K-hop Hypergraph Neural Network: A Comprehensive Aggregation Approach

Linhuang Xie[1,2], **Shihao Gao**[1,2], **Jie Liu**[3,4], **Ming Yin**[5], **Taisong Jin**[1,2*]

[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing,
Ministry of Education of China, Xiamen University, China
[2]School of Informatics, Xiamen University, China
[3]School of Information Science, North China University of Technology, China
[4]China Language Intelligence Research Center, Capital Normal University, China
[5]School of Electronic Science and Engineering, South China Normal University, China
linhuangxie@gmail.com, gaoshihao@stu.xmu.edu.cn, liujxxxy@126.com, m.yin@scnu.edu.cn, jintaisong@xmu.edu.cn

Abstract

The powerful capability of HyperGraph Neural Networks (HGNNs) in modeling intricate, high-order relationships among multiple data samples stems primarily from their ability to aggregate both the direct neighborhood features of individual nodes and those associated with hyperedges. However, the limited scope of feature propagation in existing HGNNs significantly reduces the utilization of hypergraph information, exacerbating over-squashing and over-smoothing issues. To this end, we propose a novel ***K***-hop **H**yper**G**raph **N**eural **N**etwork (KHGNN) to facilitate the interactions of distant nodes and hyperedges. Specifically, the bisection nested convolution based on HyperGINE is employed to extract features from nodes, hyperedges, and structures along all shortest paths between nodes or hyperedges, providing representations of long-distance relationships. With these comprehensive path features, nodes and hyperedges are guided to aggregate distant information while learning their complex relationships. The extensive experiments, particularly on long-range graph datasets, demonstrate that the proposed method achieves SOTA performance compared to existing HGNNs and graph neural networks.

Introduction

Graph Neural Networks (GNNs) that effectively handle non-Euclidean structured data have been widely applied in tasks such as computational chemistry (Yu et al. 2020), social networks (Cao et al. 2020), and anomaly detections (Yang et al. 2024). However, GNNs are inherently limited by their exclusive focus on pairwise relationships between data samples, which restricts the expressive power.

Unlike GNNs, Hypergraph Neural Networks (HGNNs) can capture higher-order non-Euclidean relationships among real-world entities through unpaired connections involving any number of nodes on hyperedges. Since the introduction of the **clique expansion (CE)** technique by HGNN (Feng et al. 2019), most of the existing HGNNs (Chien et al. 2022; Huang and Yang 2021; Wang et al. 2023) define various message-passing methods based on this technique. As shown in Fig. 1(a) and Fig. 2(a1), CE involves a two-stage message passing process: **(1)** Node-to-hyperedge: Numerous nodes are compressed into hyperedges with limited feature capacity, resulting in information loss. **(2)** Hyperedge-to-node: Identical hyperedge features are assigned to all neighboring nodes, diminishing the discriminability of node features. The aforementioned slow diffusion exacerbates the issues of **over-squashing** and **over-smoothing** (Hu et al. 2019; Topping et al. 2021; Alon and Yahav 2020), reducing the utilization of hypergraph information.

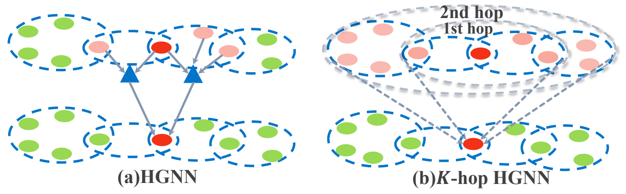

Figure 1: (a) For the existing hypergraph neural networks, node features are first aggregated by hyperedges and then propagated to neighboring nodes. (b) The proposed K-hop HGNN introduces a wider range of node aggregation by utilizing path features, facilitating feature propagation.

GNNs have extended the concept of direct message passing to K-hop message passing (Wang et al. 2021; Zhang and Li 2021), thereby enhancing the expressive power of them (Feng et al. 2022; Abboud et al. 2022). In particular, leveraging path features that encapsulate contextual information to guide K-hop message passing between nodes can explicitly provide rich semantic information for relationship modeling, leading to promising performance in real-world applications (Michel et al. 2023; Sun et al. 2022).

However, there is few research specifically dedicated to path-based K-hop hypergraph neural networks. Considering the following reasons, it is intractable to design K-hop HGNNs: **(1)** Inherent **structural differences** between hypergraphs, characterized by hyperedges with varying numbers of nodes, and normal graphs render the methods used in K-hop GNNs unsuitable for HGNNs. **(2)** Diverging from K-hop GNNs (Michel et al. 2023; Sun et al. 2022), merely aggregating K-hop node neighborhoods while **disregarding hyperedges** representing higher-order relations constrains the expressive capacity of HGNNs. **(3)** Due to differences in feature space, the commonly used **summation operators** only aggregate either node (Sun et al. 2022; Michel

*Corresponding author.

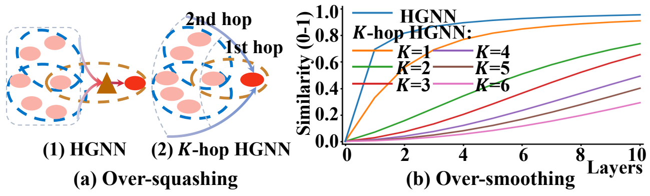

Figure 2: (a) For HGNN, the left nodes are squashed into hyperedges before being transmitted to the red node. K-hop HGNN mitigates over-squashing by enabling direct propagating across multiple hops. (b) As the number of **parameter-free** message passing layers increases, adjacent nodes in HGNN rapidly become similar (smooth) on Peptides-func (Dwivedi et al. 2022b), where nodes are initialized with distinct one-hot encodings. K-hop HGNN leverages distant nodes to maintain distinctiveness.

et al. 2023) or edge features (Feng et al. 2022; Ying et al. 2021) as path features, which makes it hard to effectively represent path information. **(4)** As the length of paths increases, the number of paths grows exponentially, demanding **substantial computational resources**. This restricts the use of complex operators for extracting path features.

Inspired by advances in K-hop message passing, we propose a novel hypergraph neural network, termed **KHGNN**, to learn representations of nodes and hyperedges with larger receptive fields. The proposed model employs a bisection nested convolution strategy, which iteratively performs convolutions at the centers of the shortest paths, combining information from nodes, hyperedges, and structures into path features with lower computational costs. Thus, nodes and hyperedges are guided by path features to learn long-range higher-order relations while aggregating information from an expanded neighborhood, eliminating the need for separately designing aggregation strategies. KHGNN mitigates over-squashing by enabling direct interaction among nodes through higher-order relationships, thereby avoiding information compression by hyperedges (see Fig. 2(a)). The over-smoothing issue is addressed through the discriminative information provided by distant nodes (see Fig. 2(b)).

