RESEARCH-ARTICLE

Consistency and Invariance Guided Multi-View Hypergraph Learning for Robust Hyperedge Prediction

Accepted: 24 December 2025
Revised: 25 June 2025
Received: 22 January 2025

Consistency and Invariance Guided Multi-View Hypergraph Learning for Robust Hyperedge Prediction

CHANGYUAN TIAN, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, China
LI JIN*, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, China
ZEQUN ZHANG*, Aerospace Information Research Institute, Chinese Academy of Sciences, China
ZHICONG LU, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, China
WEN SHI, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, China
JIANHUA YIN, Shandong University, China
SHIYAO YAN, University of the Chinese Academy of Sciences, China
ZHI GUO, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, China

Hypergraphs, by extending traditional graphs with hyperedges, enable the modeling and prediction of complex higher-order interactions that go beyond simple pairwise interactions. Hyperedge prediction, an evolution of link prediction, aims to identify potential higher-order interactions—such as those in social media group chats—by recognizing and predicting hyperedges. Recently, hypergraph neural networks (HGNNs) have advanced hyperedge prediction by structuring higher-order interactions into a hypergraph, enabling effective capture of higher-order relations through information propagation across the hypergraph. However, existing methods primarily focus on developing complex HGNNs, underestimating the

*Corresponding author

Authors' Contact Information: Changyuan Tian, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China, tianchangyuan21@mails.ucas.edu.cn; Li Jin, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, Beijing, China, jinlimails@gmail.com; Zequn Zhang, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing, China, zqzhang1@mail.ie.ac.cn; Zhicong Lu, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China, nazaritelzc@gmails.com; Wen Shi, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, Beijing, China, shiwen@aircas.ac.cn; Jianhua Yin, Shandong University, Qingdao, China, jhyin@sdu.edu.cn; Shiyao Yan, University of the Chinese Academy of Sciences, Beijing, China, yanshiyao19@mails.ucas.ac.cn; Zhi Guo, Aerospace Information Research Institute, Chinese Academy of Sciences, Key Laboratory of Target Cognition and Application Technology, Beijing, China, guozhi@mail.ie.ac.cn.

ACM 1556-472X/2026/1-ART
https://doi.org/10.1145/3787102

inherent unreliability of the underlying hypergraph due to incompleteness and noise, leading to suboptimal and fragile performance. In this paper, we propose Multi-HyperLinker, a novel multi-view hypergraph learning framework that leverages the consistency and invariance across multiple views to capture reliable higher-order interaction patterns from historical observational data for robust hyperedge prediction. Specifically, to facilitate effective information propagation on incomplete hypergraphs, Multi-HyperLinker first synthesizes a tightly structured hypergraph and designs a consistency-guided dual-view learning strategy. To capture reliable higher-order interaction patterns on noisy hypergraphs, Multi-HyperLinker augments the hypergraphs by perturbing hyperedges to simulate variations and noise, and introduces an invariant learning strategy. Extensive experiments conducted on four real-world datasets demonstrate the superiority of Multi-HyperLinker, achieving performance improvements of up to 19.80% in hit rate compared to existing HGNN-based methods. Additionally, it exhibits enhanced robustness on incomplete and noisy hypergraphs.

CCS Concepts: • **Information systems → Data mining**; • **Computing methodologies → Supervised learning**; **Knowledge representation and reasoning**.

Additional Key Words and Phrases: Hyperedge Prediction, Hypergraph Learning, Neural Network

1 Introduction

Graphs are powerful tools for modeling complex real-world systems, where edges represent pairwise interactions between nodes. The task of link prediction within these graphs has been extensively researched, focusing on predicting the existence of interactions between node pairs. However, higher-order interactions are also prevalent in natural systems. Examples include co-authorship among multiple authors [13], group communications or shopping involving three or more individuals [14], and chemical reactions involving multiple substances [39]. To address these complexities, hypergraphs extend traditional graphs by introducing hyperedges. Hyperedges are sets that can connect more than two nodes, thereby capturing more intricate higher-order interactions. Developing effective methods for hyperedge prediction is crucial for identifying potential and future higher-order interactions, moving beyond the limitations of predicting simple pairwise interactions.

The core of hyperedge prediction lies in capturing reliable higher-order interaction patterns from observed set-structured higher-order interactions, requiring models that handle permutation-invariant set inputs and assign labels for entire sets. Traditional models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are order-sensitive and thus unsuitable for this task. A straightforward approach employs set neural networks [24, 42] that process each node in a set through a Multilayer Perceptron (MLP) or Graph Neural Network (GNN) [39, 44], treating each set as a fully connected graph. This is followed by pooling operations, such as mean, sum, or max, to generate a set representation vector that can be used by a subsequent classifier. However, these methods fall short in modeling inter-set interactions due to the assumption that sets are independent. Hypergraph neural networks (HGNNs) [1, 13] address this limitation by structuring sets into a hypergraph, enabling both intra-set and inter-set interactions through a message-passing mechanism. For example, AHP [18] utilizes HGNNs to generate node embeddings that effectively capture higher-order relations. These embeddings are pooled to obtain a set representation, which is then fed into an MLP to label the entire set. Due to their ability to model higher-order interaction patterns, HGNNs have emerged as the leading solution for hyperedge prediction, inspiring a range of HGNN-based models that offer significantly improved performance [6, 11, 18, 21].

Despite advancements achieved through developing complex HGNNs for hyperedge prediction, these methods remain constrained by the inherent unreliability of the underlying hypergraph due to incompleteness and noise. Figure 1a illustrates an example of an unreliable hypergraph, characterized by its incomplete and noisy hyperedges. This results in weak connectivity, with fragmented structures such as small disconnected components, isolated hyperedges, and solitary nodes. In such a scenario, HGNNs struggle with node-to-node information propagation (e.g., A-C and B-C), failing to capture the similarity among nodes A, B, and C, which leads to false negative predictions (e.g., {BDC} {ABC}). Additionally, the reliability of the raw hypergraph is compromised by noise hyperedges [5, 34, 45], including erroneous higher-order interactions due to monitoring or recording

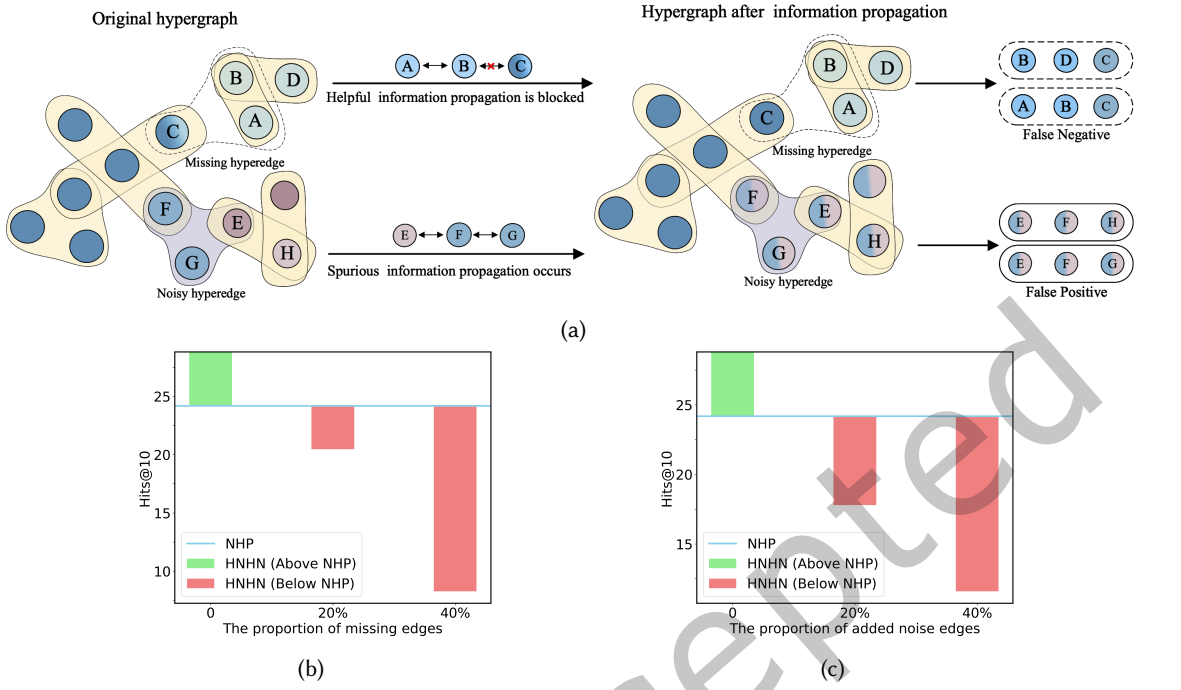


Fig. 1. (a): An unreliable hypergraph characterized by incompleteness and noise can result in the failure of information propagation by hypergraph neural networks. In the figure, the color of the nodes indicates their degree of similarity; nodes with more similar colors are more likely to form hyperedges. (b): Exacerbated incompleteness leads to the hypergraph-based approach (HNHN) lagging behind the set-based method (NHP). (c): Increased noise also contributes to HNHN lagging behind NHP.

errors. This noise can mislead HGNNs into conducting spurious information propagation, such as between nodes E-F and E-G, resulting in incorrect similarity capture among nodes E, F, and G, and thus, false positive predictions (e.g., {EFH} {EFG}). Figures 1b and 1c further demonstrate the failure of non-robust hypergraph-based methods as incompleteness and noise levels increase, resulting in their performance lagging behind set-based methods. Ultimately, incomplete observations and inescapable noise render hypergraphs unreliable. Therefore, to further advance hyperedge prediction, it is essential to shift our focus from solely developing complex HGNNs to enhancing the reliability of the hypergraphs they rely on.

In this paper, we propose Multi-HyperLinker, a novel multi-view hypergraph learning framework that leverages the consistency and invariance across multiple views to capture reliable higher-order interaction patterns for robust hyperedge prediction. Specifically, to facilitate effective information propagation on incomplete hypergraphs, we generate a synthetic hypergraph with a tighter structure using heuristic hypergraph generation method, repairing the raw hypergraph's fragmented structure. Dual-view learning is then proposed, enabling joint information propagation across both raw and synthetic hypergraphs to capture the inherent consistency within them. Furthermore, to extract reliable higher-order interaction patterns from noisy hypergraphs, we create multiple augmented hypergraphs via hyperedges perturbation strategy. Invariant learning is introduced to guide the model in learning strong, non-spurious higher-order relations by optimizing it for invariant and optimal performance across multiple augmented hypergraphs. We evaluate Multi-HyperLinker on four real-world datasets

from diverse domains, including social networks, academic collaboration networks, drug compositions. Extensive experiments conducted on four real-world datasets demonstrate the superiority of Multi-HyperLinker, with performance improvements of up to 19.80% in hit rate compared to existing HGNN-based methods. Additionally, it exhibits enhanced robustness on incomplete and noisy hypergraphs.

