Hypergraph Representation Learning with Adaptive Broadcasting and Receiving

Tianyi Ma[♠], Yiyue Qian[♣†], Zheyuan Zhang[♠], Zehong Wang[♠], Shinan Zhang[♣†], Chuxu Zhang[♦], Yanfang Ye[♠*]

[♠]*University of Notre Dame,* [♦]*University of Connecticut,* [♣]*Amazon GenAI*,

tma2@nd.edu, yyqian5@gmail.com, {zzhang42, zwang43}@nd.edu
zhangshinan@gmail.com, chuxu.zhang@uconn.edu, yye7@nd.edu.

***Abstract*—Hypergraphs, in contrast to general graphs, utilize hyperedges to connect multiple nodes, thereby inherently facilitating the representation of higher-order relational structures. To leverage the benefits of hypergraphs, several Hypergraph Neural Networks (HyGNNs) have been proposed to model hypergraph structures. Although existing HyGNNs excel at capturing complex relationships in homophilic hypergraphs, they still face challenges in modeling heterophilic hypergraphs, as most existing HyGNNs are designed based on the homophily principle. Recent studies have attempted to leverage attention mechanisms that are less reliant on the homophily principle. However, these attention mechanisms remain ineffective for nodes in heterophilic hypergraphs. To tackle the aforementioned challenges, we propose a novel Broadcast HyperGraph Neural Network (BHyGNN) to adaptively broadcast node information to learn more effective node representations in heterophilic hypergraphs. Specifically, we devise a novel Variational Broadcast Autoencoder Network to sample the broadcast and receive actions to propagate information between nodes and hyperedges. Moreover, we design an incorporation transformer mechanism to perform the estimated broadcast or receive actions to learn the hyperedge or node representations, incorporating the information from both sides. Extensive experiments over five benchmark heterophilic hypergraph datasets and six homophilic hypergraph datasets demonstrate the effectiveness of BHyGNN over all baseline methods. Our source code and datasets are available at https://github.com/Tianyi-Billy-Ma/BHyGNN.**

***Index Terms*—Heterophily Hypergraph, Hypergraph representation learning, Graph neural network.**

I. Introduction

Recently, several hypergraph neural networks (HyGNNs) [1]–[6] have been proposed to capture the rich and complex structural relationships in hypergraphs effectively. Most existing HyGNNs are based on the homophily principle, i.e., connected nodes belonging to the same class or having similar features [7]–[9]. Specifically, HyGNNs model the principle of homophily using classic message-passing neural networks (MPNNs) [10]–[16], i.e., propagating attribute features and aggregating them to neighbor nodes through various mechanisms. For instance, HyperGCN [11] transforms hypergraphs into homophilic graphs and feeds the node attribute features with homophilic graph structures into GCN [17], to learn the node embeddings. However, there are opposite settings in real-world scenarios, leading to **networks with heterophily**: connected nodes are likely from different classes or have dissimilar attribute features [18]. Node classification in heterophilic hypergraphs is more challenging than that in homophilic hypergraphs [19], [20]. In addition, heterophily has been shown to be a more common phenomenon in hypergraphs than in general graphs as it is hard to expect all nodes in a giant hyperedge to share the same label [21]. For example, in the Walmart hypergraph dataset [22], nodes represent products purchased at Walmart, each hyperedge corresponds to products purchased in a shopping trip, and node labels indicate product categories. Products with similar functions, i.e., identical labels or similar attribute features, are unlikely to be purchased together, e.g., a shopping trip for business travel may only contain either a travel size or a regular size toothpaste, but not both.

Although existing HyGNNs achieve excellent performance in homophilic hypergraphs, these works still face the following limitations in modeling heterophilic hypergraphs: (i) The assumption of high homophily by most HyGNNs limits their ability to accurately model heterophilic or low-homophilic hypergraph structures. In some cases, **MLPs**, which ignore hypergraph structures, may **outperform HyGNNs in modeling heterophilic hypergraphs.** For instance, HGNN [10] leverages a hypergraph convolution operation to propagate node information for representation learning. The propagation process produces a similar representation for nodes connected in the same hyperedge. However, in heterophilic hypergraphs, where connected nodes belong to different classes, HGNN may generate similar representations for nodes with distinct labels, potentially compromising its ability to differentiate between node classes. (ii) A few studies [23]–[25] attempt to address the heterophily issue by employing attention mechanisms on node attribute features, which are less reliant on the homophily principle. However, the attention mechanisms cannot distinguish the information within the same hypergraph structure. For example, AllSet [26] applies multiset transformer operators on node and hyperedge sides for representation learning. Although AllSet alleviates the heterophilic issue with the multiset transformer, it is a permutation-invariant propagation operator and merely

†The work is not related to the position at Amazon.
∗ Corresponding author.

learns the importance of terms on one side (either nodes or hyperedges), which may not effectively distinguish nodes in certain heterophilic cases [19]. Taking the Walmart hypergraph dataset as an example, two products from different classes, toothpaste and a flash drive, purchased on exactly the same three shopping trips, will receive identical information in propagation, resulting in similar node embeddings. Therefore, a research gap exists in designing hypergraph neural networks that can handle the heterophily issue.

To tackle the aforementioned challenges, we propose a novel hypergraph neural network called **B**roadcast **Hy**per**G**raph **N**eural **N**etwork (BHyGNN) that nodes adaptively broadcast information to hyperedges, and further discriminately receive the information from hyperedges. Specifically, to tackle the first challenge, we devise a novel **V**ariational **B**roadcast **A**utoencoder Network (VBA-Net) to estimate node actions (*broadcast* or *receive*) to facilitate propagation mechanism in learning more informative node representation on heterophilic hypergraphs. In particular, we feed the hypergraph with node and hyperedge features into a variational autoencoder to encode the hypergraph into the latent representations and decode the latent representations via the variational distributions to learn the probability of actions. Afterward, we leverage the Straight-through Gumbel-Softmax Estimator to sample the discrete actions for every node in the corresponding hyperedges. To address the second issue, we design an incorporation transformer for learning nodes or hyperedge representations from each other. Unlike AllSet, which merely adopts the information from one side (nodes or hyperedges) to update the target representations (hyperedges or nodes), we incorporate the node information and hyperedge information and employ the incorporated features into the transformer mechanism to update either the node representations or the hyperedge representations. Specifically, with the broadcast actions estimated by VBA-Net, we utilize the incorporation transformer to propagate node information adaptively, incorporated with hyperedge information, to generate the hyperedge embeddings. Afterward, we employ another VBA-Net to sample the receive actions based on node embeddings and updated hyperedge embeddings. Moreover, we utilize an incorporation transformer to execute sampled receive actions on node embeddings, incorporated with updated hyperedge embeddings to obtain the current layer node embeddings. Last, we leverage an MLP as the classifier on final node embeddings for downstream tasks. To conclude, this work makes the following contribution:

- Problem: This work proposes to leverage HyGNNs to tackle the heterophilic issue in hypergraphs.
- Novelty: We design a novel **B**roadcast **Hy**per**G**raph **N**eural **N**etwork (BHyGNN), including a VBA-Net and an incorporation transformer to allow nodes to broadcast or receive information from hyperedges adaptively.
- Effectiveness: Comprehensive experiments over both benchmark heterophilic and homophilic hypergraph datasets demonstrate the effectiveness of our model.

II. Related Work

Hypergraph Neural Networks. Unlike graphs where each edge connects two nodes [13], [14], [17], [27]–[39], hypergraphs allow hyperedges to connect any arbitrary number of nodes, which can represent higher-order relationships [2], [40]. To utilize the advantages of hypergraphs, various Hypergraph Neural Networks (HyGNNs) [7]–[9], [12], [23], [26] have been developed, aiming to model the complex relationships in hypergraphs effectively. Early HyGNNs [10], [11] attempt to employ GNNs to hypergraphs by transforming hypergraphs into graphs and further utilizing GNNs to learn the node representations. Despite the simplicity of these early studies, they usually lead to undesired losses in learning performance [26], [41]. On the other hand, recent works [19], [26] view hypergraphs as bipartite graphs and design multiset propagation rules that generate the hyperedge embeddings from node attribute features and propagate hyperedge embeddings back to nodes as final node embeddings. Inspired by existing HyGNNs, this work proposes a hypergraph propagation mechanism to propagate information between nodes and hyperedges adaptively.

Heterophilic Graph Representation Learning. In recent years, various graph representation learning models have been proposed to address the heterophily settings [42]–[44]. For instance, an early work, Geom-GCN [45], pre-computes unsupervised node embeddings and utilizes geometric relationships in the embedding space to establish a bi-level aggregation process to handle the heterophily. Additionally, a more recent study [46] introduces neural sheaf diffusion models that learn cellular sheaves to address the heterophily issue in graphs. Despite the extensive exploration of graph representation learning in heterophily settings, the domain of representation learning on heterophilic hypergraphs has not yet been thoroughly studied. Motivated by this, this work proposes a novel and effective hypergraph representation learning method to address the heterophily issue of hypergraphs.

III. Preliminary

Definition III.1. Hypergraph. Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes with size N, $\mathcal{E} = \{e_1, \ldots, e_M\}$ is the set of hyperedges with size M, and $\mathcal{X}$ is the attribute feature set. A hypergraph can be represented by an incidence matrix $\mathbf{M} \in \mathbb{R}^{N \times M}$, where $\mathbf{M}_{i,j} = 1$ if $v_i \in e_j$; otherwise, $\mathbf{M}_{i,j} = 0$. Besides, we use $d(v_i) = \sum_{e_j \in \mathcal{E}} \mathbf{M}_{i,j}$ and $d(e_j) = \sum_{v_i \in \mathcal{V}} \mathbf{M}_{i,j}$ to denote the degrees of node and hyperedge, respectively.