Our contributions are in three-folds: **(1)** We propose HyperGINE, a hypergraph neural network that leverages connection features to guide the direct aggregation of nodes or hyperedges. **(2)** We propose a novel K-hop hypergraph neural network (KHGNN), an extension of HyperGINE, that integrates three challenging-to-harmonize features on paths using a bisection nested convolutional approach as the guidance for K-hop message passing between nodes or hyperedges. To our knowledge, this is the first attempt at path-based K-hop hypergraph message passing. **(3)** We propose a novel strategy to encode structural information of different paths, which is suitable for K-hop HGNN.

Related Work

Hypergraph Neural Networks

In contemporary hypergraph research, most HGNNs draw inspiration from the clique expansion (CE) technique proposed by HGNN (Feng et al. 2019). For instance, HyperGAT (Ding et al. 2020) leveraged attention-based message passing, while Wang et al. (2023) combined hypergraph diffusion algorithms with CE. DHGNN (Jiang et al. 2019) introduced a feature-influenced dynamic hypergraph structure, and Gao et al. (2022) further defined directed hyperedge convolution. Some works (Huang and Yang 2021; Chien et al. 2022) endeavored to define a unified two-stage architecture. However, the CE architectures constrain the breadth of information accessible to nodes and hyperedges, exacerbating over-smoothing and over-squashing issues (Yu et al. 2022). While methods like KNN, clustering, node neighborhoods (Jiang et al. 2019; Gao et al. 2022), and random walk techniques (Huang et al. 2020; Huang, Liu, and Song 2019) gather information from distant nodes, they fall short in accurately capturing the hypergraph structures and learning appropriate representations of hyperedges.

K-hop Message Passing

In **graph neural networks**, K-hop message passing constructs K-hop neighborhood for each node in three primary ways: (1) Random walk-based methods establish node neighborhoods using different powers of the adjacency matrix (Abu-El-Haija et al. 2019; Chien et al. 2020; Wang et al. 2021). However, the inherent complexity of random walks leads to the neglect of important path features. (2) Shortest path-based methods define neighborhoods based on the shortest distance between nodes (Abboud et al. 2022; Yang et al. 2021; Li et al. 2020). Due to pairwise relationships, only one of the multiple shortest paths between nodes is chosen as the feature. (3) All Paths-based methods consider all possible paths between nodes (Sun et al. 2022; Michel et al. 2023). However, the complexity of the graph structure introduces significant redundancy and computational overhead.

To reduce computational costs, existing methods employ **summation operators**, either summing edge features (Feng et al. 2022; Ying et al. 2021; Brossard, Frigo, and Dehaene 2020) or node features (Sun et al. 2022; Michel et al. 2023; Yang et al. 2021), to represent path features. However, path features obtained through such an operator are incomplete for graphs, and even more so for hypergraphs with their inherently more complex structures.

For **hypergraph neural networks**, existing works either simply aggregate the K-hop neighborhoods of nodes (Huang et al. 2021, 2023), or transform them into hyperedges (Gao et al. 2022). However, these approaches overlook the utilization of path information and hyperedge features, which encapsulate intricate semantic details.

In summary, whether in graphs or hypergraphs, there remains a lack of effective methods to encode nodes, hyperedges, and structural information into path features that can guide the aggregation of information from larger neighborhoods for both nodes and hyperedges.

Method

In this section, we first introduce the hypergraph notation for later use. Then, we present HyperGINE, from which KHGNN is derived. KHGNN leverages path features obtained through bisection nested convolutions to guide the representation learning of nodes and hyperedges. Finally, we introduce a relative degree encoding scheme to further enhance the model's ability to capture structural information.

Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a hypergraph with a node set $\mathcal{V} = \{v_1, ..., v_n\}$ and an edge set $\mathcal{E} = \{e_1, ..., e_m\}$, where n and m represent the total number of nodes and hyperedges, and h_v and h_e denote their respective feature embeddings. We define the k-th hop neighboring node set of node v_i as $\mathcal{N}_{v_i}^{k}$, which includes all nodes reachable from v_i through at least k hyperedges (i.e., at a distance of at least k). $\mathcal{N}_{v_i}^{0,k} = \{\mathcal{N}_{v_i}^{t}\}_{t\in[0,k]}$ denotes the K-hop neighboring node set of node v_i. The k-th hop neighboring hyperedge set of node v_i is denoted as $\mathcal{Q}_{v_i}^{k}$, representing all hyperedges reachable from v_i through at least k nodes. Specially, $\mathcal{Q}_{v_i}^{0}$ denotes hyperedges directly connected to node v_i.

Similarly, for hyperedge e_b, we define its k-th hop neighboring hyperedge set as $\mathcal{N}_{e_b}^{k}$, representing all hyperedges reachable from hyperedge e_b through at least k nodes. $\mathcal{N}_{e_b}^{0,k} = \{\mathcal{N}_{e_b}^{t}\}_{t\in[0,k]}$ represents the K-hop neighboring hyperedge set of hyperedge e_b. The k-th hop neighboring node set of hyperedge e_b is denoted as $\mathcal{Q}_{e_b}^{k}$, where $\mathcal{Q}_{e_b}^{0}$ represents nodes directly connected to hyperedge e_b.

HyperGINE

To tackle the issues posed by slow clique expansion message passing, we extend GINE (Hu et al. 2019), whose baseline is provably one of the most expressive GNNs (Xu et al. 2019), to hypergraphs. HyperGINE is introduced, which enables the simultaneous learning of features for nodes, hyperedges, and the overall neighborhood structure. The formulation of HyperGINE is presented as follows:

$$\begin{aligned} h_{v_i}^{l} &= \mathcal{AGGN}^{l}\left(h_v^{l-1}, h_e^{l-1}, \mathcal{Q}_v^0, \mathcal{Q}_e^0, S_{v_i}\right) \\ &= \sum_{e_b\in\mathcal{Q}_{v_i}^0}\left(\sum_{v_j\in\{\mathcal{Q}_{e_b}^0 - S_{v_i}\}}\left(f(h_{v_j}^{l-1}) + g(h_{e_b}^{l-1})\right)\right), \end{aligned} \tag{1}$$

where f and g are learnable linear transformations that map node features h_v and hyperedge features h_e to appropriate feature spaces. S_{v_i} denotes the set of nodes within hyperedges that do not transmit information to node v_i, as defined by different algorithms. For instance, within the same community (hyperedge), not all residents (nodes) necessarily need to know each other.

Furthermore, hypergraph duality enables a seamless interchange of roles between nodes and hyperedges. Leveraging this property, HyperGINE directly updates hyperedge features without modification:

$$\begin{aligned} h_{e_b}^{l} &= \mathcal{AGGE}^{l}\left(h_e^{l-1}, h_v^{l-1}, \mathcal{Q}_e^0, \mathcal{Q}_v^0, S_{e_b}\right) \\ &= \sum_{v_i\in\mathcal{Q}_{e_b}^0}\left(\sum_{e_c\in\{\mathcal{Q}_{v_i}^0 - S_{e_b}\}}\left(f(h_{e_c}^{l-1}) + g(h_{v_i}^{l-1})\right)\right), \end{aligned} \tag{2}$$

and eliminates the necessity of designing separate aggregation functions for learning node and hyperedge representations. By aggregating inter-node hyperedge information, HyperGINE mitigates the effects of over-squashing and reduces potential noise that may arise when nodes are compressed into hyperedges and subsequently transmitted to other nodes, as compared to HGNN (see Fig. 2(b) K=1).

