UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Modeling Edge-Specific Node Features through Co-Representation Neural Hypergraph Diffusion

Zheng, Yijia; Worring, Marcel

DOI
10.1145/3746252.3761094

Publication date
2025

Document Version
Final published version

Published in
CIKM'25

License
CC BY

Link to publication

Citation for published version (APA):
Zheng, Y., & Worring, M. (2025). Modeling Edge-Specific Node Features through Co-Representation Neural Hypergraph Diffusion. In *CIKM'25: Proceedings of the 34th ACM International Conference on Information and Knowledge Management : November 10-14, 2025, Seoul, Republic of Korea* (pp. 4444-4454). Association for Computing Machinery. https://doi.org/10.1145/3746252.3761094

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:10 Oct 2026

Modeling Edge-Specific Node Features through Co-Representation Neural Hypergraph Diffusion

Yijia Zheng
University of Amsterdam
Amsterdam, The Netherlands
y.zheng@uva.nl

Marcel Worring
University of Amsterdam
Amsterdam, The Netherlands
m.worring@uva.nl

Abstract

Hypergraphs are widely being employed to represent complex higher-order relations in real-world applications. Most existing research on hypergraph learning focuses on node-level or edge-level tasks. A practically relevant and more challenging task, edge-dependent node classification (ENC), is still under-explored. In ENC, a node can have different labels across different hyperedges, which requires the modeling of node features unique to each hyperedge. The state-of-the-art ENC solution, WHATsNet, only outputs single node and edge representations, leading to the limitations of **entangled edge-specific features** and **non-adaptive representation sizes** when applied to ENC. Additionally, WHATsNet suffers from the common **oversmoothing issue** in most HGNNs. To address these limitations, we propose **CoNHD**, a novel HGNN architecture specifically designed to model edge-specific features for ENC. Instead of learning separate representations for nodes and edges, CoNHD reformulates within-edge and within-node interactions as a hypergraph diffusion process over node-edge co-representations. We develop a neural implementation of the proposed diffusion process, leveraging equivariant networks as diffusion operators to effectively learn the diffusion dynamics from data. Extensive experiments demonstrate that CoNHD achieves the best performance across all benchmark ENC datasets and several downstream tasks without sacrificing efficiency. Our implementation is available at https://github.com/zhengyijia/CoNHD.

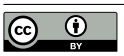
CCS Concepts

• **Mathematics of computing** → **Graph algorithms**; **Hypergraphs**; • **Computing methodologies** → **Machine learning**; • **Information systems** → **Data mining**; **Social networks**.

Keywords

Hypergraph Neural Networks; Hypergraph Diffusion

ACM Reference Format:
Yijia Zheng and Marcel Worring. 2025. Modeling Edge-Specific Node Features through Co-Representation Neural Hypergraph Diffusion. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3746252.3761094



This work is licensed under a Creative Commons Attribution 4.0 International License.
CIKM '25, Seoul, Republic of Korea
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761094

1 Introduction

Real-world applications often involve intricate higher-order relations that cannot be represented by traditional graphs with pairwise connections [6, 22, 72]. Hypergraphs, where an edge can connect more than two nodes, provide a flexible structure for representing such relations [1, 26]. To tackle hypergraph-related tasks such as node classification [8, 47, 82] and edge prediction [15, 37, 39, 54], message passing-based hypergraph neural networks (HGNNs) have become the common solution [40]. Recent research [17, 34] shows that most HGNNs can be formulated as an instantiation of the two-stage message passing framework depicted in Fig. 1(a). The first stage aggregates messages from nodes to update the edge representation, while the second stage aggregates messages from edges to update the node representation. Although message passing-based HGNNs have achieved success in various applications [14, 40, 50], the majority of research efforts have concentrated on node-level and edge-level tasks. In many real-world hypergraphs, a node's property varies with different hyperedges it belongs to. For instance, in a co-authorship network, a researcher may be the lead author in one paper but the corresponding author in another. Likewise, in a multiplayer game, a player might be the winner in one match yet the loser in another. Motivated by such scenarios, Choe et al. [19] introduce a new task namely *edge-dependent node classification* (ENC), where a node can have different labels across different hyperedges. This new task has been shown to be valuable for many downstream tasks [19], including ranking aggregation [18], node clustering [31], and product-return prediction [43].

Although many message passing-based HGNNs can be applied to ENC, Choe et al. [19] highlight that these methods overlook edge-specific node features during aggregation. To address this limitation, they propose WHATsNet, the current state-of-the-art method for ENC. WHATsNet follows the edge-dependent message passing framework as shown in Fig. 1(b), where edge-dependent representations are extracted before aggregation. The final node and edge representations are concatenated to predict the ENC labels. Adopting the dominant message passing framework to address ENC is intuitive, but does it yield the most appropriate solution?

Unlike traditional node-level or edge-level tasks, ENC allows a node to have varying labels across different hyperedges. This requires, as indicated in [19], the model to capture node features unique to each hyperedge. However, the message passing framework aggregates different edge-specific features into a single node representation, leading to the following two limitations:

(1) Entangled edge-specific features. The single node representation entangles edge-specific features from different edges, making it challenging to distinguish features corresponding to a specific target edge. This becomes particularly problematic when

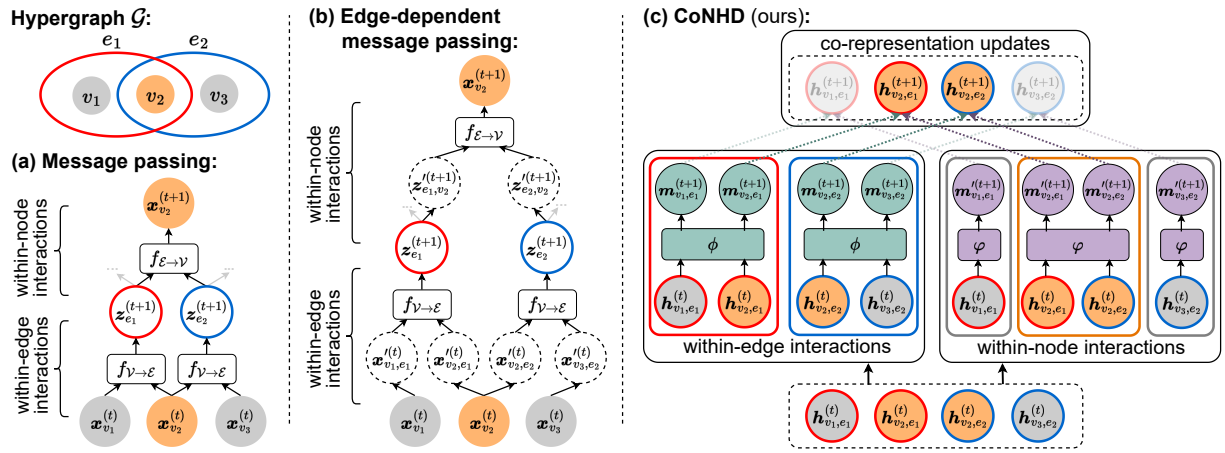


Figure 1: Different HGNN architectures. (a,b) The (edge-dependent) message passing framework aggregates (edge-dependent) messages from neighboring nodes to update a single edge representation through an aggregation function $f_{\mathcal{V}\to\mathcal{E}}$ and then from neighboring edges back to update a single node representation through an aggregation function $f_{\mathcal{E}\to\mathcal{V}}$. (c) Our proposed CoNHD redefines within-edge and within-node interactions as multi-input multi-output processes among node-edge co-representations. These interactions are modeled by two equivariant networks, ϕ and φ, which can generate diverse node-specific or edge-specific information for different node-edge pairs. The outputs from both interactions are used to update the co-representations.

the edge-specific features are highly dissimilar, as the entangled vector may obscure features specific to different edges. To verify this assumption, as shown in Figure 3, we examine the performance of WHATsNet under different node entropy levels, where higher entropy levels indicate that the node has more dissimilar labels in different edges. At low entropy levels, since a node has similar labels in neighboring edges, the prediction may rely on similar features, and therefore WHATsNet with single node representations performs well. As the entropy level increases, dissimilar edge-specific features are required to predict different labels. The performance of WHATsNet drops significantly, which supports our assumption that a single node representation with entangled edge-specific features is insufficient for predicting different ENC labels.

(2) Non-adaptive representation sizes. Storing different edge-specific features in a fixed-size node representation vector causes information loss for large-degree nodes, which interact with more neighboring hyperedges and therefore require larger representation sizes. As shown in Figure 4, WHATsNet fails to generate discriminative embeddings for node-edge pairs related to large-degree nodes. Since low-degree nodes have fewer neighboring edges and do not require large representation sizes, simply increasing the embedding dimension for all nodes leads to excessive computational costs and problems like overfitting and optimization difficulties [28, 48].

Apart from the above two limitations specific to the ENC task, WHATsNet [19] also suffers from the common **oversmoothing issue** in most HGNNs [63, 69], as demonstrated in Fig. 6. This issue hinders the utilization of long-range information and limits model performance. Unlike traditional HGNNs, hypergraph diffusion methods [24, 46, 61] obtain optimal node representations by directly optimizing a regularized objective function, ensuring convergence to the desired solution. Wang et al. [63] propose an HGNN inspired by hypergraph diffusion, demonstrating its robustness to the oversmoothing issue. However, their approach remains within the message passing framework and inherits the two aforementioned limitations when applied to ENC.