Our contributions in this work are summarized as follows:

- We introduce Multi-HyperLinker, a novel multi-hypergraph learning framework that leverages the consistency and invariance across multiple views to model reliable higher-order interaction patterns for robust hyperedge prediction, mitigating the negative impact of incomplete and noisy underlying hypergraphs.
- To address hypergraph incompleteness, we synthesize a hypergraph using hypergraph generation techniques to enforce structure and design dual-view learning to capture integrated higher-order interaction patterns by jointly propagating information on both the raw and synthetic hypergraphs.
- To reduce the impact of noise, we generate multiple augmented hypergraphs through hyperedge perturbation to simulate variations and noise, and utilize invariant learning to identify strong, non-spurious higher-order relations by ensuring the model performs optimally across these augmented hypergraphs simultaneously.
- Extensive experiments conducted on four real-world datasets demonstrate the superiority of Multi-HyperLinker, with performance improvements of up to 19.80% in hit rate compared to existing HGNN-based methods. Additionally, it exhibits enhanced robustness on incomplete and noisy hypergraphs.

2 Related Works

2.1 Hypergraph Neural Network

Hypergraph neural network (HGNN) has gained significant attention as a promising technique for modeling various higher-order interactions [3, 8–10, 12, 26–28, 31, 36]. HGNN aims to generate embeddings for nodes or hyperedges on a hypergraph while preserving structural information inherent in the hypergraph. For instance, Bai et al. [3] introduced hypergraph convolution and attention operators, enabling more effective representation learning on hypergraph. Chen et al.[9] introduced the AllSet framework, a novel HGNN that leverages compositions of multiset functions, demonstrating superior performance in hypergraph node classification compared to existing methods. Meanwhile, HGNN is also being applied in various fields like social networks [32, 40], image understanding [15, 33], and recommendation systems [37, 38] to capture complex interaction patterns effectively. For instance, Yang et al. [40] used hypergraphs to model location-based social networks and applied hypergraph learning for node embedding. Additionally, Han et al. [15] introduced the vision hypergraph neural network to capture complex interactions between image patches. In the field of recommendation systems, a series of works [37, 38] uses hypergraphs to model higher-order interactions between users and items, enhancing the performance of recommendation systems. Recently, contrastive learning (CL) has been widely adopted for hypergraph representation learning, with [35] exploring fabricated and generative augmentations, and [22] proposing tri-directional contrasts between nodes, groups, and their memberships to capture microscopic and mesoscopic structures. While these CL methods focus on discriminating between positive and negative pairs, our work leverages invariant learning to extract stable features across augmented environments, thereby enhancing robustness against spurious correlations. Despite notable advancements in HGNNs, most of these efforts have been centered around node classification tasks and downstream applications. There remains a significant gap in exploring hypergraph-based methods specifically tailored for hyperedge prediction task.

Table 1. Main Notations.

Notations	Descriptions
$\mathcal{G}$	Hypergraph
$\mathbf{H}$	Incidence matrix
$\mathcal{V}$	Node set
v	Node
$\mathcal{E}$	Observed hyperedge set
e	Hyperedge
$\mathbf{X}$	Node feature
$\mathcal{G}^{syn}$	Tight hypergraph
$\mathcal{G}^{aug}$	Augmented Hypergraphs via Hyperedge Perturbation
$\mathcal{X}$	Feature space
$\mathcal{H}$	Latent embedding space
$\mathcal{Y}$	Label space
f_ω	Hypergraph encoder
f_ϕ	Decoder
$\mathbf{X}_V$	Node representation
$\mathbf{X}_E$	Hyperedge representation
$\bar{\mathcal{E}}$	Negative hyperedge set generated by negative sampling

2.2 Hyperedge prediction

The common occurrence of higher-order interactions has drawn research interest in hyperedge prediction, resulting in the proposal of several methods in this field [6, 7, 18, 21, 39, 42–44]. We categorize these methods into set-based and hypergraph-based approaches based on their consideration of inter-set interactions. Set-based methods [39, 42, 44] typically treat hyperedges as independent sets, without explicitly addressing these interactions, and utilize set representation learning to embed the sets. In contrast, hypergraph-based methods [6, 18, 21] construct hyperedges into hypergraphs, employing HGNN to explicitly account for both intra-set and inter-set interactions. Early on, researchers focused on designing more powerful n-tuple-wise similarity functions, leading to the development of numerous set-based methods. NHP [39], a representative set-based method, expands each hyperedge into a fully connected graph and utilizes graph convolutional networks to encode this expanded graph, thereby improving the performance of hyperedge prediction. Later, driven by the development of HGNNs, researchers began considering encoding hypergraph structural information to enhance hyperedge prediction. Hwang et al. [18] employ hypergraph neural networks to encode hypergraph structural information while incorporating adversarial learning to implement learnable negative sampling. Ko et al. [21] propose CASH, a novel hyperedge prediction framework to addresses challenges in accurately aggregating nodes within hyperedges and mitigating data sparsity by employing context-aware node aggregation and self-supervised contrastive learning, outperforming existing methods. Despite advancements achieved through developing complex HGNNs for hyperedge prediction, these methods remain constrained by the inherent unreliability of the underlying hypergraph due to incompleteness and noise. These issues warrant further exploration.

3 Preliminary

3.1 Hypergraph

A hypergraph extends the concept of a traditional graph by allowing hyperedges to connect multiple nodes, making it suitable for modeling higher-order higher-order relationships beyond pairwise connections. Table 1 summarizes the key notations adopted in this paper. We formally denote a hypergraph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Theta)$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of hyperedges. Each hyperedge $e \in \mathcal{E}$ is a subset of nodes, such as $e = \{v_1, v_2, \ldots, v_k\}$. The matrix $\Theta \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is diagonal and typically has elements set to 1, representing hyperedge weights. Node features are represented by the matrix $X \in \mathbb{R}^{|\mathcal{V}|\times d}$, where $|\mathcal{V}|$ is the number of nodes in the hypergraph and d is the dimensionality of each node's feature vector. The incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$ captures the hypergraph's topology, with each row indicating a node's connections to hyperedges, and each column indicating a hyperedge's connections to nodes. Formally, for a node $v \in \mathcal{V}$ and a hyperedge $e \in \mathcal{E}$:

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

Consider a social media platform where group chats are modeled as a hypergraph. Each user represents a node in the set $\mathcal{V}$, and each group chat forms a hyperedge in the set $\mathcal{E}$, connecting all users participating in that chat. The incidence matrix $\mathbf{H}$ captures this structure, with entries indicating whether a user is part of a specific group chat.

3.2 Problem statement

Hyperedge prediction aims to predict missing or future hyperedges, and can be formalized as a binary classification problem. The hyperedge prediction model is trained on a given set of observed positive hyperedges $\mathcal{E}$, and a collection of negative hyperedges $\bar{\mathcal{E}}$ generated through negative sampling. For a candidate hyperedge e', the hyperedge prediction model assesses the probability of e' being a positive hyperedge. Formally, using f_ω to represent the encoder and f_ϕ to represent the decoder, this problem can be formulated as:

Input: Observed hypergraph $\mathcal{G}$, and a candidate hyperedge e'.

Output: Probability that the candidate hyperedge e' is a positive hyperedge, i.e., $s_{e'} = f_\phi(f_\omega(\mathcal{G}), e')$.

The goal of hyperedge prediction task is to find an encoder-decoder pair capable of assigning higher probabilities to positive hyperedges and lower probabilities to negative hyperedges.

4 Method

4.1 Overview

Multi-HyperLinker is a novel framework for hyperedge prediction that uses multiple hypergraphs to model higher-order interactions. It employs dual-view learning and invariant learning to integrate multiple hypergraphs, addressing issues of incompleteness and noise in the raw hypergraph, thus enhancing hyperedge prediction performance. Figure 2 illustrates the overall framework of Multi-HyperLinker.

4.2 Encoder-Decoder Architecture for Hyperedge Prediction

HGNN as a Hypergraph Encoder. HGNNs are a representation learning method operating on hypergraphs, effectively encoding the topological structure and node feature of hypergraphs. In this study, we employ a HGNN as the hypergraph encoder. Specifically, the hypergraph encoder, denoted as $f_\omega : \mathcal{X} \to \mathcal{H}$, maps the hypergraph $\mathcal{G}$ and its associated feature space $\mathcal{X}$ to a latent embedding space $\mathcal{H}$. By default, we instantiate the hypergraph encoder as a HNHN [11] network. This classic HGNN features two message passing steps in each

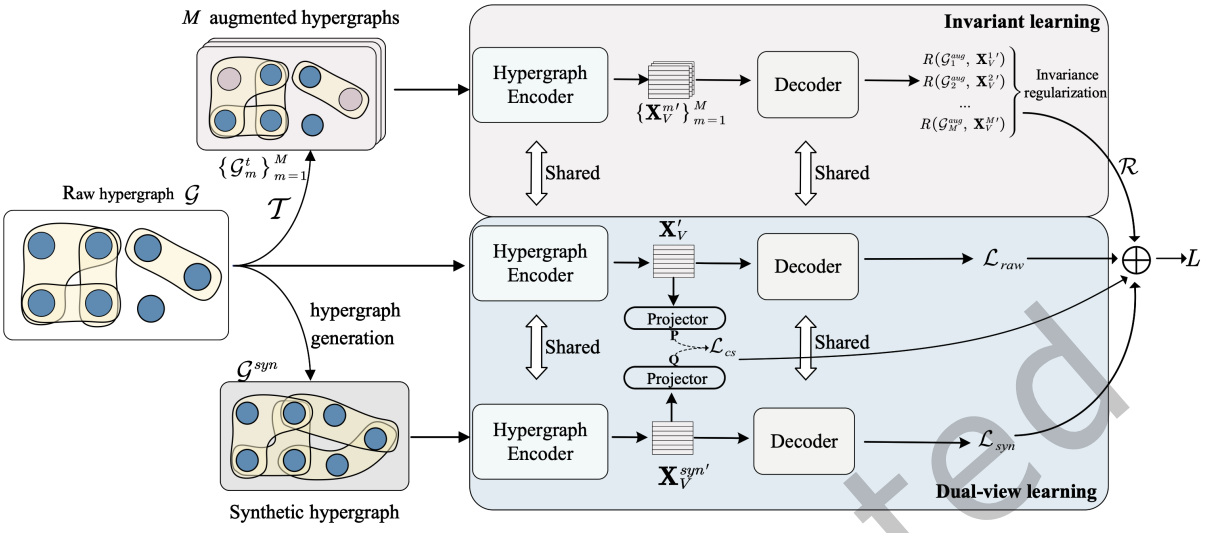


Fig. 2. The overall framework of Multi-HyperLinker. Multi-HyperLinker consists of dual-view learning and invariant learning. Dual-view learning encodes both the raw hypergraph $\mathcal{G}$ and the tightly structured synthetic hypergraph $\mathcal{G}^{syn}$ using a dual-branch network with shared weights, effectively integrating beneficial information from both. Invariant learning requires that Multi-HyperLinker simultaneously maintains optimal performance across multiple augmented hypergraphs. This ensures that the learned patterns remain invariant despite minor changes to the hypergraph and are generalizable during inference.

layer: node-to-hyperedge and hyperedge-to-node. Formally, the node-to-hyperedge step is defined as:

$$\mathbf{X}'_E = \sigma(\mathbf{H}^T \mathbf{X}_V \mathbf{W}_E + \mathbf{b}_E) \tag{2}$$

where $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ represents the incidence matrix of the hypergraph, $\mathbf{X}_V \in \mathbb{R}^{|\mathcal{V}|\times d}$ represents the node representation, $\mathbf{W}_E \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_E \in \mathbb{R}^d$ denote learnable parameters, and $\mathbf{X}'_E \in \mathbb{R}^{|\mathcal{E}|\times d}$ represents the updated hyperedge representation. $\sigma(\cdot)$ represents the non-linear activation function, typically the ReLU function by default. Following that, the hyperedge-to-node step is defined as:

$$\mathbf{X}'_V = \sigma(\mathbf{H}\mathbf{X}'_E \mathbf{W}_V + \mathbf{b}_V) \tag{3}$$

where $\mathbf{X}'_V \in \mathbb{R}^{|\mathcal{V}|\times d}$ represents the updated node representation, $\mathbf{W}_V \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_V \in \mathbb{R}^d$ denote learnable parameters.

