.*** For recommender system applications, many studies utilized hypergraphs consisting of item nodes (those being recommended) and user hyperedges (those receiving recommendations). For instance, all items that a user interacts with are connected by a hyperedge [117]. When sessions are available, hyperedges are used to connect item nodes by their context window [75, 118, 131]. Some studies leveraged multiple hypergraphs. For instance, Zhang et al. [149] incorporated user- and group-level hypergraphs. In the user-level hypergraph, a node is a user and a hyperedge connects a group of users. The group-level hypergraph consists of group nodes and their motif-induced hyperedges. [6] Ji et al. [51] constructed a hypergraph with item nodes and a hypergraph with user nodes, where their hyperedges were inferred from heuristic-based algorithms. In contrast, other studies incorporate learnable hypergraph structure [129, 130].

5.1.2 ***Application tasks.*** Hypergraph-based modeling allows natural applications of HNNs for recommendation, typically formulated as hyperedge prediction problem. HNNs have been used for sequential [74, 117], session-based [75, 118, 131], group [52, 149], conversational [153], and point-of-interest [64] recommendation.

[5] HSL may apply further sparsification processes to $\hat{\mathbf{H}} + \Delta\mathbf{H}$.

[6] Specifically, the group nodes first make *pair-wise connections* based on their shared users. Then, the group nodes that form a triadic motif are connected by a hyperedge, returning their group-level hypergraph.

5.2 Bioinformatics and medical science

5.2.1 ***Hypergraph construction.*** For bioinformatics applications, molecular-level structures are often considered as nodes. Studies connected the structures with a hyperedge based on their joint reaction [18], presence within each drug [102], presence within each herb [42], and association with each disease [42]. Some studies used multiple node types. A study considered cell line nodes and drug nodes, with hyperedge connecting those with a synergy relationship [80, 124]. Drug and their side effects were also considered as nodes, where those with drug-drug interaction were connected by a hyperedge [90]. Another study used drug nodes or target protein nodes connected by hyperedges based on their similarity in interactions or associations [100]. Lastly, several studies used kNN or learnable hyperedges to construct hypergraphs [73, 95, 107].

Some other studies used hypergraphs to model MRI data. Many of them had region-of-interest serving as a node, while a hyperedge connected the nodes using interaction strength estimation [119], k-means [50], or random-walk-based sampling [16]. On the other hand, in some studies, study subjects were nodes, and their neighbors connected by hyperedges were found by kNN [38, 85].

Lastly, electronic health records (EHR) data were often modeled with hypergraphs. Most often, nodes were either medical codes [15, 22, 127, 133, 134] or clinical events [159]. A hyperedge connected the codes or clinical events that were shared by each patient.

5.2.2 ***Application tasks.*** For bioinformatics applications, HNNs have been applied to predict interactions or associations among molecular-level structures. Thus, many of the tasks are naturally formulated as a hyperedge prediction task. Specifically, the application tasks include predictions of missing metabolic reactions [18, 138], drug-drug interactions [80, 90, 102, 124], drug-target interactions [100], drug-gene interactions [107], herb–disease associations [42], and miRNA-disease associations [95].

For MRI analysis, if regions-of-interest serve as nodes, HNNs are applied to solve a hypergraph classification problem. The task, however, may become a node prediction problem if each subject serves as a node. Alzheimer's disease classification [38], brain connectome analysis [119], autism prediction [50, 85], and brain network dysfunction prediction [16] problems have been solved with HNNs.

In analyzing EHR data, since a hyperedge consists of medical codes or clinical events of a patient, HNNs are applied for hyperedge prediction. Studies used HNNs to predict mortality [15], readmission [15], diagnosis [127], medication [127], phenotype [22, 134], clinical outcomes [22, 133, 134], and clinical pathways [159].

5.3 Time series analysis

5.3.1 ***Hypergraph construction.*** A variety of nodes have been used for time series forecast applications. Depending on the data, nodes were cities [123, 143], gas regulators [143], rail segments [143], train stations [123], stocks [77, 103, 104], or regions [76]. Studies often leveraged learnable hyperedges [76, 77, 105, 123, 128, 154] or similarity- or proximity-based hyperedges [77, 103, 104, 143].

5.3.2 ***Application tasks.*** When applying HNNs, many time series forecast problems can be formulated as node regression problems. Specifically, the prior works use HNNs to forecast taxi demands [143], gas pressures [143], vehicle speeds [143], traffic [83, 105, 120, 123, 128, 154], electricity consumptions [105, 128], meteorological measures [105, 123], stocks [77, 103, 104], and crimes [76].