To overcome the limitations of message passing for ENC, we introduce **Co**-representation **N**eural **H**ypergraph **D**iffusion (**CoNHD**), a novel diffusion-based HGNN architecture for modeling edge-specific features. Specifically, we show that the two aforementioned limitations are both related to the single-output design in message passing as shown in Fig. 2(a), which only generates a single node or edge representation. Therefore, we first extend the concept of hypergraph diffusion by utilizing node-edge co-representations, redefining the input and output of within-edge and within-node interactions as information exchanged across multiple node-edge pairs, as shown in Fig. 2(b). The co-representation design enables each node to have multiple representations, and the number of these representations scales with the node degree. We further develop a neural implementation that leverages learnable equivariant networks as diffusion operators, which can adaptively learn suitable diffusion dynamics and effectively capture diverse edge-specific features, eliminating the need for handcrafting regularization functions. **Our main contributions** are twofold:

(1) We define **co-representation hypergraph diffusion**, a new concept that generalizes hypergraph diffusion using node-edge co-representations, which offers the benefits of disentangled edge-specific features and adaptive representation sizes.
(2) We propose **CoNHD**, a neural implementation of the proposed diffusion process. This results in a novel HGNN architecture that can learn diffusion dynamics from data and effectively capture edge-specific features for addressing the ENC task.

We conduct extensive experiments to validate the effectiveness and efficiency of CoNHD, demonstrating that CoNHD achieves the best performance across ten ENC datasets as well as several downstream tasks while maintaining high efficiency.

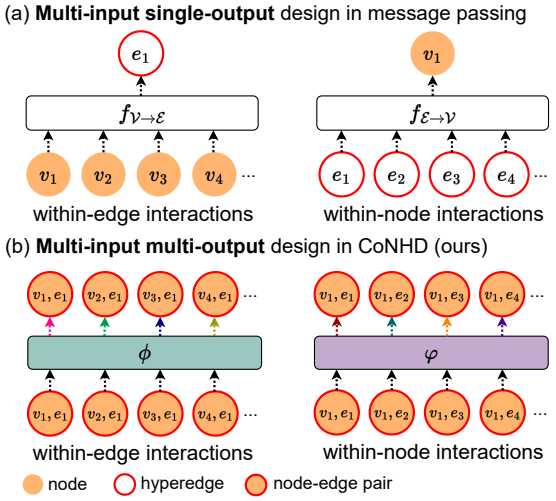


Figure 2: Comparison between the single-output design in message-passing and the multi-output design in our CoNHD method. (a) In the single-output design, information from multiple neighboring nodes or edges is aggregated into a single edge or node using the aggregation function $f_{\mathcal{V}\to\mathcal{E}}$ or $f_{\mathcal{E}\to\mathcal{V}}$, respectively. (b) In our design, information diffuses across node-edge pairs using two multi-input multi-output functions ϕ and φ. These functions are designed as equivariant, which can produce diverse outputs while maintaining element-wise consistency under permutation.

2 Related Work

Hypergraph Neural Networks. Inspired by the success of graph neural networks (GNNs) [41, 66, 67], hypergraph neural networks (HGNNs) have been proposed for modeling complex higher-order relations [21, 40]. HyperGNN [23, 25] and HCHA [5] define hypergraph convolution based on the clique expansion graph. HyperGCN [68] reduces the clique expansion graph into an incomplete graph with mediators. To directly utilize higher-order structures, HNHN [20] and HyperSAGE [3, 4] model the convolution layer as a message passing process with two aggregation stages. UniGNN [34] provides a general framework for extending GNNs to hypergraphs. AllSet [17] implements the aggregation functions in message passing as universal invariant functions. HDS^{ode} improves message passing by modeling it as an ODE-based dynamic system [69]. Recent research explores edge-dependent message passing, where edge-dependent node messages are extracted before feeding them into the aggregation process [2, 57, 63]. LEGCN [70] and MultiSetMixer [57] have multiple representations for a single node. However, these two methods model interactions as an invariant function, which only produces the same propagating messages for different node-edge pairs. This invariance design, as shown in Section 5.4, is insufficient to capture the edge-specific features for ENC. While most existing methods focus on node-level or edge-level tasks [7, 8, 13, 47] and applications [11, 53, 65, 71], the ENC task remains less explored. Choe et al. [19] explore the ENC task and propose WHATsNet, the state-of-the-art solution based on message passing. Different from our method, it employs an aggregation after the equivariant operator to produce a single node or edge representation. While message passing has become a dominant framework for addressing various hypergraph-related tasks [40], its single node and edge representation design suffers from the limitations of entangled edge-specific features and non-adaptive representation sizes when applied to ENC.

(Hyper)graph Diffusion. Different from HGNNs with trainable parameters, hypergraph diffusion is a class of non-parametric regularization methods. (Hyper)graph diffusion [12, 27] models the diffusion information as the gradients derived from minimizing a regularized target function, which regularizes the node representations within the same edge. This ensures that the learned node representations converge to the solution of the optimization target instead of an oversmoothed solution [58, 73]. The technique was first introduced to achieve local and global consistency on graphs [79, 81], and was then generalized to hypergraphs [1, 80]. Zhou et al. [80] propose a regularization function by reducing the higher-order structure in a hypergraph using clique expansion. To directly utilize the higher-order structures, Hein et al. [33] propose a regularization function based on the total variation of the hypergraph. Other regularization functions are designed to improve parallelization ability and introduce non-linearity [36, 46, 59, 60]. Some advanced optimization techniques have been investigated in hypergraph diffusion to improve efficiency [44, 76]. Recent research [12, 29, 42, 58, 63, 64] explores the neural implementation of (hyper)graph diffusion processes, which demonstrate strong robustness against the oversmoothing issue. While hypergraph diffusion methods have shown effectiveness in various tasks like ranking [45], motif clustering [56], and signal processing [51, 78], they are restricted to node representations and cannot address the ENC task.

In this paper, we extend hypergraph diffusion using node-edge co-representations and propose a neural implementation. Most related to our work is ED-HNN [63], which is designed to approximate any traditional hypergraph diffusion process. However, it still follows message passing with single node and edge representations. In contrast, our method directly models interactions among co-representations using multi-input multi-output equivariant functions, effectively capturing edge-specific features and achieving significant performance improvements on the ENC task.

3 Preliminaries

In this section, we introduce the general notations for hypergraphs and present key concepts related to message passing-based HGNNs and traditional node-representation hypergraph diffusion, which are essential for the development of our method.

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a hypergraph, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ represents a set of n nodes, and $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ represents a set of m hyperedges. Each edge $e_i \in \mathcal{E}$ is a non-empty subset of $\mathcal{V}$ and can contain an arbitrary number of nodes. $\mathcal{E}_v = \{e \in \mathcal{E} | v \in e\}$ represents the set of edges that contain node v, and $d_v = |\mathcal{E}_v|$ and $d_e = |e|$ are the degrees of node v and edge e, respectively. We use v_i^e and e_j^v to respectively denote the i-th node in edge e and the j-th edge in $\mathcal{E}_v$. $\boldsymbol{X}^{(0)} = [\boldsymbol{x}_{v_1}^{(0)}, \ldots, \boldsymbol{x}_{v_n}^{(0)}]^\top$ is the initial node feature matrix.

Since the nodes and edges in a hypergraph are inherently unordered, it is important to ensure that the outputs of the interaction

modeling functions are consistent regardless of the input ordering. This requirement is formally captured by two key properties: permutation invariance and permutation equivariance. A permutation invariant function is suitable for single-output settings, where the final output remains unchanged under input reordering. In contrast, a permutation equivariant function is well-suited for multi-output settings where permuting the inputs induces the same permutation in the multiple outputs with element-wise consistency. Here we formally give the definitions of these two properties. Let $\mathbb{S}_n$ denote the symmetric group on n elements, where each action $\pi \in \mathbb{S}_n$ acts on any input matrix $\boldsymbol{I} \in \mathbb{R}^{n\times d}$ by permuting its rows.

Definition 1 (Permutation Invariance). *A function $g : \mathbb{R}^{n\times d} \to \mathbb{R}^{d'}$ is permutation invariant if it satisfies $g(\pi \cdot \boldsymbol{I}) = g(\boldsymbol{I})$ for all $\pi \in \mathbb{S}_n$ and $\boldsymbol{I} \in \mathbb{R}^{n\times d}$.*

Definition 2 (Permutation Equivariance). *A function $g : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d'}$ is permutation equivariant if it satisfies $g(\pi \cdot \boldsymbol{I}) = \pi \cdot g(\boldsymbol{I})$ for all $\pi \in \mathbb{S}_n$ and $\boldsymbol{I} \in \mathbb{R}^{n\times d}$.*

Message Passing-based HGNNs. Message passing [17, 34] has become a standard framework for most HGNNs, which models the interactions in within-edge and within-node structures as two multi-input single-output aggregation functions $f_{\mathcal{V}\to\mathcal{E}}$ and $f_{\mathcal{E}\to\mathcal{V}}$:

$$\boldsymbol{z}_e^{(t+1)} = f_{\mathcal{V}\to\mathcal{E}}\big(\boldsymbol{X}_e^{(t)}; \boldsymbol{z}_e^{(t)}\big), \tag{1}$$

$$\boldsymbol{x}_v^{(t+1)} = f_{\mathcal{E}\to\mathcal{V}}\big(\boldsymbol{Z}_v^{(t+1)}; \boldsymbol{x}_v^{(t)}\big). \tag{2}$$

Here $\boldsymbol{x}_v^{(t)}$ and $\boldsymbol{z}_e^{(t)}$ are the node and edge representations in the (t)-th iteration. $\boldsymbol{x}_v^{(0)}$ is the initial node features, and $\boldsymbol{z}_e^{(0)}$ is typically initialized to a zero vector. $\boldsymbol{X}_e^{(t)}$ denotes the representations of nodes contained in edge e, *i.e.*, $\boldsymbol{X}_e^{(t)} = \big[\boldsymbol{x}_{v_1^e}^{(t)}, \ldots, \boldsymbol{x}_{v_{d_e}^e}^{(t)}\big]^\top$. Similarly, $\boldsymbol{Z}_v^{(t)} = \big[\boldsymbol{z}_{e_1^v}^{(t)}, \ldots, \boldsymbol{z}_{e_{d_v}^v}^{(t)}\big]^\top$ denotes the representations of edges containing node v. $f_{\mathcal{V}\to\mathcal{E}}$ and $f_{\mathcal{E}\to\mathcal{V}}$ are two invariant functions that take multiple representations from neighboring nodes or edges as inputs, but only output a single edge or node representation.

