# EDGEMASK-HGNN: LEARNING TO SPARSIFY HYPERGRAPHS FOR SCALABLE NODE CLASSIFICATION IN HYPERGRAPH NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hypergraph Neural Networks (HGNNs) have achieved remarkable performance in various learning tasks involving hypergraphs— a data model for higher-order relationships across diverse domains and applications. However, the scalability of HGNNs is limited by the computational and memory demands incurred by dense hypergraph structures. Existing unsupervised sparsifiers address the scalability issue but sacrifice downstream predictive performance. To address this, we propose **EdgeMask-HGNN**, a novel framework that introduces a learnable, task-aware sparsification mechanism to reduce the hypergraph size while preserving predictive performance. EdgeMask-HGNN offers two distinct masking: a fine-grained node-hyperedge masking and a coarse-grained hyperedge-level masking, both trained end-to-end using supervision from the downstream task. We provide theoretical analysis showing that our approach (i) yields stable model outputs under stochastic masking, and (ii) ensures convergence of retention probabilities under gradient descent. Extensive experiments on multiple node classification benchmarks demonstrate that EdgeMask-HGNN reduces or maintains memory usage on both small- and large-scale hypergraphs without sacrificing accuracy, and in some cases outperforms HGNNs trained on full hypergraphs. EdgeMask-HGNN also consistently outperforms unsupervised sparsification baselines such as random, degree-based, and spectral sparsification.

## 1 INTRODUCTION

Hypergraph Neural Networks (HGNNs) have emerged as powerful tools for learning on hypergraphs, where a single edge can connect multiple nodes (beyond pairwise connections), unlike traditional graphs. There are numerous real-world instances of such relations involving multiple entities: set of researchers collaborating on a paper (Han et al., 2009), set of products purchased in a shopping cart (Xia et al., 2021), group of legislators co-sponsoring a bill (Benson et al., 2018), or set of molecules participating together in biological processes (Gaudelet et al., 2018). With the rapid growth of data, the scale of real-world hypergraphs has also expanded dramatically. For instance, the open-source bibliographic database OpenAlex has $\sim$209 million scholarly works (hyperedges) from $\sim$13 million authors (node) across more than 450 topics[1]. A wide range of learning problems arises in this setting, from node classification (Yadati et al., 2019; Duta et al., 2023) to node clustering (Chodrow et al., 2021) to hyperlink prediction (Yadati et al., 2020). HGNNs have proven to be effective for addressing these tasks, demonstrating strong empirical performance across diverse domains such as social networks (Wang et al., 2018), recommendation systems (Wang et al., 2021), and bioinformatics (Deng et al., 2024).

Despite the success of Hypergraph neural networks, they incur high computational and memory complexity in terms of scaling to large-scale hypergraphs due to dense incidence structures. These costs arise from sparse-dense matrix multiplications during message-passing and the storage of intermediate activations during forward and backward passes. *Sparsifying the hypergraph by pruning redundant node–hyperedge links offers a promising solution by significantly reducing computational and memory overhead while retaining key structural information.*

[1] https://en.wikipedia.org/wiki/OpenAlex

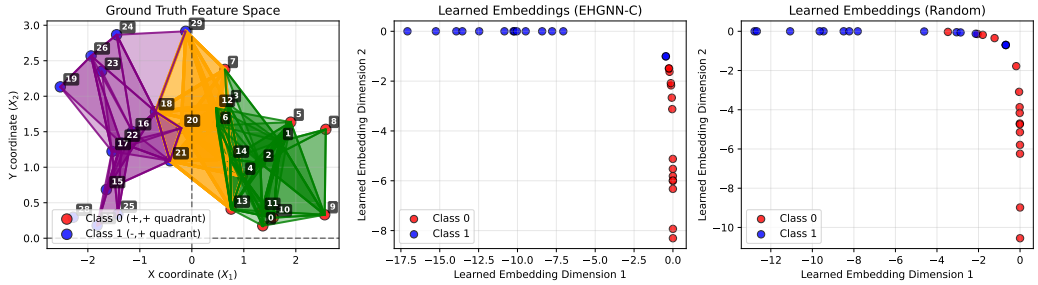



Figure 1: A synthetic 3-uniform hypergraph (#nodes=30,#edges=134) with their (x,y)-coordinates in $\mathbb{R}^2$ as ground truth node features (left). 50% hyperedges were pruned to learn node embeddings via EdgeMask-HGNN (middle) and via Random sparsification (right). The node embeddings learned by EdgeMask-HGNN are well separable, and more aligned with the true class separation.

A naïve way to sparsify would be dropping hyperedges uniformly at random, often termed as *random sparsification* in graph literature (Das et al., 2025). However, this method does not exploit the structure of the hypergraph. Inspired by graph literature, *degree-proportionate sampling* (Leskovec & Faloutsos, 2006) samples nodes based on their degree in the hypergraph, constructing a sparse subhypergraph. However, this method may not preserve the spectral properties of the hypergraph and may remove nodes that are important for message passing for downstream tasks. *spectral sparsifiers* (Soma & Yoshida, 2019) sample nodes based on hypergraph effective resistance to preserve eigenvalues and eigenvectors of the original hypergraph. However, this approach may preserve task-irrelevant hyperedges since it does not exploit label information.

**Our contributions.** (I) In this paper, we propose EdgeMask-HGNN to fill this gap of a lack of task-specific sparsification in HGNNs. EdgeMask-HGNN offers two distinct masking strategies: a fine-grained node–hyperedge masking and a coarse-grained hyperedge-level masking, both trained end-to-end using supervision from the downstream task to facilitate task-aware sparsification. Unlike unsupervised methods (e.g., random, degree-based, and spectral sparsification) that rely on fixed hypergraph structures sampled a priori, EdgeMask-HGNN dynamically selects a subset of hyperedges during training that helps the downstream HGNN perform better on the learning task. Figure 1 highlights the importance of leveraging task-specific signal during sparsification: under the same sparsification budget, embeddings learned by EdgeMask-HGNN yield better class separation than the random sparsifier that does not utilize label information.

(II) We theoretically analyze EdgeMask-HGNN to show that it has a probabilistic interpretation in terms of sampling the mode of the distribution over sparse subhypergraphs. We also show that the sparsification (a) produces stable model outputs, and (b) has an $\mathcal{O}(1/\epsilon)$ convergence rate to an $\epsilon$-confidence threshold on the learned retention probabilities.

(III) Empirical evaluation and analysis show that EdgeMask-HGNN is more effective than alternative sparsifiers such as random, degree distribution-based, and effective resistance sparsifiers, while also being competitive or superior in terms of memory and runtime. Compared to training on the full hypergraph, EdgeMask-HGNN sacrifices little accuracy and often achieves better predictive performance, while maintaining or reducing memory usage on both small- and large-scale hypergraphs.

## 2 Related Works

**Hypergraph Neural Networks (HGNNs).** Feng et al. (2019) proposed HGNN by generalizing spectral graph convolutions to hypergraphs through the hypergraph Laplacian. Subsequently, several variants have been proposed, such as HyperGCN (Yadati et al., 2019), and HNHN (Dong et al., 2020). HyperGCN reformulates hypergraph convolution using clique expansion, and HNHN introduces attention mechanisms for hyperedge importance. *spatial* HGNNs typically define two-stage message aggregation: node to hyperedge and hyperedge to node. HyperSAGE (Arya et al., 2020), HGNN+ (Gao et al., 2022), and UniGCN (Huang & Yang, 2021) belong to this category. Recently, Chien et al. (2021) showed that propagation rules of many existing spatial HGNNs can be represented as a composition of two multiset functions, and proposed two different multiset encoding functions: DeepSets and SetTransformer. In addition, several recent frameworks, such as

Wang et al. (2025); Saxena et al. (2024), focus on general and expressive message-passing designs for hypergraphs. Our work targets resource-efficient learning by identifying and preserving only task-relevant hyperedges, making it a practical augmentation for such general-purpose designs. Interested readers may refer to recent surveys for a more in-depth discussion on HGNNs (Antelmi et al., 2023; Kim et al., 2024).

