Exploring Hypergraph Condensation via Variational Hyperedge Generation and Multi-Aspectual Amelioration

Anonymous Author(s)

Abstract

Hypergraph neural networks (HyperGNNs) show promise in modeling online networks with high-order correlations. Despite notable progress, training these models on large-scale raw hypergraphs entails substantial computational and storage costs, thereby increasing the need of hypergraph size reduction. However, existing size reduction methods primarily capture pairwise association pattern within conventional graphs, making them challenging to adapt to hypergraphs with high-order correlations. To fill this gap, we introduce a novel hypergraph condensation framework, HG-Cond, designed to distill large-scale hypergraphs into compact, synthetic versions while maintaining comparable HyperGNN performance. Within this framework, we develop a Neural Hyperedge Linker to capture the high-order connectivity pattern through variational inference, achieving linear complexity with respect to the number of nodes. Moreover, We propose a multi-aspectual amelioration strategy including a Gradient-Parameter Synergistic Matching objective to holistically refine synthetic hypergraphs by coordinating improvements in node attributes, high-order connectivity, and label distributions. Extensive experiments demonstrate the efficacy of HG-Cond in hypergraph condensation, notably outperforming the original test accuracy on the 20News dataset while concurrently reducing the hypergraph size to a mere 5% of its initial scale. Furthermore, the condensed hypergraphs demonstrate robust cross-architectural generalizability and potential for expediting neural architecture search. This research represents a significant advancement in hypergraph processing, providing a scalable approach for hypergraph-based learning in resource-limited environments.

CCS Concepts

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → *Neural networks.*

Keywords

Hypergraph Condensation, High-order Correlations

1 Introduction

The ubiquity of online networks has led to the accumulation of vast high-order correlation data, which frequently extends beyond simple pairwise associations and permeates the World Wide Web, citation networks, and social media platforms. For example, a tweet connects all users who engage with it, a research paper is typically co-authored by a group of scholars, and an email is often addressed to multiple recipients. In light of these high-order correlations, hypergraphs have emerged as a flexible and effective mathematical tool for faithfully capturing multi-way relationships and facilitating more nuanced analysis of network, offering particular advantages in domains such as online community detection [28] and e-commerce recommendation [44]. Recent advancements in hypergraph neural networks (HyperGNNs) have demonstrated remarkable progress in representation learning on hypergraphs, showcasing the effectiveness of capturing and leveraging high-order relationships [13, 15]. Despite these notable advancements, the necessity to train these models on large-scale raw hypergraphs incurs substantial computational and storage costs, while simultaneously raising concerns regarding data privacy protection [38].

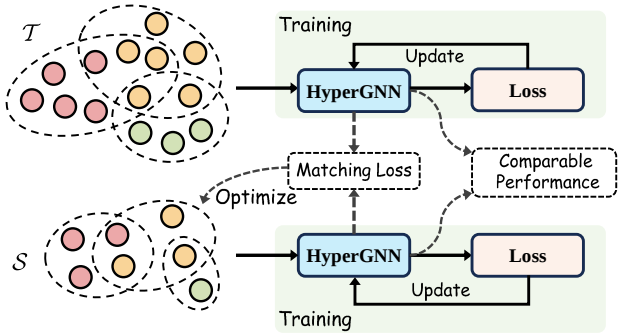


Figure 1: Our framework condenses hypergraph data to a much smaller yet informative one, such that HyperGNNs trained on condensed hypergraph $\mathcal{S}$ and target hypergraph $\mathcal{T}$ exhibit comparable performance. The synthesis of $\mathcal{S}$ necessitates matching objectives on two HyperGNNs.

In recent years, graph size reduction has emerged as a promising research direction to compress or simplify large-scale pairwise graphs while preserving their essential information content, thereby enhancing the efficiency and scalability of graph neural networks (GNNs) training. Embodying this reduction principle, graph sparsification [7, 36] centers on preserving specific structural properties, *e.g.,* principle eigenvalues, through the retention of critical nodes and a subset of pair-wise edges, while graph coarsening [5, 22] maintains all nodes but aggregates them into super nodes, with original inter-group edges consolidated into super edges. However, these methods, constrained by their reliance on sampling from existing data, inherently limit the information content of the sampled data to that of the original dataset, consequently leading to diminished performance when GNNs are trained on these reduced graphs. In contrast, graph condensation [20, 21], as a generative method, distills the original graph into a newly-constructed, smaller graph with synthesized nodes and structures. While existing graph size reduction techniques have shown efficacy in preserving or simplifying pairwise structures in conventional graphs, they encounter substantial limitations when applied to hypergraphs. To our knowledge, there still lacks a solution on hypergraph size reduction.

Indeed, the task of hypergraph size reduction presents unique challenges attributable to the intrinsic characteristics of hypergraph. **(i) Modeling complex high-order correlations:** The high-order correlations arise from the semantic relationships among multiple nodes. Inadequate modeling of these complex interactions during the reduction process may result in significant information loss,

leading to degraded performance trained on the reduced hypergraphs. As expression capacity of high-order correlation transcends pair-wise edges in conventional graphs, the complexity of modeling high-order connectivity patterns manifests in various forms, such as higher-dimensional dependencies and semantic topological features. **(ii) Managing exponential complexity:** Let $|\mathcal{V}|$ represents the number of nodes. While conventional graphs are limited to a maximum of $|\mathcal{V}|^2$ pairwise edges, the flexible node inclusion characteristic of hypergraphs permits up to $2^{|\mathcal{V}|}$ hyperedges (the number of node sets). Extant graph methods [5, 20, 21] primarily rely on pairwise edge reconstruction. However, they are not directly applicable to hypergraphs due to challenges in computational efficiency and algorithmic scalability arising from the exponential growth in potential hyperedges relative to the number of nodes.

To tackle the above challenges, we propose a novel HyperGraph Condensation framework, called HG-Cond, to condense large-real hypergraphs into small-synthetic hypergraphs, such that HyperGNNs trained on condensed hypergraph and original hypergraph exhibit comparable performance. Within this framework, we develop a Neural Hyperedge Linker to model the complex high-order correlations in hypergraphs. The Neural Hyperedge Linker employs variational inference with a Dirichlet prior and transitions the focus of prediction from the set level (hyperedge) to the node level for efficient hyperedge synthesis, thereby reducing the prediction space from intractable $O(2^{|\mathcal{V}|})$ to manageable $O(|\mathcal{V}|)$. Moreover, we develop a coreset initialization method to select representative node exemplars from the raw hypergraphs and synthesize structure through stochastic composition, serving as the initialization of reduced hypergraphs. Furthermore, we propose a multi-aspectual amelioration strategy including a Gradient-Parameter Synergistic Matching objective to holistically refine synthetic hypergraphs by coordinating improvements in node attributes, high-order connectivity, and label distributions. Extensive experiments conducted across multiple real-world hypergraphs demonstrate that HG-Cond can effectively condense various hypergraphs and achieve comparable performance to their larger counterparts. Notably, HG-Cond even outperforms the original test accuracy on the Pubmed and 20News dataset while reducing the hypergraph size to a mere 7% and 5% of its initial scale, respectively. *To the best of our knowledge, we are among the first to propose a hypergraph condensation framework to reduce the hypergraph size.*[1]

In summary, the key contributions of our work are as follows:

- We propose a novel hypergraph condensation framework to distill large-real hypergraphs into small-synthetic hypergraphs, such that HyperGNNs trained on the original and condensed hypergraphs have comparable performance.
- We design a neural hyperedge linker, a hypergraph coreset initialization, and a multi-aspectual amelioration strategy to collaboratively optimize node features, structures and labels of condensed hypergraphs.
- We conduct extensive experiments to validate the efficacy of HG-Cond in condensing diverse large-real hypergraphs. Our results reveal that condensed hypergraphs achieve performance on par with original counterparts.
- We demonstrate the generalizability of the condensed hypergraphs across different test HyperGNNs. Furthermore, we uncover a strong correlation between the performance of models trained on condensed hypergraphs and whole hypergraphs in architecture search experiments.

[1]The anonymous code can be found at https://anonymous.4open.science/r/HG-Cond.

2 Related Work

The related work of this paper falls into three categories: Hypergraph Neural Network, Graph Size Reduction and Dataset Condensation (Distillation).

2.1 Hypergraph Neural Networks

Hypergraph Neural Networks (HyperGNNs) are designed to capture structural information and derive representations of nodes and edges in hypergraphs, analogous to graph neural networks [24] applied in traditional pairwise graphs. Unlike graphs, where edges connect pairs of nodes, hypergraphs allow for hyperedges that can link multiple nodes simultaneously. Early HyperGNNs are built upon the spectral domain, such as HGNN [13] and HpLapGCN [14], which conduct the hypergraph Laplacian matrix for feature smoothing. HyperGCN [42] employs a specific strategy to transform the hypergraph into standard graphs, enabling the application of traditional graph neural networks [24] for node representation learning. Additionally, a series of spatial-based HyperGNNs are proposed, including two-stage message passing techniques utilized in HNHN [10], AllDeepSets [9] and AllSetTransformer [9].

2.2 Graph Size Reduction

Graph size reduction encompasses several approaches: graph sampling, graph coreset, graph sparsification, graph coarsening, and the recent graph condensation techniques. These methods aim to reduce the number of nodes and edges to facilitate more efficient GNN training. Graph sampling [8, 45] and graph coreset methods [32, 41] involve selecting a subset of nodes and edges from the original graph. However, the expressiveness of the resulting subgraph is inherently limited by the structure of the full-scale graph. Graph sparsification methods [16, 36], on the other hand, focus on preserving specific graph properties, such as the spectrum or principal eigenvalues, while simplifying redundant edges in the original graph. Similarly, graph coarsening methods [5, 22] aim to maintain certain graph characteristics by grouping node representations from the raw graph. A limitation of both graph sparsification and coarsening approaches is that the preserved graph properties may not always align optimally with the requirements of downstream GNN tasks. In contrast, recent graph condensation techniques [20, 21, 48] offer a novel approach to graph size reduction. These methods aim to create condensed graph representations that capture essential structural and feature information, potentially overcoming some limitations of earlier approaches.