Hypergraph Diffusion. Hypergraph diffusion learns node representations $\boldsymbol{X} = \big[\boldsymbol{x}_{v_1}, \ldots, \boldsymbol{x}_{v_n}\big]^\top$, where $\boldsymbol{x}_{v_i} \in \mathbb{R}^d$, by minimizing a hypergraph-regularized target function [49, 59]. For brevity, we use $\boldsymbol{X}_e = \big[\boldsymbol{x}_{v_1^e}, \ldots, \boldsymbol{x}_{v_{d_e}^e}\big]^\top$ to denote the representations of nodes contained in edge e. The target function is the weighted summation of some non-structural and structural regularization functions. The non-structural regularization function is independent of the hypergraph structure, which is typically defined as a squared loss function between the learned node representation vector $\boldsymbol{x}_v$ and the node attribute vector $\boldsymbol{a}_v$ (composed of initial node features $\boldsymbol{x}_v^{(0)}$ [56] or observed node labels [59]). The structural regularization functions incorporate the hypergraph structure and apply regularization to multiple node representations within the same hyperedge, which are invariant functions. Many structural regularization functions are designed by heuristics [32, 33, 60, 80]. For instance, the clique expansion (CE) regularization functions [80], defined as $\Omega_{\text{CE}}(\boldsymbol{X}_e) := \sum_{v,u\in e}\|\boldsymbol{x}_v - \boldsymbol{x}_u\|_2^2$, encourage the representations of all nodes in the same hyperedge to become similar. Alternatively, the total variation (TV) functions, defined as $\Omega_{\text{TV}}(\boldsymbol{X}_e) := \max_{v,u\in e}\|\boldsymbol{x}_v - \boldsymbol{x}_u\|^p (p \in \{1, 2\})$, focus on reducing the discrepancy between the most dissimilar nodes within an edge. Without making a choice among these functions, here we discuss the general form of hypergraph diffusion, which can be defined as:

Definition 3 (Node-Representation Hypergraph Diffusion). *Given a non-structural regularization function $\mathcal{R}_v(\cdot; \boldsymbol{a}_v) : \mathbb{R}^d \to \mathbb{R}$ and a structural regularization function $\Omega_e(\cdot) : \mathbb{R}^{d_e\times d} \to \mathbb{R}$, the node-representation hypergraph diffusion learns representations by solving the following optimization problem*

$$\boldsymbol{X}^\star = \arg\min_{X}\left\{\sum_{v\in\mathcal{V}}\mathcal{R}_v\big(\boldsymbol{x}_v; \boldsymbol{a}_v\big) + \lambda\sum_{e\in\mathcal{E}}\Omega_e\big(\boldsymbol{X}_e\big)\right\}. \tag{3}$$

Here $\Omega_e(\cdot)$ is also referred to as the edge regularization function. $\boldsymbol{X}^\star$ denotes the matrix of all learned node representations, which can be used for predicting the node labels.

4 Methodology

In this section, we propose a new hypergraph diffusion process based on node-edge co-representations, and then develop CoNHD, a learnable neural implementation of the proposed diffusion process. This leads to the novel HGNN architecture shown in Fig. 1(c).

4.1 Co-Representation Hypergraph Diffusion

In this section, we introduce the co-representation hypergraph diffusion process for modeling edge-specific features in ENC. We first formally define the ENC task following [19].

Definition 4 (Edge-Dependent Node Classification (ENC)). *Given **(1)** a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, **(2)** a label space $\mathcal{C}$, **(3)** observed edge-dependent node labels for $\mathcal{E}' \subset \mathcal{E}$ (i.e., $y_{v,e} \in \mathcal{C}, \forall v \in e, \forall e \in \mathcal{E}'$), and **(4)** an initial node feature matrix $\boldsymbol{X}^{(0)}$, the ENC task is to predict the unobserved edge-dependent node labels for $\mathcal{E} \setminus \mathcal{E}'$ (*i.e., $y_{v,e} \in \mathcal{C}, \forall v \in e, \forall e \in \mathcal{E} \setminus \mathcal{E}'$).*

In ENC, the label $y_{v,e}$ is associated with both node v and edge e. We extend hypergraph diffusion to learn a co-representation $\boldsymbol{h}_{v,e} \in \mathbb{R}^d$ for each node-edge pair (v, e). Let $\boldsymbol{H} = \big[\ldots, \boldsymbol{h}_{v,e}, \ldots\big]^\top$ be the matrix containing co-representation vectors of all node-edge pairs. We use $\boldsymbol{H}_e = \big[\boldsymbol{h}_{v_1^e,e}, \ldots, \boldsymbol{h}_{v_{d_e}^e,e}\big]^\top$ and $\boldsymbol{H}_v = \big[\boldsymbol{h}_{v,e_1^v}, \ldots, \boldsymbol{h}_{v,e_{d_v}^v}\big]^\top$ to denote the co-representations associated with edge e and node v, respectively. With these notations, the co-representation hypergraph diffusion is defined as:

Definition 5 (Co-Representation Hypergraph Diffusion). *Given a non-structural regularization function $\mathcal{R}_{v,e}^{co}(\cdot; \boldsymbol{a}_{v,e}) : \mathbb{R}^d \to \mathbb{R}$, structural regularization functions $\Omega_e^{co}(\cdot) : \mathbb{R}^{d_e\times d} \to \mathbb{R}$ and $\Omega_v^{co}(\cdot) : \mathbb{R}^{d_v\times d} \to \mathbb{R}$, the co-representation hypergraph diffusion learns node-edge co-representations by solving the following optimization problem*

$$\begin{aligned}\boldsymbol{H}^\star = \arg\min_{\boldsymbol{H}}\Bigg\{&\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}_v}\mathcal{R}_{v,e}^{\text{co}}\big(\boldsymbol{h}_{v,e}; \boldsymbol{a}_{v,e}\big)\\ &+\lambda\sum_{e\in\mathcal{E}}\Omega_e^{\text{co}}\big(\boldsymbol{H}_e\big) + \gamma\sum_{v\in\mathcal{V}}\Omega_v^{\text{co}}\big(\boldsymbol{H}_v\big)\Bigg\}.\end{aligned} \tag{4}$$

Here $\mathcal{R}_{v,e}^{\text{co}}(\cdot; \boldsymbol{a}_{v,e})$ is a squared loss function following traditional hypergraph diffusion, and $\boldsymbol{a}_{v,e}$ can be any related attributes of the

node-edge pair (v, e) (*e.g.*, node features or edge features). $\Omega_e^{\text{co}}(\cdot)$ and $\Omega_v^{\text{co}}(\cdot)$ are referred to as the co-edge and co-node regularization functions, respectively. They apply regularization to co-representations associated with the same node or edge, which can be implemented as any invariant structural regularization functions designed for traditional node-representation hypergraph diffusion [32, 33, 80]. Instead of making a choice from these handcrafted functions, in Section 4.2, we will develop a neural implementation that can adaptively learn suitable diffusion dynamics from data.

Depending on whether the regularization functions are differentiable, we can solve Eq. 4 using one of two standard optimization methods: gradient descent (GD) or alternating direction method of multipliers (ADMM) [9]. We adopt the GD-based implementation throughout our experiments, while we also provide an ADMM-based implementation in our source code for completeness. We initialize $\boldsymbol{h}_{v,e}^{(0)} = \boldsymbol{a}_{v,e}$, and solve it using GD with a step size α:

$$\begin{aligned}\boldsymbol{h}_{v,e}^{(t+1)} =& \boldsymbol{h}_{v,e}^{(t)} - \alpha\Big(\nabla\mathcal{R}_{v,e}^{\text{co}}\big(\boldsymbol{h}_{v,e}^{(t)};\boldsymbol{a}_{v,e}\big) \\ &+ \lambda\big[\nabla\Omega_e^{\text{co}}\big(\boldsymbol{H}_e^{(t)}\big)\big]_v + \gamma\big[\nabla\Omega_v^{\text{co}}\big(\boldsymbol{H}_v^{(t)}\big)\big]_e\Big),\end{aligned} \tag{5}$$

where ∇ is the gradient operator. $[\cdot]_v$ and $[\cdot]_e$ represent the gradient vector associated with node v and edge e, respectively. For example, $\big[\nabla\Omega_e^{\text{co}}(\boldsymbol{H}_e^{(t)})\big]_v$ represents the gradient w.r.t. $\boldsymbol{h}_{v,e}^{(t)}$. Similar to the traditional hypergraph diffusion, we refer to $\nabla\Omega_e^{\text{co}}(\cdot)$ as the *co-edge diffusion operator*, which models within-edge interactions among co-representations and generates information that should "diffuse" to each node-edge pair. $\nabla\Omega_v^{\text{co}}(\cdot)$ is referred to as the *co-node diffusion operators*. We now reveal a critical property of the diffusion operators.