**Unsupervised and Spectral Hypergraph Sparsification.** Graph sparsification has a long legacy, starting with Benczúr & Karger (1996) for preserving cuts via edge sampling. Subsequently, significant progress was made by Spielman & Teng (2011) and Spielman & Srivastava (2008), who introduced spectral sparsification via effective resistance. These methods ensure that a small subset of edges preserves global graph properties. Among earlier works on hypergraph sparsification, Deveci et al. (2013) proposed hyperedge sampling heuristics to reduce hyperedges while preserving cut structure for hypergraph partitioning tasks. Going beyond heuristics, Soma & Yoshida (2019) extended spectral sparsification to hypergraphs, defining a nonlinear Laplacian quadratic form and constructing $\epsilon$-spectral sparsifiers of size $\mathcal{O}(n^3/\epsilon^2)$ in polynomial time. Subsequent works such as Bansal et al. (2019); Kapralov et al. (2021); Lee (2023) attempted to improve this bound. Most recently, Kapralov et al. (2022) proved a significant result showing that it is possible to obtain an $\epsilon$-spectral sparsifier of linear size.

Unlike EdgeMask-HGNN, these methods are unsupervised in nature and do not incorporate node labels into their sparsification decision. Since sparsification is done a priori to training, they are downstream task-unaware. Hence, they may not be suitable for representation learning tasks on hypergraphs that require preserving task-relevant hyperedges.

## 3 Problem Statement

We consider the semi-supervised node classification task on a hypergraph $\mathcal{H} \triangleq (V, E, \boldsymbol{X}) \equiv (\boldsymbol{H}, \boldsymbol{X})$, where $V$ is the set of nodes, $E$ is the set of hyperedges, $\boldsymbol{H}$ is the incidence matrix of $\mathcal{H}$, and $\boldsymbol{X} \in \mathbb{R}^{n \times F}$ is the node feature matrix. A hyperedge $e \in E$ is a subset of $V$. Let $V_L, V_U$ be the set of labeled and unlabeled nodes, respectively, and $\boldsymbol{y}_L \in \{1, \ldots, C\}^{|V_L|}$ be the label vector for the labeled nodes. The goal is to learn a *model* $f_{\theta^*} : V_U \to \{1, \ldots, C\}$ that predicts labels for nodes in $V_U$ based on the labels $\boldsymbol{y}_L$ of the labeled nodes by minimizing a loss function $\mathcal{L}_{\text{task}}$ (e.g. cross-entropy): $\theta^* = \arg\min_\theta \mathcal{L}_{\text{task}}(f_\theta(\boldsymbol{H}, \boldsymbol{X}), \boldsymbol{y}_L)$.

Given an budget, the goal of supervised sparsification is to construct a sparse hypergraph $\hat{\mathcal{H}} = (\tilde{\boldsymbol{H}}, \boldsymbol{X})$ such that the prediction accuracy of the downstream task can be preserved or even improved by using $\hat{\mathcal{H}}$ (instead of $\mathcal{H}$) as input. This can be formulated as the following optimization problem:

$$\theta^* = \arg\min_\theta \mathcal{L}_{\text{task}}(f_\theta(\tilde{\boldsymbol{H}}, \boldsymbol{X}), \boldsymbol{y}_L),$$

where $\tilde{\boldsymbol{H}}$ is the masked incidence matrix produced by a learnable sparsification module. The downstream task loss could be any classification loss, e.g., the cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|V_L|} \sum_{v \in V_L} \sum_{c=1}^{|C|} \boldsymbol{Y}_{vc} \log \hat{\boldsymbol{Y}}_{vc},$$

where $Y_{vc}$ indicates the true probability of node $v$ belonging to class $c \in C$, and the predicted probabilities expressed as $\hat{\boldsymbol{Y}} = f_\theta(\tilde{\boldsymbol{H}}, \boldsymbol{X})$. Due to the uniqueness of hypergraphs, the budget constraint could be in the form of # incidence-pairs or # hyperedges. Let $k$ denotes the #node-hyperedge connections retained after sparsification. As edges in a hypergraph may contain more than 2 nodes, we shall denote by $\kappa$ the corresponding #hyperedges retained after sparsification. We denote the # node-hyperedge connections before sparsification by $t = \sum_{e \in \mathcal{H}} |e|$.

## 4 EdgeMask-HGNN: Learnable Hypergraph Sparsification

In this section, we introduce EdgeMask-HGNN, a task-aware, learnable sparsification framework for hypergraph neural networks. The main idea is to devise a differentiable module that can learn to selectively mask hyperedges based on their relevance to the downstream learning task. Unlike

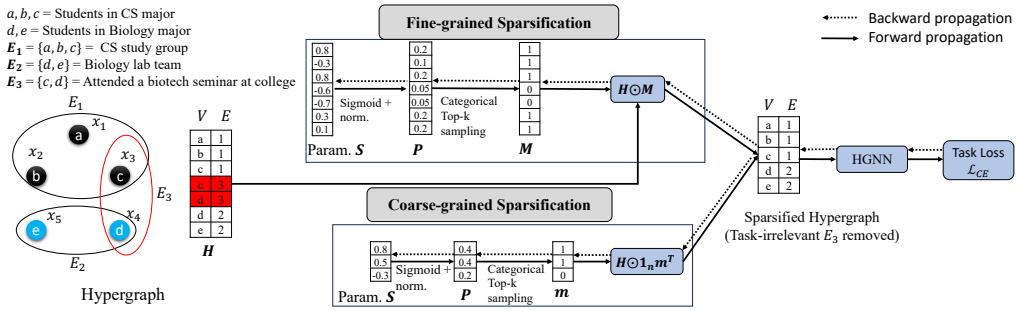



Figure 2: EdgeMask-HGNN on a small example where a task-irrelevant edge (red) is pruned.

prior hypergraph pruning methods that use fixed heuristics, EdgeMask-HGNN is trained end-to-end, allowing the model to jointly optimize both the hypergraph structure and parameters of HGNNs. Figure 2 illustrates the schematic of EdgeMask-HGNN with an example.

Since hyperedges may contain more than two nodes, two types of learnable masking strategies emerge: *Incidence-level Masking* and *Edge-Level Masking*. Incidence-level Masking learns a score for each (node, hyperedge) pair in the hypergraph, whereas Edge-Level Masking learns a score for each hyperedge. The scores are used to compute the sparsified hypergraph, which in turn is used to train the downstream HGNN. Based on the task loss of the downstream HGNN, the gradients are backpropagated down to compute the updated incidence-level or edge-level scores.