In particular, when $S_{v_i} = \{v_i\}$, symmetric normalization of Eq. 1 yields corresponding matrix form:

$$\begin{aligned} X_v^l = D_v^{-\frac{1}{2}} H \left(D_e^{-1}(H^T D_v^{-\frac{1}{2}} X_v^{l-1}\Theta_1 - X_e\Theta_2) + X_e\Theta_2\right) \\ - H\mathrm{diag}(D_e^{-1})\mathbf{1}^T \odot D_v^{-\frac{1}{2}} X_v^{l-1}\Theta_1, \end{aligned} \tag{3}$$

where X_v and X_e represent features of all nodes and hyperedges, respectively. D_v and D_e are diagonal matrices of node and hyperedge degrees. H denotes the incidence matrix in the hypergraph, and Θ denotes learnable parameters. $\odot$ denotes the Hadamard product. Thus, HGNN (Feng et al. 2019) is viewed as a special case of HyperGINE. For detailed proofs and discussions, please refer to Appendix D.

K-hop Hypergraph Neural Network

In this section, we introduce KHGNN, as illustrated in Fig. 3, which guides higher-order aggregation of nodes or hyperedges by integrating features from nodes, hyperedges, and structure along the paths. KHGNN comprises two essential components: K-hop node representation learning and K-hop hyperedge representation learning.

***K*-hop Node Representation Learning** Unlike in normal graphs where edges establish fixed connections between two nodes, paths on hypergraphs interweave between nodes and structurally complex hyperedges, forming fundamental features. Therefore, HyperGINE, which accounts for node, hyperedge, and structure features, is selected as the foundational model of KHGNN.

To alleviate the computational and parameter demands imposed by numerous paths, KHGNN introduces the **bisection nested convolution**, inspired by bisection search. From a top-down perspective, higher-order paths undergo recursive partitioning around a central position. From a bottom-up standpoint, central nodes (hyperedges) utilize HyperGINE convolution to iteratively merge features from the left and right segments of the path, simultaneously injecting structural characteristics. During this process, the convolution of the central node (hyperedge) not only updates its features but also amalgamates features from multiple paths to generate comprehensive higher-order path features.

Specifically, when k is **even**, the set of central nodes on all shortest paths between nodes v_i and v_d, where the shortest path length is k, is defined as $\{\mathcal{N}_{v_i}^{\frac{k}{2}} \cap \mathcal{N}_{v_d}^{\frac{k}{2}}\}$. Let $v_j \in \{\mathcal{N}_{v_i}^{\frac{k}{2}} \cap \mathcal{N}_{v_d}^{\frac{k}{2}}\}$ represent one of these central nodes, it is evident that both v_i and v_d are contained within $\mathcal{N}_{v_j}^{\frac{k}{2}}$. The hidden state $\hat{h}_{v_j}^{l,\frac{k}{2}}$ obtained from the $\frac{k}{2}$-hop bisection nested convolution effectively combines all the shortest paths that are divided into two segments by node v_j between $\mathcal{N}_{v_j}^{\frac{k}{2}}$. By injecting the structural features using HyperGINE as the base model, it provides a comprehensive representation of these shortest paths and constructs hyperedges for $\mathcal{N}_{v_j}^{\frac{k}{2}}$ to reveal higher-order position relationships. At this stage, the k-th hop aggregation function at l-th layer is expressed as:

$$\hat{h}_{v_i}^{l,k} = \mathcal{AGGN}_k^l \left(h_v^{l-1}, \hat{h}_v^{l,\frac{k}{2}}, \mathcal{N}_v^{\frac{k}{2}}, \mathcal{N}_v^{\frac{k}{2}}, S_{v_i}^k \right)$$
$$= \sum_{v_j \in \mathcal{N}_{v_i}^{\frac{k}{2}}} \left(\sum_{v_d \in \{\mathcal{N}_{v_j}^{\frac{k}{2}} - S_{v_i}^k\}} \left(f(h_{v_d}^{l-1}) + g(\hat{h}_{v_j}^{l,\frac{k}{2}}) \right) \right), \quad (4)$$

where $\mathcal{AGGN}$ is defined as in Eq. 1. Since the shortest path length between node v_i and $\mathcal{N}_{v_j}^{\frac{k}{2}}$ ranges from 0 to k, the set difference $S_{v_i} = \mathcal{N}_{v_i}^{0,k-1}$ represents that node v_i selectively aggregates only the nodes along the shortest path. In particular, when $k = 0$, node v_i establishes self-connection, inheriting the feature from the previous layer (i.e. $\hat{h}_{v_i}^{l,0} = h_{v_i}^{l-1}$).

Similarly, when k is **odd**, the set of central hyperedges for all shortest paths between nodes v_i and v_d is denoted as $\{\mathcal{Q}_{v_i}^{\frac{k-1}{2}} \cap \mathcal{Q}_{v_d}^{\frac{k-1}{2}}\}$. For a hyperedge $e_j \in \{\mathcal{Q}_{v_i}^{\frac{k-1}{2}} \cap \mathcal{Q}_{v_d}^{\frac{k-1}{2}}\}$, we have $v_i, v_d \in \mathcal{Q}_{e_j}^{\frac{k-1}{2}}$. The hidden state $\hat{h}_{e_j}^{l,\frac{k-1}{2}}$ from the $\frac{k-1}{2}$-hop nested convolution combines the shortest paths divided by hyperedge e_j while incorporating structural information of hyperedge e_j as the path features. In this case, the k-th hop message passing for nodes v_i is represented as:

$$\hat{h}_{v_i}^{l,k} = \mathcal{AGGN}_k^l \left(h_v^{l-1}, \hat{h}_e^{l,\frac{k-1}{2}}, \mathcal{Q}_v^{\frac{k-1}{2}}, \mathcal{Q}_e^{\frac{k-1}{2}}, S_{v_i}^k \right), \quad (5)$$

where $\mathcal{AGGN}$ also follows Eq. 1, and $S_{v_i}^k = \mathcal{N}_{v_i}^{0,k-1}$ denotes the set of nodes within a distance less than k from the node v_i within any hyperedge.

Subsequently, the update function $\mathcal{UPD}$ assigns weights to the representations of different neighborhoods associated with each node. The feature embedding $h_{v_i}^l$ of node v_i is generated at the l-th layer by the following process:

$$h_{v_i}^l = \mathcal{UPD}^l \left(\{\hat{h}_{v_i}^{l,t}\}_{t\in[0,K]} \right) = MLP \left(\sum_{t=0}^{K} \lambda_t \hat{h}_{v_i}^{l,t} \right), \quad (6)$$

where λ are the learnable parameters, and MLP denotesdenotes a two-layer perceptron.