Proposition 1. *In the co-representation hypergraph diffusion with permutation invariant co-edge and co-node regularization functions, the corresponding co-edge and co-node diffusion operators are permutation equivariant.*

Proof. We analyze $\nabla\Omega_e^{\text{co}}(\cdot)$ here, while the analysis for $\nabla\Omega_v^{\text{co}}(\cdot)$ is analogous. Since Ω_e^{co} is permutation invariant, for any $\pi \in \mathbb{S}_n$ we have $\Omega_e^{\text{co}}(\boldsymbol{P}_\pi\boldsymbol{H}) = \Omega_e^{\text{co}}(\boldsymbol{H})$, where $\boldsymbol{P}_\pi$ is the corresponding row permutation matrix of action π. Due to the linearity of the permutation action, the Jacobian matrix of $\boldsymbol{P}_\pi\boldsymbol{H}$ with respect to $\boldsymbol{H}$ is $\boldsymbol{P}_\pi$. By applying the chain rule we have

$$\nabla\Omega_e^{\text{co}}(\boldsymbol{P}_\pi\boldsymbol{H}) = \boldsymbol{P}_\pi\frac{\partial\Omega_e^{\text{co}}(\boldsymbol{P}_\pi\boldsymbol{H})}{\partial(\boldsymbol{H})} = \boldsymbol{P}_\pi\frac{\partial\Omega_e^{\text{co}}(\boldsymbol{H})}{\partial(\boldsymbol{H})} = \boldsymbol{P}_\pi\nabla\Omega_e^{\text{co}}(\boldsymbol{H}).$$

This completes the proof. □

This property shows that the diffusion operators derived from the co-representation diffusion process not only reformulate the within-edge and within-node interactions as multi-output functions, but also satisfy the equivariance property that ensures the diverse output results commute according to the input ordering.

Next, we state the relation between the co-representation hypergraph diffusion process and the traditional node-representation hypergraph diffusion process.

Proposition 2. *The traditional node-representation hypergraph diffusion is a special case of the co-representation hypergraph diffusion, while the converse does not hold.*

Proof. For each $v \in \mathcal{V}$, we introduce a set of auxiliary variables $\{\boldsymbol{h}_{v,e_i} | e_i \in \mathcal{E}_v\}$, satisfying $\boldsymbol{h}_{v,e_i} = \boldsymbol{h}_{v,e_j}$ for any $e_i, e_j \in \mathcal{E}_v$. Let $\boldsymbol{H}$ denote the collection of all auxiliary variables. Then the original problem in Eq. 3 can be reformulated as the following constrained optimization problem:

$$\begin{aligned}\arg\min_{\boldsymbol{H}} \quad & \left\{\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}_v}\frac{1}{d_v}\mathcal{R}_v(\boldsymbol{h}_{v,e};\boldsymbol{a}_v) + \lambda\sum_{e\in\mathcal{E}}\Omega_e(\boldsymbol{H}_e)\right\}, \\ \text{s.t.} \quad & \boldsymbol{h}_{v,e_i} = \boldsymbol{h}_{v,e_j}, \quad \forall v \in \mathcal{V},\ \forall e_i, e_j \in \mathcal{E}_v.\end{aligned} \tag{6}$$

Let $\boldsymbol{H}^\star$ be an optimal solution to Eq. (6). Then the solution to the original problem satisfies $\boldsymbol{x}_v^\star = \boldsymbol{h}_{v,e}^\star$ for any $e \in \mathcal{E}_v$.

We can set $\mathcal{R}_{v,e}^{\text{co}}(\cdot;\boldsymbol{a}_{v,e}) = \frac{1}{d_v}\mathcal{R}_v(\cdot;\boldsymbol{a}_v)$ and $\Omega_e^{\text{co}}(\cdot) = \Omega_e(\cdot)$ in Eq. 4, and set $\Omega_v^{\text{co}}(\cdot)$ as the CE regularization functions [80], *i.e.*, $\Omega_v^{\text{co}}(\boldsymbol{H}_v) = \Omega_{\text{CE}}(\boldsymbol{H}_v) = \sum_{e_i,e_j\in\mathcal{E}_v}\|\boldsymbol{h}_{v,e_i} - \boldsymbol{h}_{v,e_j}\|_2^2$. Then Eq. 4 can be reformulated as follows:

$$\begin{aligned}\arg\min_{\boldsymbol{H}} \Bigg\{ & \sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}_v}\frac{1}{d_v}\mathcal{R}_v(\boldsymbol{h}_{v,e};\boldsymbol{a}_v) \\ & + \lambda\sum_{e\in\mathcal{E}}\Omega_e(\boldsymbol{H}_e) + \gamma\sum_{v\in\mathcal{V}}\Omega_{\text{CE}}(\boldsymbol{H}_v)\Bigg\}.\end{aligned} \tag{7}$$

Here $\Omega_{\text{CE}}(\cdot)$ is exactly the exterior penalty function [74] for the given equality constraints in Eq. 6. Thus as $\gamma \to \infty$, Eq. 7 yields the same optimal solutions as Eq. 6. The converse does not hold, since the node-representation hypergraph diffusion enforces a single node representation for each node and cannot accommodate multiple co-representations. This completes the proof. □

Node-representation hypergraph diffusion is equivalent to imposing a strict constraint that all the co-representations $\boldsymbol{h}_{v,e_i}$ associated with the same node v must be identical, resulting in a single unified node representation. Our method relaxes this hard constraint by co-node regularization functions, allowing multiple co-representations associated with the same node to be different while still being constrained by certain regularization terms.

4.2 Neural Implementation

In this section, we propose **Co**-representation **N**eural **H**ypergraph **D**iffusion (**CoNHD**), which is a neural implementation of the diffusion process defined in Definition 5 without the need for handcrafting regularization functions.

Since $\mathcal{R}_{v,e}^{co}(\boldsymbol{h}_{v,e}^{(t)};\boldsymbol{a}_{v,e}) = \frac{1}{2}\|\boldsymbol{h}_{v,e}^{(t)} - \boldsymbol{a}_{v,e}\|^2$ is a squared loss function, we have $\nabla\mathcal{R}_{v,e}^{\text{co}}(\boldsymbol{h}_{v,e}^{(t)};\boldsymbol{a}_{v,e}) = \boldsymbol{h}_{v,e}^{(t)} - \boldsymbol{a}_{v,e}$. Eq. 5 can be rewritten as:

$$\begin{aligned}\boldsymbol{h}_{v,e}^{(t+1)} =& (1-\alpha)\boldsymbol{h}_{v,e}^{(t)} - \alpha\lambda\big[\nabla\Omega_e^{\text{co}}\big(\boldsymbol{H}_e^{(t)}\big)\big]_v \\ & - \alpha\gamma\big[\nabla\Omega_v^{\text{co}}\big(\boldsymbol{H}_v^{(t)}\big)\big]_e + \alpha\boldsymbol{a}_{v,e}.\end{aligned} \tag{8}$$

Therefore, $\boldsymbol{h}_{v,e}^{(t+1)}$ is a linear combination of the co-representation in the last step $\boldsymbol{h}_{v,e}^{(t)}$, within-edge and within-node diffusion information $\big[\nabla\Omega_e^{\text{co}}\big(\boldsymbol{H}_e^{(t)}\big)\big]_v$ and $\big[\nabla\Omega_v^{\text{co}}\big(\boldsymbol{H}_v^{(t)}\big)\big]_e$, and initial features $\boldsymbol{h}_{v,e}^{(0)} = \boldsymbol{a}_{v,e}$. To avoid handcrafting regularization functions and manual choice of the factors α, λ, and γ, we define two networks, ϕ and φ, to approximate the two interaction processes, and a linear

layer ψ to approximate the co-representation update process. The $(t+1)$-th layer can be represented as:

$$\boldsymbol{M}_e^{(t+1)} = \phi\big(\boldsymbol{H}_e^{(t)}\big),\ \boldsymbol{M}_v'^{(t+1)} = \varphi\big(\boldsymbol{H}_v^{(t)}\big), \tag{9}$$

$$\boldsymbol{h}_{v,e}^{(t+1)} = \psi\big(\big[\boldsymbol{h}_{v,e}^{(t)}, \boldsymbol{m}_{v,e}^{(t+1)}, \boldsymbol{m}'^{(t+1)}_{v,e}, \boldsymbol{h}_{v,e}^{(0)}\big]\big). \tag{10}$$

Here, ϕ and φ serve as the neural implementation of the diffusion operators, which should satisfy the permutation equivariance property of the co-edge and co-node diffusion operators stated in Proposition 1. For the implementation of the diffusion operators, we explore two popular equivariant neural networks, UNB [52, 63] and ISAB [17]. Notably, CoNHD is a general HGNN framework allowing different equivariant network implementations for the diffusion operators, not limited to the two investigated in this work. $\boldsymbol{M}_e^{(t)} = \left[\boldsymbol{m}^{(t)}_{v_1^e,e}, \ldots, \boldsymbol{m}^{(t)}_{v_{d_e}^e,e}\right]^\top$ and $\boldsymbol{M}_v'^{(t)} = \left[\boldsymbol{m}'^{(t)}_{v,e_1^v}, \ldots, \boldsymbol{m}'^{(t)}_{v,e_{d_v}^v}\right]$ are the within-edge and within-node diffusion information generated using the neural diffusion operators ϕ and φ. The function $\psi(\cdot)$, implemented as a linear layer, collects diffusion information and updates the co-representations.

Previous research based on traditional hypergraph diffusion only explores modeling the composition of within-edge and within-node interactions as an equivariant function, while each interaction is still an invariant aggregation function [63]. Differently, WHATsNet [19] explores utilizing the equivariant module in both interactions. However, the multiple outputs serve only as intermediate results, with an aggregation module applied at the end, where the composition is still an invariant aggregation function and only a single node or edge representation is updated. In contrast, our method reformulates within-edge and within-node interactions as two distinct equivariant functions, ensuring diverse update information for different node-edge pairs, which is helpful for modeling node/edge-specific features and improves ENC performance as shown in Table 4.

To demonstrate the expressiveness of CoNHD, we compare it with the message passing framework defined in Eq. 1-2, which, as previously noted, is followed by most HGNNs [17, 34], including the state-of-the-art ENC solution WHATsNet [19]. Following [19], we regard the concatenation of node and edge representations as the final embeddings, which can be used to predict ENC labels. This leads to the following proposition.