### 4.1 Fine-grained Incidence-level Masking (EHGNN-F)

Fine-grained incidence-level masking focuses on learning a soft selection probability for individual node–edge connections so as to achieve a fine-grained control over the sparsified hypergraph structure. For each incidence pair $(v, e)$, we maintain a learnable score parameter $s_{v,e}$, which is converted to a soft selection probability as

$$p_{v,e} = \sigma(s_{v,e}) / \sum_{v,e} \sigma(s_{v,e}), \tag{1}$$

where $\sigma$ is the sigmoid function. $\boldsymbol{p} = (p_{v,e})$ can be interpreted as the marginal probabilities of a categorical distribution over all incidence entries, from which we aim to sample a prescribed number of active pairs, denoted by constraint $k$. In other words, if the sparsifier samples $(v, e)$, the hard binary mask $\hat{m}_{v,e} = 1$ and 0, otherwise. Mathematically,

$$\hat{\boldsymbol{m}} \sim \text{Top-k } \texttt{Categorical}(\boldsymbol{p}), \ \sum_{v,e} \hat{m}_{v,e} = k. \tag{2}$$

This sampling process is not differentiable during the backward pass. To enable differentiable learning, we apply a straight-through estimator (Bengio et al., 2013). During the forward pass, we compute $p_{v,e}$ for each node–edge pair and use top-$k$ sampling to generate a hard binary mask $\hat{m}_{v,e}$, where only the top-$k$ values are retained. During the backward pass, we define the final mask as:

$$m_{v,e} = \text{stop\_grad}(\hat{m}_{v,e}) + p_{v,e} - \text{stop\_grad}(p_{v,e}). \tag{3}$$

This ensures that the forward pass uses the hard mask $\hat{m}_{v,e}$, while the backpropagation treats it as if it were the continuous probability $p_{v,e}$, allowing gradient-based optimization. The sparse matrix is:

$$\tilde{\boldsymbol{H}} = \boldsymbol{H} \odot \boldsymbol{M}, \tag{4}$$

where $\boldsymbol{M} \in \{0, 1\}^{n \times m}$ is the sampled incidence-level mask, and $\odot$ indicates element-wise multiplication. To avoid notational conflict, in subsequent theoretical sections we define $\boldsymbol{P} \in [0, 1]^{n \times m}$ as the matrix of marginal retention probabilities, and write: $\boldsymbol{\mathcal{M}} \sim q(\boldsymbol{\mathcal{M}} \mid \boldsymbol{P})$ to denote a probability distribution over binary incidence matrices, where $\boldsymbol{P}$ specifies the marginal inclusion probabilities such that $p_{v,e} = \mathbb{P}(\boldsymbol{\mathcal{M}}_{v,e} = 1)$. A realization of this random variable ($\boldsymbol{\mathcal{M}}$) is denoted by $\boldsymbol{M} \in \{0, 1\}^{n \times m}$, used in the sparsified incidence matrix $\tilde{\boldsymbol{H}}$ (Equation 4).

**Variant of EHGNN-F conditioned on node features.** EHGNN-F is feature-agnostic, each node–hyperedge incidence $(v, e)$ has a free learnable parameter $s_{v,e}$, which do not depend on node features, rather they are optimized via task loss. In semi-supervised setting, this may cause only the incidences near labeled nodes to benefit from supervision. To address this issue, feature-conditioned EHGNN-F (**EHGNN-F (cond.)**) learns the scorer as a parameter of node features as follows:

$$s_{v,e} = \texttt{MLP}(\boldsymbol{X}_v || \hat{\boldsymbol{X}}_e), \tag{5}$$

where $\boldsymbol{X}_v$ is the feature vector of node $v$ and the aggregated embedding of hyperedge $e$ is $\hat{\boldsymbol{X}}_e = \frac{1}{|e|}\sum_{v\in e} \boldsymbol{X}_v$. The shared MLP parameters allow limited labeled nodes to shape sparsification decisions across the whole hypergraph, alleviating the lack of supervision issue.

To summarize, EHGNN-F promotes competitive selection among node–edge connections, which ensures that only the most informative relationships are retained under a global sparsity constraint. This selective pressure improves the model's ability to identify task-relevant structure while avoiding over-retention of redundant or noisy incidences.

### 4.2 Coarse-grained Edge-Level Masking (EHGNN-C)

Although Incidence-level masking allows fine-grained control, it may result in a large parameter size since the network keeps one parameter per incidence pair. Coarse-grained Edge-Level masking alleviates this by maintaining one parameter per edge. For instance, if the input hypergraph is $k$-uniform (every edge contains $k$ nodes), edge-level scoring reduces the model parameters by $k$.

The learnable edge score parameter $s$ is converted to soft probabilities via sigmoid function:

$$p_e = \sigma(s_e) / \sum_e \sigma(s_e) \tag{6}$$

We treat $\boldsymbol{p} = (p_e)$ as the marginal probabilities of a categorical distribution over all hyperedges and sample binary hyperedge masks from it:

$$\boldsymbol{m} \sim \text{Top-k } \texttt{Categorical}(\boldsymbol{p}),\ \sum_e m_e = \kappa.$$

The binary mask $m_e \in \{0, 1\}$ determines whether the hyperedge $e$ is retained or dropped. This allows efficient sparsification of hyperedges with a low-parameter overhead and is especially well suited for large-scale hypergraphs. The sparsified incidence matrix becomes:

$$\tilde{\boldsymbol{H}} = \boldsymbol{H} \odot \mathbf{1}_n \boldsymbol{m}^T, \tag{7}$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is a vector of ones, $\boldsymbol{m}^T \in \mathbb{R}^{1\times m}$ is the binary mask.

**Variant of EHGNN-C conditioned on node features.** EHGNN-C is feature-agnostic, each edge has a free learnable parameter $s_e$, which do not depend on its constituent node features. In semi-supervised setting, this may cause only the edges containing labeled nodes to benefit from supervision. To address this issue, feature-conditioned EHGNN-C (**EHGNN-C (cond.)**) learns the scorer as a parameter of node features using a permutation-invariant operator, such as mean pooling:

$$\hat{\boldsymbol{X}}_e = \frac{1}{|e|}\sum_{v\in e} \boldsymbol{X}_v, \quad s_e = \texttt{MLP}(\hat{\boldsymbol{X}}_e) \tag{8}$$

**Training and Inference.** The masked incidence $\tilde{\boldsymbol{H}}$ is passed into any HGNN architecture (e.g., HyperGCN, UniGNN, etc.) and trained end-to-end with a task loss $\mathcal{L}_{\text{task}}$. This yields a sparse hypergraph $\tilde{\boldsymbol{H}}$ adapted to the node classification task. During inference, instead of stochastic sampling, we use deterministic top-$k$ (or top-$\kappa$) selection from $\boldsymbol{p}$ to select incidences (or edges) of $\tilde{\boldsymbol{H}}$.

Unlike prior sparsification approaches for graphs or hypergraphs, EdgeMask-HGNN introduces a fully learnable, task-aware sparsification module at two granularities, operating end-to-end with downstream supervision. The fine-grained variant (EHGNN-F) allows localized selection of informative node–edge connections, while the coarse-grained variant (EHGNN-C) offers a more parameter-efficient alternative by scoring entire hyperedges via permutation-invariant pooling. Although masking strategies have been explored in the graph sparsification literature, such as $L_0$-based sparsifiers (Ye & Ji, 2021), EdgeMask-HGNN addresses the unique structure of hypergraphs, where sparsification potentially operates over exponentially many possible incidence patterns.

## 5 THEORETICAL GUARANTEES

We provide a formal analysis of EHGNN-F by examining stability, and convergence behavior. The proofs are presented in the Appendix for brevity.