Along the hypergraph, information propagates between nodes and hyperedges, updating node representations at each layer. By stacking multiple layers, nodes can interact with more distant nodes within the hypergraph.

For brevity, the process of encoding the hypergraph $\mathcal{G}$ by the encoder $f_\omega(\cdot)$ can be formalized as follows:

$$\mathbf{X}'_V = f_\omega(\mathcal{G}, \mathbf{X}_V) \tag{4}$$

Hyperedge Decoder: Pooling and Scoring. The decoder aims to assign labels to candidate hyperedges. It first applies pooling techniques to derive hyperedge representations from the node representations learned by the hypergraph encoder. Then, it uses a scorer to estimate the probability of a hyperedge being real. We denote the decoder as $f_\phi : \mathcal{H} \to \mathcal{Y}$, where $\mathcal{Y}$ denotes the label space.

Building upon the previous approach [18, 39], in order to effectively capture the divergence between the nodes within the hyperedge, the pooling operation is implemented as MaxMin pooling. Formally,

$$\mathbf{x}_{e'} = \max\min(\{\mathbf{x}_v\}_{v\in e'}) = [\max\{\mathbf{x}_v^j\}_{v\in e'} - \min\{\mathbf{x}_v^j\}_{v\in e'}]_{j=0}^{d} \tag{5}$$

where d represents the dimension of node representations, $\mathbf{x}_v$ denotes the representation of node v, and $\mathbf{x}_v^j$ indicates the j-th dimension of $\mathbf{x}_v$. e' represents the candidate hyperedge, and $\mathbf{x}_{e'}$ stands for the representation of candidate hyperedge e'.

The scorer is implemented using a two-layer MLP and can be formalized as:

$$s_{e'} = \text{sigmoid}(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_e + \mathbf{b}_1) + \mathbf{b}_2) \tag{6}$$

where $\sigma(\cdot)$ represents the the ReLU function, and $s_{e'}$ denotes the probability that the candidate hyperedge e' is the true hyperedge.

For brevity, the process of decoding can be formalized as follows:

$$s_{e'} = f_\phi(\mathbf{X}'_V, e') \tag{7}$$

4.3 Constructing Multiple Hypergraphs

In hyperedge prediction, the raw hypergraph processed by the HGNN is often incomplete and noisy, limiting its ability to encode reliable higher-order interaction patterns. To address this challenge, we introduce multiple complementary hypergraphs. This approach enables the integration and refinement of information from diverse structures, unlocking the HGNN's potential.

Synthetic Hypergraph. The incompleteness of the raw hypergraph creates a fragmented structure that impedes information propagation in HGNN. Inspired by hypergraph generation techniques, we aim to synthesize an auxiliary hypergraph that embodies the characteristics of a complete hypergraph. This approach focuses on creating a tight structure that facilitates comprehensive interactions among nodes. While complex hypergraph generation techniques, such as deep generative networks, are available, we opt for a simpler approach by applying the K-Nearest Neighbors (KNN) method on pre-defined node representations. This straightforward technique has been effectively used in previous studies [13, 15, 19]. This method involves identifying $k-1$ nearest neighbors for each node to form a hyperedge, thereby constructing a tight hypergraph. The tight hypergraph is denoted as $\mathcal{G}^{syn}(\mathcal{V}, \mathcal{E}')$. Formally, for each node $v_i \in \mathcal{V}$, we define:

$$NN(v_i) = \{v_i, v_{i_1}, v_{i_2}, \dots, v_{i_{k-1}}\} \tag{8}$$

where $NN(v_i)$ is the set containing v_i and its $k-1$ nearest neighbors. The set of hyperedges $\mathcal{E}'$ is then given by:

$$\mathcal{E}' = \{NN(v_i) \mid v_i \in \mathcal{V}\} \tag{9}$$

here, $\mathcal{E}'$ represents the hyperedges of the tight hypergraph.

Both the synthetic hypergraph with a tight structure and the raw hypergraph are jointly encoded using dual-view learning, as introduced in Section 4.4.

Hypergraph Augmentations. Relying solely on a single raw hypergraph in an HGNN can lead to overfitting to noise. To mitigate this, we reduce dependency on the raw hypergraph by utilizing multiple hypergraph variants through hypergraph augmentation methods. Specifically, as illustrated in Figure 3, we perturb hyperedges through three strategies: removing original nodes from the hyperedge, introducing irrelevant nodes, and substituting genuine nodes with random ones. These perturbation methods effectively simulate the characteristics of noisy

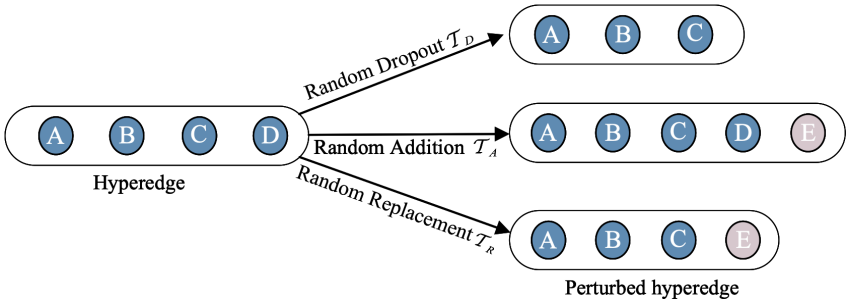


Fig. 3. The diagram of hyperedge perturbation process $\mathcal{T}$

hyperedges commonly encountered in real-world scenarios. Formally, the augmentation process is denoted as follows:

$$\mathcal{G}^{aug} = \mathcal{T}(\mathcal{G}, r) \tag{10}$$

where $\mathcal{G}$ represents the raw hypergraph, r can signify the dropout ratio, addition ratio, or replacement ratio for a hyperedge. $\mathcal{T}(\cdot)$ denotes the augmentation process, which can be instantiated as $\mathcal{T}_D, \mathcal{T}_A, \mathcal{T}_R$; and $\mathcal{G}^{aug}$ represents the generated hypergraph. Through the hypergraph augmentation process, we can obtain M randomly generated hypergraphs $\{\mathcal{G}_m^{aug}\}_{m=1}^{M}$.

The diverse hypergraph augmentations utilized by HGNN are seamlessly integrated through invariant learning, as detailed in Section 4.5.

4.4 Dual-view Learning

Dual-branch network. Dual-view learning jointly encodes both the raw hypergraph $\mathcal{G}$ and the synthetic hypergraph $\mathcal{G}^{syn}$ with its tight structure, integrating beneficial information from both. This is achieved through a dual-branch network with shared weights. Specifically, the hypergraph encoder f_ω is applied to encode both views:

$$\mathbf{X}'_V = f_\omega(\mathcal{G}, \mathbf{X}_V) \tag{11}$$

$$\mathbf{X}_V^{syn'} = f_\omega(\mathcal{G}^{syn}, \mathbf{X}_V) \tag{12}$$

where, $\mathcal{G}$ and $\mathcal{G}^{syn}$ represent the raw hypergraph and the synthetic hypergraph; $\mathbf{X}_V$ denotes the initial node representations; $\mathbf{X}'_V$ and $\mathbf{X}_V^{syn'}$ represent the updated node representations of $\mathcal{G}$ and $\mathcal{G}^{syn}$ respectively.

Once the node representations are obtained, the decoder f_ϕ is utilized to predict the probability of forming a hyperedge. We consider the observed hyperedges as positive samples. To ensure effective model training, negative samples need to be generated via negative sampling. Following prior research [6, 39], we replace half of the nodes in positive hyperedges with random nodes to create negative hyperedges. We call the set of these negative hyperedges $\overline{\mathcal{E}}$. We want the observed hyperedges to get high scores and negative hyperedges to get low scores. Therefore, the training objective can be expressed as follows:

$$\mathcal{L}_{\text{raw}} = 1 - \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} f_\phi\left(\mathbf{X}'_V, e\right) + \frac{1}{|\bar{\mathcal{E}}|}\sum_{\bar{e}\in\overline{\mathcal{E}}} f_\phi\left(\mathbf{X}'_V, \bar{e}\right) \tag{13}$$

Simultaneously, for the synthetic hypergraph, there exists a similar objective:

$$\mathcal{L}_{syn} = 1 - \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} f_\phi\left(\mathbf{X}_V^{syn'}, e\right) + \frac{1}{|\bar{\mathcal{E}}|}\sum_{\bar{e}\in\bar{\mathcal{E}}} f_\phi\left(\mathbf{X}_V^{syn'}, \bar{e}\right) \tag{14}$$

Consistency loss. We need to guide the encoder f_ω to prioritize learning common information over divergent information from dual views. This can be achieve by imposing constraints that promote the consistency of the embedding spaces between the two views. To facilitate this, we introduce a consistency loss. Specifically, we start by employing a simple MLP to map representations $\mathbf{X}_V'$ and $\mathbf{X}_V^{syn'}$ to another common embedding space. Then, cosine distance is utilized to measure the discrepancy between $\mathbf{X}_V$ and $\overline{\mathbf{X}}_V$.

$$\mathbf{P} = \mathrm{MLP}(\mathbf{X}_V') \tag{15}$$

$$\mathbf{Q} = \mathrm{MLP}(\mathbf{X}_V^{syn'}) \tag{16}$$

$$\mathcal{L}_{cs} = 1 - \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\frac{\mathbf{p}_i \cdot \mathbf{q}_i}{\| \mathbf{p}_i \| \| \mathbf{q}_i \|} \tag{17}$$

where $\mathbf{p}_i$ and $\mathbf{q}_i$ represent the i-th row vectors of matrices $\mathbf{P}$ and $\mathbf{Q}$, respectively. The MLP is implemented using a linear layer followed by a ReLU activation function.

We combine the supervised loss with the consistency loss, resulting in the following loss function:

$$\mathcal{L} = \mathcal{L}_{raw} + \alpha \cdot \mathcal{L}_{syn} + \beta \cdot \mathcal{L}_{cs} \tag{18}$$

where α represents the trade-off coefficient for the supervised loss of the tight view, and β controls the strength of the consistency loss, thereby influencing the extent to which the model prioritizes learning common information over divergent information in dual views.