2.3 Dataset Condensation

Dataset condensation aims to create a small, representative dataset that captures the essential knowledge from a larger target dataset. The goal is to train models on this condensed dataset that perform comparably to those trained on the full dataset. This approach has various applications [27, 34], such as rapid architecture search [29]

and efficient data replay in continual learning [4], where minimizing resource consumption is crucial. Early methods like DD [39] and DC-KRR [30] use meta-learning frameworks to calculate meta-gradients and solve bi-level distillation objectives. Later approaches, including DC [47], DM [46], and MTT [6], avoid unrolled optimization by designing surrogate functions to match gradients, feature distributions, and training trajectories, respectively. These methods essentially aim to create a condensed dataset that effectively mimics the larger target dataset from different perspectives. While the aforementioned works primarily focused on image data, recent research has extended these concepts to structured graph data. GCOND [21] was the first to adapt gradient matching for graph data, incorporating a graph structure learning module to synthesize pairwise edges. DosCond [20] introduced single-step gradient matching for node synthesis and a probabilistic graph model for structure condensation. SFGC [48] employed a graph neural tangent kernel to bypass iterative GNN training, proposing a structure-free graph condensation method that eliminates the need to synthesize graph structures.

3 Preliminaries

This section commences with an overview of the hypergraph structure and subsequently introduces HyperGNNs as the backbone for condensing hypergraph datasets. Following this, we provide a precise definition of the hypergraph size reduction problem.

Definition 3.1 (Hypergraph). Let $\mathcal{G} = \{\mathbf{X}, \mathbf{H}, \mathbf{Y}\}$ represent a hypergraph, where $\mathbf{X} \in \mathbb{R}^{N\times d}$ denotes the d-dimensional features of node set $\mathcal{V}$ including N number of nodes, and $\mathbf{Y} \in \{0, ..., C-1\}^N$ denotes the node labels with C-classes. The topology of $\mathcal{G}$, *i.e.*, the hyperedge set $\mathcal{E}$, is represented by the incidence matrix $\mathbf{H} \in \{0, 1\}^{N\times M}$. In this incidence matrix, the rows correspond to the N nodes and the columns correspond to the M hyperedges, where $N = |\mathcal{V}|$ and $M = |\mathcal{E}|$. An entry $\mathbf{H}_{ve} = 1$ in the incidence matrix indicates that node v is a member of hyperedge e.

Definition 3.2 (Hypergraph Neural Networks.). Given a hypergraph $\mathcal{G} = \{\mathbf{X}, \mathbf{H}, \mathbf{Y}\}$, a HyperGNN learns to aggregate information from the hyperedges and their incident nodes, typically through a message-passing mechanism [9]. This process facilitates the model's ability to capture and leverage high-order structural information inherent in hypergraph data. Typically, a HyperGNN comprises K layers, each of which incorporates two components: (i) Hyperedge convolution: A generalization of graph convolution that operates on hyperedges to aggregate information from connected nodes. (ii) Node feature update: A mechanism to update node features based on the aggregated information from incident hyperedges.

Definition 3.3 (Hypergraph Size Reduction). Given a large-scale hypergraph $\mathcal{T} = \{\mathbf{X}, \mathbf{H}, \mathbf{Y}\}$ comprising N nodes with d-dimensional features and M hyperedges, the objective of hypergraph size reduction is to generate a small-scale hypergraph denoted as $\mathcal{S} = \{\mathbf{X}', \mathbf{H}', \mathbf{Y}'\}$. In this condensed dataset, $\mathbf{X}' \in \mathbb{R}^{N'\times d}$ represents the features of node set $\mathcal{V}'$, $\mathbf{Y}' \in \{0, ..., C-1\}^{N'}$ denotes the node labels, and $\mathbf{H}' \in \{0, 1\}^{N'\times M'}$ represents the incidence matrix, where $N' \ll N$ and $M' \ll M$. The reduction rate r is set as $r = \frac{M'}{M} = \frac{N'}{N}$. The goal is to ensure that a hypergraph neural network trained on $\mathcal{S}$ can achieve comparable performance to one trained on the much larger hypergraph $\mathcal{T}$:

$$\min_{\mathcal{S}} \mathcal{L}_{\mathcal{T}} \left(\text{HyperGNN}_{\theta_{\mathcal{S}}}(\mathbf{X}, \mathbf{H}), \mathbf{Y}\right) \tag{1}$$

$$s.t. \quad \theta_{\mathcal{S}} = \arg\min_{\theta} \mathcal{L}_{\mathcal{S}} \left(\text{HyperGNN}_{\theta}\left(\mathbf{X}', \mathbf{H}'\right), \mathbf{Y}'\right). \tag{2}$$

In this formulation, HyperGNN_{θ} denotes a HyperGNN parameterized by θ, $\theta_{\mathcal{S}}$ represents the parameters of the HyperGNN trained on the compact synthetic dataset $\mathcal{S}$, and $\mathcal{L}_{\mathcal{T}}$ and $\mathcal{L}_{\mathcal{S}}$ are the loss functions (typically cross-entropy loss) that measures the difference between model predictions and ground-truth labels on target hypergraph $\mathcal{T}$ and condensed hypergraph $\mathcal{S}$, respectively.

4 Hypergraph Condensation

This section presents an overview of our hypergraph condensation framework, HG-Cond, followed by detailed descriptions of its components. For comprehensive training procedures and time complexity analyses, please refer to Appendix B.

4.1 Framework Overview

Given a large-scale hypergraph $\mathcal{T}$, our framework aims to synthesize a compact yet informative hypergraph $\mathcal{S}$, with the objective of minimizing the performance disparity between HyperGNNs trained on $\mathcal{T}$ and $\mathcal{S}$. In contrast to conventional size reduction methods such as sparsification and sampling, which typically derive a subset of the original node set, our approach generates a novel synthesized node set $\mathcal{V}'$ for $\mathcal{S}$. Notably, $\mathcal{V}'$ is not constrained to be a subset of $\mathcal{V}$, allowing for greater flexibility in the reduction process. To effectively capture high-order connectivity patterns and address the exponential complexity of potential hyperedges (as discussed in Section 1), we introduce an innovative **Neural Hyperedge Linker** that employs variational inference with Dirichlet prior. This method, detailed in Section 4.2, efficiently synthesizes hyperedges with the highest log-likelihood in the original hypergraph $\mathcal{T}$. Then we propose a **Hypergraph Coreset Initialization** approach for the synthetic hypergraph, as described in Section 4.3. In Section 4.4, we develop a **Multi-Aspectual Amelioration** strategy, which includes a Gradient-Parameter Synergistic Matching objective, to jointly optimize the node attributes, structures and labels within hypergraphs. The schematic representation of our hypergraph condensation procedure is illustrated in Figure 2, and the holistic algorithm pipeline is presented in Algorithm 1.

4.2 Neural Hyperedge Linker

In the condensation process of original hypergraphs, a significant challenge lies in simultaneously establishing high-order structures and modifying node features. Previous hyperedge prediction methods [19, 31, 43] primarily rely on negative sampling or adversarial training. These approaches evaluate their efficacy by comparing the potential scores of preserved existing hyperedges against pre-sampled negative candidates. However, such methods cannot be directly applied to generate the most probable hyperedges. To address this limitation, our neural hyperedge linker generates hyperedges for the new condensed hypergraph by modeling the log-likelihood of the original hypergraph. Our neural hyperedge linker supports two distinct generation schemas. (i) **Additive expansion**: This

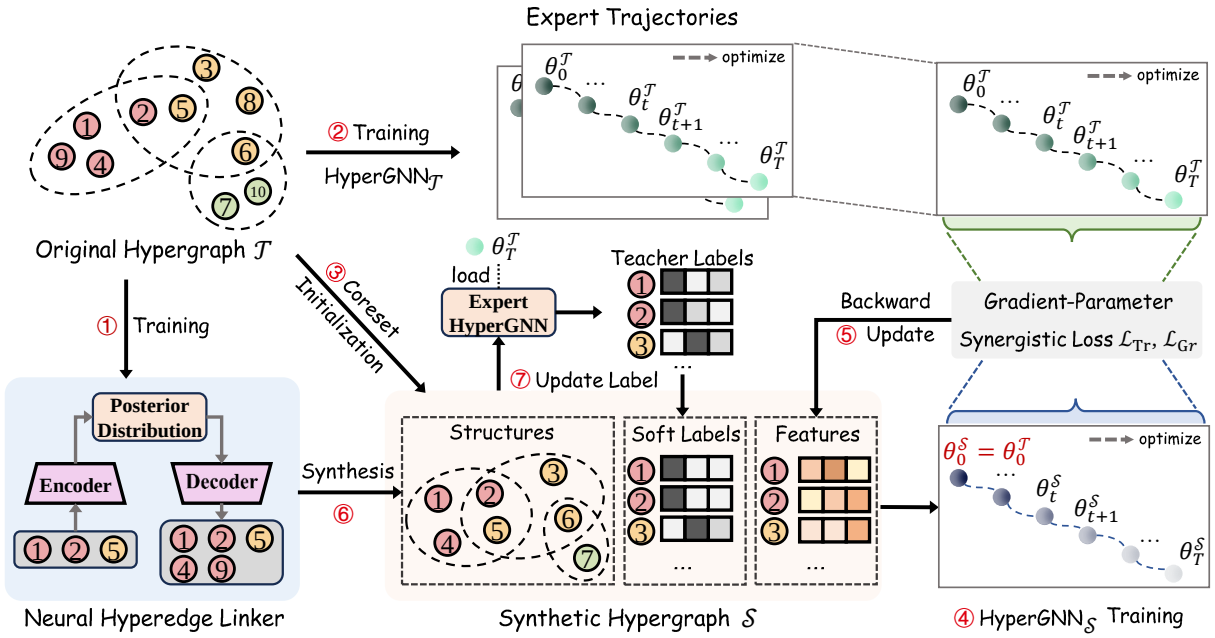


Figure 2: Our hypergraph condensation framework consists of three main phases. (i) Training the neural hyperedge linker through variational inference (①). (ii) Coreset initialization phase (③⑥) selects node exemplars from $\mathcal{T}$ and synthesizes initial structures. (iii) Condensation phase (②④⑤⑥⑦) optimizes node features $\mathbf{X}'$, structures $\mathbf{H}'$ and label distributions $\mathbf{Y}'$.