Proposition 3. *With the same embedding dimension, CoNHD is expressive enough to represent the message passing framework, while the converse does not hold.*

Proof. We provide a proof sketch here. We first show that CoNHD can express any models following the message passing framework. By initializing $\boldsymbol{h}_{v,e}^{(0)} = [\boldsymbol{x}_v^{(0)}, \boldsymbol{z}_e^{(0)}]$, we can alternately update edge and node representations using two CoNHD layers. Specifically, the first aggregates nodes-to-edge messages via ϕ to form the outputs $\boldsymbol{h}_{v,e}^{(2t+1)} = [\boldsymbol{x}_v^{(t)}, \boldsymbol{z}_e^{(t+1)}]$. The second aggregates edges-to-node messages via φ to form the outputs $\boldsymbol{h}_{v,e}^{(2(t+1))} = [\boldsymbol{x}_v^{(t+1)}, \boldsymbol{z}_e^{(t+1)}]$. Since ϕ and φ can be implemented as universal equivariant functions like UNB, CoNHD can approximate the same updates as $f_{\mathcal{V}\to\mathcal{E}}$ and $f_{\mathcal{E}\to\mathcal{V}}$ in Eq. 1-2. Conversely, message passing models cannot express CoNHD, as they generate only one representation for each node or edge, whereas CoNHD allows multiple node-edge co-representations for each node or edge. □

Proposition 3 shows that CoNHD is more expressive than all methods following the message passing framework. Notably, this gain in expressiveness does not incur additional complexity, as further analyzed in the next section.

4.3 Complexity Analysis

In this section, we analyze the time and space complexity of CoNHD compared to methods based on the dominant message passing framework, including WHATsNet.

Time Complexity. CoNHD computes within-edge and within-node interactions using UNB or ISAB operators, both with linear computational complexity relative to the input size. The complexity of both interactions is $O(\sum_{e\in\mathcal{E}}(d_e d^2) + \sum_{v\in\mathcal{V}}(d_v d^2)) = O(d^2 \sum_{e\in\mathcal{E}} d_e)$, where d is the hidden size. For the update function, the complexity is $O(d^2 \sum_{e\in\mathcal{E}} d_e)$. Therefore, the overall complexity of CoNHD is $O(Ld^2 \sum_{e\in\mathcal{E}} d_e)$, where L is the number of layers. The overall time complexity is linear to the number of node-edge pairs, *i.e.*, $\sum_{e\in\mathcal{E}} d_e$, which is the same as other HGNNs within the message passing framework. For example, WHATsNet has the complexity of $O(L \cdot (2 \cdot d^2 \sum_{e\in\mathcal{E}} d_e + \sum_{e\in\mathcal{E}} d^2 d_e + \sum_{v\in\mathcal{V}} d^2 d_v))$, which is of the same order as that of CoNHD after ignoring constant factors and low-order terms. However, WHATsNet incurs higher runtime in practice due to additional edge-dependent feature extraction and aggregation steps in each layer, as well as the involvement of indirectly connected neighbors during the message passing process.

Space Complexity. Since the number of input co-representations in each layer of our model depends on the number of node-edge pairs, *i.e.*, $\sum_{e\in\mathcal{E}} d_e$, the size of the inputs is $O(d \sum_{e\in\mathcal{E}} d_e)$. For within-edge and within-node interactions, both UNB and ISAB implementations utilize MLPs to perform feature transformation, where the size of the weights is $O(d^2)$. The sizes of the outputs for the within-edge and within-node interactions are $O(d \sum_{e\in\mathcal{E}} d_e)$. In the update process, the concatenated input is a $4d$-dimensional vector. This is then passed through a linear layer to output the updated co-representations, where the weight size is $O(4d^2)$. Therefore, the total space complexity of L layers after removing the constant factors is $O(L(d^2 + d\sum_{e\in\mathcal{E}} d_e)) = O(Ld(d + \sum_{e\in\mathcal{E}} d_e))$. The overall space complexity is linear to the number of node-edge pairs in the input hypergraph, *i.e.*, $\sum_{e\in\mathcal{E}} d_e$. which is the same as those edge-dependent message passing methods, including WHATsNet [19], which generate multiple edge-dependent node representations for each node in the calculation process.

5 Experiments

In this section, we conduct experiments to evaluate the effectiveness and efficiency of the proposed CoNHD method on the ENC task as well as several downstream tasks.

5.1 Effectiveness and Efficiency on ENC

Datasets. We conduct experiments on ten ENC datasets. These datasets include all six datasets in [19], which are Email (Email-Enron and Email-Eu), StackOverflow (Stack-Biology and Stack-Physics), and Co-authorship networks (Coauth-DBLP and Coauth-AMiner). Notably, Email-Enron and Email-Eu have relatively large node degrees, while Email-Enron has relatively large edge degrees as well. Additionally, as real-world hypergraph structures typically

Table 1: Performance of edge-dependent node classification. Bold numbers represent the best results, while underlined numbers indicate the second-best. "O.O.M." means "out of memory". Shaded cells indicate that our method significantly outperforms the best baseline (p-value < 0.05, based on the Wilcoxon signed-rank test). "A.R." denotes the average ranking among all datasets.

Method	Email-Enron		Email-Eu		Stack-Biology		Stack-Physics		Coauth-DBLP		A.R. of Micro-F1
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	
GraphSAGE	0.775 ± 0.005	0.714 ± 0.007	0.658 ± 0.001	0.564 ± 0.005	0.689 ± 0.010	0.598 ± 0.014	0.660 ± 0.011	0.523 ± 0.018	0.474 ± 0.002	0.401 ± 0.008	11.7
GAT	0.736 ± 0.056	0.611 ± 0.103	0.618 ± 0.002	0.580 ± 0.024	0.692 ± 0.015	0.628 ± 0.010	0.725 ± 0.024	0.636 ± 0.043	0.575 ± 0.005	0.558 ± 0.007	8.1
ADGN	0.790 ± 0.001	0.723 ± 0.001	0.667 ± 0.001	0.622 ± 0.006	0.714 ± 0.002	0.651 ± 0.001	0.686 ± 0.014	0.537 ± 0.019	0.505 ± 0.006	0.440 ± 0.020	8.6
HyperGNN	0.725 ± 0.004	0.674 ± 0.003	0.633 ± 0.001	0.533 ± 0.008	0.689 ± 0.002	0.624 ± 0.007	0.686 ± 0.004	0.630 ± 0.002	0.540 ± 0.004	0.519 ± 0.002	10.0
HAT	0.817 ± 0.001	0.753 ± 0.004	0.669 ± 0.001	0.638 ± 0.002	0.661 ± 0.005	0.606 ± 0.005	0.708 ± 0.005	0.643 ± 0.009	0.503 ± 0.004	0.483 ± 0.006	7.9
UniGCNII	0.734 ± 0.010	0.656 ± 0.010	0.630 ± 0.005	0.565 ± 0.013	0.610 ± 0.004	0.433 ± 0.007	0.671 ± 0.022	0.492 ± 0.016	0.497 ± 0.003	0.476 ± 0.002	13.5
AllSet	0.796 ± 0.014	0.719 ± 0.020	0.666 ± 0.005	0.624 ± 0.021	0.571 ± 0.054	0.446 ± 0.081	0.728 ± 0.039	0.646 ± 0.046	0.495 ± 0.038	0.487 ± 0.040	9.0
HDS^{ode}	0.805 ± 0.001	0.740 ± 0.006	0.651 ± 0.000	0.577 ± 0.005	0.708 ± 0.001	0.643 ± 0.004	0.737 ± 0.001	0.635 ± 0.008	0.558 ± 0.001	0.550 ± 0.002	7.3
LEGCN	0.783 ± 0.001	0.728 ± 0.007	0.639 ± 0.001	0.535 ± 0.004	0.668 ± 0.002	0.572 ± 0.006	0.701 ± 0.003	0.575 ± 0.018	0.499 ± 0.003	0.490 ± 0.002	7.5
MultiSetMixer	0.818 ± 0.001	0.755 ± 0.005	0.670 ± 0.001	0.636 ± 0.005	0.709 ± 0.001	0.643 ± 0.003	0.754 ± 0.001	0.679 ± 0.004	0.559 ± 0.001	0.554 ± 0.001	6.0
HNN	0.763 ± 0.003	0.679 ± 0.007	O.O.M.	O.O.M.	0.618 ± 0.015	0.568 ± 0.013	0.683 ± 0.005	0.617 ± 0.005	0.488 ± 0.006	0.482 ± 0.006	12.4
ED-HNN	0.778 ± 0.001	0.713 ± 0.004	0.648 ± 0.001	0.558 ± 0.004	0.688 ± 0.005	0.506 ± 0.002	0.726 ± 0.002	0.617 ± 0.006	0.514 ± 0.016	0.484 ± 0.024	9.3
WHATsNet	0.826 ± 0.001	0.761 ± 0.003	0.671 ± 0.000	0.645 ± 0.003	0.742 ± 0.002	0.685 ± 0.003	0.770 ± 0.003	0.707 ± 0.004	0.604 ± 0.003	0.592 ± 0.004	5.2
CoNHD (UNB) (*ours*)	0.905 ± 0.001	0.858 ± 0.004	0.708 ± 0.001	0.689 ± 0.001	0.748 ± 0.003	0.694 ± 0.005	0.776 ± 0.001	**0.712 ± 0.005**	**0.620 ± 0.002**	**0.604 ± 0.002**	1.9
CoNHD (ISAB) (*ours*)	**0.911 ± 0.001**	**0.871 ± 0.002**	**0.709 ± 0.001**	**0.690 ± 0.002**	**0.749 ± 0.002**	**0.695 ± 0.004**	**0.777 ± 0.001**	0.710 ± 0.004	0.619 ± 0.002	**0.604 ± 0.003**	**1.1**