***K*-hop Hyperedge Representation Learning** When dealing with hypergraphs that consist of hyperedges containing arbitrary numbers of nodes, it becomes imperative to consider the more complex topological structures and connectivity relationships compared to pairwise connections in traditional graphs. Consequently, expanding the receptive field of hyperedges is crucial for capturing both local structures and connection information of nodes while obtaining additional contextual information. Thanks to hypergraph duality and the direct message-passing characteristics of HyperGINE, KHGNN updates hyperedge features with a larger receptive field without requiring additional design of hyperedge aggregation functions.

Specifically, after applying dual transformation, hyperedges e and nodes v in the hypergraph are transformed into dual nodes e and dual hyperedges v in the dual hypergraph.

Similar to node representation learning, when the shortest path length k is **even**, the $\frac{k}{2}$-th hop hidden state $\hat{h}_{e_j}^{l,\frac{k}{2}}$ of dual node e_j, located at the center of the path, serves as the shortest path feature with structural information between dual nodes $\mathcal{N}_{e_j}^{\frac{k}{2}}$. Hyperedges with higher-order relationships are then constructed for these dual nodes. The corresponding hidden state $\hat{h}_{e_b}^{l,k}$ for the hyperedge e_b is computed by

$$\hat{h}_{e_b}^{l,k} = \mathcal{AGGE}_k^l \left(h_e^{l-1}, \hat{h}_e^{l,\frac{k}{2}}, \mathcal{N}_e^{\frac{k}{2}}, \mathcal{N}_e^{\frac{k}{2}}, S_{e_b}^k \right), \quad (7)$$

where $\mathcal{AGGE}$ adheres to the definition in Eq. 2, and $S_{e_b}^k = \mathcal{N}_{e_b}^{0,k-1}$ represents the set of hyperedges within a distance less than k from hyperedge e_b.

When k is **odd**, $\hat{h}_{v_j}^{l,\frac{k-1}{2}}$ is utilized to characterize the shortest paths through dual hyperedges v_j between dual node $\mathcal{Q}_{v_j}^{\frac{k-1}{2}}$. The k-th hop hidden state $\hat{h}_{e_b}^{l,k}$ is given by

$$\hat{h}_{e_b}^{l,k} = \mathcal{AGGE}_k^l \left(h_e^{l-1}, \hat{h}_v^{l,\frac{k-1}{2}}, \mathcal{Q}_e^{\frac{k-1}{2}}, \mathcal{Q}_v^{\frac{k-1}{2}}, S_{e_b}^k \right). \quad (8)$$

Certainly, the update function for hyperedge embedding remains consistent with Eq. 6:

$$h_{e_b}^l = \mathcal{UPD}^l(\{\hat{h}_{e_b}^{l,t}\}_{t\in[0,K]}) = MLP \left(\sum_{t=0}^{K} \lambda_t \hat{h}_{e_b}^{l,t} \right). \quad (9)$$

In Fig. 3, node v_2 aggregates its first-hop neighborhood to update feature h_{v_2} by utilizing hyperedges as paths. This convolution is also viewed as stitching together paths of length 1 around v_2 and constructing a higher-order hyperedge v_2 with the path endpoints v_1, v_3, and v_5 to represent the positional relationship between them. Consequently, the hidden state $h_{v_2}^{l,1}$ serves as a path feature to guide message passing between nodes v_1 and v_3, v_5. The convolution of hyperedges e_2 and e_4 operates in the same manner.

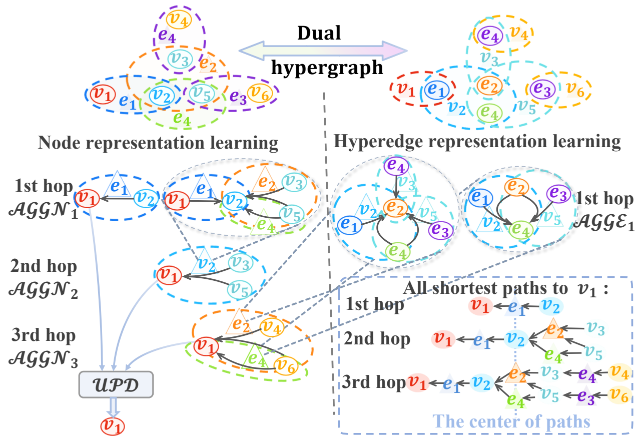
Dual
hypergraph
Node representation learning
Hyperedge representation learning
1st hop $\mathcal{AGGN}_1$
2nd hop $\mathcal{AGGN}_2$
3rd hop $\mathcal{AGGN}_3$
$\mathcal{UPD}$
1st hop $\mathcal{AGGE}_1$
All shortest paths to v_1 :
1st hop
2nd hop
3rd hop
The center of paths

Figure 3: The process of learning the 3-hop node representation for node v_1 in KHGNN. The low-order HyperGINE convolution at the center of the path, serving as input to the update function, effectively stitches together surrounding path features while incorporating structural information to construct new hyperedges.

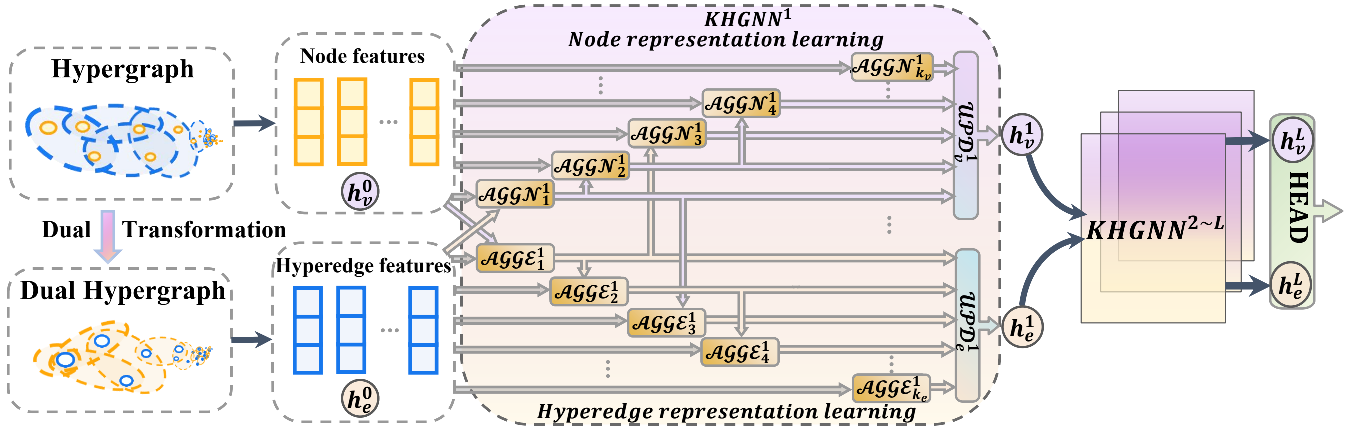

Figure 4: The overview of the proposed KHGNN framework. Firstly, the input hypergraph is transformed to obtain the dual hypergraph. Secondly, through bisection nested convolutions, the low-order convolutions serve as inputs to the update function and are simultaneously reused as path features, guiding the higher-order neighborhood aggregation of hyperedges/nodes. Finally, different heads are employed to make the hypergraph representations suitable for various downstream tasks.