Definition III.2. Hypergraph Homophily. Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ with a set of node classes $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$, the homophily score of the hypergraph $\mathcal{H}$ is measured from node and hyperedge aspects. The homophily score for node v is computed as $h(v) = |\{u : u \in \mathcal{N}_v \wedge y_v = y_u\}|/|\mathcal{N}_v|$, where y_v is the label of node v, and $\mathcal{N}_v$ is the neighbors of node v. The homophily score of a hyperedge e is the maximum ratio of nodes in hyperedge e that have the same class label: $h(e) = \max_{c \in \mathcal{C}} |\{v : v \in e \wedge y_v = c\}|/|e|$. We refer to the class that emits the maximum ratio as the majority class of

hyperedge e, and the rest of the classes are the minority classes of hyperedge e.

Problem Definition. ***Heterophilic Hypergraph Neural Network.*** *Given a heterophilic hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ with ground-truth label $\mathbf{Y}$, the objective is to build a hypergraph neural network model $f : \mathcal{V} \rightarrow \mathbb{R}^d$ to project nodes into d-dimensional embeddings for downstream tasks with $\mathbf{Y}$.*

IV. Methodology

In this section, we present the details of **B**roadcast **Hy**perGraph **N**eural **N**etwork (BHyGNN) , which is designed to handle the heterophily issue in hypergraphs through the following two key stages: (i) VBA-Net to sample the broadcast and receive actions; (ii) Incorporation transformer to execute the broadcast or receive actions for node representations. Our model propagates node information layer-wise, and we denote the previous layer representation as $\mathbf{Z}'$ and the current layer updated representation as $\mathbf{Z}$. The following sections will elaborate on the implementation details of these two stages within a single layer, as illustrated in Figure 1.

A. VBA-Net for Action Learning

Previous works [10]–[12] primarily follow the homophily principle, where nodes within a hyperedge share identical labels or similar attribute features. However, this strategy encounters limitations on heterophilic hypergraphs, where nodes linked by a hyperedge often fall into different classes or have dissimilar attribute features [19], [26]. This limitation highlights a crucial gap in the current hypergraph neural network landscape.

In light of this limitation, we propose a novel **V**ariational **B**roadcast **A**utoencoder **Net**work (VBA-Net) for heterophilic hypergraphs that enables adaptive information exchange, i.e., nodes adaptively broadcast and receive valuable information, optimizing representation learning in heterophilic hypergraphs. Take a social media platform as an example: nodes represent users, and hyperedges represent topics users can follow. Users have the option to *Broadcast* their viewpoints or *Receive* updates on each topic they follow. Node and hyperedge attribute features correspond to user viewpoints and topic updates, influenced by user-shared viewpoints, respectively. Under time constraints, users prioritize topics aligned with their interests, as indicated by similar features. Therefore, to accurately characterize a user, we integrate all the topic updates from the user who receives updates as the corresponding node representation. Following this idea, a straightforward implementation to estimate broadcast and receive actions involves employing a similarity function to compute the scores between nodes and hyperedges. Subsequently, a node v broadcasts to a hyperedge e if hyperedge e includes node v and their score exceeds a predefined threshold, same for estimating the receiving actions.

However, given the complexity of setting the optimal thresholds for actions, which requires domain-specific knowledge and may vary across node-hyperedge pairs, we advocate for a more flexible and adaptable approach for action estimation. Inspired by the well-studied variational inference generative models [47]–[49], we propose to leverage the variational autoencoder to estimate action a from two distinct action types: **B**roadcast and **R**eceive.

B. Variational Broadcast Autoencoder Network

VBA-Net contains an encoder neural network to encode the hypergraph into latent representations and a decoder neural network for action estimation. Considering the distinct behaviors associated with the two action types, we employ dual VBA-Nets, denoted as $g^{(\mathbf{B})}(\cdot)$ and $g^{(\mathbf{R})}(\cdot)$, to learn the broadcast and receive actions, separately. In each VBA-Net $g(\cdot)$, the encoder aims to learn the node latent representations $\mathbf{H}_{\mathcal{V}} \in \mathbb{R}^{N\times d}$ and hyperedge latent representations $\mathbf{H}_{\mathcal{E}} \in \mathbb{R}^{M\times d}$ via the variational distribution $\mathbf{H}_{\mathcal{V}} \sim q(\mathbf{H}_{\mathcal{V}}|\mathcal{H})$ and $\mathbf{H}_{\mathcal{E}} \sim q(\mathbf{H}_{\mathcal{E}}|\mathcal{H})$, where d is the latent representation dimension, N and M are the size of nodes and hyperedges, respectively. Unlike previous works [20], [49], which adopt a separate graph or hypergraph encoder to learn the variational distributions, we directly feed the node embeddings $\mathbf{Z}'_{\mathcal{V}} \in \mathbb{R}^{N\times d}$ and hyperedge embeddings $\mathbf{Z}'_{\mathcal{E}} \in \mathbb{R}^{M\times d}$ into VBA-Net $g^{(\mathbf{B})}(\mathbf{Z}'_{\mathcal{V}}, \mathbf{Z}'_{\mathcal{E}}, \mathcal{H})$ to encode the hypergraph. It is noteworthy that using identical node embeddings and hyperedge embeddings as inputs for both $g^{(\mathbf{B})}(\cdot)$ and $g^{(\mathbf{R})}(\cdot)$ might result in similar variational distributions for the broadcast and receive actions, hindering the learning of adaptive behaviors for heterophilic hypergraphs. Therefore, instead of the identical embedding inputs for $g^{(\mathbf{B})}(\cdot)$ and $g^{(\mathbf{R})}(\cdot)$, we first apply broadcast actions through the incorporation transformer (discussed later) to update the hyperedge embeddings $\mathbf{Z}_{\mathcal{E}}$. Subsequently, these along with the original node embedding $\mathbf{Z}'_{\mathcal{V}}$ are fed into VBA-Net $g^{(\mathbf{R})}(\mathbf{Z}'_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}}, \mathcal{H})$ to sample the receive actions.

Encoder. In each VBA-Net $g(\mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}}, \mathcal{H})$[1], we consider the normal variational posterior for latent representations, $\mathbf{H}_{\mathcal{V}} \sim q(\mathbf{H}_{\mathcal{V}}|\mathcal{H}) = \mathcal{N}(\mu_{\mathcal{V}}, \sigma^2_{\mathcal{V}})$ and $\mathbf{H}_{\mathcal{E}} \sim q(\mathbf{H}_{\mathcal{E}}|\mathcal{H}) = \mathcal{N}(\mu_{\mathcal{E}}, \sigma^2_{\mathcal{E}})$, where $\mu_{\mathcal{V}} \in \mathbb{R}^{N\times d}$ and $\sigma_{\mathcal{V}} \in \mathbb{R}^{N\times d}$ represent the Gaussian mean and standard deviation for nodes, respectively. Similarly, $\mu_{\mathcal{E}} \in \mathbb{R}^{M\times d}$ and $\sigma_{\mathcal{E}} \in \mathbb{R}^{M\times d}$ denote the Gaussian mean and standard deviation for hyperedges, respectively. We employ Multilayer Perceptrons (MLP) to parameterize the mean μ_* and the standard deviation σ_*:

$$\begin{aligned} \mu_{\mathcal{V}} &= \mathrm{MLP}^{(\mu)}_{\mathcal{V}}(\mathbf{Z}_{\mathcal{V}}), \log \sigma_{\mathcal{V}} = \mathrm{MLP}^{(\sigma)}_{\mathcal{V}}(\mathbf{Z}_{\mathcal{V}}), \\ \mu_{\mathcal{E}} &= \mathrm{MLP}^{(\mu)}_{\mathcal{E}}(\mathbf{Z}_{\mathcal{E}}), \log \sigma_{\mathcal{E}} = \mathrm{MLP}^{(\sigma)}_{\mathcal{E}}(\mathbf{Z}_{\mathcal{E}}). \end{aligned} \tag{1}$$

Decoder. After obtaining the node and hyperedge variational distributions, the decoder aims to estimate actions that guide our propagation mechanism to generate effective node and hyperedge embeddings in heterophilic hypergraphs. Specifically, we leverage the reparameterization trick [48], [49] to encode the nodes and hyperedges into the latent space:

$$\mathbf{H}_{\mathcal{V}} = \rho_{\mathcal{V}} \odot \sigma_{\mathcal{V}} + \mu_{\mathcal{V}}, \text{ and } \mathbf{H}_{\mathcal{E}} = \rho_{\mathcal{E}} \odot \sigma_{\mathcal{E}} + \mu_{\mathcal{E}}, \tag{2}$$

where $\odot$ denotes the element-wise product, $\rho_{\mathcal{V}} \sim \mathcal{N}(0, I)$ and $\rho_{\mathcal{E}} \sim \mathcal{N}(0, I)$ are the standard normal variables for nodes

[1]For simplicity, we omit the superscript in the following sections unless explicitly needed.