5.4 Computer vision

5.4.1 ***Hypergraph construction.*** Hypergraph-based modeling has also been adopted for computer vision applications. Studies used nodes to represent image patches [37], features [140], 3D shapes [6], joints [79, 135, 158], and humans [43]. To connect the nodes by a hyperedge, kNN [6, 140], Fuzzy C-Means [37], and other learnable functions [79, 114, 135] were adopted.

5.4.2 ***Application tasks.*** For computer vision tasks, studies use HNNs to solve problems including image classification [37], object detection [37], video-based person re-identification [140], image impainting [114], action recognition [158], pose estimation [79, 135], 3D shape retrieval and recogntion [6], and multi-human mesh recovery [43]. Due to the heterogeneity of the applied tasks, we find no consistent hypergraph learning task formulation

6 DISCUSSIONS

In this work, we provide a survey on hypergraph neural networks (HNNs), with a focus on how they address higher-order interactions (HOIs). We aim for the present survey to be in-depth, covering HNN encoder designs (Sec. 3), training objective designs (Sec. 4), and applications (Sec. 5). We provide a taxonomy of how HOIs are modeled by HNN architectures and objectives (Fig. 2), with how representative HNNs fit into the taxonomy (Table 2). Having reviewed the exponentially growing literature, we close the present survey with some future directions.

HNN theory. Various theoretical analyses of graph neural networks (GNNs) have been conducted [147], including studies on their graph isomorphism recognition [113, 132] and approximation abilities [54, 86]. However, given the complex nature of hypergraphs, where a hyperedge may connect an arbitrary number of nodes, directly applying these theoretical findings from graphs to hypergraphs can be non-trivial [31]. Therefore, many theoretical properties of HNNs remain yet to be unveiled, and some areas have begun to be explored, including their generalization abilities [155] and transferability [39].

Advantages of HNNs. Instead of leveraging HNNs, one could simply use GNNs for a hypergraph by reducing the structure into that of a simple pairwise graph. While many studies have empirically shown that HNNs outperform these alternatives [19, 32, 59, 122], the factors that confer HNNs the empirical advantages remain underexplored. Yoon et al. [144] investigated the effectiveness of using HOIs for heuristic classifiers. However, studies dedicated for HNNs may inspire improved HNN architectures.

Complex hypergraphs. Networks of HOIs often exhibit temporal, directional, and heterogeneous properties and are modeled by temporal [70], directed [33], and heterogeneous [46, 136] hypergraphs. Although structural patterns in them have been discussed [12, 58, 70, 88], developing and evaluating HNNs to leverage and learn such complex HOIs is still in the early stages [1, 46, 83, 110, 136, 157]. Thus, simultaneously, more benchmark datasets and tasks for their evaluation are necessary. With the proper datasets and tasks, we expect that studies will advance for HNNs to fully exploit the complex and informative nature of HOIs.

REFERENCES

[1] Shivam Agarwal, Ramit Sawhney, Megh Thakkar, Preslav Nakov, Jiawei Han, and Tyler Derr. 2022. Think: Temporal hypergraph hyperbolic network. In *ICDM*.

[2] Unai Alvarez-Rodriguez, Federico Battiston, Guilherme Ferraz de Arruda, Yamir Moreno, Matjaž Perc, and Vito Latora. 2021. Evolutionary dynamics of higher-order interactions in social networks. *Nature Human Behaviour* 5, 5 (2021), 586–595.

[3] Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. 2023. A survey on hypergraph representation learning. *Comput. Surveys* 56, 1 (2023), 1–38.

[4] Ryan Aponte, Ryan A Rossi, Shunan Guo, Jane Hoffswell, Nedim Lipka, Chang Xiao, Gromit Chan, Eunyee Koh, and Nesreen Ahmed. 2022. A hypergraph neural network framework for learning hyperedge-dependent node embeddings. *arXiv preprint arXiv:2212.14077* (2022).

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] Junjie Bai, Biao Gong, Yining Zhao, Fuqiang Lei, Chenggang Yan, and Yue Gao. 2021. Multi-scale representation learning on hypergraph for 3D shape retrieval and recognition. *IEEE Transactions on Image Processing* 30 (2021), 5327–5338.

[7] Song Bai, Feihu Zhang, and Philip HS Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637.

[8] Federico Battiston, Enrico Amico, Alain Barrat, Ginestra Bianconi, Guilherme Ferraz de Arruda, Benedetta Franceschiello, Iacopo Iacopini, Sonia Kéfi, Vito Latora, Yamir Moreno, et al. 2021. The physics of higher-order interactions in complex systems. *Nature Physics* 17, 10 (2021), 1093–1098.

[9] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. 2020. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports* 874 (2020), 1–92.