Discussion The bisection nested convolution operator in KHGNN offers several key advantages:

(1) **Enhanced Representation of High-Order Relationships:** Nodes and hyperedges encapsulate crucial structural and connectivity information that enhances one another. The robust interaction between them, depicted in Fig. 4, seamlessly integrates low-order structural and connectivity details into high-order positional hyperedges, providing rich semantic information for modeling complex relationships.

(2) **Comprehensive Path Feature Extraction:** This convolutional approach mirrors the bisection method applied to paths. By iteratively nesting downwards, it captures the features of bottom-level nodes, hyperedges, and their associated structures. As the method synthesizes paths upward through their central points, it infuses a broader structural context, ensuring a comprehensive path representation.

(3) **Optimized Efficiency:** As the aggregation's receptive field expands, the increasing number and length of paths present challenges for complex operators, which would introduce significant computational and parametric overhead. KHGNN addresses this by reusing low-order convolutions, thereby minimizing the computational and parametric costs associated with extracting numerous path features.

Relative Degree Encoding

The essence of hypergraph message passing lies in exchanging relative features between nodes via hyperedges or paths. Consequently, global node degrees are primarily utilized for normalizing the aggregated representations of nodes and hyperedges, rather than directly encoding them (Feng et al. 2019; Gao et al. 2022; Antelmi et al. 2023).

In social networks, more mutual friends between two strangers implies a higher likelihood of their acquaintance. KHGNN excels at capturing all shortest path relations between nodes. The abundance of these paths reflects the strength of relative relationships and the corresponding structural information. Relative degree encoding is introduced based on these principles.

Consider the hypergraph $\mathcal{G}$ with the k-th hop connectivity between nodes denoted as $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$, where $\mathcal{E}_k$ represents the high-order hyperedges constructed through bisection nested convolutions and $\mathcal{E}_1 = \mathcal{E}$. The k-th hop relative degree $d^k_{v_i,v_j}$ between node v_i and v_j is defined as the number of hyperedges connecting the two nodes at the k-th hop:

$$d^k_{v_i,v_j} = \begin{cases} \left|\left\{v_k \in \{\mathcal{N}^{\frac{k}{2}}_{v_i} \cap \mathcal{N}^{\frac{k}{2}}_{v_j}\}\right\}\right| & \text{if } k \text{ is even} \\ \left|\left\{e_k \in \{\mathcal{Q}^{\frac{k-1}{2}}_{v_i} \cap \mathcal{Q}^{\frac{k-1}{2}}_{v_j}\}\right\}\right| & \text{if } k \text{ is odd} \end{cases}, \quad (10)$$

where $|\cdot|$ denotes the cardinality of a set. Consequently, the aggregation function, such as Eq. 4, is modified as follows:

$$\hat{h}^{l,k}_{v_i} = \mathcal{AGGN}^l_k\left(h^{l-1}_v, \hat{h}^{l,\frac{k}{2}}_v, \mathcal{N}^{\frac{k}{2}}_v, \mathcal{N}^{\frac{k}{2}}_v, S^k_{v_i}, \hat{d}^k\right) \quad (11)$$

$$= \sum_{v_j \in \mathcal{N}^{\frac{k}{2}}_{v_i}} \left(\sum_{v_d \in \{\mathcal{N}^{\frac{k}{2}}_{v_j} - S^k_{v_i}\}} \left(f(h^{l-1}_{v_d}) + g(\hat{h}^{l,\frac{k}{2}}_{v_j}) + \hat{d}^k_{v_i,v_j}\right)\right).$$

where $\hat{d}^k_{v_i,v_j}$ is the embedding corresponding to $d^k_{v_i,v_j}$.

Computational Complexity Analysis

Consider a hypergraph $\mathcal{G}$ with N nodes and E hyperedges and the message-passing complexity per hyperedge in the hypergraph architecture is denoted as $F(n)$, where n is the average number of nodes within high-order hyperedges constructed by bisection nested convolution. In HyperGINE, $F(n) \in [n, n^2)$, with $n \ll N$ due to the inherent sparsity of real-world graphs. In KHGNN, the number of hops for node and hyperedge aggregation are denoted as K_v and K_e.

For odd hops of node representation, almost every hyperedge constructs higher-order hyperedges for the endpoints of the shortest paths represented by its bisection convolution features, leading to a time complexity of $O\left(\lceil \frac{K_v}{2} \rceil E \cdot F(n)\right)$. For even hops, the con-

	Cora	Citeseer	Pubmed	Cora-CA	Mushroom	NTU2012	ModelNet40	Rank↓
HGNN (Feng et al. 2019)	79.39 ± 1.36	72.45 ± 1.16	86.44 ± 0.44	82.64 ± 1.65	98.73 ± 0.32	87.72 ± 1.35	95.44 ± 0.33	7.71
HyperGCN (Yadati et al. 2019)	78.45 ± 1.26	71.28 ± 0.82	82.84 ± 8.67	79.48 ± 2.08	47.90 ± 1.04	56.36 ± 4.86	75.89 ± 5.26	10
UniGCNII (Huang et al. 2021)	78.81 ± 1.05	73.05 ± 2.21	88.25 ± 0.40	83.60 ± 1.14	99.96 ± 0.05	<u>89.30 ± 1.33</u>	97.84 ± 0.25	4.43
Allset (Chien et al. 2022)	78.59 ± 1.47	73.08 ± 1.20	<u>88.72 ± 0.37</u>	83.63 ± 1.47	**100.00 ± 0.00**	88.69 ± 1.24	<u>98.20 ± 0.20</u>	3.71
HGNN+ (Gao et al. 2022)	79.81 ± 1.66	73.50 ± 1.10	87.83 ± 0.35	83.42 ± 1.07	99.88 ± 0.10	88.00 ± 1.39	94.72 ± 0.37	6.42
TriCL (Lee and Shin 2023)	79.21 ± 1.23	71.91 ± 1.26	85.69 ± 1.04	82.68 ± 1.24	99.90 ± 0.07	88.90 ± 1.08	97.26 ± 0.24	7
ED-HNN (Wang et al. 2023)	<u>80.31 ± 1.35</u>	73.70 ± 1.38	**89.03 ± 0.53**	<u>83.97 ± 1.55</u>	99.89 ± 0.09	88.07 ± 1.28	97.25 ± 0.28	4.14
Hypergraph-MLP (Tang et al. 2024)	79.80 ± 1.82	<u>73.90 ± 1.57</u>	87.89 ± 0.55	82.32 ± 2.08	99.96 ± 0.07	88.42 ± 1.32	97.52 ± 0.26	5
HyperGINE	79.26 ± 0.41	73.72 ± 0.52	87.91 ± 0.28	82.88 ± 0.48	99.92 ± 0.06	88.52 ± 0.42	97.61 ± 016	5
KHGNN (ours)	**80.67 ± 0.76**	**74.80 ± 1.10**	88.47 ± 0.47	**84.25 ± 0.74**	**100.00 ± 0.00**	**89.60 ± 1.64**	**98.33 ± 0.14**	**1.29**