4.5 Invariant Learning

Operating on multiple augmented hypergraphs. Invariant learning requires that our model simultaneously maintains optimal performance across multiple augmented hypergraphs. This ensures that the learned patterns remain invariant despite minor changes to the hypergraph and are generalizable during inference. Formally, for the m-th generated hypergraph, the forward propagation process is defined as follows:

$$\mathbf{X}_V^{m\prime} = f_\omega(\mathcal{G}_m^t, \mathbf{X}_V) \tag{19}$$

$$s_{e'}^m = f_\phi(\mathbf{X}_V^{m\prime}, e') \tag{20}$$

where $\mathbf{X}_V^{m\prime}$ denotes node embeddings of hypergraph $\mathcal{G}_m^t$, and $s_{e'}^m$ represents the probability of the candidate hyperedge e' being a true hyperedge predicted by the decoder f_ϕ using $\mathbf{X}_V^{m\prime}$.

Invariant risk minimization. To force the model to perform well across multiple augmented hypergraphs simultaneously, we employ invariant risk minimization [2]. Invariant risk minimization is an optimization goal aimed at developing an encoder, f_ω, that can learn non-spurious correlations while excluding spurious ones. The learned representations $f_\omega(\mathcal{G}, \mathbf{X}_V) \in \mathcal{H}$ should meet two requirements: they must be sufficiently informative for accurate predictions, and there should exist an optimal classifier f_ϕ that performs well across representations from multiple augmented hypergraphs. The objective of invariant risk minimization can be formalized as

$$\begin{aligned} \min_{\substack{\omega:\mathcal{X}\to\mathcal{H}\\ \phi:\mathcal{H}\to\mathcal{Y}}} \quad & \textstyle\sum_{m=1}^{M} R(f_\omega \circ f_\phi, \mathcal{G}_m^t, \mathbf{X}_V) \\ \text{subject to} \quad & \phi \in \underset{\bar{\phi}:\mathcal{H}\to\mathcal{Y}}{\arg\min} R(f_{\bar{\phi}} \circ f_\omega, \mathcal{G}^{aug}, \mathbf{X}_V), \text{ for all } \mathcal{G}^{aug} \in \{\mathcal{G}_m^{aug}\}_{m=1}^M. \end{aligned} \tag{21}$$

where R represents empirical risk of the hyperedge prediction task. In our context, the specific form of $R(f_\omega \circ f_\phi, \mathcal{G}, \mathbf{X}_V)$ is given by:

$$R(f_\omega \circ f_\phi, \mathcal{G}, \mathbf{X}_V) = 1 - \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} f_\phi(f_\omega(\mathcal{G}, \mathbf{X}_V), e) + \frac{1}{|\bar{\mathcal{E}}|} \sum_{\bar{e} \in \bar{\mathcal{E}}} f_\phi(f_\omega(\mathcal{G}, \mathbf{X}_V), \bar{e}) \tag{22}$$

The objective of invariant risk minimization here indeed represents a challenging bi-level optimization problem. In practice, this objective is often relaxed for implementation purposes [2]. Specially, the hypergraph encoder f_ω and decoder f_ϕ can be considered as a single unit, called the entire encoder $f_\xi(\mathcal{G}, \mathbf{X}_V, e) = f_\phi(f_\omega(\mathcal{G}, \mathbf{X}_V), e) : \mathcal{X} \to \mathcal{Y}$. Additionally, we can make a prior assumption that the optimal classifier f_ρ is a fixed constant classifier, i.e. $f_\rho = \rho$, where ρ is a constant. Consequently, the problem transforms into finding the best entire encoder f_ξ under this fixed constant classifier f_ρ. Formally, the relaxed objective is as follows:

$$\mathcal{R} = \sum_{m=1}^{M} \underbrace{R(f_\xi, \mathcal{G}_m^t, \mathbf{X}_V)}_{\text{supervised loss}} + \underbrace{\lambda \cdot \left\| \nabla_{\rho|\rho=1.0} R(\rho \cdot f_\xi, \mathcal{G}_m^t, \mathbf{X}_V) \right\|^2}_{\text{Invariance regularization}} \tag{23}$$

In this objective, the first part pushes the model to predict well on each augmented hypergraph. The second part minimizes the gradients of the fixed pseudo classifier f_ρ parameters across all augmented hypergraphs, enabling the fixed pseudo classifier to be optimal across all augmented hypergraphs, indicating that the entire encoder learns enough invariant relationships.

4.6 Optimization

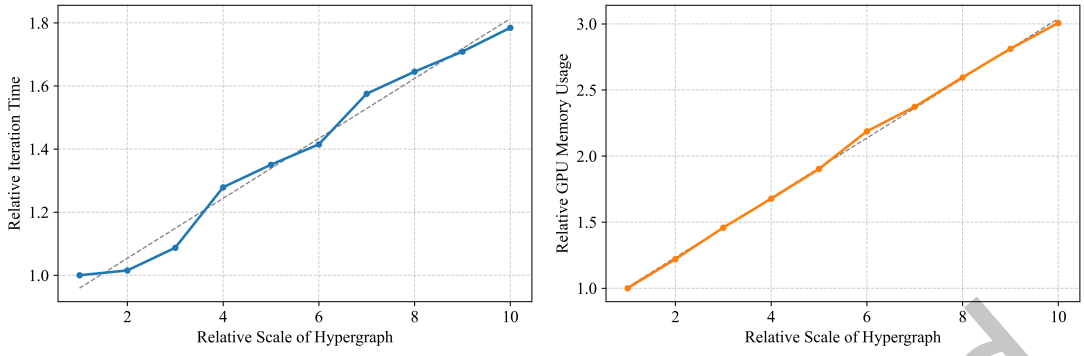
The Multi-HyperLinker employs two distinct learning strategies, each with its own optimization objective. By combining Equations (18) and (23), the overall optimization objective of the Multi-HyperLinker is formulated as follows:

$$L = \mathcal{L} + \mathcal{R} \tag{24}$$

4.7 Complexity Analysis

Space Complexity: the components of Multi-HyperLinker include the hypergraph encoder f_ω, the decoder f_ϕ, and the projector f_{mlp}. The encoder is composed of $L_{encoder}$ HNHN layers, each containing two linear transformations. Excluding bias terms, each layer requires $3d^2$ parameters, resulting in a total of $3L_{encoder}d^2$ parameters for the encoder. The decoder consists of $L_{decoder}$ linear layers, contributing $(L_{decoder} - 1)d^2 + d$ parameters in total, while the projector is implemented as a single linear layer with d^2 parameters. In addition, the model maintains node embeddings of size $|V| \times d$, where $|V|$ denotes the number of nodes—a component commonly required in hyperedge prediction methods. Therefore, the total number of parameters is $(3L_{encoder} + L_{decoder})d^2 + (|V| + 1)d$. Given that $|V| \gg d \gg L_{encoder}, L_{decoder}$, the space complexity is dominated by the node embeddings, resulting in an overall space complexity of $\mathcal{O}(|V|d)$, which scales linearly with the number of nodes in the hypergraph.

Time Complexity: the encoder consists of L_{encoder} HNHN layers, each performing bidirectional updates between nodes and hyperedges through two linear transformations. The per-layer computational complexity is $\mathcal{O}(|V|d^2 + |E|d^2)$, where $|V|$ and $|E|$ are the numbers of nodes and hyperedges. As d is fixed, d^2 can be treated as a constant factor. The decoder, which is an L_{decoder}-layer MLP, adds a smaller computational term of $\mathcal{O}(d^2)$. These computational costs are common to current hypergraph-based methods. The projector that implemented as a single linear layer contributes an additional smaller term of $\mathcal{O}(|V|d^2)$. Under our multi-view training setting, the model also processes a KNN view and M augmented views. Thus, the overall training complexity can be approximated as $\mathcal{O}((2 + M)(2|V| + |E| + 1)d^2)$, which scales linearly with the size of the hypergraph. During inference, the KNN view, the M augmented views, and the projector are not utilized; therefore, the time complexity



(a) (b)

Fig. 4. Scalability of the Multi-HyperLinker with increasing hypergraph size. (a) Relative iteration time and (b) relative GPU memory usage as functions of the relative scale of the hypergraph (measured by the number of nodes and hyperedges, synchronously increased from 5,000 to 50,000). All values are normalized to the baseline at 5,000 nodes/hyperedges (i.e., relative values from 1.0 to 10.0), demonstrating the model's linear scalability in both time and space consumption.

Table 2. Dataset statistics.

Dataset	# Nodes	# Hyperedges	Average size of hyperedges	Dentity
Steam	16,455	10,720	6.5	0.65
Citeseer	1,457	1,078	3.2	0.74
Weeplaces	6,189	4,945	2.6	0.8
NDC	3,073	12,629	6.6	4.11

is reduced to $\mathcal{O}((|V| + |E| + 1)d^2)$. Overall, the time complexity of Multi-HyperLinker is comparable to that of existing methods and scales linearly with the size of the hypergraph.

Empirical Scaling Analysis: to investigate the scalability of the model across different dataset sizes, we conduct additional empirical experiments by increasing the scale of randomly synthetic hypergraphs, specifically in terms of the number of nodes ($|V|$) and hyperedges ($|E|$). The numbers of nodes and hyperedges are synchronously increased from 5,000 to 50,000. At each scale, we record both the training time and memory consumption to comprehensively assess the time and space scalability of the proposed method. The results in Figure 4 show that both training time and GPU memory usage scale linearly with hypergraph size, indicating that our method can efficiently handle large-scale datasets.

5 Experiment

5.1 Datasets

We utilize four real-world datasets from distinct domains, including social groups, academic collaborations, location-based social networks, and drug compositions. Steam [29] dataset originates from an online gaming platform where users form groups for gaming and discussions, represented by nodes and hyperedges respectively. Citeseer [18] is a co-citation dataset, where each node represents a research paper and each hyperedge represents a set of papers cited by the same paper. Weeplaces [25] logs users' travel histories within a location-based social network, with nodes as individual users and hyperedges representing group travel. NDC [4] dataset is a drug

composition dataset. The NDC dataset originates from the National Drug Code Directory in the United States. In this dataset, each hyperedge corresponds to a specific drug, while the nodes represent the substance comprising that drug.

The details about the datasets is presented in Table 2. Building upon prior research [16, 23], we utilize density as a metric to measure the level of connectivity in the raw hypergraph. This can be formally expressed as: $density = |\mathcal{E}|/|\mathcal{V}|$. A lower density in a raw hypergraph indicates a weaker connectivity, resulting in a more fragmented structure.

All datasets are divided into training, validation, and testing sets in a ratio of 0.6:0.2:0.2. To collect negative hyperedges, following prior work [6, 39], we replace nodes within positive hyperedges with randomly selected nodes at a specified ratio, denoted as p. In our experiments, for diverse negative samples, we shuffle negative samples generated by p values of 0.1, 0.5, and 0.9. Generally, $p = 0.1$ produces challenging negative samples, while $p = 0.9$ yields easier negative samples. The ratio of positive hyperedges to negative hyperedges is 1:1.

5.2 Compared Methods

We compare our proposed method against representative and state-of-the-art deep learning-based hyperedge prediction approaches. These methods are categorized into set-based methods that do not explicitly consider the hypergraph structure and hypergraph-based methods that explicitly consider the hypergraph structure.