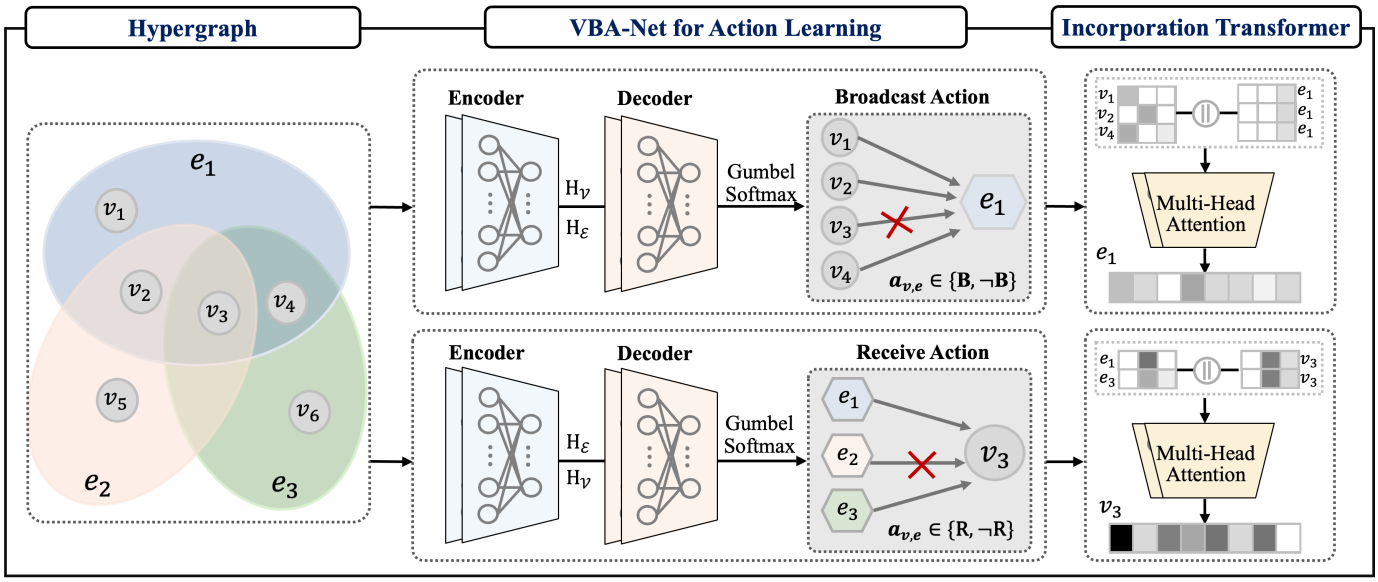

Fig. 1: The overall framework of BHyGNN: (i) VBA-Net encodes the hypergraph $\mathcal{H}$ into the latent representations $(\mathbf{H}_\mathcal{V}, \mathbf{H}_\mathcal{E})$ via variational distributions; (ii) VBA-Net estimates the probability of broadcast actions and further employs the Gumbel-Softmax to sample the discrete actions $a_{v,e}$ for each node-hyperedge pair. (iii) The incorporation transformer performs broadcast actions to learn hyperedge embeddings. (iv) With the updated hyperedge embeddings, we employ another VBA-Net to estimate the receiving actions. (v) Lastly, we adopt an incorporation transformer to execute receiving actions to obtain the node embeddings.

$\mathcal{V}$ and hyperedges $\mathcal{E}$, respectively. Afterward, the logit of an action $a_{v,e}$ is derived from the inner product between the node latent representation $\mathbf{H}_v \in \mathbb{R}^{1\times d}$ and hyperedge latent representation $\mathbf{H}_e \in \mathbb{R}^{1\times d}$. Besides, we utilize the sigmoid function to calculate the probability of action $a_{v,e}$, i.e., $P(a_{v,e}|\mathbf{H}_v, \mathbf{H}_e) = \text{Sigmoid}(\mathbf{H}_v^T \mathbf{H}_e)$.

To bridge the gap between discrete action sampling and action probability, thereby ensuring the entire process is differentiable, we employ the Straight-through Gumbel-Softmax Estimator [50], [51] to reparameterize the action distribution P, allowing for the computation of the action $a_{v,e}$ as follows:

$$\begin{aligned} a_{v,e} &= \text{Gumbel-Softmax}(p) \\ &= \frac{\exp((\log p + \epsilon)/\tau)}{\exp((\log p + \epsilon)/\tau) + \exp((\log(1-p) + (1-\epsilon))/\tau)}, \end{aligned} \tag{3}$$

where p denotes $P(a_{v,e}|\mathbf{H}_v, \mathbf{H}_e)$, $\epsilon \in \mathbb{R}$ is a Gumbel value, i.e., $\epsilon \sim \text{Gumbel}(0, 1)$, and τ is a temperature parameter. For forward propagation, we adopt discrete sampling, i.e., $a_{v,e} \in \{\mathbf{B}, \neg\mathbf{B}\}$. The computation of $a_{v,e}$ in Eq. 3 is exclusive for optimization purposes.

Action Learning. To estimate broadcast actions at the current layer, we feed the node embeddings $\mathbf{Z}'_\mathcal{V}$ and hyperedge embeddings $\mathbf{Z}'_\mathcal{E}$ from the previous layer into the VBA-Net $g^{(\mathbf{B})}(\mathbf{Z}'_\mathcal{V}, \mathbf{Z}'_\mathcal{E}, \mathcal{H})$, resulting in broadcast actions $\mathcal{B} = \{a^{(\mathbf{B})}_{v,e} : e \in \mathcal{E}, v \in e, a^{(\mathbf{B})}_{v,e} \in \{\mathbf{B}, \neg\mathbf{B}\}\}$. Then, we apply the broadcast actions $\mathcal{B}$ using the incorporation transformer to learn the current layer hyperedge embeddings $\mathbf{Z}_\mathcal{E}$. Afterward, we feed the node embeddings $\mathbf{Z}'_\mathcal{V}$ and the current layer hyperedge embeddings $\mathbf{Z}_\mathcal{E}$ into VBA-Net $g^{(\mathbf{R})}(\mathbf{Z}'_\mathcal{V}, \mathbf{Z}_\mathcal{E}, \mathcal{H})$ to sample a set of receive actions $\mathcal{R} = \{a^{(\mathbf{R})}_{v,e} : e \in \mathcal{E}, v \in e, a^{(\mathbf{R})}_{v,e} \in \{\mathbf{R}, \neg\mathbf{R}\}\}$.

VBA-Net Optimization. With the variational inference, the optimization of VBA-Nets $g^{(\mathbf{B})}(\cdot)$ and $g^{(\mathbf{R})}(\cdot)$ is achieved through the variational lower bound $\mathcal{L}^{(\mathbf{B})}_{\text{vlb}}$ and $\mathcal{L}^{(\mathbf{R})}_{\text{vlb}}$, respectively. These are computed as follows:

$$\begin{aligned} \mathcal{L}^{(*)}_{\text{vlb}} &= \mathbb{E}_{q(\mathbf{H}_\mathcal{V}|\mathcal{H})}\mathbb{E}_{q(\mathbf{H}_\mathcal{E}|\mathcal{H})}[\log P(*|\mathbf{H}_\mathcal{V}, \mathbf{H}_\mathcal{E})] \\ &- D_{\text{KL}}[q(\mathbf{H}_\mathcal{V}|\mathcal{H})|p(\mathbf{H}_\mathcal{V})] - D_{\text{KL}}[q(\mathbf{H}_\mathcal{E}|\mathcal{H})|p(\mathbf{H}_\mathcal{E})], \end{aligned} \tag{4}$$

where $\mathcal{L}^{(*)}_{\text{vlb}}$ is the variational lower bound for either broadcast actions $\mathcal{L}^{(\mathbf{B})}_{\text{vlb}}$ or receive actions $\mathcal{L}^{(\mathbf{R})}_{\text{vlb}}$, and $P(*|\mathbf{H}_\mathcal{V}, \mathbf{H}_\mathcal{E})$ denotes the probability of either broadcast actions $P(\mathcal{B}|\mathbf{H}_\mathcal{V}, \mathbf{H}_\mathcal{E})$ or receive actions $P(\mathcal{R}|\mathbf{H}_\mathcal{V}, \mathbf{H}_\mathcal{E})$, w.r.t $\mathcal{L}^{(\mathbf{B})}_{\text{vlb}}$ or $\mathcal{L}^{(\mathbf{R})}_{\text{vlb}}$. Here, $D_{\text{KL}}[q(\cdot)|p(\cdot)]$ is the Kullback-Leibler divergence between variational posterior $q(\cdot)$ and prior $p(\cdot)$, with $p(\cdot) \sim \mathcal{N}(0, I)$ as the default Gaussian prior. For simplicity, superscripts indicating specific actions ($\mathbf{B}$ or $\mathbf{R}$) for $\mathbf{H}_\mathcal{V}$ and $\mathbf{H}_\mathcal{E}$ are omitted. Besides, to ensure adequate information propagation between nodes and hyperedges in accordance with the hypergraph structure, we introduce L_1 norm regularizers for broadcast and receive actions, denoted as $\mathcal{L}^{(\mathbf{B})}_{reg} = ||\lambda\mathbf{M} - \hat{\mathbf{M}}^{(\mathbf{B})}||$ and $\mathcal{L}^{(\mathbf{R})}_{reg} = ||\lambda\mathbf{M} - \hat{\mathbf{M}}^{(\mathbf{R})}||$, respectively. Here λ is a hyper-parameter, $\mathbf{M}$ represents the incidence matrix of the hypergraph $\mathcal{H}$, $\hat{\mathbf{M}}^{(\mathbf{B})}$ and $\hat{\mathbf{M}}^{(\mathbf{R})}$ are the incidence matrices of the broadcast actions $\mathcal{B}$ and the receive actions $\mathcal{R}$, respectively. Therefore, the total variational loss for the current layer is formulated as follows:

$$\mathcal{L}_{\text{var}} = \mathcal{L}^{(\mathbf{B})}_{\text{vlb}} + \mathcal{L}^{(\mathbf{B})}_{\text{reg}} + \mathcal{L}^{(\mathbf{R})}_{\text{vlb}} + \mathcal{L}^{(\mathbf{R})}_{\text{reg}}. \tag{5}$$

Next, we introduce our novel propagation mechanism to perform the sampled actions.

TABLE I: Performance comparison (Mean accuracy % ± std) of baseline methods for node classification on heterophilic hypergraph datasets. Bolded numbers indicate the best result and underlined numbers represent the runner-up performance.