approach augments existing hyperedge structures through the incremental addition or deletion of nodes to incomplete hyperedges. (ii) **Stochastic composition**: This method synthesizes new hyperedges via a stochastic process guided by the learned log-likelihood of the original hypergraph. *To the best of our knowledge, this represents the first hypergraph generative model capable of handling both of these patterns in the generation of hypergraph structures.*

Inspired by the well-studied generative models with variational inference [25, 26, 40], our neural hyperedge linker consists of the encoder and decoder neural networks.

4.2.1 Variational Encoder. The encoder maps nodes and hyperedges into latent representations to capture the interdependencies. First, we embed the hypergraph into two sets of representations: one for nodes $\mathbf{Z}_{\mathcal{V}} = \{\mathbf{z}_v | v \in \mathcal{V}\}$ and another for hyperedges $\mathbf{Z}_{\mathcal{E}} = \{\mathbf{z}_e | e \in \mathcal{E}\}$, using a UNIGAT model [18]. Within the framework of variational inference, selecting an appropriate prior distribution is crucial for modeling the intricate relationships within the hypergraph. Observations from various datasets indicate that hyperedges often encapsulate a collection of entities sharing common attributes [9, 42]. For example, a research paper is typically co-authored by a group of researchers with aligned research interests. Analogously, the relationship between a hyperedge and its constituent nodes can be likened to the relationship between a document and its constituent words, where the composition of words defines the document's subject matter.

Drawing inspiration from the effectiveness of topic models [3] for text classification and extraction, we posit that both the prior distribution $p(\boldsymbol{\beta}^{(e)})$ and the posterior distribution $q_\phi(\boldsymbol{\beta}^{(e)}|\mathbf{z}_e)$ of hyperedge attributes adhere to Dirichlet distributions, each with K dimensions. Specifically, the posterior distribution $q_\phi(\boldsymbol{\beta}^{(e)}|\mathbf{z}_e)$ is derived from $q(\boldsymbol{\beta}^{(e)}|\boldsymbol{\alpha}^{(e)}) = \text{Dirichlet}(\alpha_1^{(e)}, \alpha_2^{(e)}, \ldots, \alpha_K^{(e)})$, where $\boldsymbol{\alpha}^{(e)} \in \mathbb{R}_+^K$ is computed via a deterministic multi-layer perceptron (MLP) parameterized by ϕ acting on $\mathbf{z}_e$, utilizing a non-negative activation:

$$\boldsymbol{\alpha}^{(e)} = \log(1 + \exp(\text{MLP}_\phi(\mathbf{z}_e))). \tag{3}$$

To mitigate the variance and address the non-differentiable issue inherent in stochastic estimations, we employ the reparameterization trick [33] to sample the posterior hyperedge traits from the Dirichlet distribution Dirichlet($\alpha^{(e)}$). Specifically, we sample each component $y_i^{(e)}$ of the hyperedge trait vector from a Gamma distribution Gamma $\left(\alpha_i^{(e)}, 1\right)$, where $\alpha_i^{(e)}$ and 1 are the shape parameter and scale parameter of the Gamma distribution, respectively:

$$y_i^{(e)} \sim \text{Gamma}\left(\alpha_i^{(e)}, 1\right) \text{ for } i = 1, 2, \ldots, K. \tag{4}$$

Following the sampling, we normalize the sampled values $y_1^{(e)}, \ldots, y_K^{(e)}$ such that their sum equals one, thereby obtaining the latent traits $\beta^{(e)}$ of hyperedge e:

$$\boldsymbol{\beta}_i^{(e)} = y_i^{(e)} / \sum_{j=1}^{K} y_j^{(e)} \quad \text{for } i = 1, 2, \ldots, K. \tag{5}$$

4.2.2 Structure Decoder. With the learned node embeddings $\mathbf{Z}_{\mathcal{V}}$ and the hypergraph variational traits $\mathcal{B}_{\mathcal{E}} = \{\boldsymbol{\beta}^{(e)} | e \in \mathcal{E}\}$, the decoder aims to reconstruct the high-order correlations present in the hypergraph. However, directly modeling the space of high-order interactions faces a combinatorial challenge, as there are $2^{|\mathcal{V}|}$ potential hyperedges (as discussed in Section 1). Inspired by strategies from representation learning, we design the decoder to predict the presence of missing nodes within hyperedges rather than generating hyperedges from scratch. This approach reduces

the prediction space from a computationally prohibitive $O(2^{|\mathcal{V}|})$ to affordable $O(|\mathcal{V}|)$. We formulate the decoding process as:

$$p(\mathbf{H}|\mathbf{Z}_{\mathcal{V}}, \hat{\mathbf{Z}}_{\mathcal{E}}) = \prod_{e=1}^{M}\prod_{v=1}^{N} p\left(\mathbf{H}_{ve} \mid \mathbf{z}_v, \hat{\mathbf{z}}_e\right) = \prod_{e=1}^{M}\prod_{v=1}^{N} \text{Sigmoid}\left(\mathbf{z}_v^T \hat{\mathbf{z}}_e\right), \quad (6)$$

where $\hat{\mathbf{z}}_e = \text{MLP}_{\varphi}(\boldsymbol{\beta}^{(e)})$ is obtained by another deterministic MLP parameterized by φ with $\boldsymbol{\beta}^{(e)}$ as the input. This formulation enables the decoder to estimate the probability of each node v being part of hyperedge e based on the learned node embeddings $\mathbf{z}_v$ and the transformed hyperedge traits $\hat{\mathbf{z}}_e$, thus facilitating the reconstruction of the high-order correlations within the hypergraph efficiently.

4.2.3 Multi-Objective Optimization. Using variational inference [23, 25], we optimize the neural hyperedge linker by maximizing the evidence lower bound (ELBO):

$$\text{ELBO} = \mathbb{E}_{\substack{q_\phi(\boldsymbol{\beta}|\mathbf{Z}_{\mathcal{E}}) \\ e\in\mathcal{E}, v\in\mathcal{V}}} \log p_{\theta,\varphi}(\mathbf{H}_{ve} \mid \mathbf{z}_v, \hat{\mathbf{z}}_e) - \text{KL}(q_\phi(\boldsymbol{\beta}^{(e)} \mid \mathbf{z}_e) \mid p(\boldsymbol{\beta}^{(e)})), \quad (7)$$

where the prior distribution of hyperedge traits follows a Dirichlet distribution parameterized by ones, i.e., $p(\boldsymbol{\beta}^{(e)}) \sim \text{Dirichlet}(\mathbf{1})$. Detailed derivation is provided in Appendix A. As $\boldsymbol{\alpha}^{(e)}$ is obtained via a deterministic function, the KL divergence term $\text{KL}(q_\phi(\boldsymbol{\beta}^{(e)}|\mathbf{z}_e) \mid p(\boldsymbol{\beta}^{(e)}))$ simplifies to $\text{KL}(q(\boldsymbol{\beta}^{(e)}|\boldsymbol{\alpha}^{(e)}) \mid p(\boldsymbol{\beta}^{(e)}))$:

$$\begin{aligned}\text{KL}\left(q\left(\boldsymbol{\beta}^{(e)}|\boldsymbol{\alpha}^{(e)}\right) \mid p\left(\boldsymbol{\beta}^{(e)}\right)\right) &= \log \frac{\Gamma\left(\sum_{i=1}^{K} \boldsymbol{\alpha}_i^{(e)}\right)}{\prod_{i=1}^{K} \Gamma\left(\boldsymbol{\alpha}_i^{(e)}\right)} \\ - \log\Gamma(K) + \sum_{i=1}^{K}\left(\boldsymbol{\alpha}_i^{(e)} - 1\right)&\left(\psi\left(\boldsymbol{\alpha}_i^{(e)}\right) - \psi\left(\sum_{j=1}^{K} \boldsymbol{\alpha}_j^{(e)}\right)\right).\end{aligned} \quad (8)$$

Here, Γ denotes the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}\,dt$, and ψ represents the digamma function $\psi(x) = \frac{d}{dx}\ln(\Gamma(x)) = \frac{\Gamma'(x)}{\Gamma(x)}$. Moreover, the parameters θ of HyperGNN for inferring $\mathbf{Z}_{\mathcal{V}}$ and $\mathbf{Z}_{\mathcal{E}}$ are additionally optimized by the node labels $\mathbf{Y}$ which navigates distinguishing node representations across different node categories. The overall loss of neural hyperedge linker is defined as:

$$\mathcal{L}_{\text{NHL}} = -\lambda\text{ELBO} + \sum_{v\in\mathcal{V}} \hat{y}_v \log y_v + (1-\hat{y}_v)\log(1-y_v), \quad (9)$$

where $\hat{y}_v = \text{Softmax}(\mathbf{W}_v\mathbf{z}_v + \mathbf{b}_v)$ is the prediction probability of node v over each category obtained through a linear layer with softmax activation, and λ is a hyperparameter to control the weight.