[10] Federico Battiston and Giovanni Petri. 2022. *Higher-order systems*. Springer.

[11] Tatyana Benko, Martin Buck, Ilya Amburg, Stephen J Young, and Sinan G Aksoy. 2024. Hypermagnet: A magnetic laplacian based hypergraph neural network. *arXiv preprint arXiv:2402.09676* (2024).

[12] Austin R Benson, Ravi Kumar, and Andrew Tomkins. 2018. Sequences of sets. In *KDD*.

[13] Ginestra Bianconi. 2021. *Higher-order networks*. Cambridge University Press.

[14] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. 2022. Hypergraph structure learning for hypergraph neural networks. In *IJCAI*.

[15] Derun Cai, Chenxi Sun, Moxian Song, Baofeng Zhang, Shenda Hong, and Hongyan Li. 2022. Hypergraph contrastive learning for electronic health records. In *SDM*.

[16] Hongmin Cai, Zhixuan Zhou, Defu Yang, Guorong Wu, and Jiazhou Chen. 2023. Discovering Brain Network Dysfunction in Alzheimer's Disease Using Brain Hypergraph Neural Network. In *MICCAI*.

[17] Lang Chai, Lilan Tu, Xianjia Wang, and Qingqing Su. 2024. Hypergraph modeling and hypergraph multi-view attention neural network for link prediction. *Pattern Recognition* 149 (2024), 110292.

[18] Can Chen, Chen Liao, and Yang-Yu Liu. 2023. Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning. *Nature Communications* 14, 1 (2023), 2375.

[19] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. 2022. You are allset: A multiset function framework for hypergraph neural networks. In *ICLR*.

[20] Uthsav Chitra and Benjamin Raphael. 2019. Random walks on hypergraphs with edge-dependent vertex weights. In *ICML*.

[21] Minyoung Choe, Sunwoo Kim, Jaemin Yoo, and Kijung Shin. 2023. Classification of edge-dependent labels of nodes in hypergraphs. In *KDD*.

[22] Hejie Cui, Xinyu Fang, Ran Xu, Xuan Kan, Joyce C Ho, and Carl Yang. 2024. Multimodal fusion of ehr in structures and semantics: Integrating clinical records and notes with hypergraph and llm. *arXiv preprint arXiv:2403.08818* (2024).

[23] Guilherme Ferraz de Arruda, Giovanni Petri, and Yamir Moreno. 2020. Social contagion models on hypergraphs. *Physical Review Research* 2, 2 (2020), 023032.

[24] Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. 2020. Structural patterns and generative models of real-world hypergraphs. In *KDD*.

[25] Yihe Dong, Will Sawin, and Yoshua Bengio. 2020. Hnhn: Hypergraph networks with hyperedge neurons. In *ICML Workshop: Graph Representation Learning and Beyond*.

[26] Boxin Du, Changhe Yuan, Robert Barton, Tal Neiman, and Hanghang Tong. 2022. Self-supervised hypergraph representation learning. In *Big Data*.

[27] Dheeru Dua, Casey Graff, et al. 2017. Uci machine learning repository. (2017).

[28] Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Liò. 2023. Sheaf hypergraph networks. In *NeurIPS*.

[29] Paul Expert and Giovanni Petri. 2022. Higher-order description of brain function. In *Higher-Order Systems*. Springer, 401–415.

[30] Song Feng, Emily Heath, Brett Jefferson, Cliff Joslyn, Henry Kvinge, Hugh D Mitchell, Brenda Praggastis, Amie J Eisfeld, Amy C Sims, Larissa B Thackray, et al. 2021. Hypergraph models of biological networks to identify genes critical to pathogenic viral response. *BMC bioinformatics* 22, 1 (2021), 287.

[31] Yifan Feng, Jiashu Han, Shihui Ying, and Yue Gao. 2024. Hypergraph isomorphism computation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 01 (2024), 1–17.

[32] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *AAAI*.

[33] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete applied mathematics* 42, 2-3 (1993), 177–201.

[34] Rui Gao, Indu Kant Deo, and Rajeev K Jaiman. 2024. A finite element-inspired hypergraph neural network: Application to fluid dynamics simulations. *J. Comput. Phys.* (2024), 112866.

[35] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. 2022. HGNN+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 45, 3 (2022), 3181–3199.

[36] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. 2020. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 44, 5 (2020), 2548–2566.

[37] Yan Han, Peihao Wang, Souvik Kundu, Ying Ding, and Zhangyang Wang. 2023. Vision hgnn: An image is more than a graph of nodes. In *ICCV*.