	SENATE		SYNTHETIC		CONGRESS		HOUSE		WALMART	
# Nodes	282	282	2,000	2,000	1,718	1,718	1,290	1,290	88,860	88,860
# Edges	315	315	1,000	1,000	83,105	83,105	340	340	69,906	69,906
Avg. $h(v)$	0.50	0.50	0.28	0.28	0.55	0.55	0.51	0.51	0.53	0.53
Avg. $h(e)$	0.55	0.55	0.33	0.33	0.79	0.79	0.58	0.58	0.75	0.75
Noise std σ	0.6	1.0	0.6	1.0	0.6	1.0	0.6	1.0	0.6	1.0
MLP [52]	62.24±6.39	50.35±3.25	50.00±1.19	37.48±0.65	79.45±1.18	65.91±1.55	77.12±3.00	64.07±2.70	62.23±0.14	44.69±0.13
HGNN [10]	60.06±2.81	49.35±2.64	42.42±1.85	37.90±1.54	90.91±0.77	88.08±1.37	61.24±1.72	57.24±1.76	77.19±0.12	61.34±0.19
HyperGCN [11]	55.00±3.02	51.82±2.49	41.61±2.03	32.51±1.24	84.81±1.56	83.32±1.18	75.62±2.03	62.43±2.68	62.02±0.67	49.61±0.42
HNHN [12]	62.18±6.99	54.71±4.15	49.67±1.45	37.11±0.93	89.71±1.17	82.97±1.66	68.36±2.21	65.16±1.88	68.68±0.95	58.01±0.46
HCHA [23]	47.71±1.37	46.20±2.84	32.50±1.73	27.18±1.44	91.04±0.64	89.81±0.91	61.28±1.54	56.98±1.42	76.55±0.15	61.83±0.19
UniGCNII [53]	60.06±5.16	52.24±4.17	49.62±1.37	37.13±1.40	92.91±1.04	89.56±6.48	78.64±1.29	65.45±1.36	72.36±1.26	63.72±0.52
AllSet [26]	65.47±3.42	51.76±4.60	52.84±0.80	42.78±0.81	93.65±1.29	88.65±3.84	78.81±1.51	65.20±1.58	78.74±0.25	65.35±0.25
ED-HNN [19]	65.53±3.10	55.47±4.87	55.96±1.34	43.59±1.54	94.20±0.98	92.07±0.75	79.01±1.00	65.70±1.98	78.15±0.42	65.07±0.84
HyGCL-ADT [54]	64.85±3.37	52.93±2.94	55.92±2.16	42.31±1.63	93.10±2.37	91.86±2.49	79.43±1.30	64.72±1.92	78.41±0.51	65.34±0.86
SheafHGNN [55]	64.35±4.72	54.32±4.26	55.42±3.82	43.97±2.73	90.72±2.41	91.07±2.97	79.75±1.84	65.93±2.30	OOM	OOM
HypeBoy [56]	63.47±3.62	53.17±3.06	53.74±1.85	42.26±2.39	92.34±1.24	90.52±2.58	79.32±1.73	65.35±2.94	76.42±0.59	64.28±0.95
Ours	**67.87**±2.13	**58.41**±3.11	**58.10**±1.44	**47.32**±2.29	**95.45**±0.62	**93.72**±0.86	**80.23**±1.31	**67.36**±2.14	**79.94**±1.31	**66.85**±1.04

C. Incorporation Transformer

A few studies [23], [26] have attempted to alleviate the heterophily issue in hypergraphs by attention-based mechanisms. However, these attention mechanisms are not effective in certain heterophilic cases [19]. For example, AllSet [26] merely propagates the information from one side to the other side in each step, making it difficult to distinguish nodes in the same hyperedges. Given two nodes v_1 and v_2, from distinct classes within the same hyperedges, i.e., $\{e, \forall e \in \mathcal{E}, v_1 \in e\} = \{e, \forall e \in \mathcal{E}, v_2 \in e\}$, they will have identical node representations from shared hyperedges. To address this issue, we design an incorporation transformer mechanism that propagates current side (node or hyperedge) information, incorporated with the previous information from the opposite side (hyperedge or node), to learn the representations for entities on the opposite side.

Representation Learning over Actions. With the broadcast actions $\mathcal{B}$ and receive actions $\mathcal{R}$ sampled from VBA-Net, we utilize an incorporation transformer to update the current layer hyperedge embeddings $\mathbf{Z}_{\mathcal{E}}$ and node embeddings $\mathbf{Z}_{\mathcal{V}}$, which is formulated as follows:

$$\mathbf{Z}_e = \mathrm{MH}(\mathbf{S}_e^{(\mathbf{B})}; \theta_e) \text{ and } \mathbf{Z}_v = \mathrm{MH}(\mathbf{S}_v^{(\mathbf{R})}; \theta_v), \tag{6}$$

where, $\mathbf{Z}_e \in \mathbb{R}^b$ and $\mathbf{Z}_v \in \mathbb{R}^b$ are current layer embeddings for hyperedge e and node v, respectively. $\mathbf{S}_e^{(\mathbf{B})} \in \mathbb{R}^{c_e \times d}$ and $\mathbf{S}_v^{(\mathbf{R})} \in \mathbb{R}^{c_v \times d}$ are the stacked incorporation embeddings $\mathcal{Z}_e^{(\mathbf{B})}$ and $\mathcal{Z}_v^{(\mathbf{R})}$ through mappings, respectively. Mathematically:

$$\mathcal{Z}_e^{(\mathbf{B})} = \{\mathbf{Z}_v' || \mathbf{Z}_e' : a_{v,e}^{(\mathbf{B})} = \mathbf{B}\}, c_e = |\mathcal{Z}_e^{(\mathbf{B})}|, \tag{7}$$

$$\mathcal{Z}_v^{(\mathbf{R})} = \{\mathbf{Z}_e || \mathbf{Z}_v' : a_{v,e}^{(\mathbf{R})} = \mathbf{R}\}, c_v = |\mathcal{Z}_v^{(\mathbf{R})}|. \tag{8}$$

Here, $||$ denotes the concatenation operation. Mention that we first leverage VBA-Net to obtain the broadcast actions $\mathcal{B}$, and then apply the incorporation transformer to update the hyperedge embeddings $\mathbf{Z}_{\mathcal{E}}$. Afterward, we employ another VBA-Net to sample the receive actions $\mathcal{R}$ and utilize the incorporation transformer to update the node embeddings $\mathbf{Z}_{\mathcal{V}}$. Besides, $\mathrm{MH}(\cdot)$ is a multi-head attention parameterized by $\theta \in \mathbb{R}^{1 \times hd'}$, which is formulated as follows:

$$\mathrm{MH}(\mathbf{S}, \theta) = ||_{i=1}^{h} \mathbf{O}^{(i)}, \mathbf{O}^{(i)} = \omega(\theta^{(i)}(\mathbf{K}^{(i)})^\top)\mathbf{V}, \tag{9}$$

where h denotes the number of heads, $\mathbf{O}^{(i)} \in \mathbb{R}^{1 \times hd'}$, ω is an activation function, $\theta^{(i)} \in \mathbb{R}^{1 \times d'}$, $\theta = ||_{i=1}^{h} \theta^{(i)}$, $\mathbf{K}^{(i)} \in \mathbb{R}^{c_* \times d'}$ and $\mathbf{V}^{(i)} \in \mathbb{R}^{c_* \times d'}$ are the key and value vectors, respectively. d' is the hidden dimension of $\mathbf{V}^{(i)}$, and c_* is either c_e for $\mathbf{S}_e^{(\mathbf{B})}$ or c_v for $\mathbf{S}_e^{(\mathbf{B})}$. Besides, we adopt two MLPs over $\mathbf{S}$ to obtain the key and value vectors, i.e., $\mathbf{K}^{(i)} = \mathrm{MLP}_K^{(i)}(\mathbf{S})$ and $\mathbf{V}^{(i)} = \mathrm{MLP}_V^{(i)}(\mathbf{S})$.

Model Optimization. In this work, we adopt node classification to evaluate the effectiveness of our designed method. We feed the final node embeddings $\mathbf{Z}_{\mathcal{V}}$ into two-layer MLPs as the classifier for the node classification task and adopt cross-entropy loss $\mathcal{L}_{\text{ce}}$ as the node classification loss. The final objective function is $\mathcal{L} = \mathcal{L}_{\text{ce}} + \alpha \sum_{l=1}^{L} \mathcal{L}_{\text{var}}^{(l)}$, where α is a trade-off hyper-parameter, and L is the number of layers.

V. EXPERIMENTS

A. Experimental Setup

Benchmark Datasets. To evaluate the effectiveness of our model, we employ benchmark heterophilic hypergraphs, i.e., Senate [57], Congress [58], House [57], and Walmart [22], and homophilic hypergraphs, including Twitter [1] and citation datasets [11] (Citeseer, DBLP, Cora, Cora-CA, and Pubmed). All heterophilic hypergraphs do not contain node attribute features. Following existing works [19], [26], we utilize a label-dependent Gaussian distribution [59] with added Gaussian noise $\mathcal{N}(0, \sigma^2)$ to create node features and choose 0.6 and 1.0 as the Gaussian noise standard deviations. Moreover, we follow ED-HGNN and SheafHGNN [19], [55] to generate synthetic heterophilic hypergraphs, denoted as SYNTHETIC. To further examine the effectiveness of our model on heterophilic settings, we utilize the hyperedge augmentation [20] to reduce

TABLE II: Performance comparison (Mean accuracy % ± std) of baseline methods for node classification on homophilic hypergraph datasets. Bolded numbers indicate the best result, and underlined numbers represent the runner-up performance.