4.2.4 Generation. Once the neural hyperedge linker is optimized using the original hypergraph $\mathcal{T}$, it can be employed to synthesize structures in the condensed hypergraph $\mathcal{S}$. The neural hyperedge linker supports two generation schemas. (i) **Additive Expansion**: In this schema, we infer the latent traits of the expected hyperedges and predict the nodes with the highest similarity scores as new incorporations into the hyperedges. This method relies on the learned embeddings to identify nodes that are most likely to be part of the new hyperedges. (ii) **Stochastic Composition**: Alternatively, we can sample a posterior Dirichlet distribution either randomly or from existing distributions in $\mathcal{T}$ and select the nodes of the hyperedge based on this sampled posterior. This approach introduces stochasticity in the generation process, allowing for diverse and potentially more varied hyperedge compositions.

In both schemas, we determine that a node v is connected to a hyperedge e when $\text{Sigmoid}\left(\mathbf{z}_v^T\hat{\mathbf{z}}_e\right) > \sigma$. The threshold σ is a hyperparameter that controls the sparsity of the hyperedges. These two generation schemas enable the neural hyperedge linker to create new structures in the condensed hypergraph $\mathcal{S}$, leveraging the learned representations and probabilistic sampling to ensure coherent and meaningful expansions. This formulation provides a clear and structured approach to generating new hyperedges, balancing deterministic node selection with stochastic sampling for flexibility and diversity in the generated hypergraphs.

4.3 Hypergraph Coreset Initialization

The initialization of the condensed hypergraph $\mathcal{S}$ plays a pivotal role in the process of hypergraph condensation. A judiciously selected initialization can substantially enhance the efficiency and efficacy of subsequent condensation steps, thereby ensuring that the resultant structure preserves the essential relationships inherent in the original hypergraph $\mathcal{T}$. To accomplish this objective, we propose a hypergraph coreset initialization method leveraging node embeddings $\mathbf{Z}_{\mathcal{V}}$ from the optimized neural hyperedge linker.

4.3.1 Node Exemplars Selection. We employ an iterative selection process for each node category $c \in \{0, ..., C-1\}$. This process involves choosing exemplars that optimize the approximation of the mean embedding of the category's training nodes by the average embedding vector of all exemplars [32]. This selection continues until the target number, m_c, is attained. The target number for each node category is determined based on the distribution of training nodes across categories in the original hypergraph $\mathcal{T}$.

4.3.2 Initial Structure Synthesis. However, a critical challenge rises in this initialization process. Due to the selection of a limited set of representative nodes from each category, the majority of hyperedges present in $\mathcal{T}$ are either absent or only partially represented in the initialized structure. Consequently, the original hypergraph structure diverges significantly from the initialized one, potentially impeding subsequent condensation steps. To address this issue, we employ the stochastic composition method delineated in Section 4.2.4 to synthesize hyperedges of $\mathcal{S}$.

For a comprehensive overview of our hypergraph coreset initialization algorithm, please refer to Algorithm 2.

4.4 Synthetic Hypergraph Amelioration

Despite the fact that the coreset initialization method retains the most salient nodes and facilitates the synthesis of plausible hyperedges, the resultant synthesized hypergraph $\mathcal{S}$ possesses information that is inherently constrained by the boundaries set by the original hypergraph $\mathcal{T}$. To mitigate the information discrepancy between $\mathcal{S}$ and $\mathcal{T}$, we propose a comprehensive amelioration target for $\mathcal{S}$, encompassing multiple aspects including node attributes, higher-order structures, and node labels.

4.4.1 Gradient-Parameter Synergistic Matching. To achieve comparable performance between HyperGNNs trained on the synthetic hypergraph $\mathcal{S}$ and those trained on the target hypergraph $\mathcal{T}$, it is essential that the models trained on $\mathcal{S}$ exhibit similar optimization trajectories and converge to comparable final parameters as those trained on $\mathcal{T}$. This alignment in optimization dynamics and parameter space serves as a guiding principle for optimizing the structure of the synthetic hypergraph $\mathcal{S}$. Based on this premise, we propose a Gradient-Parameter synergistic matching objective

that concurrently evaluates both the optimization pathway and the parameter landscape. This unified approach facilitates a comprehensive preservation of information from $\mathcal{T}$.

Our approach involves preparing a training trajectory of an expert HyperGNN trained on $\mathcal{T}$, saving both the model parameters and gradients for T iterations, denoted as $\{\boldsymbol{\theta}_t^{\mathcal{T}}\}_{t=0}^{T}$ and $\{\nabla_{\boldsymbol{\theta}_t^{\mathcal{T}}}\mathcal{L}_{\mathcal{T}}\}_{t=0}^{T}$, respectively. Concurrently, we train a student HyperGNN on $\mathcal{S}$ with identical parameter initialization $\boldsymbol{\theta}_0 : \boldsymbol{\theta}_0^{\mathcal{S}} = \boldsymbol{\theta}_0^{\mathcal{T}}$, obtaining the student parameters $\{\boldsymbol{\theta}_t^{\mathcal{S}}\}_{t=0}^{T}$ and gradients $\{\nabla_{\boldsymbol{\theta}_t^{\mathcal{S}}}\mathcal{L}_{\mathcal{S}}\}_{t=0}^{T}$. Given the training trajectories of both expert and student HyperGNNs, we design appropriate objectives to measure the variance between these two trajectories, which serve as signals to optimize the node features $\mathbf{X}'$ of condensed hypergraph $\mathcal{S}$. A crucial aspect of this optimization is the direction of parameter optimization (i.e., gradients). We quantify the variance between the gradients of the student parameters and expert parameters at step t via cosine similarity:

$$\texttt{Gradient Matching:}\quad \mathcal{L}_{\text{Gr}}^{(t)} = 1 - \frac{\nabla_{\boldsymbol{\theta}_t^{\mathcal{S}}}\mathcal{L}_{\mathcal{S}} \cdot \nabla_{\boldsymbol{\theta}_t^{\mathcal{T}}}\mathcal{L}_{\mathcal{T}}}{\|\nabla_{\boldsymbol{\theta}_t^{\mathcal{S}}}\mathcal{L}_{\mathcal{S}}\|\|\nabla_{\boldsymbol{\theta}_t^{\mathcal{T}}}\mathcal{L}_{\mathcal{T}}\|}, \tag{10}$$

where $\mathcal{L}_{\text{Gr}}^{(t)}$ is computed as the layer-wise sum of cosine similarity in gradients for each HyperGNN layer. Furthermore, to promote convergence of the student HyperGNN's performance towards that of the expert, we calculate the normalized difference between the student and expert parameters across current training trajectories:

$$\texttt{Parameter Matching:}\quad \mathcal{L}_{\text{Tr}}^{(t)} = \frac{||\boldsymbol{\theta}_t^{\mathcal{S}} - \boldsymbol{\theta}_t^{\mathcal{T}}||_2^2}{||\boldsymbol{\theta}_0 - \boldsymbol{\theta}_t^{\mathcal{T}}||_2^2}. \tag{11}$$

The optimization target incorporates these two components. We posit that gradient matching is more crucial in the early stages of training trajectories, as gradients determine the optimization direction. Conversely, as training progresses, the alignment of student and expert parameters becomes increasingly significant, ultimately determining the final performance of the student model. Based on this premise, we formulate a balanced loss objective that dynamically weights these two aspects for hypergraph synthesis:

$$\mathcal{L}_{\text{Syn}} = \mathbb{E}_{\boldsymbol{\theta}_0 \sim \mathcal{P}_{\boldsymbol{\theta}^{\mathcal{T}}}} \sum_{t=0}^{T} \left[\cos\left(\frac{\pi t}{2T}\right)\mathcal{L}_{\text{Gr}}^{(t)} + \sin\left(\frac{\pi t}{2T}\right)\mathcal{L}_{\text{Tr}}^{(t)} \right], \tag{12}$$

where $\cos(\frac{\pi t}{2T})$ decreases from 1 to 0 and $\sin(\frac{\pi t}{2T})$ increases from 0 to 1 as step t increases. This balanced approach eliminates the need for additional hyperparameters and prevents any single aspect from unduly dominating the optimization process. The objective $\mathcal{L}_{\text{Syn}}$ is estimated given different parameter initializations $\boldsymbol{\theta}_0$ from the random initialization distribution $\mathcal{P}_{\boldsymbol{\theta}^{\mathcal{T}}}$.

4.4.2 Adaptive High-order Structure Synthesis. Given the discrete nature of high-order structures in hypergraphs, we focus on updating the node features $\mathbf{X}'$ of $\mathcal{S}$ based on the gradients $\partial\mathcal{L}_{\text{Syn}}/\partial\mathbf{X}'$. As the node features evolve, the likelihood of hyperedge formations also changes. To mitigate potential instability arising from substantial alterations, we employ the additive expansion (as illustrated in §4.2.4) of the neural hyperedge linker to update existing hyperedges, represented by $\mathbf{H}'$. Furthermore, to maintain the stability of the condensed hypergraph $\mathcal{T}$, we implement a strategy wherein the high-order structure is updated only once per several feature update steps τ.