Table 1: Performance comparison on hypergraph datasets (Mean accuracy (%) ± standard deviation), with the best results in bold, second-best underlined, and the last column showing the average performance ranking.

volution of nearly every node becomes the shortest path features among some nodes, yielding a time complexity of $O\left(\left\lceil\frac{K_v}{2}\right\rceil N\cdot F(n)\right)$. Therefore, the overall time complexity of node representation learning is $O\left(\left(\left\lceil\frac{K_v}{2}\right\rceil E+\left\lceil\frac{K_v}{2}\right\rceil N\right)\cdot F(n)\right)$, a constant multiple of the complexity $O\left((E+N)\cdot F(n)\right)$ found in most HGNNs.

Similarly, time complexity for hyperedge representation learning is $O\left(\left(\left\lceil\frac{K_e}{2}\right\rceil E+\left\lceil\frac{K_e}{2}\right\rceil N\right)\cdot F(e)\right)$, where e represents the average number of hyperedges within the high-order hyperedges constructed via bisection nested convolution. The runtime of KHGNN on real-world datasets will be discussed in ablation experiments.

Experiments

In this section, we evaluate the performance of the proposed KHGNN across seven hypergraph datasets and nine graph datasets. These datasets encompass a variety of tasks, including node classification (*e.g.*, ModelNet40), link prediction (*e.g.*, PCQM-Contact), image classification (*e.g.*, MNIST), and molecular property prediction (*e.g.*, MolHIV). Detailed descriptions of the datasets and experimental hyperparameter settings can be found in Appendix A and B[1].

Comparison on Hypergraph Datasets

Datasets. KHGNN is evaluated on seven hypergraph datasets: Cora, Citeseer, Pubmed, Cora-CA (Yadati et al. 2019), Mushroom (Asuncion and Newman 2007), NTU2012 (Chen et al. 2003), and ModelNet40 (Wu et al. 2015). Following the setup used by Chien et al. (2022), the nodes in each dataset are randomly split into training, validation, and test sets with a ratio of 50/25/25.

Results. Table 1 presents the performance of various models on hypergraph datasets. The proposed KHGNN achieves SOTA results on most datasets while maintaining competitive accuracy on the Pubmed dataset. By expanding the receptive fields of nodes and hyperedges, KHGNN enhances the performance of HyperGINE in node classification, suggesting that distant nodes provide richer information and improve node differentiation. Unlike the straightforward stacking of layers in other HGNNs, KHGNN incorporates path features of nodes, hyperedges, and structures as contextual information, effectively modeling semantic relationships for more precise predictions.

Comparison on Graph Datasets

Unlike node classification tasks which rely on a limited receptive field, graph-level tasks require each node to capture as much graph information as possible to effectively represent graph embeddings. However, the majority of HGNNs overlook graph-level tasks due to their inability to provide sufficient information for nodes and hyperedges. By treating graphs as a special case of hypergraphs, KHGNN effectively models high-order relationships between distant nodes and edges through path features that encapsulate structural information, thereby boosting downstream task performance.

Result on short-range graph benchmark. Table 2 presents the experimental results of KHGNN on datasets characterized by a short average graph diameter, including ZINC, MNIST, CIFAR10, MolHIV, and MolPCB from Benchmarking GNNs (Dwivedi et al. 2023) and Open Graph Benchmark (Hu et al. 2020).

We observe that models with larger receptive fields, such as graph transformers (GT) and KHGNN, outperformed models with limited receptive fields (*e.g.*, HGNN) on small molecule datasets. Notably, our model sets new state-of-the-art performance on ZINC, MNIST, CIFAR10, and MolHIV by expanding the receptive field of both nodes and edges simultaneously. These results also demonstrate that appropriate path features can more effectively model relationships between nodes, leading to more suitable graph embeddings for downstream tasks compared to attention mechanisms. For a detailed experimental comparison with MP GNNs, including most K-hop GNNs, please refer to Appendix C.

Result on long-range graph benchmark. Table 3 presents the performance of KHGNN on Long-Range Graph Benchmark (LRGB) (Dwivedi et al. 2022b), including PascalVOC-SP, Peptides-func, Peptides-struct, and PCQM-Contact, which feature longer average graph diameters.

Traditionally, graph transformer-based GNNs with their global receptive fields outperform message-passing GNNs in modeling long-range node interactions, as demonstrated by significant performance gaps. Despite this, KHGNN, adhering to the 500K parameter budget set by Dwivedi et

[1]https://github.com/robinxlh/KHGNN

Frame	Model	ZINC	MNIST	CIFAR10	MolHIV	MolPCBA
		MAE↓	Accuracy↑	Accuracy↑	AUROC↑	AP↑
MP	GCN (Kipf and Welling 2017)	0.367 ± 0.011	90.705 ± 0.218	55.710 ± 0.381	0.7606 ± 0.0097	0.2020 ± 0.0024
	HGNN (Feng et al. 2019)	0.236 ± 0.006	97.070 ± 0.191	67.210 ± 0.349	0.7632 ± 0.0181	0.2622 ± 0.0013
	GIN (Hu et al. 2019)	0.526 ± 0.051	96.485 ± 0.252	55.255 ± 1.527	0.7778 ± 0.0130	0.2266 ± 0.0028
	GatedGCN (Bresson et al. 2018)	0.282 ± 0.015	97.340 ± 0.143	67.312 ± 0.311	0.7874 ± 0.0119	0.2620 ± 0.0010
	PNA (Corso et al. 2020)	0.188 ± 0.004	97.940 ± 0.120	70.350 ± 0.630	0.7905 ± 0.0132	0.2838 ± 0.0035
	DGN (Beaini et al. 2021)	0.168 ± 0.003	97.940 ± 0.120	72.838 ± 0.417	0.7970 ± 0.0097	0.2885 ± 0.0030
	AGENTNET (Martinkus et al. 2023)	0.144 ± 0.016	-	-	0.7833 ± 0.0069	0.2549 ± 0.0027
	GINE-ViT (He et al. 2023)	0.085 ± 0.005	98.200 ± 0.050	69.670 ± 0.400	0.7792 ± 0.0149	-
	d-DRFWL (Zhou et al. 2024)	0.077 ± 0.002	-	-	0.7818 ± 0.0219	0.2538 ± 0.0019
GT	EGT (Hussain et al. 2022)	0.108 ± 0.009	98.173 ± 0.087	68.702 ± 0.409	0.8060 ± 0.0065	**0.2961 ± 0.0024**
	GraphTrans-ViT (He et al. 2023)	0.096 ± 0.007	97.250 ± 0.230	72.110 ± 0.550	0.7755 ± 0.0208	-
	SGHormer (Zhang et al. 2024)	0.117 ± 0.032	96.850 ± 0.247	67.740 ± 0.158	0.7747 ± 0.0040	0.2743±0.0010
	GraphiT+MAP (Ma et al. 2024)	0.160 ± 0.006	98.104 ± 0.212	-	0.7690 ± 0.0110	-
MP	HyperGINE	0.118 ± 0.004	97.523 ± 0.030	69.587 ± 0.101	0.7792 ± 0.0024	0.2690 ± 0.0015
	KHGNN (ours)	**0.074 ± 0.002**	**98.494 ± 0.045**	**74.807 ± 0.265**	**0.8084 ± 0.0063**	0.2906 ± 0.0037