	TWITTER		CITESEER		DBLP		CORA		CORA-CA		PUBMED	
# Nodes	2,936	2,936	3,312	3,312	41,302	41,302	2,708	2,708	2,708	2,708	19,717	19,717
# Edges	35,502	35,502	1,070	1,079	22,363	22,363	1,072	1,072	1,579	1,579	7,963	7,963
Avg. $h(v)$	0.41	0.82	0.42	0.83	0.49	0.87	0.40	0.80	0.45	0.90	0.48	0.95
Avg. $h(e)$	0.45	0.90	0.42	0.83	0.47	0.93	0.44	0.88	0.43	0.86	0.44	0.88
Type	HET.	HOMO.	HET.	HOMO.	HET.	HOMO.	HET.	HOMO.	HET.	HOMO.	HET.	HOMO.
MLP [52]	67.38±0.64	67.38±0.64	68.05±1.17	68.05±1.17	83.76±0.19	83.76±0.19	68.45±0.78	68.45±0.78	69.45±0.97	69.45±0.97	84.45±0.31	84.45±0.31
HGNN [10]	67.17±0.73	68.52±1.16	55.10±0.79	69.32±0.50	68.53±0.19	90.38±0.18	58.64±0.85	76.11±0.85	62.94±0.97	79.42±0.80	80.91±0.30	85.82±1.08
HyperGCN [11]	55.68±3.61	69.29±0.62	56.87±1.32	69.13±1.42	65.80±4.31	88.37±0.20	56.41±1.79	73.89±1.16	56.59±1.77	75.12±1.40	64.96±2.12	86.31±3.52
HNHN [12]	62.08±2.33	67.78±0.80	66.65±0.76	68.36±1.24	81.76±0.32	86.42±0.20	62.79±1.18	71.52±1.47	64.69±1.91	72.12±1.36	83.62±0.33	85.92±0.60
HCHA [23]	67.63±1.17	72.04±0.66	53.33±0.88	68.84±1.12	67.91±0.23	90.27±0.19	55.71±1.21	75.97±0.90	63.27±1.05	79.23±0.52	75.87±0.32	83.53±0.35
UniGCNII [53]	67.66±0.93	69.57±1.39	60.38±1.21	70.21±0.97	81.80±0.30	90.53±0.17	58.08±1.44	76.25±1.53	61.66±1.70	78.20±1.64	85.24±0.38	86.31±0.20
AllSet [26]	69.16±0.97	70.64±1.16	67.91±2.17	69.71±1.00	84.06±0.21	90.69±0.12	67.54±0.91	74.26±0.76	69.90±1.19	78.30±1.50	85.04±0.36	86.38±0.37
ED-HNN [19]	69.07±1.07	73.47±0.63	66.30±0.75	69.82±2.77	83.90±0.25	90.85±0.21	67.13±1.16	75.78±1.26	70.04±1.78	79.88±0.61	86.04±0.46	87.26±0.24
HyGCL-ADT [54]	70.08±1.28	72.46±2.56	66.51±1.83	**71.03±2.74**	83.76±0.69	90.56±0.26	67.01±1.77	76.35±0.82	69.72±2.41	80.52±1.52	85.73±1.05	86.89±0.51
SheafHGNN [55]	68.95±1.43	72.87±1.90	66.16±2.86	70.57±1.98	82.17±0.46	90.29±0.27	67.42±2.38	76.19±1.61	69.82±0.98	**81.14±2.38**	84.16±1.20	85.41±0.68
HypeBoy [56]	68.77±1.41	71.36±1.29	67.05±1.42	70.98±1.94	81.72±1.94	**91.26±0.84**	66.82±2.97	**76.40±1.37**	68.39±1.49	80.04±3.52	85.29±2.85	87.94±1.01
Ours	**72.86±0.94**	**75.61±0.76**	**68.95±1.60**	70.12±1.12	**85.05±0.25**	90.93±0.31	**69.26±1.04**	75.87±0.70	**70.99±1.55**	80.72±0.58	**86.90±0.39**	**88.02±0.53**

the homophily in Twitter and citation datasets. Specifically, for each hyperedge e, we first identify the majority class c^+. Afterward, we randomly add $|e|$ nodes from nodes that are not in the original hyperedge e and do not belong to class c^+ to the hyperedge e, leading to a heterophilic hypergraph.

Baseline Methods. We adopt MLP and ten HyGNN-based models as baseline methods, including HGNN [10], HyperGCN [11], HNHN [12], HCHA [23], UniGCNII [53], AllSet [26], ED-HNN [19], HyGCL-ADT [54], SheafHGNN [55], and HypeBoy [56]. We strictly follow the settings of baseline methods from their source code to reproduce the experimental results.

Experimental Settings. We leverage accuracy as the evaluation metric to evaluate the performance of our model and baseline methods. To conduct experiments in challenging and practical settings, we split the data into training/validation/test samples using 20%/20%/60%. Moreover, we run each method ten times with 500 epochs and report the average score with standard deviation (std). All experiments are conducted under the environment of Ubuntu 22.04.3 OS plus an Intel i9-12900K CPU, four Nvidia A40 graphics cards, and 48 GB of RAM. We utilize Adam as the optimizer and run a grid search on hyperparameters for each model to obtain the best performance. We define the specific ranges to find the optimal hyperparameters for every model. For instance, the range of hidden dimensions for layers is $\{64, 128, 256, 512\}$, the range of weight decays is $\{0.0, 0.01, 0.001\}$, and the range of learning rate is $\{0.1, 0.01, 0.001, 0.0001\}$.

B. Result Analysis

Heterophilic Hypergraph. Table I shows the performance of all models over five heterophilic hypergraphs with node attribute features generated via different Gaussian noise standard deviations, i.e., 0.6 and 1.0. Mention that OOM indicates that the model encounters out-of-memory issues. According to Table I, we have the following conclusions: (i) MLP that ignores hypergraph structures can outperform several HyGNNs in heterophilic hypergraphs. This finding implies that the heterophilic hypergraph structures may negatively affect the performance of HyGNNs, which is the motivation of this work. (ii) Attention-based methods, HCHA and AllSet, demonstrate considerable performance in most datasets, but in some datasets, they can fail to generate effective node embeddings, i.e., HCHA fails to outperform feature-based model MLP in Senate, Synthetic, and House datasets. (iii) The top-performing baseline models are AllSet and ED-HNN. Both utilize an invariant set function to propagate node features to hyperedges and aggregate back from hyperedges to nodes. This indicates the importance of information from the other side in the propagation. (iv) Our model outperforms every baseline model over all benchmark heterophilic hypergraph datasets, which demonstrates the strong effectiveness of our model in heterophilic hypergraphs for node classification tasks.

Homophilic Hypergraphs. To further evaluate the effectiveness of our model, we compare baseline models with BHyGNN over benchmark homophilic hypergraph datasets. Besides, inspired by the hypergraph augmentation methods [20], we augment the benchmark homophilic hypergraph datasets to simulate the heterophily settings and report the performance of baseline methods and our models. Table II lists the performance of baseline methods and BHyGNN under the aforementioned two settings, i.e., Het. and Homo. Based on the result, we make the following conclusions: (i) Feature-based method MLP performs better than at least three HyGNNs in each augmented dataset. Besides, MLP achieves the best performance among all baseline methods in the augmented Citeseer and Cora datasets. These findings again verify that heterophilic hypergraph structures can negatively affect HyGNNs. (ii) Our model outperforms all baseline in augmented benchmark datasets and has comparable performance in original homophilic datasets.

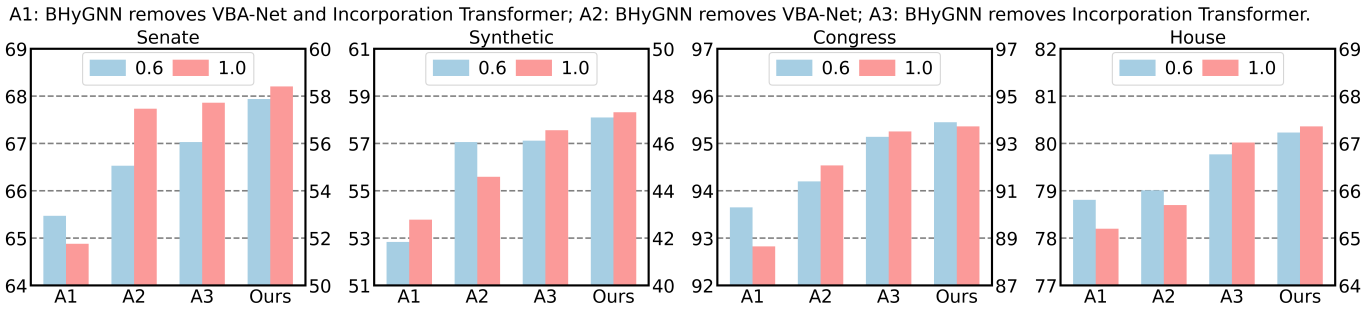

Fig. 2: Performance of different model variants over Senate, Synthetic, Congress, and House benchmark hypergraph datasets.

TABLE III: Performance comparison over HyGNNs for node classification in setting $50\%25\%25\%$.

Model	CONGRESS (1.0)	WALMART (1.0)	TWITTER (HOMO.)	Cora (HOMO.)
AllSet	92.16	65.46	73.62	78.58
ED-HNN	95.00	66.91	75.10	80.31
SheafHGNN	91.81	OOM	74.21	**81.30**
HyGCL-ADT	93.83	66.48	74.02	79.28
Ours	**96.41**	**68.82**	**77.76**	81.02

This again demonstrates its superiority in both settings.

Additional Experiments. Following existing studies [26], [55], we conduct additional experiments over setting $50\%/25\%/25\%$ for training, validation, and testing, respectively, and the result is provided in Table III. According to the table, we find that our proposed model consistently outperforms SOTA HyGNNs in heterophilic hypergraphs, i.e., Congress and Walmart, and has comparable performance in homophilic hypergraphs, which again indicates the effectiveness of our proposed model.