4.4.3 Label Distillation. Inspired by knowledge distillation techniques [37, 49], which transfer expert knowledge from large-scale neural networks to smaller ones, we adopt a similar approach for label prediction transfer. Specifically, we initialize the expert HyperGNN with its final parameters $\boldsymbol{\theta}_T^{\mathcal{T}}$ and predict node class probabilities $\mathbf{Y}'_{\boldsymbol{\theta}_T^{\mathcal{T}}}$ over the updated condensed hypergraph $\mathcal{S}$, incorporating the updated node features $\mathbf{X}'$ and high-order structures $\mathbf{H}'$ as $\mathbf{Y}'_{\boldsymbol{\theta}_T^{\mathcal{T}}} = \text{HyperGNN}_{\boldsymbol{\theta}_T^{\mathcal{T}}}(\mathbf{X}', \mathbf{H}')$. To facilitate the convergence of the student model's performance towards that of the expert, we utilize the expert HyperGNN's predictions as signals to update the labels of the condensed hypergraph S. We quantify this difference using the Kullback-Leibler Divergence:

$$\mathcal{L}_{\mathbf{Y}'} = D_{\text{KL}}(\mathbf{Y}'||\mathbf{Y}'_{\boldsymbol{\theta}_T^{\mathcal{T}}}) = \sum\nolimits_{v \in \mathcal{V}'} \mathbf{Y}'_v \log \mathbf{Y}'_v / \mathbf{Y}'_{\boldsymbol{\theta}_T^{\mathcal{T}}, v}. \tag{13}$$

Subsequently, we update the labels $\mathbf{Y}'$ based on gradient $\partial\mathcal{L}_{\mathbf{Y}'}/\partial\mathbf{Y}'$. A comprehensive schematic algorithm detailing our hypergraph condensation process is provided in Appendix B.

5 Experiments

In this section, we present a comprehensive empirical evaluation to demonstrate the efficacy of our proposed hypergraph condensation framework, HG-Cond, in hypergraph size reduction tasks. Our investigation aims to address the following five research questions: **RQ1**: To what extent does HG-Cond effectively reduce hypergraph size across diverse real-world hypergraphs? **RQ2**: How does the performance of various HyperGNNs, when trained on the reduced hypergraph and evaluated on the original hypergraph, compare to baseline size reduction methods? **RQ3**: What is the efficacy of the Neural Hyperedge Linker in capturing higher-order correlations within the hypergraph structure? **RQ4**: What is the relative contribution of each module to the overall performance of HG-Cond? **RQ5**: In what ways does the reduced hypergraph facilitate and enhance neural architecture search processes?

5.1 Experimental Settings

Datasets. We evaluate the condensation performance of our framework on five hypergraphs, *i.e.*, Cora-CA (co-authorship) [42], DBLP-CA (co-authorship) [42], Citeseer (co-citation) [42], Pubmed (co-citation) [42], 20News [9]. For all five datasets, we use the public splits and setups following the transductive settings [9]. The dataset statistics are shown in Table A1, and descriptions of these datasets can be found in Appendix C.

Baselines. We compare our method to five baselines: (1) Coreset selection methods: Random, Herding [41], K-center [12, 35]. (2) Hypergraph Coarsening methods: HyperEF [1], HyperSF [2][1]. Please note that as the state-of-the-art graph condensation methods, such as SFGC [48] and GCOND [21] require either pair-wise edge reconstruction or graph neural tangent kernel [11] to condensing graph and hence cannot applied to hypergraph datasets. The detailed baseline descriptions can be found in Appendix C.

Implementation. For each dataset, we initially employ these size reduction methods to synthesize condensed hypergraph, subsequently assessing the performance of different HyperGNN architectures by training these HyperGNNs on the condensed hypergraph

[1]Hypergraph coarsening baselines only focused on hypergraph structure, unable to handle node features. We add adaptive improvements.

Table 1: Node classification accuracy performance (ACC%±std) comparison between size reduction baselines and HG-Cond as well its variants on various types of real-world hypergraphs with different condensation ratios. indicates the performance outperforms the original test accuracy on the whole dataset training.

Dataset	Ratio (r)	Size Reduction Baselines					Hypergraph Condensation			Whole Dataset
		Random	Herding	K-Center	HyperEF	HyperSF	HG-Cond-X	HG-Cond-NHL	HG-Cond	
Cora-CA	7.00%	56.56±4.7	66.03±2.4	68.27±2.8	51.72±3.2	52.63±2.9	69.64±1.6	71.42±2.4	**75.82±1.8**	78.80±2.1
	5.00%	53.90±3.9	61.43±2.6	63.68±2.2	47.63±3.8	49.15±3.3	66.39±1.4	67.27±0.9	**69.41±1.2**	
	3.00%	44.83±5.6	55.11±3.1	56.43±2.5	44.83±2.2	44.30±2.8	57.47±1.9	58.30±0.8	**62.51±1.3**	
DBLP-CA	1.00%	59.58±3.0	82.73±0.7	80.74±0.7	65.82±3.2	63.18±2.6	85.32±1.3	87.05±0.9	**89.72±1.4**	91.37±0.3
	0.50%	51.44±2.6	78.51±1.4	74.81±1.2	60.24±1.8	62.70±1.8	79.08±1.8	81.28±0.5	**83.34±1.6**	
	0.10%	36.36±3.0	58.87±2.2	54.77±3.7	51.84±1.9	50.72±1.5	62.73±2.8	63.83±2.4	**68.50±1.8**	
Citeseer	7.00%	43.76±5.9	57.98±2.6	59.61±2.2	46.32±2.7	48.91±1.9	62.58±1.3	62.84±1.6	**65.97±0.8**	71.21±3.9
	5.00%	41.43±6.3	55.48±2.0	57.43±1.8	42.21±3.7	45.82±2.5	58.62±1.0	59.39±0.7	**62.75±1.3**	
	3.00%	34.50±6.4	51.79±2.2	53.12±2.2	38.28±0.8	35.85±2.4	54.61±1.4	56.80±1.1	**58.13±1.3**	
Pubmed	7.00%	74.74±2.0	83.51±0.4	83.53±0.5	69.14±0.5	71.83±1.3	84.79±0.8	85.20±0.7	**88.76±0.3**	88.55±0.3
	5.00%	70.97±2.7	82.15±0.5	82.37±0.3	69.20±1.7	70.25±0.7	82.85±0.4	82.72±0.8	**86.38±1.3**	
	3.00%	63.58±3.0	80.61±0.6	81.23±0.6	68.72±0.5	68.94±0.7	81.74±0.4	82.56±0.5	**83.07±1.4**	
20News	5.00%	72.61±1.3	77.70±0.8	77.37±0.9	68.33±1.5	67.21±1.2	79.46±0.7	79.84±0.5	**81.26±0.7**	80.68±0.6
	3.00%	67.89±2.7	77.00±1.0	76.68±0.8	67.53±0.8	67.48±0.6	78.52±0.3	79.08±0.6	**79.95±0.5**	
	1.00%	64.29±3.2	74.93±1.4	74.67±1.2	65.04±2.8	64.92±1.9	76.83±0.7	77.84±1.3	**78.64±0.8**	

and evaluating them on the original hypergraph. In the *condensation* phase, teacher HyperGNNs and student HyperGNNs all employ AllDeepSets [9]. In the *evaluation* phase, we train a HyperGNN on the condensed hypergraph and evaluate its performance on the test part of original hypergraph. All HyperGNNs used in our experiments are all 2-layer models. We report the average value of ten independent runs for each dataset. Additional implementation details are illustrated in Appendix C.

5.2 Size Reduction Performance (RQ1)

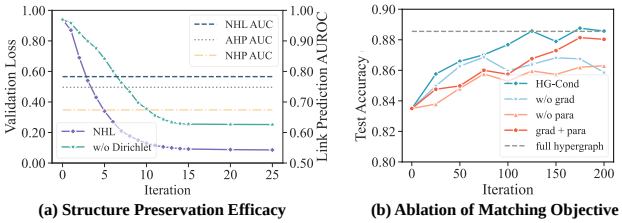
We conduct a comprehensive comparison of our proposed HG-Cond against baselines across various reduction ratios for node classification tasks, as presented in Table 1. Our approach demonstrates superior performance, achieving state-of-the-art results across all experimental scenarios and delivering improvements ranging from 2.46% to 9.63% within equivalent reduction ratios. Notably, we attain lossless condensation outcomes in Pubmed and 20News datasets, surpassing the performance achieved when training on the original, uncondensed hypergraph at a 7% and 5% reduction ratios, respectively. For the remaining datasets, our HG-Cond method effectively condenses hypergraphs, enabling HyperGNNs to achieve comparable performance relative to training on the complete hypergraphs. These findings substantiate that our proposed HG-Cond provides informative supervision signals derived from training trajectories and efficiently captures high-order connectivity through our novel neural hyperedge linker. Consequently, this allows us to obtain an optimal substitute for the original large-scale hypergraph.

5.3 Cross-Architecture Generalization (RQ2)

We conducted a comprehensive evaluation of the condensed hypergraphs generated by our HG-Cond method and baseline approaches across various HyperGNN architectures. These models are trained on the condensed hypergraphs and subsequently tested on them. The results, presented in Table 2, demonstrate that our condensed hypergraphs do not exhibit overfitting to the specific HyperGNN

Table 2: Generalization of reduced hypergraphs across HyperGNN architectures. Performance evaluated via node classification accuracy (%). 'Whole training' indicates use of entire training set. AllDeepSets indicates that the synthetic hypergraph is condensed via AllDeepSets.