Table 2: Performance comparison of KHGNN and the SOTA models, including message passing-based GNNs (MP) and graph transformer-based GNNs (GT) on datasets from (Dwivedi et al. 2023) and (Hu et al. 2020), which are short-range graph benchmarks. Results are reported as Mean ± st. over 5 random seeds.

Model	PascalVOC-SP	Peptides-func	Peptides-struct	PCQM-Contact
	F1↑	AP↑	MAE↓	MRR↑
GCN (2017)	12.68 ± 0.60	59.30 ± 0.23	34.96 ± 0.13	32.34 ± 0.06
HGNN (2019)	27.38 ± 0.28	60.57 ± 0.54	27.57 ± 0.26	32.85 ± 0.09
GINE (2019)	12.65 ± 0.76	54.98 ± 0.79	35.47 ± 0.45	31.80 ± 0.27
GatedGCN (2018)	28.73 ± 2.19	58.64 ± 0.77	34.20 ± 0.13	32.18 ± 0.11
PathNN (2023)	-	68.16 ± 0.26	25.45 ± 0.32	-
OptBasisGNN (2023)	33.83 ± 0.61	61.92 ± 0.75	25.61 ± 0.19	32.42 ± 0.45
GUMP (2024)	-	68.43 ± 0.37	25.64 ± 0.23	34.13 ± 0.15
d-DRFWL (2024)	-	59.53 ± 0.48	25.94 ± 0.38	-
SAN+LapPE (2023)	32.30 ± 0.39	63.84 ± 1.21	26.83 ± 0.43	33.50 ± 0.03
SAN+RWSE (2022a)	32.16 ± 0.27	64.39 ± 0.75	25.45 ± 0.12	33.41 ± 0.06
GPS (2022)	37.48 ± 1.09	65.35 ± 0.41	25.00 ± 0.05	33.37 ± 0.06
Specformer (2023)	35.64 ± 0.85	66.86 ± 0.64	25.50 ± 0.14	33.73 ± 0.27
HyperGINE	29.29 ± 0.12	61.06 ± 0.25	26.40 ± 0.03	33.13 ± 0.05
KHGNN (ours)	**38.30 ± 0.70**	**70.11 ± 0.30**	**24.77 ± 0.14**	**35.60 ± 0.04**

Table 3: Results of KHGNN on 4 tasks from Long-Range Graph Benchmark, shown as percentages.

al (2022b), transcends the locality constraints of conventional message passing by enabling information flow along all shortest paths. Unlike PathNN (Michel et al. 2023), which constructs path features exclusively from nodes, KHGNN leverages its bisection nested convolution operator to capture the comprehensive features of nodes, edges, and structure in paths, thereby enabling more precise modeling of long-range interactions. This capability to encode multiple relational pathways empowers KHGNN to exceed the performance of graph transformers, resulting in SOTA outcomes on all evaluated datasets.

Ablation Study

Finally, a series of ablation studies are conducted on Peptides-fun, ZINC and Citeseer to assess the contributions of various components of KHGNN to its performance, with results summarized in Table 4.

The results reveal that increasing the number of hops for node aggregation (K_v) and edge aggregation (K_e) consistently enhances KHGNN's performance on Peptides-func, ZINC, and Citeseer. The most significant improvement is observed on Peptides-func, attributed to its longer average graph diameter and the heightened importance of long-range interactions. Additionally, the training time of KHGNN scales linearly with K_v and K_e across all datasets, as detailed in the Complexity Analysis, without experiencing explosive growth as the receptive field expands.

K_v	K_e	Relative Encoding	Peptides-func		ZINC		Citeseer	
			AP↑	Time(s)	MAE↓	Time(s)	Acc(%)↑	Time(s)
1		✔	0.5965	7.9	0.1309	14.7	73.46	0.2
2		✔	0.6438	9.1	0.1153	19.4	73.76	0.2
2	1	✔	0.6525	10.5	0.1089	23.8	73.93	0.2
3	1	✔	0.6745	11.6	0.0830	27.4	74.15	0.3
4	1	✔	0.6866	12.8	0.0776	29.9	74.27	0.4
4	2	✔	0.6903	14.3	0.0758	34.1	74.45	0.4
5	2	✔	0.6951	15.7	0.0755	37.6	74.67	0.5
6	2	✔	**0.7011**	17.1	**0.0743**	39.7	**74.80**	0.6
6	2		0.6922	13.4	0.1196	35.6	74.59	0.4

Table 4: Ablation study results for K_v-hop node representation learning and K_e-hop hyperedge representation learning, along with the training time for each epoch.

Conclusion

In this paper, we have proposed a K-hop hypergraph neural network (KHGNN), marking the first attempt to utilize path features to guide hypergraph message passing. Our approach departs from existing HGNNs by utilizing a comprehensive bisection aggregation that synthesizes information from nodes, hyperedges, and structures along all shortest paths, thereby facilitating more effective long-range message propagation. Experimental results demonstrate that KHGNN exhibits superior expressive power in both hypergraph and graph tasks, particularly achieving significant performance improvements in modeling long-range dependencies.

Acknowledgements

This work is supported by National Science and Technology Major Project (No. 2020AAA0109703), the Joint Fund Key Program of the National Natural Science Foundation of China(No. U23B2029), in part by the National Natural Science Foundation of China (Nos. 62072386, 62376101), in part by Yunnan Provincial Major Science and Technology Special Plan Project (No. 202402AD080001), in part by Henan Province key research and development project (No. 231111212000), the Open Foundation of Henan Key Laboratory of General Aviation Technology (No. ZHKF-230212) and the Open Foundation the Key Laboratory of Oracle Information Processing of Ministry of Education (No. OIP2024E002).

References

Abboud, R.; Dimitrov, R.; Ceylan; and Ilkan, I. 2022. Shortest path networks for graph property prediction. In *Proceedings of Learning on Graphs Conference*.

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; and Galstyan, A. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*.

Alon, U.; and Yahav, E. 2020. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*.

Antelmi, A.; Cordasco, G.; Polato, M.; Scarano, V.; Spagnuolo, C.; and Yang, D. 2023. A survey on hypergraph representation learning. *ACM Computing Surveys*.