Set-based methods:

- DeepSet [42]: DeepSet introduces a learnable family of permutation-invariant functions designed for set tasks, making it applicable to hyperedge prediction.
- NHP [39]: NHP treats each hyperedge as a fully connected graph and employs graph convolutional neural networks to learn representations on these fully connected graphs.
- Hyper-SAGNN [44]: Hyper-SAGNN views each hyperedge as a graph, using self-attention-based graph neural networks to learn representations.

Hypergraph-based methods:

- HNHN [11]: HNHN is a representative framework for hypergraph representation learning, employing hypergraph convolution networks for nodes and hyperedges, showcasing strong performance.
- AHP [18]: AHP employs hypergraph neural networks to encode hypergraphs while introducing an adversarial training-based method for hyperedge prediction.
- CHESHIRE [6]: CHESHIRE is a hyperedge prediction method designed specifically to address missing reactions within metabolic networks. It treats the incidence matrix of the hypergraph as node features, extends hyperedges to fully connected graphs, and encodes nodes using Chebyshev spectral graph convolutional networks.
- Cash [21]: Cash applies self-supervised contrastive learning and context-aware node aggregation to enhance the representations of nodes and hyperedges, making it the current state-of-the-art model for hyperedge prediction.

5.3 Implementation Details

All baseline implementations are sourced from official repositories on GitHub. For a fair comparison, the dimensions for both node and hyperedge embeddings are set to 64 across all methods. In our implementation, we utilize the PyTorch[1] framework. The layer of the hypergraph neural network HNHN is set to 2 in our experiments, with a decoder comprising two layers of MLP using ReLU non-linear activation. For the Citeseer dataset, the batch size is 64, while for the other datasets, it is set to 1024. We employ the Adam [20] optimizer with a learning rate

[1]https://pytorch.org/

Table 3. Performance comparisons (%) on four real-world datasets. **Bold** indicates the best performance, and underline represents the second-best performance. The term *improvement* refers to the improvement achieved by our proposed Multi-HyperLinker compared to the best-performing hypergraph-based baseline.

Category	Dataset	Steam		Citeseer		Weeplaces		NDC	
		Hits@50	Hits@100	Hits@10	Hits@30	Hits@50	Hits@100	Hits@50	Hits@100
Set-based	DeepSet	$10.06_{\pm 0.43}$	$16.08_{\pm 1.17}$	$0_{\pm 0}$	$27.35_{\pm 2.27}$	$15.47_{\pm 0.99}$	$21.86_{\pm 0.64}$	$46.10_{\pm 2.79}$	$57.66_{\pm 2.01}$
	HyperSAGNN	$11.27_{\pm 1.08}$	$17.37_{\pm 0.89}$	$11.44_{\pm 2.21}$	$25.49_{\pm 2.29}$	$14.60_{\pm 1.46}$	$21.42_{\pm 0.39}$	$47.68_{\pm 1.44}$	$55.80_{\pm 0.86}$
	NHP	$\underline{22.18}_{\pm 0.73}$	$\underline{28.54}_{\pm 0.48}$	$24.19_{\pm 3.22}$	$36.37_{\pm 1.42}$	$13.90_{\pm 1.82}$	$20.52_{\pm 3.56}$	$34.50_{\pm 2.01}$	$64.78_{\pm 2.43}$
Hypergraph-based	AHP	$6.70_{\pm 1.35}$	$10.67_{\pm 1.08}$	$18.88_{\pm 3.45}$	$30.98_{\pm 1.27}$	$18.73_{\pm 1.86}$	$28.59_{\pm 2.51}$	$48.05_{\pm 4.17}$	$57.97_{\pm 1.23}$
	HNHN	$18.88_{\pm 2.44}$	$25.87_{\pm 2.57}$	$\underline{29.76}_{\pm 1.13}$	$37.05_{\pm 2.28}$	$28.41_{\pm 0.46}$	$33.06_{\pm 0.91}$	$66.21_{\pm 3.74}$	$\underline{79.72}_{\pm 2.09}$
	CHESHIRE	$9.16_{\pm 0.86}$	$14.13_{\pm 0.50}$	$13.79_{\pm 1.58}$	$\underline{38.91}_{\pm 0.58}$	$6.10_{\pm 1.04}$	$6.97_{\pm 2.08}$	$\underline{67.21}_{\pm 0.38}$	$78.63_{\pm 1.53}$
	Cash	$15.00_{\pm 0.74}$	$21.11_{\pm 0.61}$	$27.13_{\pm 2.29}$	$36.90_{\pm 0.58}$	$\underline{29.28}_{\pm 1.07}$	$\underline{34.84}_{\pm 0.99}$	$60.08_{\pm 2.04}$	$72.91_{\pm 1.45}$
	Multi-HyperLinker	$\mathbf{22.62}_{\pm 0.86}$	$\mathbf{30.55}_{\pm 1.23}$	$\mathbf{32.40}_{\pm 1.94}$	$\mathbf{44.50}_{\pm 0.71}$	$\mathbf{31.19}_{\pm 0.36}$	$\mathbf{38.27}_{\pm 0.93}$	$\mathbf{70.77}_{\pm 3.02}$	$\mathbf{82.38}_{\pm 2.88}$
	Improvement	19.80%	18.09%	8.87%	14.36%	6.52%	9.81%	5.30%	3.34%

Category	Dataset	Steam		Citeseer		Weeplaces		NDC	
		AP	AUC	AP	AUC	AP	AUC	AP	AUC
Set-based	DeepSet	$64.40_{\pm 0.88}$	$61.83_{\pm 1.09}$	$57.87_{\pm 0.87}$	$56.11_{\pm 1.30}$	$56.44_{\pm 0.42}$	$46.72_{\pm 0.48}$	$87.21_{\pm 1.51}$	$84.66_{\pm 1.76}$
	HyperSAGNN	$65.47_{\pm 0.31}$	$62.47_{\pm 0.32}$	$59.06_{\pm 0.73}$	$54.04_{\pm 1.43}$	$55.37_{\pm 0.55}$	$46.26_{\pm 0.36}$	$86.75_{\pm 0.22}$	$83.17_{\pm 0.42}$
	NHP	$\underline{71.90}_{\pm 0.20}$	$67.79_{\pm 0.18}$	$69.35_{\pm 1.21}$	$63.46_{\pm 1.07}$	$57.72_{\pm 2.24}$	$53.92_{\pm 2.74}$	$91.93_{\pm 1.14}$	$92.94_{\pm 1.16}$
Hypergraph-based	AHP	$61.37_{\pm 1.10}$	$61.65_{\pm 0.58}$	$65.95_{\pm 3.45}$	$62.01_{\pm 1.89}$	$67.36_{\pm 2.25}$	$65.83_{\pm 2.50}$	$88.21_{\pm 2.47}$	$87.08_{\pm 2.90}$
	HNHN	$71.81_{\pm 0.61}$	$\underline{69.30}_{\pm 1.79}$	$\underline{71.58}_{\pm 0.68}$	$65.66_{\pm 0.43}$	$69.58_{\pm 0.69}$	$65.17_{\pm 1.42}$	$94.51_{\pm 0.48}$	$93.61_{\pm 0.91}$
	CHESHIRE	$64.11_{\pm 1.79}$	$63.99_{\pm 1.73}$	$68.69_{\pm 0.53}$	$\underline{68.21}_{\pm 0.62}$	$67.79_{\pm 3.01}$	$\underline{70.48}_{\pm 1.61}$	$\underline{95.92}_{\pm 0.28}$	$\underline{95.38}_{\pm 0.40}$
	Cash	$66.84_{\pm 0.39}$	$62.49_{\pm 0.47}$	$70.76_{\pm 0.43}$	$66.49_{\pm 0.07}$	$\underline{71.32}_{\pm 0.67}$	$66.61_{\pm 1.00}$	$93.01_{\pm 1.66}$	$93.71_{\pm 1.45}$
	Multi-HyperLinker	$\mathbf{74.42}_{\pm 1.49}$	$\mathbf{70.98}_{\pm 1.23}$	$\mathbf{74.40}_{\pm 0.41}$	$\mathbf{68.47}_{\pm 1.50}$	$\mathbf{74.73}_{\pm 0.79}$	$\mathbf{72.17}_{\pm 1.45}$	$\mathbf{96.13}_{\pm 0.32}$	$\mathbf{95.90}_{\pm 0.32}$
	Improvement	3.63%	2.42%	3.94%	0.38%	4.78%	2.40%	0.22%	0.55%

of 5e-4 and L_2 regularization set at 3e-5. The number of epochs is set to 500, and we apply an early stopping strategy with a patience of 30 to prevent overfitting. For the hyperparameters α, β, and λ, we conduct a search within the sets {1e-3, 1e-2, 1e-1, 1, 10}, {5e-5, 5e-4, 5e-3, 5e-2, 5e-1}, and {1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1}, respectively. The number of augmented hypergraphs M is set to 2 across all datasets, and the hyperedge replacement ratio r is set to 0.2. The hyperparameter k used to construct the tight view is searched within the range of {4, 8, 16, 32, 64, 128, 256}. For each dataset, all methods run 5 times, and we report their average performance and standard deviations on the test set.

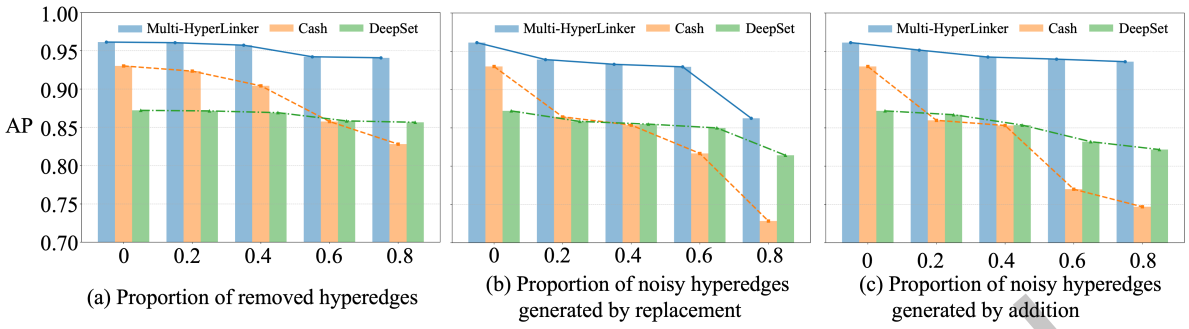
5.4 Metrics

For a thorough model evaluation, we utilize basic metrics: average precision (AP) and area under the curve (AUC) computed from prediction scores. Additionally, following [17], we employ the more challenging ranking metric Hits@K, expecting the model to rank positive hyperedges higher than almost all negative ones. Hits@K measures the proportion of positive edges ranked within the top K positions among these randomly sampled negatives. It assesses how effectively the model ranks positive hyperedges against randomly sampled negative ones. For the small dataset Citeseer dataset, K values are {10, 30}; for other datasets, K values are {50, 100}. In our experiment, Hits@K is also utilized for model selection.