C. Ablation Study

We conduct ablation studies to analyze the contributions of different components on four benchmark hypergraph datasets for node classification. Specifically, we remove VBA-Net and transformer function (A1), VBA-Net (A2), and transformer function (A3), respectively, as illustrated in Figure 2. First, we remove the VBA-Net and transformer function (A1), which means we employ AllSet to obtain the node embeddings for node classification. We conclude that BHyGNN is effective as the performance of A1 drops significantly on all benchmark hypergraph datasets. Afterward, we remove the VBA-Net from our model (A2), which means we directly feed the hypergraph into the transformer function for representation learning. The decreased performance of A2 shows the effectiveness of the VBA-Net module. Moreover, we remove the transformer function from our model (A3), which means we merely employ the VBA-Net to sample actions and feed the hypergraph with actions into AllSet to obtain the node embeddings. The decline of A3 demonstrates that the transformer function has contributed to our model.

D. Hyperparameter Analysis

We conduct experiment to evaluate the rationality of our model with respect to the hyperparameter λ, as shown in Table IV. According to the results, the performance with different λ values is related to the heterophilic level of the hypergraph dataset. Specifically, for heterophilic hypergraphs, i.e., Senate and Synthetic, the best performance is achieved when λ is 0.5 and 0.3, respectively. For homophilic hypergraphs, i.e., Twitter and Citeseer, the best performance is obtained when $\lambda = 0.9$. This result empirically verifies the rationality of the hyperparameter λ in our model.

TABLE IV: Performance of BHyGNN with different λ values over heterophilic and homophilic hypergraphs.

λ	SENATE (1.0)	SYNTHETIC (1.0)	TWITTER (Homo.)	CITESEER (Homo.)
0.1	$55.72_{\pm 2.49}$	$46.72_{\pm 1.96}$	$70.94_{\pm 1.74}$	$68.73_{\pm 1.26}$
0.3	$57.93_{\pm 2.15}$	$\mathbf{47.32}_{\pm \mathbf{2.29}}$	$72.43_{\pm 1.60}$	$69.17_{\pm 0.94}$
0.5	$\mathbf{58.41}_{\pm \mathbf{3.11}}$	$46.69_{\pm 2.31}$	$73.14_{\pm 1.38}$	$69.63_{\pm 1.28}$
0.7	$57.82_{\pm 2.48}$	$46.04_{\pm 2.45}$	$74.92_{\pm 0.77}$	$69.95_{\pm 1.35}$
0.9	$57.48_{\pm 2.01}$	$44.97_{\pm 1.93}$	$\mathbf{75.61}_{\pm \mathbf{0.76}}$	$\mathbf{70.12}_{\pm \mathbf{1.12}}$

TABLE V: Hiden dimension sensitivity test over HOUSE (0.6).

	64	128	256	512
AllSet	$77.98_{\pm 2.01}$	$78.39_{\pm 2.32}$	$78.16_{\pm 1.92}$	$78.81_{\pm 1.51}$
ED-HNN	$78.11_{\pm 2.05}$	$78.60_{\pm 1.74}$	$78.92_{\pm 1.53}$	$79.01_{\pm 1.00}$
Ours	$\mathbf{79.10}_{\pm \mathbf{1.52}}$	$\mathbf{79.72}_{\pm \mathbf{1.61}}$	$\mathbf{80.02}_{\pm \mathbf{1.05}}$	$\mathbf{80.23}_{\pm \mathbf{1.31}}$

Besides, we conduct sensitivity experiments for hidden dimension d among heterophilic hypergraph House (0.6), and the result is listed in Table V. According to the table, all models achieve the best performance at high hidden dimensions (512). Our 64-width BHyGNN can outperform AllSet and ED-HNN with 512-width. This implies that our model is better tolerant of low hidden dimensions.

E. Complexity Analysis

Time Complexity. The time complexity for feeding the node and hyperedge feature matrix into VBA-Net takes $\mathcal{O}(N+M)$, where N is the size of nodes and M is the size of hyperedges. Then, sampling the actions takes $\mathcal{O}(L)$, where L is the number of incidents in the incidence matrix of hypergraph $\mathcal{H}$, i.e., $L = \sum_{e\in\mathcal{E}} d(e) = \sum_{v\in\mathcal{V}} d(v) = \sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}} H_{v,e}$. The time complexity of aggregation from node embeddings to hyperedge embedding also takes $\mathcal{O}(L)$ in the worst case, i.e., we sample broadcast for each action. Therefore, the time complexity to compute hyperedge embeddings is $\mathcal{O}(N+M+2L)$. Since the aggregation from hyperedge back to nodes adopts the same

TABLE VI: Performance and efficiency comparison. Training and inference time are reported in $(10^{-2}s)$.

Dataset	Metric	ED-HNN	BHyGNN
SENATE (0.6)	Training	**2.39 ± 1.23**	2.84 ± 1.58
	Inference	**0.91 ± 0.07**	1.12 ± 0.02
	Accuracy	65.53 ± 3.10	**67.87 ±2.13**
SYNTHETIC (0.6)	Training	**3.94 ± 1.44**	5.08 ± 1.71
	Inference	**1.56 ± 0.03**	2.75 ± 0.04
	Accuracy	55.96 ± 1.34	**58.10 ± 1.44**
CONGRESS (0.6)	Training	48.75 ± 2.10	**27.73 ± 2.04**
	Inference	14.52 ± 0.05	**9.65 ± 0.06**
	Accuracy	94.20 ± 0.98	**95.45 ± 0.62**
HOUSE (0.6)	Training	**3.39 ± 1.29**	4.04 ± 1.73
	Inference	**1.31 ± 0.04**	2.23 ± 0.04
	Accuracy	79.01 ± 1.00	**80.23 ± 1.31**
WALMART (0.6)	Training	46.04 ± 1.84	**25.41 ± 1.74**
	Inference	18.36 ± 0.09	**10.73 ± 0.04**
	Accuracy	78.15 ± 0.42	**79.94 ± 1.31**

framework, the total time complexity of our model BHyGNN is $\mathcal{O}(2N + 2M + 4L)$. Compared with baseline methods, our model has similar time complexity among all baseline methods.

Space Complexity. In each layer, BHyGNN contains two VBA-Net and two incorporation transformer modules. Each VBA-Net takes $\mathcal{O}(2Nd + 2Md)$ to store Gaussian mean for nodes $\mu_{\mathcal{V}} \in \mathbb{R}^{N\times d}$, Gaussian mean for hyperedges $\mu_{\mathcal{E}} \in \mathbb{R}^{M\times d}$, Gaussian standard deviation for nodes $\sigma_{\mathcal{V}} \in \mathbb{R}^{N\times d}$ and Gaussian standard deviation for hyperedges $\sigma_{\mathcal{E}} \in \mathbb{R}^{M\times d}$. Here, d is the dimension of the embeddings. Afterward, the space to store the encoded node and hyperedge embeddings is $\mathcal{O}(Nd + Md)$. The output of VBA-Net is a sampled action set that requires $\mathcal{O}(L)$ space. Therefore, the space complexity of each VBA-Net is $\mathcal{O}(3Nd + 3Md + L)$.

For incorporation transformer $T^{(\mathbf{B})}$, storing incorporation embedding set $\mathcal{Z}\_e^{(\mathbf{B})}$ and matrix $\mathbf{S}\_e^{(\mathbf{B})}$ takes $\mathcal{O}(2dL)$ and $\mathcal{O}(dL)$, respectively. The output of incorporation transformer $T^{(\mathbf{B})}$ is the hyperedge embeddings $\mathbf{Z}_{\mathcal{E}} \in \mathbb{R}^{M\times d}$, which takes $\mathcal{O}(Md)$ space. Therefore, the space complexity of incorporation transformer $T^{(\mathbf{B})}$ is $\mathcal{O}(3dL + Md)$. Similarly, the space complexity of incorporation transformer $T^{(\mathbf{R})}$ is $\mathcal{O}(3dL+Nd)$. To sum it up, the space complexity of BHyGNN for each layer is $\mathcal{O}(7Nd + 7Md + L + 6dL)$.

F. Efficency Analysis

We conduct an efficiency experiment and report the result in Table VI. According to the table, our proposed method BHyGNN has comparable training time and inference time over small hypergraph datasets, i.e., Senate, Synthetic, and House, with ED-HNN. For large hypergraphs, Congress and Walmart, BHyGNN is more efficient than ED-HNN in terms of training and inference time. Moreover, BHyGNN outperforms ED-HNN over all heterophilic hypergraphs.

G. Embedding Visualization

To examine the effectiveness of our model intuitively, we render the embeddings of House (0.6) and House (0.1) by BHyGNN, AllSet, and ED-HNN in Figure 3. Each unique color represents the embeddings corresponding to a specific class. According to figure, BHyGNN shows smaller overlapping areas compared with other baseline methods, which again demonstrates the effectiveness of BHyGNN for node classification in heterophilic hypergraphs.

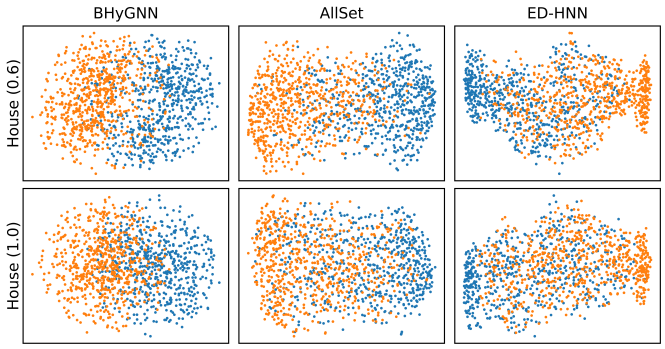

Fig. 3: Embedding visualization of AllSet, ED-HNN and BHyGNN over dataset HOUSE (0.6) and HOUSE (1.0).