[38] Xiaoke Hao, Jiawang Li, Mingming Ma, Jing Qin, Daoqiang Zhang, Feng Liu, Alzheimer's Disease Neuroimaging Initiative, et al. 2024. Hypergraph convolutional network for longitudinal data analysis in Alzheimer's disease. *Computers in Biology and Medicine* 168 (2024), 107765.

[39] Mikhail Hayhoe, Hans Matthew Riess, Michael M Zavlanos, Victor Preciado, and Alejandro Ribeiro. 2023. Transferable Hypergraph Neural Networks via Spectral Similarity. In *LOG*.

[40] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *CVPR*.

[41] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

[42] Lun Hu, Menglong Zhang, Pengwei Hu, Jun Zhang, Chao Niu, Xueying Lu, Xiangrui Jiang, and Yupeng Ma. 2024. Dual-channel hypergraph convolutional network for predicting herb–disease associations. *Briefings in Bioinformatics* 25, 2 (2024), bbae067.

[43] Buzhen Huang, Jingyi Ju, Zhihao Li, and Yangang Wang. 2023. Reconstructing groups of people with hypergraph relational reasoning. In *ICCV*.

[44] Jie Huang, Chuan Chen, Fanghua Ye, Jiajing Wu, Zibin Zheng, and Guohui Ling. 2019. Hyper2vec: Biased random walk for hyper-network embedding. In *DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA*.

[45] Jing Huang and Jie Yang. 2021. Unignn: a unified framework for graph and hypergraph neural networks. In *IJCAI*.

[46] Xingyue Huang, Miguel Romero Orth, Pablo Barceló, Michael M Bronstein, and İsmail İlkan Ceylan. 2024. Link prediction with relational hypergraphs. *arXiv preprint arXiv:2402.04062* (2024).

[47] Hyunjin Hwang, Seungwoo Lee, Chanyoung Park, and Kijung Shin. 2022. Ahp: Learning to negative sample for hyperedge prediction. In *SIGIR*.

[48] Iacopo Iacopini, Giovanni Petri, Andrea Baronchelli, and Alain Barrat. 2022. Group interactions modulate critical mass dynamics in social convention. *Communications Physics* 5, 1 (2022), 64.

[49] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *Technologies* 9, 1 (2020), 2.

[50] Junzhong Ji, Yating Ren, and Minglong Lei. 2022. FC–HAT: Hypergraph attention network for functional brain network classification. *Information Sciences* 608 (2022), 1301–1316.

[51] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In *KDD*.

[52] Renqi Jia, Xiaofei Zhou, Linhua Dong, and Shirui Pan. 2021. Hypergraph convolutional network for group recommendation. In *ICDM*.

[53] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic hypergraph neural networks.. In *IJCAI*.

[54] Nicolas Keriven and Gabriel Peyré. 2019. Universal invariant and equivariant graph neural networks. In *NeurIPS*.

[55] Jinwoo Kim, Saeyoon Oh, Sungjun Cho, and Seunghoon Hong. 2022. Equivariant hypergraph neural networks. In *ECCV*.

[56] Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. 2021. Transformers generalize deepsets and can be extended to graphs & hypergraphs. In *NeurIPS*.

[57] Sunwoo Kim, Fanchen Bu, Minyoung Choe, Jaemin Yoo, and Kijung Shin. 2023. How transitive are real-world group interactions?-Measurement and reproduction. In *KDD*.

[58] Sunwoo Kim, Minyoung Choe, Jaemin Yoo, and Kijung Shin. 2023. Reciprocity in directed hypergraphs: measures, findings, and generators. *Data Mining and Knowledge Discovery* 37, 6 (2023), 2330–2388.

[59] Sunwoo Kim, Shinhwan Kang, Fanchen Bu, Soo Yong Lee, Jaemin Yoo, and Kijung Shin. 2024. HypeBoy: Generative self-supervised representation learning on hypergraphs. In *ICLR*.

[60] Sunwoo Kim, Dongjin Lee, Yul Kim, Jungho Park, Taeho Hwang, and Kijung Shin. 2023. Datasets, tasks, and training methods for large-scale hypergraph learning. *Data Mining and Knowledge Discovery* 37, 6 (2023), 2216–2254.

[61] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. In *NeurIPS*.

[62] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[63] Yunyong Ko, Hanghang Tong, and Sang-Wook Kim. 2023. Enhancing hyperedge prediction with context-aware self-supervised learning. *arXiv preprint arXiv:2309.05798* (2023).

[64] Yantong Lai, Yijun Su, Lingwei Wei, Gaode Chen, Tianci Wang, and Daren Zha. 2023. Multi-view spatial-temporal enhanced hypergraph network for next poi recommendation. In *DASFAA*.