**Stability and Robustness.** We prove that the model output remains stable under stochastic masking. Let $p_{v,e} \in [0,1]$ represent the marginal probability of retaining the connection $(v,e)$. A binary mask $\boldsymbol{M} \in \{0,1\}^{n\times m}$ is sampled as a realization of a random variable $\boldsymbol{\mathcal{M}} \sim q(\boldsymbol{\mathcal{M}} \mid \boldsymbol{P})$. Each realization corresponds to a subhypergraph $\tilde{\boldsymbol{H}} = \boldsymbol{H} \odot \boldsymbol{M}$, and defines a distribution over the subhypergraphs.

**Theorem 5.1** (Perturbation Stability)**.** *If the HGNN $f_\theta(\boldsymbol{X}, \cdot)$ is L-Lipschitz w.r.t. the Frobenius norm,*

$$\mathbb{E}\left[\left\| f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}) - f_\theta(\boldsymbol{X}, \mathbb{E}[\tilde{\boldsymbol{H}}]) \right\|_F\right] \leq L \cdot \sqrt{\sum_{v,e} H_{v,e}^2 p_{v,e}(1-p_{v,e})}. \tag{9}$$

Here the expectation $\mathbb{E}[\tilde{\boldsymbol{H}}]$ quantifies the average connectivity across sparsified subhypergraphs sampled from the model. This theorem suggests that, as the learned mask becomes more deterministic ($p_{v,e} \to 1$ or $0$), the model's output becomes more stable.

**Convergence of retention probabilities $p_{v,e}$.**

**Theorem 5.2** (Convergence of Fine-Grained Mask Parameters)**.** *Let the mask be defined by probabilities $\boldsymbol{P} = \sigma(\boldsymbol{S}) \in [0,1]^{n\times m}$, where each entry $p_{v,e} = \sigma(s_{v,e})$ is a sigmoid-transformed logit.*

*If the gradient of the loss with respect to each $p_{v,e}$ maintains a fixed sign and the logits are updated via gradient descent, then for any small $\epsilon > 0$, the time to convergence (either $p_{v,e} < \epsilon$ or $p_{v,e} > 1-\epsilon$) follows*

$$\tau \geq \mathcal{O}\left(1/\epsilon\right)$$

This theorem suggests that one can ensure $p_{v,e}(\tau)$ reaches within $\epsilon$ of its limiting value or at least $(1-\epsilon)$; it suffices to train for #epochs proportional to $1/\epsilon$.

## 6 EXPERIMENTS

We focus on semi-supervised node classification task in the transductive setting. Note that, we have also evaluated EdgeMask-HGNN on unsupervised node clustering task (see Appendix G). We randomly split the nodes into training/validation/test samples using 50%/25%/25% splitting percentages Chien et al. (2021). Unless stated otherwise, we have used HGNN (Feng et al., 2019) as a backbone in our implementation. We follow Chien et al. (2021) to set the hyperparameters of various base HGNN models. We average the results of 10 experiments using multiple random splits and initializations. All experiments are run on a system with 1 TB RAM and NVIDIA H200 GPU with 141 GB of HBM3e memory.

**Datasets.** The hypergraph datasets and their statistics are presented in Table 1. It includes 3 recently proposed heterophilic benchmarks (Actor, Twitch, Pokec) from Li et al. (2025). The rest of the datasets are well-known in the hypergraph learning literature, originating from works such as Yadati et al. (2019); Chien et al. (2021) where they are discussed in detail.

**Baselines.** The baseline algorithms include i) *Full*: the HGNN model is trained on the entire hypergraph, ii) *Degdist*: top-$k$ nodes are sampled from the distribution Top-$k$ `Categorical`($\boldsymbol{d}$) with $d_v$ being the normalized degree of node $v$. All node, hyperedge connections involving the sampled nodes are kept in the sparsified hypergraph, iii) *Random*: drops a node-hyperedge connection $(u,e)$ if an i.i.d uniform random variable $r \sim \mathcal{U}[0,1]$ satisfies $r < k/|E|$, and finally, iv) *Spectral*: We sample top-$\kappa$ hyperedges based on their effective resistance. We approximated effective resistance

Table 1: Dataset statistics.

| | Cora | Citeseer | Pubmed | Cora-CA | DBLP-CA | 20News | Mushroom | NTU2012 | ModelNet40 | Yelp | House | Walmart | Actor | Twitch | Pokec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert V\rvert$ | 2708 | 3312 | 19717 | 2708 | 41302 | 16242 | 8124 | 2012 | 12311 | 50758 | 1290 | 88860 | 16255 | 16812 | 14998 |
| $\lvert E\rvert$ | 1579 | 1079 | 7963 | 1072 | 22363 | 100 | 298 | 2012 | 12311 | 679302 | 341 | 69906 | 10164 | 2627 | 2406 |
| # feature | 1433 | 3703 | 500 | 1433 | 1425 | 100 | 22 | 100 | 100 | 1862 | 100 | 100 | 50 | 7 | 65 |
| # class | 7 | 6 | 3 | 7 | 6 | 4 | 2 | 67 | 40 | 9 | 2 | 11 | 3 | 2 | 2 |

$R(e) \approx \sum_{v \in e} \left(L^{\dagger}\right)_{vv}$ for efficiency purposes. Here $L^{\dagger}$ indicates the Moore–Penrose pseudoinverse of the Hypergraph Laplacian (Zhou et al., 2006). Note that there are no task-relevant methods in the HGNN literature, hence our comparison had to be limited to unsupervised approaches. The model parameters, hyperparameters of the algorithms, ablation studies, convergence of retention probabilities, and node-clustering experiments can be found in the Appendix. Our source codes: `https://github.com/toggled/ehgnn`.

### 6.1 Experimental evaluation

**I. Effectiveness and Scalability.** Table 2 highlights the effectiveness of EdgeMask-HGNN over the baselines. We observe several things:

*Supervised vs Unsupervised sparsifiers:* On most datasets, EHGNN variants consistently outperform Random and degree-based sparsification. This highlights the importance of task-aware learning of sparsification masks, which can retain task-relevant incidences or hyperedges while discarding noisy ones. On large-scale datasets, spectral methods face memory issues, which highlight the computational challenges to preserving the Laplacian spectrum.

*Full training vs Sparsification:* On several (8 out of 15) datasets, EdgeMask-HGNNs actually outperforms full HGNN training, such as ModelNet40, Actor, DBLP-CA, House, Pokec, Pubmed, Walmart, and Yelp. This demonstrates that pruning of irrelevant pairs can enhance generalization by improving the signal-to-noise ratio during message passing.

*Fine-grained vs Coarse-grained EdgeMasking and variants:* EHGNN-C is slightly better than EHGNN-F, often outperforming on datasets like 20news, CiteSeer, Cora, Cora-CA, DBLP-CA, and PubMed. Since co-authorship and co-citation datasets contain semantically coherent hyperedges (papers) and rich bag-of-words features, the model does not need fine-grained incidence-level filtering to extract meaningful training signals. Instead, pruning at the hyperedge level is sufficient. In contrast, EHGNN-F excels on ModelNet40, NTU2012 (where class counts are large), and Walmart, House, Actor, and Pokec, where hyperedges are more noisy, and may group together items or people that aren't strongly related to the prediction task. In such conditions, fine-grained incidence pruning provides a stronger signal-to-noise filtering.

**II. Memory efficiency.** Table 3 reports peak GPU memory across methods at the same sparsity budget. Note that, peak GPU usage during HGNN training is dominated by activations (messages, pooled edge features, and autograd buffers) that scale with # incidences after sparsification $k = \sum_{e \in \tilde{\mathcal{H}}} |e|$, and feature dimension $d$, not by the parameter overhead (see Table 6, Appendix B).