Dataset	Method	Architecture			
		HyperGCN	HGNN	HNHN	AllDeepSets
Cora-CA (r = 7.00%)	Herding	65.03	66.87	64.29	66.03
	K-center	66.27	69.11	66.79	68.27
	HyperSF	50.80	52.91	51.84	52.63
	HG-Cond	**73.64**	**77.15**	**71.97**	**75.82**
	Whole Training	77.54	79.69	75.21	78.80
DBLP-CA (r = 1.00%)	Herding	81.39	82.59	78.84	82.73
	K-center	79.44	88.65	77.10	80.74
	HyperSF	61.74	63.29	59.36	63.18
	HG-Cond	**88.52**	**89.83**	**84.68**	**89.72**
	Whole Training	89.17	90.99	86.95	91.37
Citeseer (r = 7.00%)	Herding	57.62	57.90	57.47	57.98
	K-center	59.35	59.42	60.29	59.61
	HyperSF	50.31	49.07	48.27	48.91
	HG-Cond	**66.17**	**66.73**	**65.80**	**65.97**
	Whole Training	70.73	71.84	71.53	71.21
Pubmed (r = 7.00%)	Herding	81.92	81.63	82.80	83.51
	K-center	81.76	81.58	82.31	83.53
	HyperSF	69.86	70.22	70.08	71.83
	HG-Cond	**83.47**	**87.90**	**86.32**	**88.76**
	Whole Training	82.92	86.72	85.83	88.55
20News (r = 5.00%)	Herding	75.53	75.71	74.83	77.70
	K-center	74.92	75.48	75.63	77.37
	HyperSF	65.42	66.97	66.50	67.21
	HG-Cond	**79.82**	**80.04**	**79.85**	**81.26**
	Whole Training	79.23	78.94	80.15	80.68



(a) Structure Preservation Efficacy (b) Ablation of Matching Objective

Figure 3: (a) Evaluation of structure preservation efficacy on the Pubmed dataset in comparison with alternative hyperedge link prediction baselines. (b) Ablation analysis of the gradient-parameter synergistic matching objective conducted on the Pubmed dataset ($r = 7\%$).

architecture employed during the condensation phase. Instead, they exhibit robust generalization capabilities across diverse HyperGNN architectures. HyperGNNs trained on our condensed hypergraphs consistently and significantly outperform those trained on hypergraphs reduced by other baseline methods. Notably, our condensed hypergraphs achieve lossless performance in 8 out of 20 experimental cases, a finding that underscores the potential for widespread real-world application of hypergraph condensation techniques.

5.4 Structure Preservation Efficacy (RQ3)

To assess the efficacy of our proposed neural hyperedge linker (NHL) in capturing the intrinsic high-order connectivity patterns within hypergraphs, we conduct a comparative analysis against several baseline methods: (i) We evaluate a variant of our approach that employs a Gaussian distribution as the prior, in contrast to our primary method which utilizes a Dirichlet distribution (denoted as 'w/o Dirichlet'). (ii) We benchmark our method against established techniques without node labels, *i.e.*, NHP [43] and AHP [19], which rely on raw Graph Convolutional Network (GCN) layers and adversarial training, respectively. Following [19], we implement size-based sampling for negative hyperedges. Figure 3(a) illustrates the validation loss trajectories of our neural hyperedge linker and its variant without the Dirichlet prior, alongside the final AUROC performance metrics for link prediction across all methods on the Pubmed dataset. The results provide compelling evidence that our method significantly outperforms previous link prediction approaches by a substantial margin. Moreover, our approach demonstrates superior training efficiency compared to the variant using a Gaussian prior, which is susceptible to posterior collapse.

5.5 Ablation Study (RQ4)

Evaluating high-order correlation acquisition. To assess the significance of higher-order correlations in condensing small-informative hypergraphs, we conduct a comparative analysis of HG-Cond against two variant models: (i) HG-Cond-X, which optimizes solely node features, with the backbone HyperGNN trained exclusively on node features utilizing self-loops as structural elements. (ii) HG-Cond-NHL, which selects coreset nodes and their associated hyperedges, maintaining them unaltered during the optimization of node features and labels. The results presented in Table 1 yield two key insights: (1) Higher-order correlations play a crucial role in enabling HyperGNNs to effectively learn node representations. (2) The optimization of higher-order correlations during the condensation process yields substantial performance enhancements.

Table 3: Architecture Search Performance. Methods evaluated using Pearson correlation between validation and test accuracies of searched architectures. 'Whole' denotes architecture search using the entire dataset.

Dataset	Pearson Correlation				
	Herding	K-center	HyperSF	HG-Cond	Whole
Cora-CA	0.42	0.39	0.23	**0.69**	0.74
Citeseer	0.40	0.43	0.21	**0.58**	0.68
Pubmed	0.51	0.48	0.31	**0.84**	0.82

Evaluating gradient-parameter synergistic matching objective. To investigate the effects of the gradient-parameter synergistic matching objective in the optimization phase of synthetic hypergraphs, we formulate three variants: (1) elimination of gradient matching (w/o grad), (2) elimination of parameter matching (w/o para), and (3) simple summation of these two objectives (grad + para). As depicted in Figure 3(b), our proposed joint matching objective demonstrates enhanced performance in condensed hypergraphs, attributable to the efficacious guidance provided by the gradient and parameter matching objectives during various optimization stages of the student HyperGNNs.

5.6 Architecture Search (RQ5)

Our focus is on architecture search for AllSetTransformer, given its more complex hyperparameter settings compared to AllDeepSets. The search space encompasses: (i) Hidden dimensions: {32, 64, 128, 256, 512}, (ii) Number of heads: {1, 2, 4, 8}, (iii) Activation functions {Sigmoid, ReLU, Linear, Softplus, LeakyReLU}. This yields 100 architectural variants. Table 3 presents the Pearson correlation between validation accuracies on reduced hypergraphs and test accuracies on raw hypergraph test sets for all variants. The search process was conducted on Cora-CA, Citeseer, and Pubmed datasets, utilizing 7.00% reduced rate hypergraphs generated by each method. Results indicate that reduced hypergraphs synthesized by HG-Cond exhibit significantly higher Pearson correlations compared to baselines, even surpassing the model trained on the entire dataset in the Pubmed case. These findings underscore the efficacy of HG-Cond in producing reduced hypergraphs that retain the essential structural and predictive characteristics of their large-scale counterparts.

6 Conclusion

This study investigated a crucial and innovative challenge: the condensation of large-scale, real-world hypergraphs into smaller, more informative structures. The aim is to enable HyperGNNs trained on these condensed hypergraphs to achieve performance comparable to those trained on the original, larger hypergraphs. To tackle this issue, we proposed a novel hypergraph condensation framework called HG-Cond. Within this framework, we integrated a coreset approach to initialize the condensed hypergraph. Additionally, we developed a multi-aspectual amelioration objective, which utilized the gradient and training trajectories of a HyperGNN to optimize both synthetic node attributes, structures, and labels. Furthermore, we introduced a neural hyperedge linker to synthesize hyperedges as a function of node attributes, thereby reducing the number of spaces and enhancing inference efficiency. Our comprehensive experimental results demonstrated the effectiveness of our proposed framework in hypergraph condensation tasks.

References

[1] Ali Aghdaei and Zhuo Feng. 2022. HyperEF: Spectral hypergraph coarsening by effective-resistance clustering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[2] Ali Aghdaei, Zhiqiang Zhao, and Zhuo Feng. 2021. Hypersf: Spectral hypergraph coarsening via flow-based local clustering. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[4] Zalán Borsos, Mojmir Mutny, and Andreas Krause. 2020. Coresets via bilevel optimization for continual learning and streaming. *Advances in neural information processing systems* 33 (2020), 14879–14890.

[5] Chen Cai, Dingkang Wang, and Yusu Wang. 2021. Graph Coarsening with Neural Networks. In *International Conference on Learning Representations*.

[6] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4750–4759.

[7] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. 2021. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*. PMLR, 1695–1706.

[8] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 257–266.

[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*.

[10] Yihe Dong, Will Sawin, and Yoshua Bengio. 2020. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278* (2020).

[11] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. 2019. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems* 32 (2019).

[12] Reza Zanjirani Farahani and Masoud Hekmatfar. 2009. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media.

[13] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.

[14] Sichao Fu, Weifeng Liu, Yicong Zhou, and Liqiang Nie. 2019. HpLapGCN: Hypergraph p-Laplacian graph convolutional networks. *Neurocomputing* 362 (2019), 166–174.

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

[16] Zheng Gong, Guifeng Wang, Ying Sun, Qi Liu, Yuting Ning, Hui Xiong, and Jingyu Peng. 2023. Beyond homophily: robust graph anomaly detection via neural sparsification. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2104–2113.

[17] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. 2019. Cascade-bgnn: Toward efficient self-supervised representation learning on large-scale bipartite graphs. *arXiv preprint arXiv:1906.11994* (2019).

[18] Jing Huang and Jie Yang. 2021. UniGNN: a Unified Framework for Graph and Hypergraph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

[19] 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*. 2237–2242.

[20] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. 2022. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 720–730.

[21] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. Graph Condensation for Graph Neural Networks. In *International Conference on Learning Representations*.

[22] Yu Jin, Andreas Loukas, and Joseph JaJa. 2020. Graph coarsening with preserved spectral properties. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4452–4462.

[23] Diederik P Kingma. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[24] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[25] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[26] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *International conference on machine learning*. PMLR, 1945–1954.

[27] Shiye Lei and Dacheng Tao. 2023. A comprehensive survey of dataset distillation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[28] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. 2009. Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 527–536.

[29] Brian Moser, Federico Raue, Jörn Hees, and Andreas Dengel. 2022. Less is more: Proxy datasets in nas approaches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1953–1961.

[30] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. 2021. Dataset Meta-Learning from Kernel Ridge-Regression. In *International Conference on Learning Representations*.

[31] 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 24*. Springer, 607–619.

[32] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[33] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*. PMLR, 1278–1286.

[34] Noveen Sachdeva and Julian McAuley. 2023. Data Distillation: A Survey. *Transactions on Machine Learning Research* (2023).

[35] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

[36] Daniel A Spielman and Shang-Hua Teng. 2011. Spectral sparsification of graphs. *SIAM J. Comput.* 40, 4 (2011), 981–1025.