[65] Dongjin Lee and Kijung Shin. 2023. I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In *AAAI*.

[66] Geon Lee, Fanchen Bu, Tina Eliassi-Rad, and Kijung Shin. 2024. A survey on hypergraph mining: Patterns, tools, and generators. *arXiv preprint arXiv:2401.08878* (2024).

[67] Geon Lee, Minyoung Choe, and Kijung Shin. 2021. How do hyperedges overlap in real-world hypergraphs?-patterns, measures, and generators. In *WWW*.

[68] Geon Lee, Jihoon Ko, and Kijung Shin. 2020. Hypergraph motifs: concepts, algorithms, and discoveries. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2256–2269.

[69] Geon Lee, Soo Yong Lee, and Kijung Shin. 2024. VilLain: Self-supervised learning on hypergraphs without features via virtual label propagation. In *WWW*.

[70] Geon Lee and Kijung Shin. 2023. Temporal hypergraph motifs. *Knowledge and Information Systems* 65, 4 (2023), 1549–1586.

[71] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*.

[72] Fangyuan Lei, Jiahao Huang, Jianjian Jiang, Da Huang, Zhengming Li, and Chang-Dong Wang. 2024. Unveiling the potential of long-range dependence with mask-guided structure learning for hypergraph. *Knowledge-Based Systems* 284 (2024), 111254.

[73] Wei Li, Bin Xiang, Fan Yang, Yu Rong, Yanbin Yin, Jianhua Yao, and Han Zhang. 2023. Scmhnn: A novel hypergraph neural network for integrative analysis of single-cell epigenomic, transcriptomic and proteomic data. *Briefings in Bioinformatics* 24, 6 (2023), bbad391.

[74] Yicong Li, Hongxu Chen, Xiangguo Sun, Zhenchao Sun, Lin Li, Lizhen Cui, Philip S Yu, and Guandong Xu. 2021. Hyperbolic hypergraphs for sequential recommendation. In *CIKM*.

[75] Yinfeng Li, Chen Gao, Hengliang Luo, Depeng Jin, and Yong Li. 2022. Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation. In *SIGIR*.

[76] Zhonghang Li, Chao Huang, Lianghao Xia, Yong Xu, and Jian Pei. 2022. Spatial-temporal hypergraph self-supervised learning for crime prediction. In *ICDE*.

[77] Sihao Liao, Liang Xie, Yuanchuang Du, Shengshuang Chen, Hongyang Wan, and Haijiao Xu. 2024. Stock trend prediction based on dynamic hypergraph spatio-temporal network. *Applied Soft Computing* 154 (2024), 111329.

[78] Luotao Liu, Feng Huang, Xuan Liu, Zhankun Xiong, Menglu Li, Congzhi Song, and Wen Zhang. 2023. Multi-view contrastive learning hypergraph neural network for drug-microbe-disease association prediction. In *IJCAI*.

[79] Shengyuan Liu, Pei Lv, Yuzhen Zhang, Jie Fu, Junjin Cheng, Wanqing Li, Bing Zhou, and Mingliang Xu. 2020. Semi-dynamic hypergraph neural network for 3d pose estimation.. In *IJCAI*.

[80] Xuan Liu, Congzhi Song, Shichao Liu, Menglu Li, Xionghui Zhou, and Wen Zhang. 2022. Multi-way relation-enhanced hypergraph representation learning for anti-cancer drug synergy prediction. *Bioinformatics* 38, 20 (2022), 4782–4789.

[81] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2022), 5879–5900.

[82] Zexi Liu, Bohan Tang, Ziyuan Ye, Xiaowen Dong, Siheng Chen, and Yanfeng Wang. 2024. Hypergraph transformer for semi-supervised classification. *ICASSP*.

[83] Xiaoyi Luo, Jiaheng Peng, and Jun Liang. 2022. Directed hypergraph attention network for traffic forecasting. *IET Intelligent Transport Systems* 16, 1 (2022), 85–98.

[84] Jing Ma, Mengting Wan, Longqi Yang, Jundong Li, Brent Hecht, and Jaime Teevan. 2022. Learning causal effects on hypergraphs. In *KDD*.

[85] Mohammad Madine, Islem Rekik, and Naoufel Werghi. 2020. Diagnosing autism using t1-w mri with multi-kernel learning and hypergraph neural network. In *ICIP*.

[86] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. 2019. On the universality of invariant networks. In *ICML*.

[87] Harry Mickalide and Seppe Kuehn. 2019. Higher-order interaction between species inhibits bacterial invasion of a phototroph-predator microbial community. *Cell systems* 9, 6 (2019), 521–533.