*On small-scale datasets:* On small datasets, all methods exhibit similar peak memory because the entire incidence structure fits comfortably on a GPU. The incidence and hyperedge counts are not large enough to pose scalability challenges, thus, sparsification strategies bring little practical dif-

Table 2: Accuracy ($\pm$ std) across different datasets for each algorithm at sparsity = 50%. OOM=Out-of-Memory. Bold (underline) denotes the best (2nd best) result per dataset not counting Full.

| Algorithms | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | CiteSeer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| Full | 81.40 ± 0.04 | 94.88 ± 0.06 | 97.76 ± 0.31 | 87.67 ± 0.24 | 64.58 ± 0.03 | 69.93 ± 0.46 | 82.25 ± 0.22 | 91.14 ± 0.04 |
| Random | 55.26 ± 0.66 | 64.03 ± 1.01 | 84.19 ± 1.25 | 59.80 ± 2.55 | 65.86 ± 0.67 | 27.29 ± 1.14 | 42.72 ± 3.33 | 52.63 ± 1.90 |
| Degdist | 59.02 ± 0.60 | 50.25 ± 1.05 | 74.19 ± 1.25 | 44.53 ± 1.45 | 63.57 ± 0.16 | 40.39 ± 0.20 | 52.64 ± 2.68 | 57.64 ± 0.26 |
| Spectral | 72.80 ± 0.02 | 82.53 ± 0.03 | **97.74 ± 0.12** | 65.45 ± 0.43 | 63.93 ± 0.03 | 35.85 ± 0.28 | 64.14 ± 0.28 | OOM |
| EHGNN-F | 78.00 ± 0.04 | 95.25 ± 0.08 | 96.42 ± 0.41 | **87.40 ± 0.18** | 76.99 ± 0.03 | 67.92 ± 0.43 | 79.38 ± 0.39 | 88.00 ± 0.10 |
| EHGNN-C | **81.10 ± 0.28** | 94.90 ± 0.08 | 96.23 ± 1.13 | 87.36 ± 0.30 | 74.07 ± 0.05 | **69.03 ± 0.33** | **82.25 ± 0.28** | **91.33 ± 0.04** |
| EHGNN-F (cond.) | 78.54 ± 0.08 | **95.52 ± 0.05** | 94.92 ± 0.61 | 87.28 ± 0.54 | **78.75 ± 0.04** | 68.43 ± 0.48 | 75.95 ± 0.59 | 86.28 ± 0.09 |
| EHGNN-C (cond.) | 74.82 ± 0.26 | 94.52 ± 0.10 | 97.46 ± 0.30 | 85.33 ± 0.41 | 77.55 ± 0.04 | 67.05 ± 0.16 | 75.39 ± 0.34 | 87.11 ± 0.18 |

| | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Avg. Rank |
|---|---|---|---|---|---|---|---|---|
| Full | 78.35 ± 0.34 | 73.99 ± 0.44 | 58.55 ± 0.05 | 85.61 ± 0.05 | 51.22 ± 0.05 | 94.78 ± 0.02 | 30.48 ± 0.68 | |
| Random | 41.57 ± 1.28 | 61.73 ± 0.71 | 53.34 ± 0.29 | 46.24 ± 0.31 | 49.88 ± 0.78 | 55.70 ± 0.20 | 29.78 ± 0.28 | 6.29 |
| Degdist | 59.32 ± 1.01 | 52.14 ± 5.18 | 54.15 ± 0.05 | 49.04 ± 0.03 | 50.44 ± 0.59 | 56.28 ± 0.14 | 27.82 ± 0.65 | 6.00 |
| Spectral | 56.90 ± 0.12 | 57.59 ± 0.54 | 52.92 ± 0.06 | 49.07 ± 0.00 | 51.05 ± 0.12 | 65.90 ± 0.03 | OOM | 5.21 |
| EHGNN-F | 74.00 ± 0.43 | 87.93 ± 0.22 | 58.70 ± 0.04 | 85.59 ± 0.03 | 50.67 ± 0.03 | 95.16 ± 0.01 | **30.57 ± 0.46** | 2.58 |
| EHGNN-C | **77.52 ± 0.28** | 74.74 ± 0.17 | 58.46 ± 0.09 | **85.69 ± 0.05** | **51.10 ± 0.04** | 93.72 ± 0.09 | 30.12 ± 0.60 | **2.21** |
| EHGNN-F (cond.) | 73.65 ± 0.44 | **100.00 ± 0.00** | **59.14 ± 0.07** | 85.55 ± 0.07 | 50.89 ± 0.04 | **98.38 ± 0.01** | 29.13 ± 0.44 | 2.43 |
| EHGNN-C (cond.) | 75.86 ± 0.47 | 73.07 ± 0.95 | 59.01 ± 0.05 | 85.48 ± 0.04 | 50.67 ± 0.03 | 94.73 ± 0.04 | 29.17 ± 0.18 | 3.29 |

Table 3: Maximum GPU memory usage (in GB) across datasets for each algorithm at sparsity = 50%. Avg. Rank computed over all datasets with OOM (Out-of-memory) treated as the worst rank.

| Algorithms | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | CiteSeer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| Full | 1.3 ± 0.0 (1.3) | 1.3 ± 0.0 (1.3) | 1.0 ± 0.0 (1.0) | 0.9 ± 0.0 (0.9) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 0.9 ± 0.0 (0.9) | 3.9 ± 0.0 (3.9) |
| Random | 1.0 ± 0.0 (1.0) | 1.0 ± 0.0 (1.0) | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.0 ± 0.0 (1.0) | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.5 ± 0.0 (1.5) |
| Degdist | 1.1 ± 0.0 (1.1) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.5 ± 0.0 (1.5) |
| Spectral | 8.0 ± 0.1 (8.1) | 7.7 ± 1.4 (8.5) | 3.0 ± 0.1 (3.0) | 1.2 ± 0.0 (1.2) | 12.0 ± 2.1 (13.3) | 1.5 ± 0.0 (1.5) | 1.3 ± 0.0 (1.3) | 64.6 ± 19.8 (76.1) |
| EHGNN-F | 1.1 ± 0.0 (1.1) | 1.1 ± 0.0 (1.1) | 1.0 ± 0.0 (1.0) | 0.8 ± 0.0 (0.8) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 0.9 ± 0.0 (0.9) | 1.7 ± 0.0 (1.7) |
| EHGNN-C | 1.2 ± 0.1 (1.3) | 1.2 ± 0.1 (1.2) | 1.0 ± 0.0 (1.0) | 0.9 ± 0.0 (0.9) | 1.2 ± 0.1 (1.2) | 0.9 ± 0.0 (0.9) | 0.9 ± 0.0 (0.9) | 2.3 ± 0.5 (2.6) |
| EHGNN-F (cond.) | 1.1 ± 0.0 (1.1) | 1.1 ± 0.0 (1.1) | 1.0 ± 0.0 (1.0) | 0.8 ± 0.0 (0.8) | 1.1 ± 0.0 (1.1) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 5.5 ± 0.0 (5.5) |
| EHGNN-C (cond.) | 1.1 ± 0.0 (1.1) | 1.1 ± 0.0 (1.1) | 1.0 ± 0.0 (1.0) | 0.8 ± 0.0 (0.8) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 0.9 ± 0.0 (0.9) | 2.4 ± 0.0 (2.4) |

| | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Avg. Rank |
|---|---|---|---|---|---|---|---|---|
| Full | 0.9 ± 0.0 (0.9) | 0.9 ± 0.0 (0.9) | 1.0 ± 0.0 (1.0) | 1.2 ± 0.0 (1.2) | 1.0 ± 0.0 (1.0) | 13.3 ± 0.0 (13.3) | 98.9 ± 21.2 (111.2) | 4.25 |
| Random | 0.8 ± 0.0 (0.8) | 0.8 ± 0.0 (0.8) | 0.9 ± 0.0 (0.9) | 1.0 ± 0.0 (1.0) | 0.9 ± 0.0 (0.9) | 5.2 ± 0.7 (5.6) | 38.9 ± 0.1 (38.9) | 1.92 |
| Degdist | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 0.9 ± 0.0 (0.9) | 1.1 ± 0.0 (1.1) | 0.9 ± 0.0 (0.9) | 8.0 ± 0.0 (8.0) | 104.4 ± 48.1 (130.1) | 2.83 |
| Spectral | 1.3 ± 0.0 (1.3) | 1.1 ± 0.0 (1.1) | 8.6 ± 1.2 (9.3) | 15.5 ± 2.5 (17.0) | 10.4 ± 1.4 (11.3) | 135.8 ± 2.2 (135.2) | OOM | 7.75 |
| EHGNN-F | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.0 ± 0.0 (1.0) | 1.1 ± 0.0 (1.1) | 1.0 ± 0.0 (1.0) | 7.2 ± 0.0 (7.2) | 110.0 ± 47.2 (135.7) | 3.00 |
| EHGNN-C | 0.9 ± 0.0 (0.9) | 0.9 ± 0.0 (0.9) | 1.0 ± 0.0 (1.0) | 1.2 ± 0.0 (1.2) | 1.0 ± 0.0 (1.0) | 7.4 ± 0.0 (7.4) | 104.9 ± 22.3 (117.8) | 3.50 |
| EHGNN-F (cond.) | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.0 ± 0.0 (1.0) | 1.2 ± 0.0 (1.2) | 1.0 ± 0.0 (1.0) | 7.1 ± 0.0 (7.1) | 138.2 ± 0.3 (138.1) | 3.83 |
| EHGNN-C (cond.) | 0.9 ± 0.0 (0.9) | 0.8 ± 0.0 (0.8) | 1.0 ± 0.0 (1.0) | 1.1 ± 0.0 (1.1) | 1.0 ± 0.0 (1.0) | 83.3 ± 65.5 (101.7) | 118.1 ± 37.3 (139.5) | 3.92 |

ference in this regime. We also find that Spectral methods require noticeably large memory in this regime.

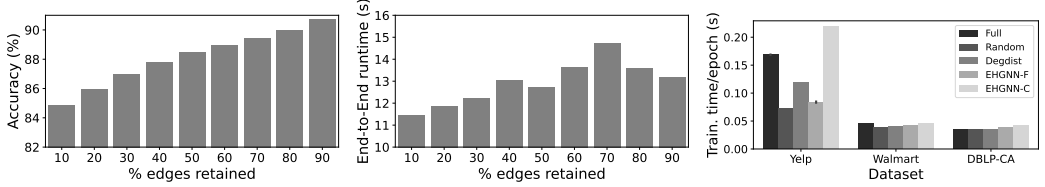
*On large, but less-dense hypergraphs (DBLP-CA, Walmart):* The differences emerge clearly on large hypergraphs where the activations dominate the memory footprint. Spectral sparsification is the most memory-hungry due to Laplacian computations on dense adjacency matrix. Random and degree-based pruning yield good memory savings, but their accuracy significantly degrades compared to Full training. EHGNN-F and EHGNN-C are better alternatives, since with a similar memory footprint as Random and degree-based pruning, they yield better accuracy.

*On large, but more-dense hypergraph (Yelp):* As density increases, the additional buffers for mask parameters and stochastic sampling cause peak activation memory to also increase. As a result, all EHGNN variants consume more memory than the full HGNN on Yelp. The effect is pronounced for feature-conditioned variants, where pooling and scorer MLPs introduce extra activations, while coarse EHGNN-C without conditioning shows smaller overhead but still exceeds the Full baseline.

*Fine- versus coarse-grained masking:* Among our methods, EHGNN-F attains the best average rank (3.00) and shows the most stable reductions overall, since incidence-level pruning directly lowers the number of active node–edge pairs $k$. EHGNN-C (3.50) can also save memory by keeping fewer parameters and retaining smaller edges, but due to randomness in sampling, sometimes it may cause large activations (large $k$) by retaining large edges, consequently increasing the memory footprint.

To summarize, memory footprint is determined by the # retained incidences $k$ and per-layer activations. EHGNN-F directly reduces $k$ and is therefore the most reliable way to shrink peak memory. EHGNN-C reduces $m$ (#edges) but can retain large hyperedges, keeping $k$ high. Feature conditioning at the hyperedge level adds pooling/MLP activations and can further increase peak usage.

**III. Accuracy/Runtime vs sparsity trade-off.** We analyze the trade-off between accuracy (on DBLP-CA) and sparsity due to edge pruning by our proposed EdgeMask-HGNN in Figure 3a, while the impact of sparsity on the end-to-end runtime (Training + Evaluation time) is presented in Figure 3b. We observe that with more edges retained in the sparsification, the accuracy increases at the cost of a higher runtime.




(a) Accuracy vs. Sparsity (b) End-to-end runtime vs. Sparsity (c) Training time

Figure 3: (a-b) Impact of sparsity on EHGNN-F's performance (DBLP-CA dataset) (c) Training time efficiency of the methods.

**IV. Runtime efficiency.** Figure 3c compares the average training time/epoch across three large-scale datasets. On Yelp, the full model incurs the highest cost (∼0.17s), while EHGNN-F reduces

Table 4: End-to-end execution time (in seconds) as mean ± std across datasets for each algorithm at sparsity = 50%.

| Algorithms | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | Citeseer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| Full | 18.37 ± 0.37 | 17.43 ± 1.08 | 17.39 ± 0.99 | 16.98 ± 1.10 | 16.90 ± 0.40 | 8.20 ± 0.14 | 10.39 ± 1.11 | 12.75 ± 0.33 |
| Random | 17.99 ± 0.38 | **16.90 ± 0.32** | 17.02 ± 0.44 | **16.78 ± 0.26** | **16.91 ± 0.32** | **7.57 ± 0.25** | **8.92 ± 0.61** | **11.29 ± 0.12** |
| Deg | 16.84 ± 0.40 | 17.10 ± 0.32 | **16.96 ± 0.27** | 16.92 ± 0.36 | 18.35 ± 0.55 | 7.76 ± 0.17 | 9.92 ± 0.63 | 12.43 ± 0.44 |
| Spectral | 74.98 ± 0.30 | 45.34 ± 1.28 | 25.17 ± 1.02 | 18.38 ± 1.35 | 88.38 ± 0.21 | 8.42 ± 0.19 | 10.66 ± 1.26 | 18.64 ± 0.32 |
| EHGNN-F | 17.89 ± 0.49 | 18.52 ± 0.72 | 18.22 ± 0.79 | 18.06 ± 0.72 | 18.50 ± 0.79 | 9.16 ± 0.49 | 10.84 ± 0.34 | 12.93 ± 0.70 |
| EHGNN-C | 17.50 ± 0.61 | 19.08 ± 0.34 | 17.52 ± 0.34 | 18.27 ± 0.32 | 19.28 ± 0.19 | 8.86 ± 0.24 | 11.15 ± 0.79 | 13.76 ± 0.78 |
| EHGNN-F (cond.) | 19.18 ± 0.13 | 18.39 ± 0.46 | 18.15 ± 0.47 | 17.90 ± 0.44 | 18.57 ± 0.73 | 9.32 ± 0.21 | 10.68 ± 0.43 | 14.51 ± 0.50 |
| EHGNN-C (cond.) | **16.22 ± 1.27** | 18.53 ± 0.37 | 18.15 ± 0.32 | 17.89 ± 0.36 | 18.49 ± 0.24 | 8.64 ± 0.45 | 11.90 ± 0.62 | 13.77 ± 0.43 |

| | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Avg. Rank |
|---|---|---|---|---|---|---|---|---|
| Full | 10.14 ± 1.23 | 18.23 ± 1.13 | 15.99 ± 1.09 | 17.25 ± 0.77 | 11.58 ± 3.04 | 22.82 ± 3.70 | 84.42 ± 1.25 | |
| Random | **8.94 ± 0.39** | 17.84 ± 0.26 | **7.62 ± 0.47** | **17.61 ± 0.46** | **7.98 ± 0.93** | **19.30 ± 0.35** | **35.88 ± 0.28** | **1.53** |
| Deg | 9.65 ± 0.10 | **16.01 ± 1.86** | 7.66 ± 0.51 | 17.64 ± 0.34 | 12.47 ± 4.02 | 20.19 ± 0.32 | 59.95 ± 2.60 | 2.13 |
| Spectral | 10.04 ± 0.26 | 18.48 ± 0.64 | 12.83 ± 0.38 | 98.80 ± 0.89 | 9.70 ± 0.24 | 62.17 ± 0.94 | OOM | 5.43 |
| EHGNN-F | 11.41 ± 0.52 | 17.36 ± 0.58 | 17.72 ± 0.29 | 18.50 ± 0.84 | 12.58 ± 3.32 | 20.84 ± 0.57 | 41.97 ± 2.92 | 4.47 |
| EHGNN-C | 10.28 ± 0.66 | 17.91 ± 0.72 | 15.91 ± 2.28 | 17.95 ± 0.30 | 14.32 ± 4.27 | 21.15 ± 0.45 | 39.75 ± 0.36 | 4.53 |
| EHGNN-F (cond.) | 10.95 ± 0.34 | 19.11 ± 0.56 | 17.90 ± 0.73 | 18.56 ± 0.15 | 14.19 ± 3.93 | 21.79 ± 0.71 | 162.61 ± 11.45 | 5.43 |
| EHGNN-C (cond.) | 11.43 ± 1.02 | 17.73 ± 0.26 | 17.71 ± 1.12 | 18.13 ± 0.26 | 11.60 ± 3.38 | 22.52 ± 0.73 | 109.33 ± 36.23 | 4.37 |

Table 5: Comparing various state-of-the-art HGNNs and their EHGNN-F enhanced counterparts (sparsity = 50%).

| **Models** | **Actor** | **Cora** | **Cora-CA** | **House** | **ModelNet40** | **NTU** | **PubMed** |
|---|---|---|---|---|---|---|---|
| AllSetTrans. | 68.77 ± 0.60 | **76.93 ± 0.41** | **83.34 ± 0.77** | **100.00 ± 0.00** | **97.80 ± 0.07** | **90.30 ± 0.71** | **88.58 ± 0.12** |
| AllSetTrans.+EHGNN-F | **85.73 ± 0.36** | 76.54 ± 0.75 | 79.41 ± 0.44 | 99.94 ± 0.14 | 97.43 ± 0.11 | 88.91 ± 0.43 | 88.49 ± 0.16 |
| CE-GAT | 70.84 ± 4.40 | **74.39 ± 0.67** | 73.26 ± 0.54 | 96.78 ± 2.69 | 90.65 ± 0.15 | **78.33 ± 1.13** | **84.76 ± 0.16** |
| CE-GAT+EHGNN-F | **74.67 ± 1.76** | 73.65 ± 0.70 | **73.83 ± 1.15** | **99.57 ± 0.28** | **92.27 ± 0.29** | 77.46 ± 1.06 | 84.45 ± 0.69 |
| CE-GCN | 57.27 ± 0.37 | 52.51 ± 0.19 | 50.66 ± 0.40 | 51.98 ± 0.39 | 43.21 ± 0.37 | 35.20 ± 0.43 | 61.01 ± 0.06 |
| CE-GCN+EHGNN-F | **62.76 ± 0.03** | **53.29 ± 0.12** | **52.44 ± 0.49** | **52.63 ± 0.31** | **44.58 ± 0.62** | **35.55 ± 0.45** | **61.08 ± 0.05** |

training time substantially, with Random being the fastest. On DBLP-CA, the EHGNN variants take slightly longer than Full HGNN. This is because our methods introduce mask-learning steps (scoring, sampling, straight-through estimation) that add a fixed computational overhead. Table 4 shows that, on small datasets (ModelNet40, NTU2012, Cora, Citeseer), message passing itself is inexpensive, so this additional overhead dominates, leading to a slightly higher runtime than Full. *On larger datasets, such as Walmart and Yelp, EHGNN achieves training times comparable to or better than Full, as incidence-level sparsification reduces the dominant message-passing cost.*

**V. Adaptability to existing HGNNs.** EdgeMask-HGNN is model-agnostic and easily adaptable to different hypergraph architectures. To demonstrate this, we have employed EHGNN-F into AllSet-Transformer, CE-GCN, and CE-GAT architectures Chien et al. (2021) and present the results in Table 5. We observe that EHGNN-F achieves comparable and sometimes better performance across these three HGNN backbones. This indicates that the sparsification mechanism is flexible and can act as a plug-in module without significantly degrading existing HGNNs' representational power.

## 7 Conclusion and Future works

We introduced EdgeMask-HGNN, a novel task-aware sparsification framework that effectively reduces memory overhead without sacrificing predictive performance of HGNNs. We proposed two learnable masking strategies: fine-grained masking and coarse-grained masking– both trained end-to-end using feedback from downstream tasks. Furthermore, EdgeMask-HGNN is theoretically grounded in terms of stability and convergence of the learned sparsifiers.

Extensive experiments across diverse and challenging benchmarks and multiple HGNN backbones underscore the adaptability and effectiveness of EdgeMask-HGNN. In particular, the fine-grained variant not only improves accuracy over full hypergraph training in many cases, but also achieves slightly smaller execution time on large hypergraphs with a comparable memory footprint. Finally, our approach consistently outperforms unsupervised and spectral baselines in accuracy, with a comparable or better memory footprint.

## REFERENCES


Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. A survey on hypergraph representation learning. *ACM Computing Surveys*, 56 (1):1–38, 2023.

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

Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 910–928. IEEE, 2019.

András A Benczúr and David R Karger. Approximating st minimum cuts in $\tilde{\mathcal{O}}(n^2)$ time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 47–55, 1996.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115 (48):E11221–E11230, 2018.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264*, 2021.

Philip S Chodrow, Nate Veldt, and Austin R Benson. Generative hypergraph clustering: From blockmodels to modularity. *Science Advances*, 7(28):eabh1303, 2021.

Siddhartha Shankar Das, Naheed Anjum Arafat, Muftiqur Rahman, SM Ferdous, Alex Pothen, and Mahantesh M Halappanavar. Sgs-gnn: A supervised graph sparsification method for graph neural networks. *arXiv preprint arXiv:2502.10208*, 2025.