[37] Yijun Tian, Shichao Pei, Xiangliang Zhang, Chuxu Zhang, and Nitesh V Chawla. 2023. Knowledge distillation on graphs: A survey. *arXiv preprint arXiv:2302.00219* (2023).

[38] Pengfei Wang, Dian Jiao, Leyou Yang, Bin Wang, and Ruiyun Yu. 2024. Hypergraph-based Truth Discovery for Sparse Data in Mobile Crowdsensing. *ACM Transactions on Sensor Networks* 20, 3 (2024), 1–23.

[39] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2018. Dataset Distillation. *arXiv preprint arXiv:1811.10959* (2018).

[40] Tianxin Wei, Yuning You, Tianlong Chen, Yang Shen, Jingrui He, and Zhangyang Wang. 2022. Augmentations in hypergraph contrastive learning: Fabricated and generative. *Advances in neural information processing systems* 35 (2022), 1909–1922.

[41] Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*. 1121–1128.

[42] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems* 32 (2019).

[43] 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*. 1705–1714.

[44] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the web conference 2021*. 413–424.

[45] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*.

[46] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6514–6523.

[47] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. Dataset Condensation with Gradient Matching. In *International Conference on Learning Representations*.

[48] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. 2023. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems* 36 (2023).

[49] Yuanxin Zhuang, Lingjuan Lyu, Chuan Shi, Carl Yang, and Lichao Sun. 2022. Data-Free Adversarial Knowledge Distillation for Graph Neural Networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. 2441–2447.

A Theoretical Background

In Section 4.2, we present the evidence lower bound of our Neural Hyperedge Linker, as formalized in Equation 7. In the following, we elaborate on the derivation of this objective. The goal of our Neural Hyperedge Linker is to capture complex high-order connectivity patterns. Given the node and hyperedge representations derived from the HyperGNN, we seek to maximize the log-likelihood function $\log p(\mathbf{H}|\mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}})$. By integrating over the latent variable $\boldsymbol{\beta}$, which represents the underlying characteristics governing the high-order interactions between nodes and hyperedges, the objective can be derived as:

$$\begin{aligned}\log p(\mathbf{H}|\mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}}) &= \log \int p_{\theta,\varphi}(\mathbf{H} \mid \mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}}, \boldsymbol{\beta}) p(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta} \\ &= \log \int \frac{q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}})}{q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}})} p_{\theta,\varphi}(\mathbf{H} \mid \mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}}, \boldsymbol{\beta}) p(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta}.\end{aligned} \tag{14}$$

Since the stochastic variable $\boldsymbol{\beta}$ is obtained by the deterministic encoder given the hyperedge representations $\mathbf{Z}_{\mathcal{E}}$, the above expression can be derived as following:

$$\begin{aligned}&\log p(\mathbf{H}|\mathbf{Z}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{E}}) \\ &= \log \int \frac{q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}})}{q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}})} p_{\theta,\varphi}(\mathbf{H} \mid \mathbf{Z}_{\mathcal{V}}, \boldsymbol{\beta}) p(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta} \\ &\overset{(i)}{\geq} \int q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}}) \log \frac{p_{\theta,\varphi}(\mathbf{H} \mid \mathbf{Z}_{\mathcal{V}}, \boldsymbol{\beta}) p(\boldsymbol{\beta})}{q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}})} \mathrm{d}\boldsymbol{\beta} \\ &= \int q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}}) \log p_{\theta,\varphi}(\mathbf{H} \mid \mathbf{Z}_{\mathcal{V}}, \boldsymbol{\beta}) + \int q_\phi(\boldsymbol{\beta}) \log \frac{p(\boldsymbol{\beta})}{q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}})} \mathrm{d}\boldsymbol{\beta} \\ &= \mathbb{E}_{q_\phi(\boldsymbol{\beta}|\mathbf{Z}_{\mathcal{E}})} \log p_{\theta,\varphi}(\mathbf{H} \mid \mathbf{Z}_{\mathcal{V}}, \boldsymbol{\beta}) - \mathrm{KL}(q_\phi(\boldsymbol{\beta} \mid \mathbf{Z}_{\mathcal{E}}) \mid p(\boldsymbol{\beta})) \\ &= \mathbb{E}_{\substack{q_\phi(\boldsymbol{\beta}|\mathbf{Z}_{\mathcal{E}}) \\ e\in\mathcal{E}, v\in\mathcal{V}}} \log p_{\theta,\varphi}(\mathbf{H}_{ve} \mid \mathbf{z}_v, \boldsymbol{\beta}^{(e)}) - \mathrm{KL}(q_\phi(\boldsymbol{\beta}^{(e)} \mid \mathbf{z}_e) \mid p(\boldsymbol{\beta}^{(e)})) \\ &\overset{(ii)}{=} \mathbb{E}_{\substack{q_\phi(\boldsymbol{\beta}|\mathbf{Z}_{\mathcal{E}}) \\ e\in\mathcal{E}, v\in\mathcal{V}}} \log p_{\theta,\varphi}(\mathbf{H}_{ve} \mid \mathbf{z}_v, \hat{\mathbf{z}}_e) - \mathrm{KL}(q_\phi(\boldsymbol{\beta}^{(e)} \mid \mathbf{z}_e) \mid p(\boldsymbol{\beta}^{(e)})),\end{aligned} \tag{15}$$

where Step (i) follows from the application of Jensen's inequality, while step (ii) results from the deterministic decoding of the latent traits of hyperedges into the representation $\hat{\mathbf{z}}_e$. The final expression is referred to as the Evidence Lower Bound (ELBO). We maximize the loglikelihood of high-order structures within original hypergraphs by maximizing the ELBO.

B Algorithm and Time Complexity Analysis

Figure 2 provides a succinct overview of the comprehensive condensation procedure employed in HG-Cond. To enhance clarity and facilitate comprehension, we delineate the systematic protocol for the overall synthesis of hypergraph $\mathcal{S}$ in Algorithm 1. This section presents a time complexity analysis of the hypergraph condensation procedure. Primarily, the time complexity associated with training a HyperGNN is typically $O(M|e| + N)$ for most architectures, such as UNIGAT and AllDeepSets, where M and N denote the number of hyperedges and nodes in the hypergraph, respectively, and $|e|$ represents the average number of nodes within a hyperedge. The time complexity for training our neural hyperedge linker is equivalent to that of training HyperGNNs, as it transitions the prediction from the set of nodes (i.e., hyperedge) to the node level. Subsequently, the coreset initialization method, as elucidated in Algorithm 2, initializes the condensed hypergraph $\mathcal{S}$. This process entails distance calculations between nodes and category centers, resulting in a time complexity of $O(NN')$, where N' represents the number of nodes in the condensed hypergraph. During the optimization of node features, high-order structures, and label distributions, the time complexity for each optimization iteration is $O(M'|e| + N' + |\boldsymbol{\theta}| + \frac{M'N'}{\tau})$. Here, $O(M'|e| + N')$ denotes the inference time of the student HyperGNN, $O(|\boldsymbol{\theta}|)$ represents the time complexity of calculating the gradient-parameter synergistic objective, and $O(\frac{M'N'}{\tau})$ signifies the time complexity of updating hypergraph structures. Considering T iterations and R different parameter initializations of student and teacher HyperGNNs, we multiply $O(M'|e|+N'+|\boldsymbol{\theta}|+\frac{M'N'}{\tau})$ by TR. In conclusion, we observe that the time complexity of hypergraph condensation increases linearly with the number of nodes in the original hypergraph.

Algorithm 1: Overall synthesis procedure of condensed hypergraph $\mathcal{S}$

Input: Original hypergraph $\mathcal{T} = \{\mathbf{X}, \mathbf{H}, \mathbf{Y}\}$ with N nodes, A neural hyperedge linker and node embeddings $\{\mathbf{z}_v | v \in \mathcal{V}\}$ both trained by $\mathcal{L}_{\text{NHL}}$, condensed hypergraph node size N', hyperedge size M', structure update interval τ

Output: Condensed hypergraph $\mathcal{S} = \{\mathbf{X}', \mathbf{H}', \mathbf{Y}'\}$

▷ Train a neural hyperedge linker (§4.2) with the optimization target (Eqn. (7))
▷ Initialize condensed hypergraph $\mathcal{S}$ through algorithm 2
while not converged **do**
  ▷ Initialize the parameters of expert HyperGNN $\boldsymbol{\theta}_0^{\mathcal{T}}$ and student HyperGNN $\boldsymbol{\theta}_0^{\mathcal{S}}$ randomly $\boldsymbol{\theta}_0^{\mathcal{S}} = \boldsymbol{\theta}_0^{\mathcal{T}}$
  ▷ Train expert HyperGNN on $\mathcal{T}$ and obtain trajectory with parameters $\{\boldsymbol{\theta}_t^{\mathcal{T}}\}_{t=0}^{T}$ and gradients $\{\nabla_{\boldsymbol{\theta}_t^{\mathcal{T}}} \mathcal{L}_{\mathcal{T}}\}_{t=0}^{T}$
  ▷ Train student HyperGNN on $\mathcal{S}$ and obtain trajectory with parameters $\{\boldsymbol{\theta}_t^{\mathcal{S}}\}_{t=0}^{T}$ and gradients $\{\nabla_{\boldsymbol{\theta}_t^{\mathcal{S}}} \mathcal{L}_{\mathcal{S}}\}_{t=0}^{T}$
  ▷ Measure the variance of the two trajectories $\mathcal{L}_{\text{Syn}}$ (Eqn. (12))
  ▷ Update node attributes $\mathbf{X}'$ through the gradient $\partial \mathcal{L}_{\text{Syn}} / \partial \mathbf{X}'$
  if *Update Times % τ == 0* **then**
    ▷ Update structure $\mathbf{H}'$ of $\mathcal{S}$ by the additive expansion of neural hyperedge linker
  ▷ Infer the expert prediction $\mathbf{Y}'_{\boldsymbol{\theta}_T^{\mathcal{T}}} = \text{HyperGNN}_{\boldsymbol{\theta}_T^{\mathcal{T}}}(\mathbf{X}', \mathbf{H}')$
  ▷ Update node labels $\mathbf{Y}'$ through the gradient $\partial \mathcal{L}_{\mathbf{Y}'} / \partial \mathbf{Y}'$ (Eqn. (13))