[88] Heechan Moon, Hyunju Kim, Sunwoo Kim, and Kijung Shin. 2023. Four-set hypergraphlets for characterization of directed hypergraphs. *arXiv preprint arXiv:2311.14289* (2023).

[89] Manon A Morin, Anneliese J Morrison, Michael J Harms, and Rachel J Dutton. 2022. Higher-order interactions shape microbial interactions as microbial community complexity increases. *Scientific Reports* 12, 1 (2022), 22640.

[90] Duc Anh Nguyen, Canh Hao Nguyen, Peter Petschner, and Hiroshi Mamitsuka. 2022. Sparse: A sparse hypergraph neural network for learning multiple types of latent combinations to accurately predict drug-drug interactions. *Bioinformatics* 38 (2022), i333–i341.

[91] OpenAI. 2023. Gpt-4 technical report. (2023).

[92] Theodore Papamarkou, Tolga Birdal, Michael Bronstein, Gunnar Carlsson, Justin Curry, Yue Gao, Mustafa Hajij, Roland Kwitt, Pietro Liò, Paolo Di Lorenzo, et al. 2024. Position paper: challenges and opportunities in topological deep learning. *arXiv preprint arXiv:2402.08871* (2024).

[93] Mathilde Papillon, Sophia Sanborn, Mustafa Hajij, and Nina Miolane. 2023. Architectures of topological deep learning: A survey on topological neural networks. *arXiv preprint arXiv:2304.10031* (2023).

[94] Prasanna Patil, Govind Sharma, and M Narasimha Murty. 2020. Negative sampling for hyperlink prediction in networks. In *PAKDD*.

[95] Wei Peng, Zhichen He, Wei Dai, and Wei Lan. 2024. Mhclmda: Multihypergraph contrastive learning for mirna–disease association prediction. *Briefings in Bioinformatics* 25, 1 (2024), bbad524.

[96] Konstantin Prokopchik, Austin R Benson, and Francesco Tudisco. 2022. Nonlinear feature diffusion on hypergraphs. In *ICML*.

[97] Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2023. Adaptive expansion for hypergraph learning. (2023).

[98] Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2024. Adaptive temperature enhanced dual-level hypergraph contrastive learning. In *WWW*.

[99] Shilin Qu, Weiqing Wang, Yuan-Fang Li, Xin Zhou, and Fajie Yuan. 2023. Hypergraph node representation learning with one-stage message passing. *arXiv preprint arXiv:2312.00336* (2023).

[100] Ding Ruan, Shuyi Ji, Chenggang Yan, Junjie Zhu, Xibin Zhao, Yuedong Yang, Yue Gao, Changqing Zou, and Qionghai Dai. 2021. Exploring complex and heterogeneous correlations on hypergraph for the prediction of drug-target interactions. *Patterns* 2, 12 (2021).

[101] Khaled Mohammed Saifuddin, Mehmet Emin Aktas, and Esra Akbas. 2023. Topology-guided hypergraph transformer network: Unveiling structural insights for improved representation. *arXiv preprint arXiv:2310.09657* (2023).

[102] Khaled Mohammed Saifuddin, Briana Bumgardner, Farhan Tanvir, and Esra Akbas. 2023. Hygnn: Drug-drug interaction prediction via hypergraph neural network. In *ICDE*.

[103] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *AAAI*.

[104] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Ratn Shah. 2020. Spatiotemporal hypergraph convolution network for stock movement forecasting. In *ICDM*.

[105] Zongjiang Shang and Ling Chen. 2024. Mshyper: Multi-scale hypergraph transformer for long-range time series forecasting. *arXiv preprint arXiv:2401.09261* (2024).

[106] Bohan Tang, Zexi Liu, Keyue Jiang, Siheng Chen, and Xiaowen Dong. 2024. Hypergraph node classification With graph neural networks. *arXiv preprint arXiv:2402.05569* (2024).

[107] Wen Tao, Yuansheng Liu, Xuan Lin, Bosheng Song, and Xiangxiang Zeng. 2023. Prediction of multi-relational drug-gene interaction via dynamic hypergraph contrastive learning. *Briefings in Bioinformatics* 24, 6 (2023), bbad371.

[108] Lev Telyatnikov, Maria Sofia Bucarelli, Guillermo Bernardez, Olga Zaghen, Simone Scardapane, and Pietro Lio. 2023. Hypergraph neural networks through the lens of message passing: a common perspective to homophily and architecture design. *arXiv preprint arXiv:2310.07684* (2023).

[109] Hao Tian and Reza Zafarani. 2024. Higher-order networks representation and learning: A survey. *arXiv preprint arXiv:2402.19414* (2024).