5.5 Main Result

Table 3 displays the comparative results of overall performance. From this, we can observe the following:

Multi-HyperLinker demonstrates outstanding performance across various datasets. Multi-HyperLinker outperforms other baselines across all datasets, showcasing its superiority. This is largely attributed to Multi-HyperLinker's ability to capture reliable higher-order interaction patterns through multi-hypergraph learning.



(a) Proportion of removed hyperedges

(b) Proportion of noisy hyperedges generated by replacement

(c) Proportion of noisy hyperedges generated by addition

Fig. 5. (a): Robustness experiments for incompleteness. (b): Robustness experiments for noise hyperedges generated by random replacement. (c): Robustness experiments for noise hyperedges generated by random addition.

Specifically, on the most challenging dataset with the lowest density, Steam, our method exhibits the most significant 19.80% improvement compared to the hypergraph-based baselines and also surpasses all set-based methods. This clearly illustrates that Multi-HyperLinker significantly enhances hyperedge prediction performance on the unreliable underlying hypergraph.

Set-based methods excel in low-density datasets. Set-based methods outperform or are comparable to hypergraph-based methods, except for Multi-HyperLinker, in low-density datasets such as Steam. However, in high-density datasets, set-based methods are significantly inferior to hypergraph-based methods. This indicates that set-based methods are more suitable for scenarios where hypergraph reliability is low.

Existing hypergraph-based methods show pronounced advantages in high-density datasets. Hypergraph-based methods perform poorly on low-density data due to the high unreliability of hypergraphs. However, as the dataset density increases, the superiority of hypergraph-based methods becomes more pronounced. This suggests that explicitly encoding hypergraph structures is a double-edged sword: encoding unreliable hypergraph structural information can have negative effects, whereas encoding reliable hypergraph structural information can yield positive benefits. Multi-HyperLinker is capable of encoding reliable hypergraph information through dual-view learning and invariant learning, thereby achieving consistent superiority across all datasets.

5.6 Robustness Experiment

To further investigate the performance of our proposed Multi-HyperLinker under unreliability factors like incompleteness and noise, we conduct robustness experiments targeting these issues. Specifically, for incompleteness, we use the dense NDC dataset and exacerbate its incompleteness by randomly deleting hyperedges at varying proportions to observe performance changes. For noise, we explore two main noise patterns - random addition and random replacement, introduced in Section 4.3. We increase the noise level in the dense NDC dataset by randomly perturbing hyperedges, either by adding unrelated nodes or replacing existing nodes with unrelated ones, at varying proportions. We also compared our results with two state-of-the-art methods: the hypergraph-based Cash[21] and the set-based DeepSet[42].

Figure 5(a) demonstrates that as incompleteness intensifies, the performance of Multi-HyperLinker, Cash, and DeepSet declines. Notably, in terms of predictive performance, Multi-HyperLinker consistently outperforms other methods. Cash, another hypergraph-based approach, initially surpasses DeepSet under mild incompleteness but falls behind as incompleteness increases. In terms of robustness, both Multi-HyperLinker and DeepSet exhibit a gradual decline, whereas Cash experiences the most severe performance degradation. Since DeepSet is inherently less affected by the unreliability of the hypergraph structure, Multi-HyperLinker still achieves comparable

Table 4. Ablation Study.

Dataset	Steam		Citeseer		Weeplaces		NDC	
	Hits@50	Hits@100	Hits@10	Hits@30	Hits@50	Hits@100	Hits@50	Hits@100
Multi-HyperLinker	**22.62**$_{\pm 0.86}$	**30.55** $_{\pm 1.23}$	**32.40**$_{\pm 1.94}$	**44.50**$_{\pm 0.71}$	**31.19**$_{\pm 0.36}$	**38.27**$_{\pm 0.93}$	**70.77**$_{\pm 3.02}$	**82.38**$_{\pm 2.88}$
Multi-HyperLinker$_{-IL}$	18.91$_{\pm 1.55}$	26.21$_{\pm 0.53}$	28.68$_{\pm 0.88}$	37.83$_{\pm 1.79}$	29.82$_{\pm 1.03}$	37.66$_{\pm 0.71}$	69.60$_{\pm 1.46}$	82.01$_{\pm 1.29}$
Multi-HyperLinker$_{-DL}$	18.25$_{\pm 1.05}$	25.42$_{\pm 0.56}$	26.82$_{\pm 1.44}$	36.74$_{\pm 0.38}$	27.47$_{\pm 0.29}$	32.79$_{\pm 0.17}$	67.55$_{\pm 0.79}$	78.93$_{\pm 1.40}$
Dataset	Steam		Citeseer		Weeplaces		NDC	
	AP	AUC	AP	AUC	AP	AUC	AP	AUC
Multi-HyperLinker	**74.42**$_{\pm 1.49}$	**70.98**$_{\pm 1.23}$	**74.40**$_{\pm 0.41}$	**68.47**$_{\pm 1.50}$	**74.73**$_{\pm 0.79}$	**72.17**$_{\pm 1.45}$	**96.13**$_{\pm 0.32}$	**95.90**$_{\pm 0.32}$
Multi-HyperLinker$_{-IL}$	72.32$_{\pm 0.72}$	69.13$_{\pm 0.90}$	72.38$_{\pm 0.90}$	67.20$_{\pm 2.11}$	74.29$_{\pm 0.29}$	71.08$_{\pm 0.93}$	95.93$_{\pm 0.33}$	95.61$_{\pm 0.50}$
Multi-HyperLinker$_{-DL}$	70.89$_{\pm 0.43}$	67.88$_{\pm 0.36}$	71.59$_{\pm 0.46}$	66.29$_{\pm 1.08}$	67.72$_{\pm 1.57}$	60.53$_{\pm 2.93}$	95.91$_{\pm 0.35}$	95.66$_{\pm 0.50}$

robustness to DeepSet. Moreover, Multi-HyperLinker attains significantly better performance than set-based methods while maintaining similar robustness. These results demonstrate that our proposed Multi-HyperLinker effectively alleviates the challenges of incompleteness through multi-view hypergraph learning.

Similarly, Figures 5(b) and 5(c) illustrate that as noise levels increase, the performance of Multi-HyperLinker, Cash, and DeepSet declines to varying degrees. CASH shows the poorest robustness to noisy hyperedges, whether through replacement or addition, indicating greater sensitivity of hypergraph-based methods to noise. Multi-HyperLinker, however, demonstrates robustness comparable to DeepSet, with gradual performance degradation, while consistently maintaining superior performance. This suggests that our multi-hypergraph framework and invariant learning strategy effectively enhance the noise resistance of hypergraph-based methods.

5.7 Ablation Experiment

To investigate the impact of Multi-HyperLinker's two parts on prediction performance—namely, dual-view learning and invariant learning, we further perform ablation experiments. We create two variants of Multi-HyperLinker for analysis: variant Multi-HyperLinker$_{-DL}$ by removing the synthetic hypergraph and dual-view learning, and variant Multi-HyperLinker$_{-IL}$ by removing the augmented hypergraphs and invariant learning. We compare the performance of Multi-HyperLinker, Multi-HyperLinker$_{-DL}$, and Multi-HyperLinker$_{-IL}$ across four datasets: Steam, Citeseer, Weeplaces, and NDC.

Table 4 presents the results of the ablation experiments. We can observe that removing either the dual-view learning or the invariant learning led to a decrease in model performance across all four datasets. This comparative results indicates that both the dual-view learning and invariant learning significantly enhance the model's performance in hyperedge prediction. On the one hand, the removal of dual-view learning leads to a significant performance drop, indicating that hypergraph incompleteness is a primary hindrance for hypergraph-based methods. On the other hand, the notable performance decline across all datasets due to the removal of the invariant learning component highlights its critical role in mitigating issues arising from noise in hypergraph-based methods. Ultimately, our proposed Multi-HyperLinker effectively addresses the challenges posed by the unreliability of the raw hypergraph and emerges as the top performer compared to two variants, Multi-HyperLinker$_{-DL}$ and Multi-HyperLinker$_{-IL}$.

5.8 Sensitivity Analysis

In this section, we conduct an analysis of the model's sensitivity to various hyperparameters, examining their impact on prediction performance. We focus on two categories of hyperparameters: those related to the dual-view learning and those related to invariant learning. For the dual-view learning, we analyze the scale α of the

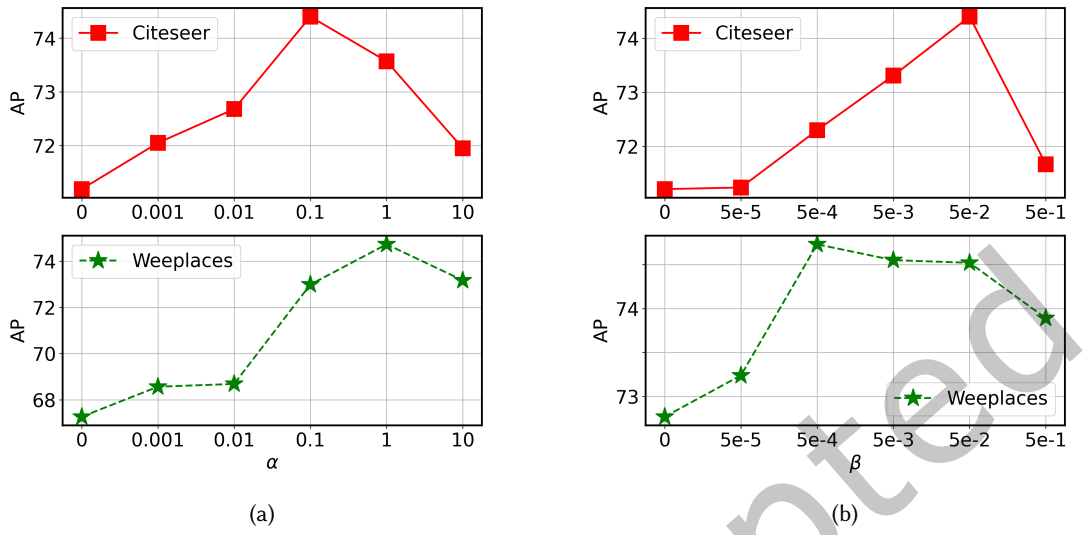


Fig. 6. (a): Sensitivity analysis of α on Citeseer and Weeplaces datasets. (b): Sensitivity analysis of β on Citeseer and Weeplaces datasets.

supervised loss of the synthetic view, the strength β of the consistency loss and the hyperedge size k of the synthetic view. For the hyperparameters related to invariant learning, we investigate the number M of augmented hypergraphs and the strength λ of invariance regularization. In practice, we maintain fixed values for other hyperparameters while varying a specific one to observe the model's AP metric changes on the Citeseer and Weeplaces datasets.

The scale α of synthetic view

α controls the scale of the supervised loss for the synthetic hypergraph, as introduced in equation (18). Figure 6a illustrates the variation of AP with changing α. As α increases, the model's performance initially rises and then declines. A small α limits the model's focus on the synthetic hypergraph. Conversely, excessively large α prioritizes the synthetic hypergraph, amplifying the adverse effects of noise from the synthetic hypergraph, consequently resulting in suboptimal performance. This suggests the necessity of an appropriate scale for α, with 0.1 to 1 being reasonable default value.