Method	Coauth-AMiner		Cora-Outsider		DBLP-Outsider		Citeseer-Outsider		Pubmed-Outsider		A.R. of Macro-F1
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	
GraphSAGE	0.441 ± 0.013	0.398 ± 0.012	0.520 ± 0.009	0.518 ± 0.007	0.490 ± 0.029	0.427 ± 0.083	0.704 ± 0.005	0.704 ± 0.005	0.677 ± 0.003	0.663 ± 0.002	11.8
GAT	0.623 ± 0.006	0.608 ± 0.009	0.531 ± 0.009	0.521 ± 0.008	0.563 ± 0.003	0.548 ± 0.003	0.704 ± 0.011	0.702 ± 0.011	0.677 ± 0.003	0.670 ± 0.002	7.9
ADGN	0.452 ± 0.009	0.415 ± 0.014	0.533 ± 0.007	0.524 ± 0.005	0.559 ± 0.005	0.548 ± 0.001	0.706 ± 0.008	0.705 ± 0.008	0.669 ± 0.003	0.667 ± 0.002	9.0
HyperGNN	0.566 ± 0.002	0.551 ± 0.004	0.532 ± 0.015	0.528 ± 0.013	0.571 ± 0.005	0.566 ± 0.005	0.696 ± 0.006	0.696 ± 0.006	0.658 ± 0.003	0.654 ± 0.002	9.5
HAT	0.543 ± 0.002	0.533 ± 0.003	0.548 ± 0.015	0.544 ± 0.017	0.588 ± 0.002	0.586 ± 0.002	0.691 ± 0.018	0.690 ± 0.019	0.676 ± 0.003	0.673 ± 0.003	6.8
UniGCNII	0.520 ± 0.001	0.507 ± 0.001	0.519 ± 0.019	0.509 ± 0.023	0.540 ± 0.004	0.537 ± 0.006	0.674 ± 0.018	0.671 ± 0.023	0.621 ± 0.004	0.617 ± 0.006	13.3
AllSet	0.577 ± 0.005	0.570 ± 0.002	0.523 ± 0.018	0.502 ± 0.016	0.585 ± 0.008	0.515 ± 0.013	0.686 ± 0.010	0.681 ± 0.009	0.679 ± 0.006	0.660 ± 0.010	10.1
HDS^{ode}	0.561 ± 0.003	0.552 ± 0.003	0.537 ± 0.009	0.529 ± 0.010	0.554 ± 0.004	0.548 ± 0.002	0.703 ± 0.008	0.703 ± 0.008	0.669 ± 0.004	0.664 ± 0.005	7.1
LEGCN	0.520 ± 0.002	0.511 ± 0.003	0.698 ± 0.008	0.689 ± 0.008	0.676 ± 0.016	0.675 ± 0.016	0.733 ± 0.015	0.731 ± 0.016	0.703 ± 0.002	0.698 ± 0.002	7.4
MultiSetMixer	0.593 ± 0.005	0.585 ± 0.005	0.542 ± 0.013	0.538 ± 0.011	0.561 ± 0.004	0.552 ± 0.003	0.706 ± 0.007	0.705 ± 0.007	0.668 ± 0.001	0.666 ± 0.001	5.6
HNN	0.543 ± 0.002	0.533 ± 0.002	0.522 ± 0.008	0.354 ± 0.008	0.527 ± 0.006	0.409 ± 0.083	0.527 ± 0.028	0.436 ± 0.094	0.673 ± 0.006	0.668 ± 0.006	11.9
ED-HNN	0.503 ± 0.006	0.479 ± 0.008	0.532 ± 0.011	0.511 ± 0.014	0.599 ± 0.002	0.559 ± 0.013	0.709 ± 0.007	0.709 ± 0.007	0.668 ± 0.008	0.656 ± 0.009	11.1
WHATsNet	0.632 ± 0.004	0.625 ± 0.006	0.526 ± 0.014	0.519 ± 0.014	0.587 ± 0.004	0.582 ± 0.008	0.711 ± 0.010	0.710 ± 0.009	0.677 ± 0.004	0.670 ± 0.004	4.9
CoNHD (UNB) (*ours*)	0.646 ± 0.003	0.640 ± 0.004	0.769 ± 0.028	0.767 ± 0.028	0.884 ± 0.011	0.883 ± 0.011	0.827 ± 0.013	**0.826 ± 0.013**	0.896 ± 0.003	0.895 ± 0.003	1.8
CoNHD (ISAB) (*ours*)	**0.650 ± 0.003**	**0.646 ± 0.004**	**0.800 ± 0.019**	**0.797 ± 0.020**	**0.903 ± 0.002**	**0.902 ± 0.002**	**0.828 ± 0.010**	**0.826 ± 0.010**	**0.899 ± 0.004**	**0.898 ± 0.004**	**1.1**

contain more noise [10] than benchmark datasets, we construct four new datasets (Cora-Outsider, DBLP-Outsider, Citeseer-Outsider, and Pubmed-Outsider) by transforming the outsider identification task [77] into the ENC task. In these datasets, we randomly replace half of the nodes in each edge with other nodes, and the task is to predict whether each node belongs to the corresponding edge.

Baselines. We compare CoNHD (with two diffusion operator implementations, UNB and ISAB) to ten baseline HGNN methods, including five models following the traditional message passing framework (HyperGNN [23], HAT [35], UniGCNII [34], AllSet [17], and HDS^{ode} [69]) and five models that utilize edge-dependent node information (LEGCN [70], MultiSetMixer [57], HNN [2], ED-HNN [63], and WHATsNet [19]). Since a hypergraph can also be viewed as a bipartite graph, we add three traditional GNN methods (GraphSAGE [30], GAT [62], and a graph diffusion-based method ADGN [29]) as our baselines. We follow the experiment setup in [19].

Effectiveness. As shown in Table 1, CoNHD achieves the best performance across all datasets in both Micro-F1 and Macro-F1. Notably, CoNHD shows substantial improvements on Email-Enron and Email-Eu, which have relatively large-degree nodes or edges. Baseline methods using single node or edge representations can cause information loss for large degree nodes or edges in these datasets. In contrast, the number of co-representations in CoNHD is adaptive to the node and edge degrees. Additionally, CoNHD achieves very significant improvements on four outsider identification datasets. This suggests that mixing features from noise outsiders into an entangled edge representation significantly degrades the performance. Our method, with the co-representation design, can differentiate features from normal nodes and outsiders, leading to superior results. On simpler datasets with very low node and edge degrees, such as Stack-Physics, the improvement is less pronounced. In these datasets, each hyperedge only contains a very limited number of nodes (about 2 on average, similar to normal graphs) and cannot fully demonstrate the ability of different HGNNs in modeling complex higher-order interactions. Nevertheless, our method still consistently achieves the best performance on these datasets with statistically significant improvement (p-value < 0.5) in most cases. The performance gap between the two diffusion operator implementations (UNB and ISAB) is minimal, while the transformer-based ISAB implementation overall demonstrates better performance. Unless otherwise specified, we adopt the better ISAB implementation for most subsequent experiments.

To quantify the diversity of ENC labels of the same node and explore its impact on model performance, we introduce a measure called node entropy. Specifically, for a node v, the node entropy $H(v)$ is defined as $H(v) := -\sum_{c\in C} p_v(c)\log p_v(c)$, where $p_v(c) = \sum_{e\in\mathcal{E}_v}\mathbb{I}(y_{v,e}=c)/d_v$. A higher node entropy value indicates that the node labels in different edges tend to be different. The edge entropy is defined similarly. We sort the entropy values in ascending order and divide them into ten equal-frequency levels. We calculate the model performance in each level. As shown in Fig. 3, CoNHD demonstrates advantages compared to message passing methods

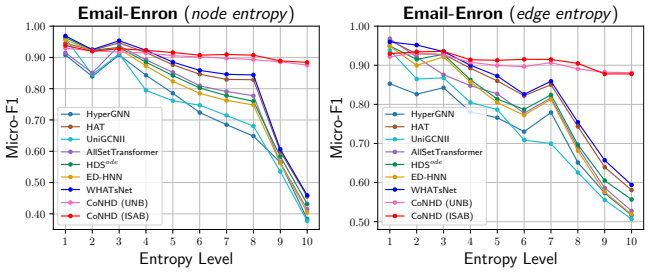


Figure 3: Comparison of performance under different node/edge entropy levels. As node/edge entropy increases, the performance of message passing methods drops significantly, whereas CoNHD still maintains high performance.

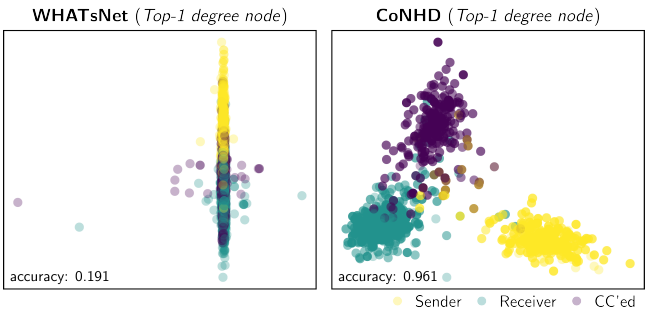


Figure 4: Visualization of embeddings of the top-1 degree node (user) within different hyperedges (emails) in Email-Enron using LDA. The embeddings learned by CoNHD exhibit clearer distinctions based on labels in different hyperedges compared to the embeddings learned by WHATsNet.

like WHATsNet as the entropy level increases. This suggests that using single node representations is insufficient to capture different node/edge-specific features for predicting different ENC labels.

To examine whether CoNHD learns separable embeddings for the same node across different edges, we follow [19] and use LDA to visualize embeddings of the largest-degree node in Email-Enron. Fig. 4 shows that CoNHD learns more separable embeddings than WHATsNet. Compared to WHATsNet, CoNHD provides adaptive representation sizes by introducing co-representations, which can avoid information loss for large-degree nodes.