[110] Loc Hoang Tran and Linh Hoang Tran. 2020. Directed hypergraph neural network. *arXiv preprint arXiv:2008.03626* (2020).

[111] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

[112] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR*.

[113] Clement Vignac, Andreas Loukas, and Pascal Frossard. 2020. Building powerful and equivariant graph neural networks with structural message-passing. In *NeurIPS*.

[114] Gourav Wadhwa, Abhinav Dhall, Subrahmanyam Murala, and Usman Tariq. 2021. Hyperrealistic image inpainting with hypergraphs. In *WACV*.

[115] Changlin Wan, Muhan Zhang, Wei Hao, Sha Cao, Pan Li, and Chi Zhang. 2021. Principled hyperedge prediction with structural spectral features and neural networks. *arXiv preprint arXiv:2106.04292* (2021).

[116] Fuli Wang, Karelia Pena-Pena, Wei Qian, and Gonzalo R Arce. 2024. T-hypergnns: Hypergraph neural networks via tensor representations. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[117] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *SIGIR*.

[118] Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. 2021. Session-based recommendation with hypergraph attention networks. In *SDM*.

[119] Junqi Wang, Hailong Li, Gang Qu, Kim M Cecil, Jonathan R Dillman, Nehal A Parikh, and Lili He. 2023. Dynamic weighted hypergraph convolutional network for brain functional connectome analysis. *Medical Image Analysis* 87 (2023), 102828.

[120] Jingcheng Wang, Yong Zhang, Lixun Wang, Yongli Hu, Xinglin Piao, and Baocai Yin. 2022. Multitask hypergraph convolutional networks: A heterogeneous traffic prediction framework. *IEEE Transactions on Intelligent Transportation Systems* 23, 10 (2022), 18557–18567.

[121] Maolin Wang, Yaoming Zhen, Yu Pan, Zenglin Xu, Ruocheng Guo, and Xiangyu Zhao. 2024. Tensorized hypergraph neural networks. In *SDM*.

[122] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. 2023. Equivariant hypergraph diffusion neural operators. In *ICLR*.

[123] Shun Wang, Yong Zhang, Xuanqi Lin, Yongli Hu, Qingming Huang, and Baocai Yin. 2024. Dynamic Hypergraph Structure Learning for Multivariate Time Series Forecasting. *IEEE Transactions on Big Data* 01 (2024), 1–13.

[124] Wei Wang, Gaolin Yuan, Shitong Wan, Ziwei Zheng, Dong Liu, Hongjun Zhang, Juntao Li, Yun Zhou, and Xianfang Wang. 2024. A granularity-level information fusion strategy on hypergraph transformer for predicting synergistic effects of anticancer drugs. *Briefings in Bioinformatics* 25, 1 (2024), bbad522.

[125] Yuxin Wang, Quan Gan, Xipeng Qiu, Xuanjing Huang, and David Wipf. 2023. From hypergraph energy functions to hypergraph neural networks. In *ICML*.

[126] Tianxin Wei, Yuning You, Tianlong Chen, Yang Shen, Jingrui He, and Zhangyang Wang. 2022. Augmentations in hypergraph contrastive learning: Fabricated and generative. In *NeurIPS*.

[127] Jialun Wu, Kai He, Rui Mao, Chen Li, and Erik Cambria. 2023. MEGACare: Knowledge-guided multi-view hypergraph predictive framework for healthcare. *Information Fusion* 100 (2023), 101939.

[128] Jinming Wu, Qi Qi, Jingyu Wang, Haifeng Sun, Zhikang Wu, Zirui Zhuang, and Jianxin Liao. 2023. Not only pairwise relationships: fine-grained relational modeling for multivariate time series forecasting. In *IJCAI*.

[129] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *SIGIR*.

[130] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-supervised hypergraph transformer for recommender systems. In *KDD*.

[131] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI*.

[132] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? (2019).

[133] Ran Xu, Mohammed K Ali, Joyce C Ho, and Carl Yang. 2023. Hypergraph transformers for ehr-based clinical predictions. *AMIA Summits on Translational Science Proceedings* 2023 (2023), 582.

[134] Ran Xu, Yue Yu, Chao Zhang, Mohammed K Ali, Joyce C Ho, and Carl Yang. 2022. Counterfactual and factual reasoning over hypergraphs for interpretable clinical predictions on ehr. In *ML4H*.

[135] Xixia Xu, Qi Zou, and Xue Lin. 2022. Adaptive hypergraph neural network for multi-person pose estimation. In *AAAI*.