The strength β of the consistency loss

β controls the consistency between representations from the raw hypergraph and those from the synthetic hypergraph, as introduced in equation (18). As depicted in 6b, the model's performance exhibits an initial increase followed by a decline as the consistency strength increases. Early gains suggest improved learning through aligned representations, but too much consistency may create unhelpful representations, resulting in declining performance. We also observe that the model is more sensitive to β on Citeseer than on Weeplaces, possibly because Weeplaces has a larger number of nodes, allowing for a more stable estimation of representation consistency. A recommended starting range for β is from $5e-4$ to $5e-2$.

The hyperedge size k of synthetic view.

The value of k reflects the size of each hyperedge in the synthetic hypergraph, where a larger k signifies more interactions between nodes. Figure 7 illustrates the impact of hyperedge size k on model performance. As k

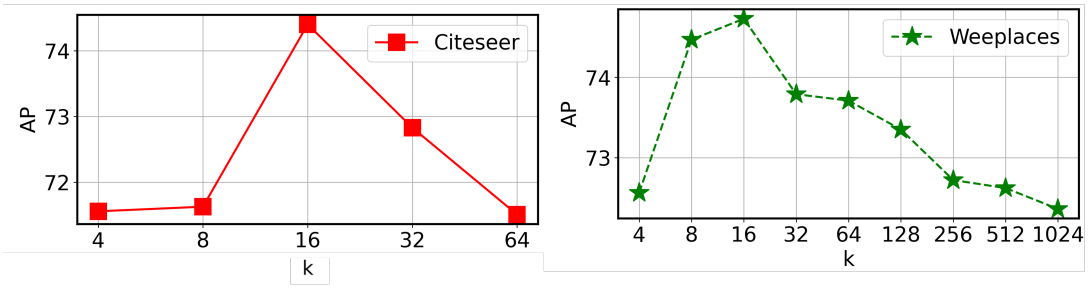


Fig. 7. Sensitivity analysis of k on Citeseer and Weeplaces datasets.

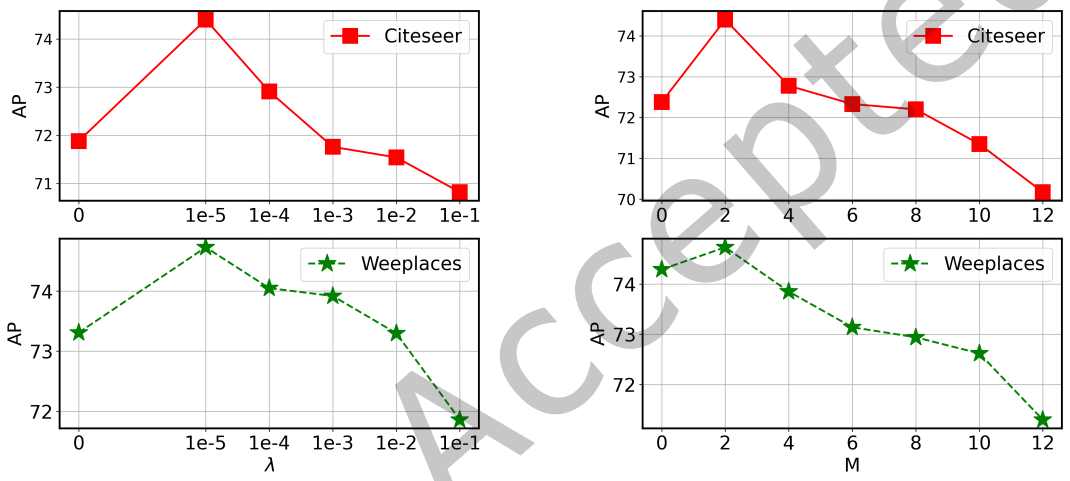


Fig. 8. (a): Sensitivity analysis of λ on Citeseer and Weeplaces datasets. (b): Sensitivity analysis of M on Citeseer and Weeplaces datasets.

increases, the model performance initially improves before declining. Synthetic hypergraph with small k do not introduce sufficient informative interactions, thus failing to significantly enhance the encoding of the hypergraph. Conversely, synthetic hypergraph with overly large k introduces excessive noisy interactions, compromising the hypergraph encoding. Therefore, achieving an appropriately sized synthetic hypergraph is necessary, and an empirical starting point can be 16.

The strength λ of the invariance regularization

λ controls the strength of invariance regularization. From Figure 8a, we observe that as λ increases, the model's performance initially improves and then decreases. A too-small λ is not sufficient to compel the model to learn invariant relationships, while an excessively large λ leads to decreased performance due to an overemphasis on invariance. Therefore, an appropriate λ value is necessary, and $1e-5$ serves as a reasonable starting point.

The number M of the augmented hypergraphs

Table 5. Performance comparison on the Weeplaces dataset using different hypergraph construction methods: no hypergraph (NHP, set-based baseline), random, ground-truth, and KNN-based synthetic hypergraphs. **Bold** indicates the best result in each metric; gray indicates the worst.

Hypergraph Used	hits@50 (%)	hits@100 (%)	AP (%)	AUC (%)
No Hypergraph (NHP)	13.90	20.52	57.72	53.92
Random Hypergraph	**9.61**	**13.75**	**53.25**	**50.93**
Ground-truth Hypergraph	**28.41**	**33.06**	**69.59**	**65.17**
KNN Hypergraph (k=4)	17.90	27.40	66.84	62.38
KNN Hypergraph (k=16)	21.26	31.90	67.68	63.96
KNN Hypergraph (k=128)	19.72	30.54	65.61	61.83

M stands for the number of generated augmented hypergraphs. Figure 8b illustrates the impact of varying M on the model's performance. With a small M, the model's learned invariance might not generalize well and could be affected by spurious correlations. On the other hand, when M is too large, achieving optimal performance across all M augmented hypergraphs becomes challenging, resulting in suboptimal performance. While the performance of Multi-HyperLinker is somewhat sensitive to the choice of M, setting $M = 2$ consistently yields optimal results across different datasets.

5.9 Effectiveness of Synthetic Hypergraph

To quantitatively assess the feasibility and effectiveness of the synthetic hypergraph constructed via KNN, we conduct comparative experiments on the Weeplaces dataset. Specifically, we compare four settings: (1) no hypergraph (NHP, set-based baseline), (2) random hypergraph, (3) ground-truth hypergraph, and (4) KNN-generated synthetic hypergraph with different values of k. The detailed results are shown in Table 5.

The results demonstrate that hypergraph quality significantly impacts model performance, with the ground-truth hypergraph achieving the best results and the random hypergraph performing the worst. KNN-based synthetic hypergraphs consistently outperform the random setting, and an appropriate choice of k (e.g., $k = 16$) yields the best results among KNN variants. Although KNN-based hypergraphs do not reach the performance of ground-truth hypergraphs, they provide a practical and effective alternative when ground-truth information is unavailable.

5.10 Evaluation under Diverse Negative Sampling Strategies

To provide a more comprehensive evaluation, in addition to random replacement negative sampling with varying replacement ratios ($p = 0.1/0.5/0.9$), we further adopt three negative sampling strategies: Sized Negative Sampling (SNS), Motif Negative Sampling (MNS), and Clique Negative Sampling (CNS), following [30, 41]. These strategies allow us to assess model performance under a wider range of negative samples.

Table 6 reports the performance of our Multi-HyperLinker and several state-of-the-art baselines (NHP, CHESHIRE, Cash) on the Steam (low density) and NDC (high density) datasets under these negative sampling protocols. Multi-HyperLinker consistently achieves the best or second-best results across all metrics and sampling strategies, demonstrating strong generalization. Notably, while set-based methods like NHP perform better on sparse hypergraphs, and hypergraph-based methods excel on dense structures, our method adapts well to both scenarios and outperforms all baselines on average. These results confirm the effectiveness of our approach under diverse negative sampling settings.

Table 6. Performance comparison of different methods under various negative sampling strategies on the Steam and NDC datasets. The AVG columns represent the average performance across all negative sampling strategies. **Bold** values indicate the best performance among all methods for each metric and sampling strategy, while underlined values denote the second-best performance.

Steam Dataset								
Method	Hits@50 (%)				Hits@100 (%)			
	SNS	MNS	CNS	AVG	SNS	MNS	CNS	AVG
NHP	29.71	22.79	3.44	18.65	36.94	30.36	6.94	24.75
CHESHIRE	7.01	15.81	3.29	8.71	12.07	21.44	6.19	13.23
Cash	17.54	19.62	2.54	13.23	25.68	26.68	5.91	19.42
Multi-HyperLinker	**31.30**	**24.44**	**3.82**	**19.85**	**36.94**	**33.26**	**8.35**	**26.18**
Method	AP (%)				AUC (%)			
	SNS	MNS	CNS	AVG	SNS	MNS	CNS	AVG
NHP	77.96	74.40	51.97	68.11	**74.24**	72.30	49.55	65.36
CHESHIRE	60.93	65.08	**53.02**	59.67	60.43	59.12	**53.55**	57.70
Cash	73.45	72.80	49.57	65.27	71.55	69.80	46.62	62.66
Multi-HyperLinker	**78.01**	**75.67**	52.50	**68.73**	73.80	**72.49**	51.23	**65.84**
NDC Dataset								
Method	Hits@50 (%)				Hits@100 (%)			
	SNS	MNS	CNS	AVG	SNS	MNS	CNS	AVG
NHP	**86.87**	45.08	4.01	45.32	**87.66**	58.84	5.73	50.74
CHESHIRE	75.76	60.79	16.04	50.86	78.38	66.77	17.58	54.24
Cash	75.89	58.96	13.14	49.33	80.02	63.72	21.22	54.99
Multi-HyperLinker	84.04	**61.62**	**26.46**	**57.37**	86.10	**67.25**	**34.46**	**62.60**
Method	AP (%)				AUC (%)			
	SNS	MNS	CNS	AVG	SNS	MNS	CNS	AVG
NHP	**96.39**	89.43	55.55	80.45	**94.63**	**86.92**	54.80	78.78
CHESHIRE	95.42	88.38	67.39	83.73	93.95	83.29	65.07	80.77
Cash	95.44	89.06	67.23	83.91	94.04	84.81	60.32	79.72
Multi-HyperLinker	96.05	**90.06**	**75.06**	**87.06**	94.15	86.82	**69.66**	**83.54**

5.11 Analysis of Hyperedge Perturbation Strategies

To further investigate the impact of different hyperedge perturbation strategies in invariant learning, we compare three commonly used approaches: *add*, *remove*, and *replace*. These strategies are designed to simulate different types of noise that may occur in real-world hypergraphs, such as missing or redundant connections. We evaluate their performance on the NDC dataset under varying hyperedge retention ratios (100%, 50%, and 20%), which correspond to different levels of hypergraph sparsity. A lower retention ratio indicates a sparser hypergraph structure. The performance is measured by hit@50, hit@100, and their average, as summarized in Table 7.