H. Model Analysis over Heterophilic Hypergraphs

We leverage a detailed example in social media to demonstrate the benefit of BHyGNN over heterophilic hypergraphs. For simplicity, we compare BHyGNN with one of the SOTA baseline methods, i.e., AllSet. Given a social network hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$, where $\mathcal{V} = \{A, B, C, D, E\}$ and the hyperedge set $\mathcal{E}$ includes hyperedges $e_1 = \{A, B, C, D\}, e_2 = \{A, D, E\}, e_3 = \{A, B, D, E\}$, and $e_4 = \{B, C, E\}$, the downstream task is to predict whether the user likes outdoor activities. Specifically, user A is interested in galaxy and landscape photography, user B enjoys landscape photography, user C is interested in galaxy photography, and users D and E like food photography. Then, we have nodes ABC in class 1 (like outdoor activities) and DE in class 0. Clearly, the $\mathcal{H}$ is a heterophilic hypergraph with score 0.63.

At each layer, AllSet learn the node embedding $\mathbf{Z}_v$ as:

$$\mathbf{Z}_v = \text{Aggr}(\{\mathbf{Z}_e : v \in e\}), \text{ where} \tag{10}$$

$$\mathbf{Z}_e = \text{Aggr}(\{\mathbf{Z}'_v : v \in e\}). \tag{11}$$

Here, $\text{Aggr}(\cdot)$ is a pooling function such as mean pooling (AllDeepSets) or pooling by multihead attention (AllSetTransformer). In the above social media hypergraph $\mathcal{H}$, AllSet learns node embeddings for nodes A and D as:

$$\mathbf{Z}_{\text{A}} = \mathbf{Z}_{\text{D}} = \text{Aggr}(\{\mathbf{Z}_{e_1}, \mathbf{Z}_{e_2}, \mathbf{Z}_{e_3}\}), \tag{12}$$

which means AllSet cannot differentiate them. Moreover, as node A and node D are from different classes, AllSet cannot effectively differentiate nodes from different classes in a heterophilic setting.

However, our proposed method, BHyGNN, can effectively handle the heterophilic issue via our novel VBA-Net and incorporation transformer. Our model leverages VBA-Net to learn the broadcast action for each node-hyperedge pair in the hypergraph $\mathcal{H}$. For instance, in hyperedge e_1, as nodes A, B

and C share similar photography interests (compared with node D), VBA-Net samples broadcast actions $\mathbf{B}$, $\mathbf{B}$, $\mathbf{B}$, and $\neg\mathbf{B}$ for nodes A, B, C, and D, respectively. Here, action $\mathbf{B}$ denotes broadcast information from node to hyperedge. Formally, the hyperedge embedding $\mathbf{Z}_{e_1}$ is computed as:

$$\mathbf{Z}_{e_1} = \text{Aggr}(\{\mathbf{Z}'_A||\mathbf{Z}'_{e_1}, \mathbf{Z}'_B||\mathbf{Z}'_{e_1}, \mathbf{Z}'_C||\mathbf{Z}'_{e_1}\}), \quad (13)$$

where $\mathbf{Z}'_{\text{A}}$, $\mathbf{Z}'_{\text{B}}$, and $\mathbf{Z}'_{\text{C}}$ are the previous layer node embeddings for node A, B, and C, respectively. The learned hyperedge embedding $\mathbf{Z}_{e_1}$ by BHyGNN is more informative to depict nodes that are interested in outdoor photography, i.e., galaxy and landscape photography, than the hyperedge embedding obtained by AllSet. Therefore, our designed module, VBA-Net, is able to tackle the heterophilic issue in hypergraphs from an adaptive broadcast perspective.

Next, we illustrate the advantages of our incorporation transformer in heterophilic settings. We remove VBA-Net module to analyze the pure benefit of the incorporation transformer. The incorporation transformer updates the node embeddings for nodes A and D as:

$$\mathbf{Z}_{\text{A}} = \text{Aggr}(\{\mathbf{Z}_{e_1}||\mathbf{Z}'_{\text{A}}, \mathbf{Z}_{e_2}||\mathbf{Z}'_{\text{A}}, \mathbf{Z}_{e_3}||\mathbf{Z}'_{\text{A}}\}), \quad (14)$$

$$\mathbf{Z}_{\text{D}} = \text{Aggr}(\{\mathbf{Z}_{e_1}||\mathbf{Z}'_{\text{D}}, \mathbf{Z}_{e_2}||\mathbf{Z}'_{\text{D}}, \mathbf{Z}_{e_3}||\mathbf{Z}'_{\text{D}}\}). \quad (15)$$

Intuitively, the learned embeddings for nodes A and D are distinct, which further facilitates our model for differentiating nodes in heterophilic hypergraphs. Therefore, our proposed incorporation transformer can tackle the heterophilic issue in hypergraphs from the aggregation aspect.

VI. Conclusion

This paper introduces a novel Broadcast Hypergraph Neural Network, called BHyGNN, to address the limitations of existing HyGNNs for heterophilic hypergraphs. We first introduce a novel Variational Broadcast Autoencoder Network (VBA-Net) to estimate the broadcast and receive actions for nodes to facilitate the representation learning in heterophilic hypergraphs. Then we devise an incorporation transformer that performs the sampled actions to adaptively propagate node information incorporated with hyperedge information to learn informative representations in the heterophilic hypergraphs. Extensive experiments on benchmark datasets demonstrate the effectiveness of BHyGNN over baseline methods in both heterophilic and homophilic hypergraphs.

Acknowledgments

The work of T. Ma, Z. Zhang, Z. Wang, C. Zhang, and Y. Ye was partially supported by the NSF under grants IIS-2533550, IIS-2321504, IIS-2340346, IIS-2334193, IIS-2217239, CNS-2426514, and CMMI-2146076, ND-IBM Tech Ethics Lab Program, Notre Dame Strategic Framework Research Grant (2025), and Notre Dame Poverty Research Package (2025). Any expressed opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] T. Ma, Y. Qian, C. Zhang, and Y. Ye, "Hypergraph contrastive learning for drug trafficking community detection," in *ICDM*, 2023.
[2] T. Ma, Y. Qian, S. Zhang, C. Zhang, and Y. Ye, "Adaptive expansion for hypergraph learning," *arXiv preprint arXiv:2502.15564*, 2025.
[3] N. Yadati, "Neural message passing for multi-relational ordered and recursive hypergraphs," in *NeurIPS*, 2020.
[4] J. Zhang, F. Li, X. Xiao, T. Xu, Y. Rong, J. Huang, and Y. Bian, "Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs," *arXiv preprint arXiv:2203.16939*, 2022.
[5] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, "Semi-supervised hypergraph node classification on hypergraph line expansion," in *CIKM*, 2022.
[6] Y. Qian, Y. Zhang, Y. Ye, and C. Zhang, "Adapting meta knowledge with heterogeneous information network for covid-19 themed malicious repository detection," in *IJCAI*, 2021.
[7] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is homophily a necessity for graph neural networks?" in *ICLR*, 2022.
[8] O. Platonov, D. Kuznedelev, M. Diskin, A. Babenko, and L. Prokhorenkova, "A critical look at the evaluation of gnns under heterophily: are we really making progress?" in *ICLR*, 2023.
[9] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," in *NeurIPS*, 2021.
[10] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *AAAI*, 2019.
[11] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "Hypergcn: A new method for training graph convolutional networks on hypergraphs," in *NeurIPS*, 2019.
[12] Y. Dong, W. Sawin, and Y. Bengio, "Hnhn: Hypergraph networks with hyperedge neurons," *arXiv preprint arXiv:2006.12278*, 2020.
[13] Z. Wang, Z. Zhang, T. Ma, N. V. Chawla, C. Zhang, and Y. Ye, "Beyond message passing: Neural graph pattern machine," in *ICML*, 2025.
[14] Z. Wang, Z. Liu, T. Ma, J. Li, Z. Zhang, X. Fu, Y. Li, Z. Yuan, W. Song, Y. Ma *et al.*, "Graph foundation models: A comprehensive survey," *arXiv preprint arXiv:2505.15116*, 2025.
[15] Z. Wang, Z. Zhang, N. Chawla, C. Zhang, and Y. Ye, "Gft: Graph foundation model with transferable tree vocabulary," in *NeurIPS*, 2024.
[16] Z. Wang, Z. Zhang, T. Ma, N. V. Chawla, C. Zhang, and Y. Ye, "Towards graph foundation models: Learning generalities across graphs via task-trees," in *ICML*, 2025.
[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[18] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *NeurIPS*, 2020.
[19] P. Wang, S. Yang, Y. Liu, Z. Wang, and P. Li, "Equivariant hypergraph diffusion neural operators," in *ICLR*, 2023.
[20] T. Wei, Y. You, T. Chen, Y. Shen, J. He, and Z. Wang, "Augmentations in hypergraph contrastive learning: Fabricated and generative," in *NeurIPS*, 2022.
[21] N. Veldt, A. R. Benson, and J. Kleinberg, "Higher-order homophily is combinatorially impossible," *SIAM Review*, 2022.
[22] I. Amburg, N. Veldt, and A. Benson, "Clustering in graphs and hypergraphs with categorical edge labels," in *WWW*, 2020.
[23] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, 2021.
[24] K. Ding, J. Wang, J. Li, D. Li, and H. Liu, "Be more with less: Hypergraph attention networks for inductive text classification," in *EMNLP*, 2020.
[25] R. Zhang, Y. Zou, and J. Ma, "Hyper-SAGNN: a self-attention based graph neural network for hypergraphs," in *ICLR*, 2020.
[26] E. Chien, C. Pan, J. Peng, and O. Milenkovic, "You are allset: A multiset function framework for hypergraph neural networks," in *ICLR*, 2022.
[27] Y. Qian, Y. Zhang, Y. Ye, and C. Zhang, "Distilling meta knowledge on heterogeneous graph for illicit drug trafficker detection on social media," in *NeurIPS*, 2021.
[28] Y. Zhang, Y. Qian, Y. Fan, Y. Ye, X. Li, Q. Xiong, and F. Shao, "dstylegan: Generative adversarial network based on writing and photography styles for drug identification in darknet markets," in *ACSAC*, 2020.
[29] Y. Zhang, Y. Qian, Y. Ye, and C. Zhang, "Adapting distilled knowledge for few-shot relation reasoning over knowledge graphs," in *SDM*, 2022.