[136] Naganand Yadati. 2020. Neural message passing for multi-relational ordered and recursive hypergraphs. In *NeurIPS*.

[137] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *NeurIPS*.

[138] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. 2020. NHP: Neural hypergraph link prediction. In *CIKM*.

[139] Jielong Yan, Yifan Feng, Shihui Ying, and Yue Gao. 2024. Hypergraph dynamic system. In *ICLR*.

[140] Yichao Yan, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, Ying Tai, and Ling Shao. 2020. Learning multi-granular hypergraphs for video-based person re-identification. In *CVPR*.

[141] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. 2022. Hypergraph learning with line expansion. In *CIKM*.

[142] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. 2022. Semi-supervised hypergraph node classification on hypergraph line expansion. In *CIKM*.

[143] Jaehyuk Yi and Jinkyoo Park. 2020. Hypergraph convolutional recurrent neural network. In *KDD*.

[144] Se-eun Yoon, Hyungseok Song, Kijung Shin, and Yung Yi. 2020. How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In *WWW*.

[145] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. 2021. Identity-aware graph neural networks. In *AAAI*.

[146] Shan Yu, Hongdian Yang, Hiroyuki Nakahara, Gustavo S Santos, Danko Nikolić, and Dietmar Plenz. 2011. Higher-order interactions characterized in cortical activity. *Journal of neuroscience* 31, 48 (2011), 17514–17526.

[147] Bingxu Zhang, Changjun Fan, Shixuan Liu, Kuihua Huang, Xiang Zhao, Jincai Huang, and Zhong Liu. 2023. The expressive power of graph neural networks: A survey. *arXiv preprint arXiv:2308.08235* (2023).

[148] Jiying Zhang, Yuzhao Chen, Xi Xiao, Runiu Lu, and Shu-Tao Xia. 2022. Learnable hypergraph laplacian for hypergraph learning. In *ICASSP*.

[149] Junwei Zhang, Min Gao, Junliang Yu, Lei Guo, Jundong Li, and Hongzhi Yin. 2021. Double-scale self-supervised hypergraph learning for group recommendation. In *CIKM*.

[150] Jiying Zhang, Fuyang Li, Xi Xiao, Tingyang Xu, Yu Rong, Junzhou Huang, and Yatao Bian. 2022. Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs. *ICML Workshop on Topology, Algebra, and Geometry in Machine Learning*.

[151] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2020. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *ICLR*.

[152] Zizhao Zhang, Yifan Feng, Shihui Ying, and Yue Gao. 2022. Deep hypergraph structure learning. *arXiv preprint arXiv:2208.12547* (2022).

[153] Sen Zhao, Wei Wei, Xian-Ling Mao, Shuai Zhu, Minghui Yang, Zujie Wen, Dangyang Chen, and Feida Zhu. 2023. Multi-view hypergraph contrastive policy learning for conversational recommendation. In *SIGIR*.

[154] Yusheng Zhao, Xiao Luo, Wei Ju, Chong Chen, Xian-Sheng Hua, and Ming Zhang. 2023. Dynamic hypergraph structure learning for traffic flow forecasting. In *ICDE*.

[155] Luo Zhezheng, Mao Jiayuan, Tenenbaum Joshua B., and Kaelbling Leslie, Pack. 2023. On the expressiveness and generalization of hypergraph neural networks. In *LOG*.

[156] Peng Zhou, Zongqian Wu, Xiangxiang Zeng, Guoqiu Wen, Junbo Ma, and Xiaofeng Zhu. 2023. Totally dynamic hypergraph neural network. In *IJCAI*.

[157] Xue Zhou, Bei Hui, Ilana Zeira, Hao Wu, and Ling Tian. 2023. Dynamic relation learning for link prediction in knowledge hypergraphs. *Applied Intelligence* 53, 22 (2023), 26580–26591.

[158] Yuxuan Zhou, Zhi-Qi Cheng, Chao Li, Yanwen Fang, Yifeng Geng, Xuansong Xie, and Margret Keuper. 2022. Hypergraph transformer for skeleton-based action recognition. *arXiv preprint arXiv:2211.09590* (2022).

[159] Fanglin Zhu, Shunyu Chen, Yonghui Xu, Wei He, Fuqiang Yu, Xu Zhang, and Lizhen Cui. 2022. Temporal hypergraph for personalized clinical pathway recommendation. In *BIBM*.

[160] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS*.

[161] Minhao Zou, Zhongxue Gan, Yutong Wang, Junheng Zhang, Dongyan Sui, Chun Guan, and Siyang Leng. 2024. Unig-encoder: A universal feature encoder for graph and hypergraph node classification. *Pattern Recognition* 147 (2024), 110115.