Asuncion, A.; and Newman, D. 2007. UCI machine learning repository.

Beaini, D.; Passaro, S.; Létourneau, V.; Hamilton, W.; Corso, G.; and Liò, P. 2021. Directional graph networks. In *ICML*.

Bo, D.; Shi, C.; Wang, L.; and Liao, R. 2023. Specformer: Spectral Graph Neural Networks Meet Transformers. In *ICLR*.

Bresson, X.; and Laurent, T. 2018. Residual gated graph convnets. In *ICLR*.

Brossard, R.; Frigo, O.; and Dehaene, D. 2020. Graph convolutions that can finally model local structure. *arXiv preprint arXiv:2011.15069*.

Cao, Q.; Shen, H.; Gao, J.; Wei, B.; and Cheng, X. 2020. Popularity prediction on social platforms with coupled graph neural networks. In *WSDM*.

Chen, D.-Y.; Tian, X.-P.; Shen, Y.-T.; and Ouhyoung, M. 2003. On visual similarity based 3D model retrieval. In *Computer Graphics Forum*.

Chien, E.; Pan, C.; Peng, J.; and Milenkovic, O. 2022. You are allset: A multiset function framework for hypergraph neural networks. In *ICLR*.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2020. Adaptive universal generalized pagerank graph neural network. In *ICLR*.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Veličković, P. 2020. Principal neighbourhood aggregation for graph nets. In *NeurIPS*.

Ding, K.; Wang, J.; Li, J.; Li, D.; and Liu, H. 2020. Be more with less: Hypergraph attention networks for inductive text classification. In *EMNLP*.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2023. Benchmarking graph neural networks. *JMLR*.

Dwivedi, V. P.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2022a. Graph neural networks with learnable structural and positional representations. In *ICLR*.

Dwivedi, V. P.; Rampášek, L.; Galkin, M.; Parviz, A.; Wolf, G.; Luu, A. T.; and Beaini, D. 2022b. Long range graph benchmark. In *NeurIPS*.

Feng, J.; Chen, Y.; Li, F.; Sarkar, A.; and Zhang, M. 2022. How powerful are k-hop message passing graph neural networks. In *NeurIPS*.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph neural networks. In *AAAI*.

Gao, Y.; Feng, Y.; Ji, S.; and Ji, R. 2022. HGNN+: General hypergraph neural networks. *TPAMI*.

Guo, Y.; and Wei, Z. 2023. Graph neural networks with learnable and optimal polynomial bases. In *ICML*.

He, X.; Hooi, B.; Laurent, T.; Perold, A.; LeCun, Y.; and Bresson, X. 2023. A generalization of vit/mlp-mixer to graphs. In *ICML*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Huang, J.; Chen, C.; Ye, F.; Hu, W.; and Zheng, Z. 2020. Nonuniform hyper-network embedding with dual mechanism. *ACM Transactions on Information Systems (TOIS)*.

Huang, J.; Lei, F.; Jiang, J.; Zeng, X.; Ma, R.; and Dai, Q. 2023. Multi-order hypergraph convolutional networks integrated with self-supervised learning. *Complex & Intelligent Systems*.

Huang, J.; Lei, F.; Wang, S.; Wang, S.; and Dai, Q. 2021. Hypergraph convolutional network with hybrid higher-order neighbors. In *PRCV*.

Huang, J.; Liu, X.; and Song, Y. 2019. Hyper-path-based representation learning for hyper-networks. In *CIKM*.

Huang, J.; and Yang, J. 2021. Unignn: A unified framework for graph and hypergraph neural networks. In *IJCAI*.

Hussain, M. S.; Zaki, M. J.; and Subramanian, D. 2022. Global self-attention as a replacement for graph convolution. In *SIGKDD*.

Jiang, J.; Wei, Y.; Feng, Y.; Cao, J.; and Gao, Y. 2019. Dynamic hypergraph neural networks. In *IJCAI*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Lee, D.; and Shin, K. 2023. I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In *AAAI*.

Li, P.; Wang, Y.; Wang, H.; and Leskovec, J. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *NeurIPS*.

Ma, G.; Wang, Y.; and Wang, Y. 2024. Laplacian canonization: A minimalist approach to sign and basis invariant spectral embedding. In *NeurIPS*.

Martinkus, K.; Papp, P. A.; Schesch, B.; and Wattenhofer, R. 2023. Agent-based graph neural networks. In *ICLR*.

Michel, G.; Nikolentzos, G.; Lutzeyer, J. F.; and Vazirgiannis, M. 2023. Path neural networks: Expressive and accurate graph neural networks. In *ICML*.

Qiu, H.; Bian, Y.; and Yao, Q. 2024. Graph unitary message passing. *arXiv preprint arXiv:2403.11199*.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a general, powerful, scalable graph transformer. In *NeurIPS*.

Sun, Y.; Deng, H.; Yang, Y.; Wang, C.; Xu, J.; Huang, R.; Cao, L.; Wang, Y.; and Chen, L. 2022. Beyond homophily: Structure-aware path aggregation graph neural network. In *IJCAI*.

Tang, B.; Chen, S.; and Dong, X. 2024. Hypergraph-MLP: Learning on Hypergraphs without Message Passing. In *ICASSP*.

Topping, J.; Di Giovanni, F.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2021. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*.

Wang, G.; Ying, R.; Huang, J.; and Leskovec, J. 2021. Multi-hop attention graph neural network. In *IJCAI*.

Wang, P.; Yang, S.; Liu, Y.; Wang, Z.; and Li, P. 2023. Equivariant hypergraph diffusion neural operators. In *ICLR*.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *ICLR*.

Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; and Talukdar, P. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *NeurIPS*.

Yang, G.; Luo, Z.; Gao, J.; Lai, Y.; Yang, K.; He, Y.; and Li, S. 2024. A multilevel guidance-exploration network and behavior-scene matching method for human behavior anomaly detection. In *ACM MM*.

Yang, Y.; Wang, X.; Song, M.; Yuan, J.; and Tao, D. 2021. Spagan: Shortest path graph attention network. In *IJCAI*.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? In *NeurIPS*.

Yu, S.; Feng, Y.; Zhang, D.; Bedru, H. D.; Xu, B.; and Xia, F. 2020. Motif discovery in networks: A survey. *Computer Science Review*.

Yu, Z.; Li, J.; Chen, L.; and Zheng, Z. 2022. Unifying multi-associations through hypergraph for bundle recommendation. *Knowledge-Based Systems*.

Zhang, H.; Li, J.; Chen, L.; and Zheng, Z. 2024. SGHormer: An Energy-Saving Graph Transformer Driven by Spikes. *arXiv preprint arXiv:2403.17656*.

Zhang, M.; and Li, P. 2021. Nested graph neural networks. In *NeurIPS*.

Zhou, J.; Feng, J.; Wang, X.; and Zhang, M. 2024. Distance-restricted folklore weisfeiler-leman GNNs with provable cycle counting power. In *NeurIPS*.