Efficiency. The performance and training time on Email-Enron and Email-Eu are illustrated in Fig. 5, tested on a single NVIDIA A100 GPU. As full-batch training is impractical for large hypergraphs, we only compare models using mini-batch training. The best baseline WHATsNet trades efficiency for performance, while CoNHD not only achieves the best performance but also maintains high efficiency. Different from message passing, in CoNHD, the within-edge and within-node interactions are designed as parallel without inter-dependency, which is helpful for restricting inputs from only direct neighbors to the target node-edge pair and improving efficiency. Additionally, CoNHD eliminates the edge-dependent feature extraction and aggregation steps in edge-dependent message passing methods, further increasing efficiency.

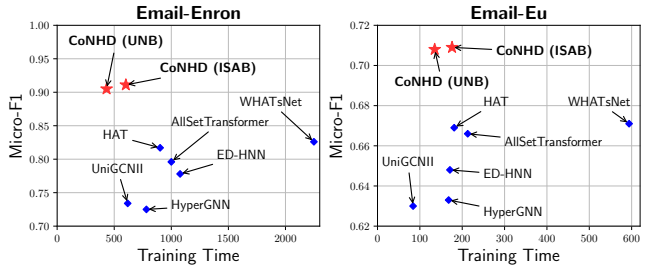


Figure 5: Comparison of the performance and training time (minutes). CoNHD demonstrates significant improvements in terms of Micro-F1 while maintaining good efficiency.

5.2 Performance of Constructing Deep HGNNs

Oversmoothing is a well-known issue in deep HGNNs [63, 69], which hinders the utilization of long-range information and limits performance. Diffusion-based HGNNs are shown to be more robust against this issue [12, 63]. To evaluate the performance of CoNHD in constructing deep HGNNs, we conduct experiments on ENC with varying numbers of HGNN layers. As deeper HGNNs significantly increase GPU memory usage, we experiment on Citeseer-Outsider, a relatively small dataset, to ensure computational feasibility.

As shown in Fig. 6, the performance of WHATsNet drops sharply beyond 4 layers, while two diffusion-based methods, EDHNN and HDS^{ode}, remain stable but show no notable gains with deeper architectures. In contrast, CoNHD continues to improve with more layers, and the performance converges after 16 layers. This suggests that CoNHD can effectively leverage long-range information to enhance performance. Compared to other diffusion-based HGNNs, the co-representation design in CoNHD allows the same node to have distinct representations when interacting within different hyperedges, ensuring the updated information to different node-edge pairs remains diverse and thereby preventing the collapse into oversmoothed representations.

5.3 Application to Downstream Tasks

The ENC task has been shown to be beneficial for many downstream applications [19]. Following [19], we evaluate whether the ENC labels predicted by CoNHD can improve three downstream tasks: Ranking Aggregation (Halo, AMiner), Clustering (DBLP, AMiner), and Product Return Prediction (Etail). The ENC labels are first predicted and then used as additional input features to enhance the downstream prediction performance. Since the improvements depend not only on model performance in ENC but also on the relevance between downstream tasks and ENC, we report both the ENC prediction results and the downstream task performance.

Consistent with results in Section 5.1, Table 2 shows that CoNHD consistently achieves superior performance on ENC label prediction across all datasets. Table 3 further demonstrates that using predicted ENC labels as additional information improves downstream task performance compared to cases where these labels are not used. CoNHD also outperforms WHATsNet across all three downstream tasks, benefiting from more accurate ENC label prediction. In the ranking aggregation task, using predicted labels even surpasses the cases using ground truth labels. This suggests that the ground

Table 2: Performance of predicting ENC labels on downstream datasets.

Method	Halo		AMiner		DBLP		Etail	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
WHATsNet	0.377 ± 0.002	0.352 ± 0.006	0.631 ± 0.027	0.561 ± 0.044	0.625 ± 0.092	0.553 ± 0.128	0.622 ± 0.004	0.461 ± 0.007
CoNHD (*ours*)	**0.396** ± 0.003	**0.381** ± 0.007	**0.661** ± 0.027	**0.605** ± 0.040	**0.768** ± 0.094	**0.740** ± 0.127	**0.751** ± 0.008	**0.696** ± 0.008

Table 3: Performance on downstream tasks using the predicted ENC labels.

(a) Ranking Aggregation (Acc.↑)

Method	Halo	AMiner
RW [18] w/o Labels	0.532 ± 0.000	0.654 ± 0.000
RW [18] w/ WHATsNet	0.714 ± 0.004	0.693 ± 0.001
RW [18] w/ CoNHD	**0.723** ± 0.003	**0.695** ± 0.001
RW [18] w/ GroundTruth	0.711 ± 0.000	0.675 ± 0.000

(b) Clustering (NMI↑)

Method	DBLP	AMiner
RDC-Spec [31] w/o Labels	0.163 ± 0.000	0.338 ± 0.000
RDC-Spec [31] w/ WHATsNet	0.184 ± 0.028	0.352 ± 0.034
RDC-Spec [31] w/ CoNHD	**0.196** ± 0.022	**0.354** ± 0.016
RDC-Spec [31] w/ GroundTruth	0.221 ± 0.000	0.359 ± 0.000

(c) Product Return Prediction (F1↑)

Method	Etail
HyperGO [43] w/o Labels	0.718 ± 0.000
HyperGO [43] w/ WHATsNet	0.723 ± 0.003
HyperGO [43] w/ CoNHD	**0.733** ± 0.004
HyperGO [43] w/ GroundTruth	0.738 ± 0.000

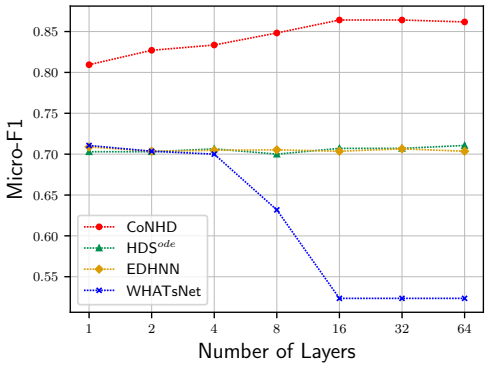


Figure 6: Performance of HGNNs with varying numbers of layers on the Citeseer-Outsider dataset. CoNHD achieves the best performance across all settings.

truth labels might contain noise, while the predicted labels better capture the underlying smooth structure of the label space and further enhance the downstream task performance.

5.4 Effectiveness of the Equivariant Design

CoNHD reformulates interactions as multi-output functions by introducing co-representations as shown in Fig. 2, thereby enabling the use of equivariant functions. To show the importance of the equivariant design, we apply a mean aggregation to the equivariant outputs, reducing ϕ and φ to invariant functions with identical outputs for different node-edge pairs. We conduct experiments on Email-Enron and Email-Eu.

As shown in Table 4, CoNHD with two equivariant operators achieves the best performance, significantly outperforming the variant with two invariant operators. Furthermore, variants with just one equivariant operator still outperform the fully invariant model, indicating that equivariance benefits both within-edge and within-node interactions. We also notice that the performance gap between the full equivariant model and the variant with only the equivariant within-edge operator ϕ is not significant. This might imply that within-edge interactions can provide the majority of the information needed for predicting the ENC labels in these datasets.

Table 4: Effectiveness of the equivariance in two diffusion operators ϕ and φ. We use ✓ and ✗ to denote whether the corresponding operator is equivariant or invariant, respectively. Shaded cells indicate the variants with equivariance significantly outperform the one with only invariant operators.

Method	ϕ	φ	Email-Enron		Email-Eu	
			Micro-F1	Macro-F1	Micro-F1	Macro-F1
CoNHD (UNB)	✗	✗	0.827 ± 0.000	0.769 ± 0.004	0.673 ± 0.000	0.645 ± 0.001
	✗	✓	0.876 ± 0.001	0.817 ± 0.006	0.698 ± 0.001	0.677 ± 0.002
	✓	✗	0.903 ± 0.001	0.855 ± 0.004	0.707 ± 0.000	0.688 ± 0.002
	✓	✓	**0.905** ± 0.001	**0.858** ± 0.004	**0.708** ± 0.001	**0.689** ± 0.001
CoNHD (ISAB)	✗	✗	0.829 ± 0.001	0.765 ± 0.007	0.673 ± 0.001	0.647 ± 0.002
	✗	✓	0.878 ± 0.001	0.823 ± 0.005	0.698 ± 0.001	0.678 ± 0.003
	✓	✗	0.910 ± 0.001	0.870 ± 0.003	0.707 ± 0.001	0.689 ± 0.001
	✓	✓	**0.911** ± 0.001	**0.871** ± 0.002	**0.709** ± 0.001	**0.690** ± 0.002

6 Conclusion

In this paper, we develop CoNHD, a novel diffusion-based HGNN for modeling edge-specific features in ENC. CoNHD reformulates within-edge and within-node interactions as multi-output equivariant diffusion processes among node-edge co-representations, which disentangles edge-specific features and provides adaptive representation sizes. Our experiments demonstrate that CoNHD achieves the best performance on ten benchmark ENC datasets and several downstream tasks without sacrificing efficiency. We further show the robustness of CoNHD against the oversmoothing issue and validate the effectiveness of the equivariant design. Future work could explore extending CoNHD to more complex scenarios, such as dynamic hypergraphs [75] and multi-modal hypergraphs [38], where existing approaches mostly rely on traditional message passing-based HGNNs [16, 55]. CoNHD has the potential to improve representation quality by modeling edge-specific features in these complex settings.

Acknowledgments

This work is supported by the AI4Intelligence project with file number KICH1.VE01.20.011, partly financed by the Dutch Research Council (Nederlandse Organisatie voor Wetenschappelijk Onderzoek, NWO).

GenAI Usage Disclosure

During the writing process, we employed ChatGPT solely for language refinement and grammatical corrections. All technical concepts, experimental work, and analytical content were independently conducted and written by the authors without relying on generative AI for idea generation or content creation.

References

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

[2] 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:2212.14077* (2022).

[3] Devanshu Arya, Deepak K Gupta, Stevan Rudinac, and Marcel Worring. 2020. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv:2010.04558* (2020).