[30] Y. Qian, Y. Zhang, Q. Wen, Y. Ye, and C. Zhang, "Rep2vec: Repository embedding via heterogeneous graph adversarial contrastive learning," in *KDD*, 2022.
[31] Y. Qian, Y. Zhang, N. Chawla, Y. Ye, and C. Zhang, "Malicious repositories detection with adversarial heterogeneous graph contrastive learning," in *CIKM*, 2022.
[32] Q. Wen, Z. Ouyang, J. Zhang, Y. Qian, Y. Ye, and C. Zhang, "Disentangled dynamic heterogeneous graph learning for opioid overdose prediction," in *KDD*, 2022.
[33] Y. Qian, T. Ma, C. Zhang, and Y. Ye, "Adaptive graph enhancement for imbalanced multi-relation graph learning," in *WSDM*, 2025.
[34] Q. Wen, Z. Ouyang, C. Zhang, Y. Qian, C. Zhang, and Y. Ye, "Gcvr: reconstruction from cross-view enable sufficient and robust graph contrastive learning," in *UAI*, 2024.
[35] Y. Qian, *Graph Representation Learning Techniques for the Combat Against Online Abusive Activity*. University of Notre Dame, 2024.
[36] Y. Qian, P. Chen, S. Cui, and D. Chen, "Universal ring-of-abusers detection via multi-modal heterogeneous graph learning," 2023.
[37] Y. Qian, C. Zhang, Y. Zhang, Q. Wen, Y. Ye, and C. Zhang, "Co-modality imbalanced graph contrastive learning," in *NeurIPS*, 2022.
[38] Y. Ye, Y. Fan, S. Hou, Y. Zhang, Y. Qian, S. Sun, Q. Peng, M. Ju, W. Song, and K. Loparo, "Community mitigation: A data-driven system for covid-19 risk assessment in a hierarchical manner," in *CIKM*, 2020.
[39] Y. Ye, S. Hou, Y. Fan, Y. Zhang, Y. Qian, S. Sun, Q. Peng, M. Ju, W. Song, and K. Loparo, "α-satellite: An ai-driven system and benchmark datasets for dynamic covid-19 risk assessment in the united states," *IEEE Journal of Biomedical and Health Informatics*, 2020.
[40] Y. Qian, S. Zhang, L. Chen, D. Socolinsky, N. Sokhandan, S. Cui, D. Chen, and S. Sathyanarayana, "Enhancing e-commerce representation learning via hypergraph contrastive learning and interpretable llm-driven analysis," in *WWW*, 2025.
[41] I. E. Chien, H. Zhou, and P. Li, "hs^2: Active learning over hypergraphs with pointwise and pairwise queries," in *PMLR*, 2019.
[42] J. Zhu, R. A. Rossi, A. Rao, T. Mai, N. Lipka, N. K. Ahmed, and D. Koutra, "Graph neural networks with heterophily," in *AAAI*, 2021.
[43] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *ICLR*, 2021.
[44] T. Wang, D. Jin, R. Wang, D. He, and Y. Huang, "Powerful graph convolutional networks with adaptive propagation mechanism for homophily and heterophily," in *AAAI*, 2022.
[45] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*, 2020.
[46] C. Bodnar, F. Di Giovanni, B. Chamberlain, P. Liò, and M. Bronstein, "Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns," in *NeurIPS*, 2022.
[47] E. Vértes and M. Sahani, "Flexible and accurate inference and learning for deep generative models," in *NeurIPS*, 2018.
[48] A. L. Caterini, A. Doucet, and D. Sejdinovic, "Hamiltonian variational auto-encoder," in *NeurIPS*, 2018.
[49] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
[50] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *ICLR*, 2016.
[51] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *ICLR*, 2017.
[52] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.
[53] J. Huang and J. Yang, "Unignn: a unified framework for graph and hypergraph neural networks," in *IJCAI*, 2021.
[54] Y. Qian, T. Ma, C. Zhang, and Y. Ye, "Dual-level hypergraph contrastive learning with adaptive temperature enhancement," in *WWW*, 2024, pp. 859–862.
[55] I. Duta, G. Cassarà, F. Silvestri, and P. Liò, "Sheaf hypergraph networks," in *NeurIPS*, 2024.
[56] S. Kim, S. Kang, F. Bu, S. Y. Lee, J. Yoo, and K. Shin, "Hypeboy: Generative self-supervised representation learning on hypergraphs," in *ICLR*, 2024.
[57] P. S. Chodrow, N. Veldt, and A. R. Benson, "Hypergraph clustering: from blockmodels to modularity," *Science Advances*, 2021.
[58] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, "Simplicial closure and higher-order link prediction," *Proceedings of the National Academy of Sciences*, 2018.
[59] Y. Deshpande, S. Sen, A. Montanari, and E. Mossel, "Contextual stochastic block models," in *NeurIPS*, 2018.
[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.
[61] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *ICML*, 2019.
[62] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," in *ICLR*, 2021.

APPENDIX

A. Model Design of Incorporation Transformer

In this section, we first formulate the classic multi-head attention [60] and further analyze our model design, compared with the multi-head attention. Given a set of n query vectors, each with dimension d_q, $\mathbf{Q} \in \mathbb{R}^{n\times d_q}$, the self-attention function, i.e., $Att(\mathbf{Q},\mathbf{K},\mathbf{V};\omega) = w(\mathbf{Q}\mathbf{K}^T)V$, maps queries $\mathbf{Q}$ to outputs using n_v key-value paris $\mathbf{K} \in \mathbb{R}^{n_v\times d_q}$, $\mathbf{V} \in \mathbb{R}^{n_v\times d_v}$ with an activation function w. The multi-head attention is formulated as follows:

$$\text{Multihead}(\mathbf{Q},\mathbf{K},\mathbf{V};\lambda,\omega) = ||_{i=1}^{h}\mathbf{O}^{(i)}W_O, \text{ where} \quad (16)$$

$$\mathbf{O}^{(i)} = \text{Att}(\mathbf{Q}W_Q^{(i)},\mathbf{K}W_K^{(i)},\mathbf{V}W_V^{(i)};\omega). \quad (17)$$

Here, $\lambda = {W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}}_{i=1}^{h}$ is a set of learnbale parameters, with $W_Q^{(i)}, W_K^{(i)} \in \mathbb{R}^{d_q\times d_q^M}$, $W_V^{(i)} \in \mathbb{R}^{d_v\times d_v^M}$, and $W_O \in \mathbb{R}^{hd_v^M\times d}$. h is the number of heads, and a typical choice for the dimension hyperparameters is $d_q^M = d_q/h, d_v^M = d_v/h$, and $d = d_q$. Our Multi-head attention mechanism in Eq. 9 is formulated as follows:

$$\text{MH}_h(\mathbf{S}_e;\theta_e) = ||_{i=1}^{h}\mathbf{O}^{(i)}, \text{where} \quad (18)$$

$$\mathbf{O}^{(i)} = \text{Softmax}(\theta_e^{(i)}(\mathbf{K}^{(i)})^T)\mathbf{V}^{(i)}. \quad (19)$$

Here, $\mathbf{S}_e \in \mathbb{R}^{n_v\times d_v}$ is a matrix embeddings of n_v nodes, $\theta_e \in \mathbb{R}^{1\times d_q}$ is a learnable parameter matrix, $\theta_e = ||_{i=1}^{h}\theta_e^{(i)}$, $\mathbf{K}^{(i)} = \text{MLP}_K^{(i)}(\mathbf{S}_e) \in \mathbb{R}^{n_v\times d_q^M}$ and $\mathbf{V}^{(i)} = \text{MLP}_V^{(i)}(\mathbf{S}_e) \in \mathbb{R}^{n_v\times d_v^M}$. For clarification, we substitute the dimension notations here to match the dimension notations in Eq 16. With the output of $\text{MH}(\cdot)$, we further apply layer normalizations $\text{LN}(\cdot)$ and an additional MLP layer to obtain the hyperedge embedding:

$$\mathbf{Z}_e = \text{LN}(Q + \text{MLP}(Q)), \text{ where} \quad (20)$$

$$Q = \text{LN}(\theta_e + \text{MH}(\mathbf{S}_e;\theta_e)). \quad (21)$$

In Eq. 18, θ_e can be viewed as a seed vector Q, and $\text{MH}(\cdot)$ considers interactions between the seed vector θ_e and n_v nodes, to compress n nodes into one with their attention similarities between query and keys. Compare Eq. 18 with Eq 16, we replace $\mathbf{Q}W_Q^{(i)}$ with $\theta_e^{(i)}$ as $\theta_e^{(i)}$ is already a learnable matrix, and employ another weight matrix W_Q is unnecessary. Besides, we drop the weight matrix W_O, as it can be effectively taken care of by $\mathbf{V}^{(i)}$. In Eq. 20, adding the learnable seed vector θ_e to $\text{MH}(\cdot)$ is inspired by the common strategy in aggregation scheme, i.e., pooling by multihead attention [61] and GMT [62]. As discussed in GMT, the learnable seed vector θ_e can be directly optimized end-to-end.