As shown in Table 7, the *replace* strategy consistently achieves the best average performance across all settings. This is likely because replacing a certain proportion of hypergraph connections simultaneously removes some essential links and introduces unnecessary ones, resulting in more comprehensive and challenging perturbations. Additionally, we observe that *remove* is more effective in dense hypergraphs, while *add* performs better in sparse

Table 7. Performance comparison of different hyperedge perturbation strategies (*add*, *remove*, *replace*) under various hyperedge retention ratios on the NDC dataset. The best results in each column are highlighted in bold.

Hyperedge perturbation	100% Retention			50% Retention			20% Retention		
	hit@50	hit@100	*avg*	hit@50	hit@100	*avg*	hit@50	hit@100	*avg*
add	65.81%	80.00%	*72.91%*	64.02%	78.45%	*71.24%*	60.92%	74.73%	*67.83%*
remove	69.54%	**82.60%**	*76.07%*	**65.35%**	80.38%	*72.87%*	57.33%	74.57%	*65.95%*
replace	**70.77%**	82.38%	***76.58%***	67.45%	**79.15%**	***73.30%***	**63.37%**	**75.24%**	***69.31%***

settings, indicating that the dominant noise type varies with hypergraph density. Overall, *replace* is the most effective and robust strategy.

6 Conclusion

In this paper, we transition our focus from solely developing complex HGNNs to enhancing the reliability of the underlying hypergraphs. We propose Multi-HyperLinker, a novel multi-view hypergraph learning framework that leverages the consistency and invariance across multiple views to capture reliable higher-order interaction patterns for robust hyperedge prediction. To address the inefficacy of information propagation caused by hypergraph incompleteness, our Multi-HyperLinker synthesizes a hypergraph with a more cohesive structure and employs dual-view learning to capture inherent consistency, thereby enhancing encoding efficiency. To mitigate the impact of noisy hypergraphs, we generate multiple augmented hypergraphs and apply invariant learning to capture robust, non-spurious higher-order relations, significantly enhancing robustness against noise interference. Extensive experiments across four real-world datasets demonstrate the superiority of our proposed Multi-HyperLinker. In future work, we will extend Multi-HyperLinker to inductive and temporal scenarios, while exploring more comprehensive schemes for hypergraph construction and candidate generation to capture diverse node relationships, thereby enhancing the practical feasibility and efficiency of our framework.

Acknowledgments

This research was funded by the National Natural Science Foundation of China through Grants 62206267, 62172261, and 62303439.

References

[1] Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. 2023. A survey on hypergraph representation learning. *Comput. Surveys* 56, 1 (2023), 1–38. https://dl.acm.org/doi/10.1145/3605776

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* (2019). https://arxiv.org/abs/1907.02893

[3] Song Bai, Feihu Zhang, and Philip H.S. Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637. doi:10.1016/j.patcog.2020.107637

[4] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences* (2018). doi:10.1073/pnas.1800683115

[5] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. 2022. Hypergraph Structure Learning for Hypergraph Neural Networks.. In *IJCAI*. 1923–1929.

[6] Can Chen, Chen Liao, and Yang-Yu Liu. 2023. Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning. *Nature Communications* 14, 1 (2023), 2375. https://www.nature.com/articles/s41467-023-38110-7

[7] Can Chen and Yang-Yu Liu. 2023. A Survey on Hyperlink Prediction. *IEEE Transactions on Neural Networks and Learning Systems* (2023), 1–17. doi:10.1109/TNNLS.2023.3286280

[8] Hao Chen, Linyan Li, Fuyuan Hu, Fan Lyu, Liuqing Zhao, Kaizhu Huang, Wei Feng, and Zhenping Xia. 2023. Multi-semantic hypergraph neural network for effective few-shot learning. *Pattern Recognition* 142 (2023), 109677. doi:10.1016/j.patcog.2023.109677

[9] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. 2022. You are AllSet: A Multiset Function Framework for Hypergraph Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=hpBTIv2uy_E

[10] Chaoran Cui, Xiaojie Li, Chunyun Zhang, Weili Guan, and Meng Wang. 2023. Temporal-Relational hypergraph tri-Attention networks for stock trend prediction. *Pattern Recognition* 143 (2023), 109759. doi:10.1016/j.patcog.2023.109759

[11] Yihe Dong, Will Sawin, and Yoshua Bengio. 2020. HNHN: Hypergraph Networks with Hyperedge Neurons. *ICML Graph Representation Learning and Beyond Workshop* (2020). https://arxiv.org/abs/2006.12278

[12] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 3558–3565. doi:10.1609/aaai.v33i01.33013558

[13] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. 2020. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 5 (2020), 2548–2566. https://ieeexplore.ieee.org/document/9264674

[14] Lei Guo, Hongzhi Yin, Tong Chen, Xiangliang Zhang, and Kai Zheng. 2021. Hierarchical hyperedge embedding-based representation learning for group recommendation. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–27.

[15] Yan Han, Peihao Wang, Souvik Kundu, Ying Ding, and Zhangyang Wang. 2023. Vision HGNN: An Image is More than a Graph of Nodes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 19878–19888. https://openaccess.thecvf.com/content/ICCV2023/html/Han_Vision_HGNN_An_Image_is_More_than_a_Graph_of_ICCV_2023_paper.html

[16] Shuguang Hu, Xiaowei Wu, and T-H. Hubert Chan. 2017. Maintaining Densest Subsets Efficiently in Evolving Hypergraphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 929–938. doi:10.1145/3132847.3132907

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 22118–22133. https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf

[18] Hyunjin Hwang, Seungwoo Lee, Chanyoung Park, and Kijung Shin. 2022. AHP: Learning to Negative Sample for Hyperedge Prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Madrid</city>, <country>Spain</country>, </conf-loc>) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2237–2242. doi:10.1145/3477495.3531836

[19] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic Hypergraph Neural Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 2635–2641. doi:10.24963/ijcai.2019/366

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[21] Yunyong Ko, Hanghang Tong, and Sang-Wook Kim. 2023. Enhancing Hyperedge Prediction with Context-Aware Self-Supervised Learning. arXiv:2309.05798 [cs.LG]

[22] Dongjin Lee and Kijung Shin. 2023. I'm Me, We're Us, and I'm Us: Tri-directional Contrastive Learning on Hypergraphs. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 7 (Jun. 2023), 8456–8464. doi:10.1609/aaai.v37i7.26019

[23] Geon Lee, Minyoung Choe, and Kijung Shin. 2021. How Do Hyperedges Overlap in Real-World Hypergraphs? - Patterns, Measures, and Generators. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 3396–3407. doi:10.1145/3442381.3450010

[24] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 3744–3753.

[25] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting Geographical Neighborhood Characteristics for Location Recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (Shanghai, China) *(CIKM '14)*. Association for Computing Machinery, New York, NY, USA, 739–748. doi:10.1145/2661829.2662002

[26] Yifan Lu, Mengzhou Gao, Huan Liu, Zehao Liu, Wei Yu, Xiaoming Li, and Pengfei Jiao. 2023. Neighborhood overlap-aware heterogeneous hypergraph neural network for link prediction. *Pattern Recognition* 144 (2023), 109818. doi:10.1016/j.patcog.2023.109818

[27] Zhicong Lu, Li Jin, Peiguang Li, Yu Tian, Linhao Zhang, Sirui Wang, Guangluan Xu, Changyuan Tian, and Xunliang Cai. 2024. Rethinking the reversal curse of LLMs: a prescription from human knowledge reversal. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 7518–7530.

[28] Zhicong Lu, Changyuan Tian, PeiguangLi PeiguangLi, Li Jin, Sirui Wang, Wei Jia, Ying Shen, and Guangluan Xu. 2025. PIPER: Benchmarking and Prompting Event Reasoning Boundary of LLMs via Debiasing-Distillation Enhanced Tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 28591–28613.

[29] Mark O'Neill, Elham Vaziripour, Justin Wu, and Daniel Zappala. 2016. Condensing Steam: Distilling the Diversity of Gamer Behavior. In *Proceedings of the 2016 Internet Measurement Conference* (Santa Monica, California, USA) *(IMC '16)*. Association for Computing Machinery, New York, NY, USA, 81–95. doi:10.1145/2987443.2987489

[30] Prasanna Patil, Govind Sharma, and M. Narasimha Murty. 2020. Negative Sampling for Hyperlink Prediction in Networks. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part II* (Singapore, Singapore). Springer-Verlag, Berlin, Heidelberg, 607–619. doi:10.1007/978-3-030-47436-2_46

[31] Changyuan Tian, Zhicong Lu, Zequn Zhang, Heming Yang, Wei Cao, Zhi Guo, Xian Sun, and Li Jin. 2025. HyperMixer: Specializable Hypergraph Channel Mixing for Long-term Multivariate Time Series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 20885–20893.

[32] Yu Tian, Xingliang Huang, Ruigang Niu, Hongfeng Yu, Peijin Wang, and Xian Sun. 2022. Hypertron: Explicit Social-Temporal Hypergraph Framework for Multi-Agent Forecasting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1356–1362. doi:10.24963/ijcai.2022/189 Main Track.

[33] Yu Tian, Xian Sun, Ruigang Niu, Hongfeng Yu, Zicong Zhu, Peijin Wang, and Kun Fu. 2022. Fully-weighted HGNN: Learning efficient non-local relations with hypergraph in aerial imagery. *ISPRS Journal of Photogrammetry and Remote Sensing* 191 (2022), 263–276. doi:10.1016/j.isprsjprs.2022.07.001

[34] Shun Wang, Yong Zhang, Xuanqi Lin, Yongli Hu, Qingming Huang, and Baocai Yin. 2024. Dynamic Hypergraph Structure Learning for Multivariate Time Series Forecasting. *IEEE Transactions on Big Data* 10, 4 (2024), 556–567. doi:10.1109/TBDATA.2024.3362188

[35] Tianxin Wei, Yuning You, Tianlong Chen, Yang Shen, Jingrui He, and Zhangyang Wang. 2022. Augmentations in hypergraph contrastive learning: fabricated and generative. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 139, 14 pages.

[36] Hanrui Wu, Nuosi Li, Jia Zhang, Sentao Chen, Michael K. Ng, and Jinyi Long. 2024. Collaborative contrastive learning for hypergraph node classification. *Pattern Recognition* 146 (2024), 109995. doi:10.1016/j.patcog.2023.109995

[37] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*. 70–79. https://dl.acm.org/doi/10.1145/3477495.3532058

[38] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-Supervised Hypergraph Transformer for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 2100–2109. doi:10.1145/3534678.3539473

[39] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. 2020. NHP: Neural Hypergraph Link Prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 1705–1714. doi:10.1145/3340531.3411870

[40] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2147–2157. doi:10.1145/3308558.3313635

[41] Taehyung Yu, Soo Yong Lee, Hyunjin Hwang, and Kijung Shin. 2024. Prediction Is NOT Classification: On Formulation and Evaluation of Hyperedge Prediction. In *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*. 349–356. doi:10.1109/ICDMW65004.2024.00051

[42] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep Sets. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf

[43] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. 2018. Beyond Link Prediction: Predicting Hyperlinks in Adjacency Space. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). doi:10.1609/aaai.v32i1.11780

[44] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2020. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryeHuJBtPH

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

Received 22 January 2025; revised 25 June 2025; accepted 24 December 2025