[4] Devanshu Arya, Deepak K Gupta, Stevan Rudinac, and Marcel Worring. 2024. Adaptive neural message passing for inductive learning on hypergraphs. *IEEE Trans. Pattern Anal. Mach. Intell.* (2024).

[5] Song Bai, Feihu Zhang, and Philip HS Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognit.* (2021).

[6] 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. *Phys. Rep.* (2020).

[7] Ali Behrouz, Farnoosh Hashemi, Sadaf Sadeghian, and Margo Seltzer. 2023. CAT-walk: Inductive hypergraph learning via set walks. In *NeurIPS*.

[8] Tatyana Benko, Martin Buck, Ilya Amburg, Stephen J Young, and Sinan G Aksoy. 2024. HyperMagNet: A Magnetic Laplacian based Hypergraph Neural Network. *arXiv:2402.09676* (2024).

[9] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* (2011).

[10] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. 2022. Hypergraph Structure Learning for Hypergraph Neural Networks. In *IJCAI*.

[11] Yuan Cao, Lei Li, Xiangru Chen, Xue Xu, Zuojin Huang, and Yanwei Yu. 2024. Hypergraph Hash Learning for Efficient Trajectory Similarity Computation. In *CIKM*.

[12] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. 2021. Grand: Graph neural diffusion. In *ICML*.

[13] Can Chen, Chen Liao, and Yang-Yu Liu. 2023. Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning. *Nat. Commun.* (2023).

[14] Yin Chen, Xiaoyang Wang, and Chen Chen. 2024. Hyperedge Importance Estimation via Identity-aware Hypergraph Attention Network. In *CIKM*.

[15] Zirui Chen, Xin Wang, Chenxu Wang, and Jianxin Li. 2022. Explainable link prediction in knowledge hypergraphs. In *CIKM*.

[16] Zhangtao Cheng, Jienan Zhang, Xovee Xu, Goce Trajcevski, Ting Zhong, and Fan Zhou. 2024. Retrieval-augmented hypergraph for multimodal social media popularity prediction. In *KDD*.

[17] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. 2022. You are AllSet: A Multiset Function Framework for Hypergraph Neural Networks. In *ICLR*.

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

[19] Minyoung Choe, Sunwoo Kim, Jaemin Yoo, and Kijung Shin. 2023. Classification of Edge-dependent Labels of Nodes in Hypergraphs. In *KDD*.

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

[21] Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Lio. 2023. Sheaf Hypergraph Networks. In *NeurIPS*.

[22] Barakeel Fanseu Kamhoua, Lin Zhang, Kaili Ma, James Cheng, Bo Li, and Bo Han. 2021. Hypergraph convolution based attributed hypergraph clustering. In *CIKM*.

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

[24] Kimon Fountoulakis, Pan Li, and Shenghao Yang. 2021. Local hyper-flow diffusion. In *NeurIPS*.

[25] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. 2022. HGNN+: General hypergraph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* (2022).

[26] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. 2020. Hypergraph learning: Methods and practices. *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).

[27] David F Gleich and Michael W Mahoney. 2015. Using local spectral methods to robustify graph-based learning algorithms. In *KDD*.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[29] Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. 2023. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *ICLR*.

[30] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[31] Koby Hayashi, Sinan G Aksoy, Cheong Hee Park, and Haesun Park. 2020. Hypergraph random walks, laplacians, and clustering. In *CIKM*.

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

[33] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. 2013. The total variation on hypergraphs-learning on hypergraphs revisited. In *NeurIPS*.

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

[35] Hyunjin Hwang, Seungwoo Lee, and Kijung Shin. 2021. HyFER: A Framework for Making Hypergraph Learning Easy, Scalable and Benchmarkable. In *WWW Workshop on Graph Learning Benchmarks*.

[36] Stefanie Jegelka, Francis Bach, and Suvrit Sra. 2013. Reflection methods for user-friendly submodular optimization. In *NeurIPS*.

[37] Jaehyeong Jo, Jinheon Baek, Seul Lee, Dongki Kim, Minki Kang, and Sung Ju Hwang. 2021. Edge representation learning with hypergraphs. In *NeurIPS*.

[38] Eun-Sol Kim, Woo Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. 2020. Hypergraph attention networks for multimodal learning. In *CVPR*.

[39] 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*.

[40] Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. 2024. A survey on hypergraph neural networks: an in-depth and step-by-step guide. In *KDD*.

[41] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[42] Fuyang Li, Jiying Zhang, Xi Xiao, Dijun Luo, et al. 2022. A Simple Hypergraph Kernel Convolution based on Discounted Markov Diffusion Process. In *NeurIPS Workshop on New Frontiers in Graph Learning*.

[43] Jianbo Li, Jingrui He, and Yada Zhu. 2018. E-tail product return prediction via hypergraph-based local graph cut. In *KDD*.

[44] Pan Li, Niao He, and Olgica Milenkovic. 2020. Quadratic decomposable submodular function minimization: Theory and practice. *J. Mach. Learn. Res.* (2020).

[45] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous hypergraph clustering with applications. In *NeurIPS*.

[46] Meng Liu, Nate Veldt, Haoyu Song, Pan Li, and David F Gleich. 2021. Strongly local hypergraph diffusions for clustering and semi-supervised learning. In *WWW*.

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

[48] Gongxu Luo, Jianxin Li, Hao Peng, Carl Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021. Graph Entropy Guided Node Embedding Dimension Selection for Graph Neural Networks. In *IJCAI*.

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

[50] Khaled Mohammed Saifuddin, Corey May, Farhan Tanvir, Muhammad Ifte Khairul Islam, and Esra Akbas. 2023. Seq-hygan: Sequence classification via hypergraph attention network. In *CIKM*.

[51] Michael T Schaub, Yu Zhu, Jean-Baptiste Seby, T Mitchell Roddenberry, and Santiago Segarra. 2021. Signal processing on higher-order networks: Livin'on the edge... and beyond. *Signal Process.* (2021).

[52] Nimrod Segol and Yaron Lipman. 2020. On Universal Equivariant Set Networks. In *ICLR*.

[53] Zhiyao Shu, Xiangguo Sun, and Hong Cheng. 2024. When llm meets hypergraph: A sociological analysis on personality via online social networks. In *CIKM*.

[54] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Qing Meng, Wang Han, and Jiuxin Cao. 2021. Multi-level hyperedge distillation for social linking prediction on sparsely observed networks. In *WWW*.

[55] Xiangguo Sun, Hongzhi Yin, Bo Liu, Qing Meng, Jiuxin Cao, Alexander Zhou, and Hongxu Chen. 2022. Structure learning via meta-hyperedge for dynamic rumor detection. *IEEE Trans. Knowl. Data Eng.* (2022).

[56] Yuuki Takai, Atsushi Miyauchi, Masahiro Ikeda, and Yuichi Yoshida. 2020. Hypergraph clustering based on pagerank. In *KDD*.

[57] 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:2310.07684* (2023).

[58] Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. 2022. GRAND++: Graph neural diffusion with a source term. In *ICLR*.

[59] Francesco Tudisco, Austin R Benson, and Konstantin Prokopchik. 2021. Nonlinear higher-order label spreading. In *WWW*.
[60] Francesco Tudisco, Konstantin Prokopchik, and Austin R Benson. 2021. A nonlinear diffusion method for semi-supervised learning on hypergraphs. *arXiv:2103.14867* (2021).
[61] Nate Veldt, Austin R Benson, and Jon Kleinberg. 2023. Augmented sparsifiers for generalized hypergraph cuts. *J. Mach. Learn. Res.* (2023).
[62] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[63] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. 2023. Equivariant Hypergraph Diffusion Neural Operators. In *ICLR*.
[64] Yuxin Wang, Quan Gan, Xipeng Qiu, Xuanjing Huang, and David Wipf. 2023. From hypergraph energy functions to hypergraph neural networks. In *ICML*.
[65] Chunyu Wei, Jian Liang, Bing Bai, and Di Liu. 2022. Dynamic hypergraph learning for collaborative filtering. In *CIKM*.
[66] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *ACM Comput. Surv.* (2022).
[67] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* (2020).
[68] 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*.
[69] Jielong Yan, Yifan Feng, Shihui Ying, and Yue Gao. 2024. Hypergraph dynamic system. In *ICLR*.
[70] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. 2022. Semi-supervised hypergraph node classification on hypergraph line expansion. In *CIKM*.
[71] Guang Yang, Meiqi Tu, Zelong Li, Jinquan Hang, Taichi Liu, Ruofeng Liu, Yi Ding, Yu Yang, and Desheng Zhang. 2024. Behavior-Aware Hypergraph Convolutional Network for Illegal Parking Prediction with Multi-Source Contextual Information. In *CIKM*.
[72] Mingdai Yang, Zhiwei Liu, Liangwei Yang, Xiaolong Liu, Chen Wang, Hao Peng, and Philip S Yu. 2023. Group identification via transitional hypergraph convolution with cross-view self-supervised learning. In *CIKM*.
[73] Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. 2021. Graph neural networks inspired by classical iterative algorithms. In *ICML*.
[74] Özgür Yeniay. 2005. Penalty function methods for constrained optimization with genetic algorithms. *Math. Comput. Appl.* (2005).
[75] Nan Yin, Fuli Feng, Zhigang Luo, Xiang Zhang, Wenjie Wang, Xiao Luo, Chong Chen, and Xian-Sheng Hua. 2022. Dynamic hypergraph convolutional network. In *ICDE*.
[76] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and TH Hubert Chan. 2017. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *ICML*.
[77] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2020. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *ICLR*.
[78] Songyang Zhang, Zhi Ding, and Shuguang Cui. 2019. Introducing hypergraph signal processing: Theoretical foundation and practical applications. *IEEE Internet Things J.* (2019).
[79] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *NeurIPS*.
[80] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2007. Learning with hypergraphs: Clustering, classification, and embedding. In *NeurIPS*.
[81] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.
[82] 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 Recognit.* (2024).