C Additional Experiment Details

C.1 Dataset Description

We conduct extensive experiments to evaluate HG-Cond on five real-world hypergraphs. The statistics of these datasets are shown in Table A1. Detailed descriptions of datasets are provided below:

Algorithm 2: Coreset initialization of condensed hypergraph $\mathcal{S}$

Input: Original hypergraph $\mathcal{T} = \{\mathbf{X}, \mathbf{H}, \mathbf{Y}\}$ with N nodes, A neural hyperedge linker and node embeddings $\{\mathbf{z}_v | v \in \mathcal{V}\}$ both trained by $\mathcal{L}_{\text{NHL}}$, condensed hypergraph node size N', hyperedge size M'

Output: Initialization of condensed hypergraph $\mathcal{S}$

for category $c = 1$ to C **do**

▷ select the embeddings $\mathcal{Z}_c = \{\mathbf{z}_v | v \in \mathcal{V}, y_v = c\}$ of nodes belong to class c

$\mu_c = \frac{1}{|\mathcal{Z}_c|} \sum_{\mathbf{z}_v \in \mathcal{Z}_c} \mathbf{z}_v$ // Current class mean

$n_c = \frac{N'}{N} \times |\mathcal{Z}_c|$ // condensed node size of class c

for $i = 1$ to n_c **do**

$$p_{c,i} \leftarrow \underset{\mathbf{z}_v \in \mathcal{Z}_c}{\operatorname{argmin}} \left\| \mu_c - \frac{1}{i} \left[\mathbf{z}_v + \sum_{j=1}^{i-1} p_{c,i} \right] \right\|$$

$P_c \leftarrow (p_{c,1}, ..., p_{c,n_c})$

$\mathbf{X}' = [P_1, ..., P_C]$ // condensed node embedding of $\mathcal{S}$

$\mathbf{Y}' = [\underbrace{1, ..., 1}_{n_1}, ..., \underbrace{C, ..., C}_{n_C}]$ // condensed node labels of $\mathcal{S}$

▷ Synthesize high-order relationships

for $i = 1$ to M' **do**

▷ Sample hyperedge latent traits $\alpha^{(e)}$ through K-center methods [12]

▷ Select nodes that satisfy Sigmoid $\left(\mathbf{z}_v^T \hat{\mathbf{z}}_e\right) > \sigma$ as the constituent nodes for this hyperedge

- **Cora-CA** [42]: This dataset represents a citation network where nodes correspond to academic articles and hyperedges represent authors. Nodes within a hyperedge signify the articles in which the respective author has contributed.
- **DBLP-CA** [42]: Similar in structure to Cora-CA, this dataset also represents a citation network where nodes correspond to academic articles and hyperedges represent authors.
- **Citeseer** [42]: Another citation network, where nodes represent academic articles and hyperedges correspond to source articles. Nodes encompassed by a hyperedge denote the articles cited by the respective source article.
- **Pubmed** [42]: Similar in structure to Citeseer, this dataset also represents a citation network. Nodes represent articles, and hyperedges denote source articles. The nodes within a hyperedge indicate the articles cited by the respective source article.

Table A1: Details of dataset statistics. $|e|$ denotes the average node number in a hyperedge.

Dataset	# Node	# Hyperedge	# Feature	Avg. $\lvert e \rvert$	# Node Class
Cora-CA	2,708	1,072	1,433	4.28	7
DBLP-CA	41,302	22,363	1,425	4.45	6
Citeseer	3,327	1,079	3,703	3.20	6
Pubmed	19,717	7,963	500	4.35	3
20News	16,242	100	100	654.51	4

- **20News** [9]: The 20News is a representation of the popular 20Newsgroups dataset using hypergraph structure. In the hypergraph, documents are typically represented as nodes, while the newsgroup categories and important terms or topics serve as hyperedges connecting related documents.

C.2 Baseline Description

In this subsection, we present an extensive introduction to the baselines in hypergraph size reduction. The evaluated methods can be categorized into two distinct classes: coreset methods and coarsening methods, each distinguished by their operational mechanisms.

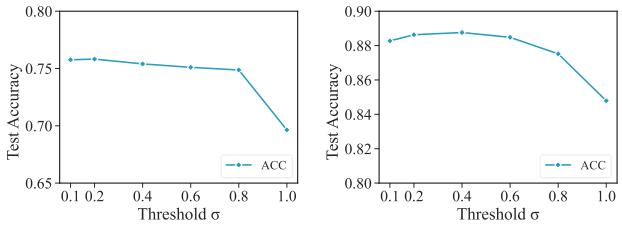
Coreset methods. For coreset approaches, we initially employ an AllDeepSets model trained on the original hypergraphs to obtain the final embeddings of each node prior to the final linear classifier. Subsequently, we select nodes from the original hypergraph and induce a sub-hypergraph from these selected nodes, which serves as the reduced hypergraph. The following methods are considered: (i) **Random** method employs a stochastic approach to node selection. (ii) **Herding** method is frequently utilized in continual learning [41], this technique selects samples that exhibit the closest proximity to the cluster center, based on the Euclidean distance of their embeddings. (iii) **K-center** method [12] selects center samples with the objective of minimizing the maximum distance between a sample and its nearest center.

Coarsening methods. Two prominent coarsening methods are evaluated: (i) **HyperSF** [2] is a spectral hypergraph coarsening method, which aims to preserve the original spectral properties of hypergraphs. HyperSF employs a strongly-local max-flow-based clustering algorithm to identify sets of nodes that minimize the ratio cut. (ii) **HyperEF** [1] decomposes large hypergraphs into multiple node clusters with minimal inter-cluster hyperedges. HyperEF approximates hyperedge effective resistances by searching within the Krylov subspace. It is noteworthy that both HyperSF and HyperEF were originally designed for hypergraphs without node features. To adapt these methods to our hypergraph size reduction tasks, we modify the distance calculation by incorporating the Euclidean distance of embeddings where applicable.

C.3 Hyperparameter Settings

Neural Hyperedge Linker. For the neural hyperedge linker, we employ a two-layer UNIGAT [18] as the backbone to derive representations of hyperedges and nodes. The dropout rate is set to 0.1, with 256 hidden units. We optimize the dimension K of prior and posterior Dirichlet distributions across {16, 32, 64} for each dataset. In optimizing the Neural Hyperedge Linker loss, we implement size-based negative sampling [19] to sample negative hyperedges for optimizing the log-likelihood of the incidence matrix $\mathbf{H}$ in the raw hypergraph.

Condensation Phase. For each dataset, we utilize a two-layer AllDeepSets [9] as the backbone to condense the hypergraph, maintaining 256 hidden units. The condensed hypergraphs are optimized over 500 iterations. We fine-tune the learning rate for node features within {0.01, 0.001, 0.0005, 0.0001} and for node labels within {0, 0.001, 0.0001}, where a label learning rate of 0 indicates fixed node labels from the original hypergraphs. For each iteration, we optimize



(a) Cora (r=7%) **(b) Pubmed (r=7%)**

Figure 4: Test accuracy under different hyperedge sparsity hyperparameter σ.

the length T of training trajectories within 200, 500, based on the convergence of the HyperGNN trained on the original hypergraphs. Besides, we update condensed hypergraphs' high-order structure at intervals τ, tuned in {5, 10, 20} and the hyperedge sparsity parameter σ is tuned in the range of {0.2, 0.4, 0.6}.

Evaluation Phase. During evaluation, we train HyperGNNs with diverse architectures on the condensed hypergraphs, validating and testing them on the original hypergraphs. We set the dropout rate to 0 and weight decay to 0.0005 when training various HyperGNNs. The training process extends to 500 epochs, with early stopping implemented after 20 epochs.

C.4 More Experiment Results

Effects of Sparsity Hyperparameter σ. Figure 4 illustrates the impact of varying the hyperedge sparsity hyperparameter σ on test accuracy for two datasets: Cora and Pubmed, both with a reduction rate r = 7%. For the Cora dataset, we observe there is a notable decline in test accuracy when σ = 1.0, i.e., each node has only self-loop structures. This circumstance is consistent in Pubmed when $\sigma > 0.6$. These results indicate that excessive sparsity may negatively impact the model's predictive capabilities.

Running Time. We present an analysis of HG-Cond's computational time across varying condensation rates for the Cora-CA and 20News datasets, as illustrated in Table A2. For Cora-CA, we examine condensation rates of 7.00%, 5.00%, and 3.00%, while for 20News, we investigate rates of 5.00%, 3.00%, and 1.00%. The runtime measurements are based on 100 iterations, conducted on a single NVIDIA A6000 GPU, complemented by two 2.87GHz AMD EPYC 7543 32-core processors. Across all reduction rates and datasets, the complete condensation process (encompassing 500 iterations) required to generate a 1.00% condensed hypergraph of DBLP-CA consumes most time costs, approximately 3.5 hours. The computational expenditure is justified by the significant advantages offered by the generated condensed hypergraphs.

Table A2: Running time of HG-Cond for 100 iterations across different condensation rates in Cora-CA and 20News.

r	7.00%	5.00%	3.00%	r	5.00%	3.00%	1.00%
Cora-CA	128.6s	87.8s	69.3s	20News	861.3s	625.9s	513.7s