Chao Deng, Hong-Dong Li, Li-Shen Zhang, Yiwei Liu, Yaohang Li, and Jianxin Wang. Identifying new cancer genes based on the integration of annotated gene sets via hypergraph neural networks. *Bioinformatics*, 40:i511–i520, 2024.

Mehmet Deveci, Kamer Kaya, and Ümit V Çatalyürek. Hypergraph sparsification and its application to partitioning. In *2013 42nd International Conference on Parallel Processing*, pp. 200–209. IEEE, 2013.

Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *Graph Representation Learning and Beyond workshop*, 2020.

Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Liò. Sheaf hypergraph networks. *Advances in Neural Information Processing Systems*, 36:12087–12099, 2023.

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

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

Thomas Gaudelet, Noël Malod-Dognin, and Natasa Przulj. Higher-order molecular organization as a source of biological function. *Bioinformatics*, 34(17):i944–i953, 2018.

Yi Han, Bin Zhou, Jian Pei, and Yan Jia. Understanding importance of collaborations in co-authorship networks: a supportiveness analysis approach. In *SIAM International Conference on Data Mining (SDM)*, pp. 1112–1123, 2009.

Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 598–611, 2021.

Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1159–1170. IEEE, 2022.

Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. A survey on hypergraph neural networks: An in-depth and step-by-step guide. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6534–6544. Association for Computing Machinery, 2024.

James R Lee. Spectral hypergraph sparsification via chaining. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pp. 207–218, 2023.

Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, 2006.

Ming Li, Yongchun Gu, Yi Wang, Yujie Fang, Lu Bai, Xiaosheng Zhuang, and Pietro Lio. When hypergraph meets heterophily: New benchmark datasets and baseline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18377–18384, 2025.

Siddhant Saxena, Shounak Ghatak, Raghu Kolla, Debashis Mukherjee, and Tanmoy Chakraborty. Dphgnn: A dual perspective hypergraph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2548–2559. Association for Computing Machinery, 2024.

Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2570–2581. SIAM, 2019.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 563–568, 2008.

Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. Session-based recommendation with hypergraph attention networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pp. 82–90. SIAM, 2021.

Yaxiong Wang, Li Zhu, Xueming Qian, and Junwei Han. Joint hypergraph learning for tag-based image retrieval. *IEEE Transactions on Image Processing*, 27(9):4437–4451, 2018.

Yifan Wang, Gonzalo R Arce, and Guangmo Tong. Generalization performance of hypergraph neural networks. In *Proceedings of the ACM on Web Conference 2025*, pp. 1273–1291, 2025.

Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI Conference on Artificial Intelligence*, pp. 4503–4511, 2021.

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

Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 1705–1714, 2020.

Yang Ye and Shihao Ji. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):905–916, 2021.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19, 2006.

# APPENDIX

## A THEORETICAL ANALYSIS

We first prove the following lemma that will be used later in Stability proof.

**Lemma A.1** (Expectation of Masked Structure)**.** *Let $\boldsymbol{P} \in [0,1]^{n\times m}$ be the matrix of marginal inclusion probabilities, i.e., $p_{v,e} = \mathbb{P}(\boldsymbol{\mathcal{M}}_{v,e} = 1)$. Then,*

$$\mathbb{E}[\tilde{\boldsymbol{H}}] = \boldsymbol{H} \odot \boldsymbol{P}.$$

*Proof.*

$$\begin{aligned} \mathbb{E}[\tilde{H}_{v,e}] &= \mathbb{E}[H_{v,e} \cdot \mathcal{M}_{v,e}] \\ &= H_{v,e} \cdot \mathbb{E}[\mathcal{M}_{v,e}] \\ &= H_{v,e} \cdot p_{v,e} \\ \Rightarrow \quad \mathbb{E}[\tilde{\boldsymbol{H}}] &= \boldsymbol{H} \odot \boldsymbol{P} \end{aligned}$$

□

**Theorem 5.1** (Perturbation Stability)**.** *If $f_\theta(\boldsymbol{X}, \cdot)$ is L-Lipschitz w.r.t. the Frobenius norm,*

$$\mathbb{E}\left[\left\| f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}) - f_\theta(\boldsymbol{X}, \mathbb{E}[\tilde{\boldsymbol{H}}])\right\|_F\right] \leq L \cdot \sqrt{\sum_{v,e} H_{v,e}^2 p_{v,e}(1-p_{v,e})}.$$

*Proof.* Since we assumed the HGNN $f_\theta$ to be $L$-Lipschitz,

$$\|f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}_1) - f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}_2)\|_F \leq L \cdot \|\tilde{\boldsymbol{H}}_1 - \tilde{\boldsymbol{H}}_2\|_F$$

for all incidence matrices $\tilde{\boldsymbol{H}}_1, \tilde{\boldsymbol{H}}_2$.

Let $\bar{\boldsymbol{H}} = \mathbb{E}[\tilde{\boldsymbol{H}}]$. As per Lemma A.1, $E[\tilde{\boldsymbol{H}}] = \boldsymbol{H} \odot \boldsymbol{P}$. Thus

$$\|f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}) - f_\theta(\boldsymbol{X}, \bar{\boldsymbol{H}})\|_F \leq L \cdot \|\tilde{\boldsymbol{H}} - \bar{\boldsymbol{H}}\|_F$$

Taking expectation

$$\mathbb{E}_{\tilde{\boldsymbol{H}}}\left[\|f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}) - f_\theta(\boldsymbol{X}, \bar{\boldsymbol{H}})\|_F\right] \leq L \cdot \mathbb{E}_{\tilde{\boldsymbol{H}}}\left[\|\tilde{\boldsymbol{H}} - \bar{\boldsymbol{H}}\|_F\right]$$

By Jensen's inequality, for any matrix $\boldsymbol{A}$

$$\begin{aligned} \mathbb{E}[\|\boldsymbol{A}\|] &\leq \sqrt{\mathbb{E}[\|\boldsymbol{A}\|^2]} \\ \Rightarrow \mathbb{E}[\|\tilde{\boldsymbol{H}} - \bar{\boldsymbol{H}}\|_F] &\leq \sqrt{\mathbb{E}[\|\tilde{\boldsymbol{H}} - \bar{\boldsymbol{H}}\|_F^2]} \end{aligned}$$

Since the entries $\tilde{H}_{v,e}$ are independent:

$$\mathbb{E}[(\tilde{H}_{v,e} - \bar{H}_{v,e})^2] = \text{Var}[\tilde{H}_{v,e}] = H_{v,e}^2 \cdot p_{v,e}(1-p_{v,e})$$

It follows that,

$$\mathbb{E}\left[\|\tilde{\boldsymbol{H}} - \bar{\boldsymbol{H}}\|_F^2\right] = \sum_{v,e} \text{Var}[\tilde{H}_{v,e}] = \sum_{v,e} H_{v,e}^2 p_{v,e}(1-p_{v,e})$$

Thus,

$$\begin{aligned} &\mathbb{E}_{\tilde{\boldsymbol{H}}}\left[\|f_\theta(\boldsymbol{X}, \tilde{\boldsymbol{H}}) - f_\theta(\boldsymbol{X}, \mathbb{E}[\tilde{\boldsymbol{H}}])\|_F\right] \\ &\leq L \cdot \sqrt{\sum_{v,e} H_{v,e}^2 \cdot p_{v,e}(1-p_{v,e})} \end{aligned}$$

□

**Theorem 5.2** (Convergence of Fine-Grained Mask Parameters)**.** *Let the mask be defined by probabilities $\boldsymbol{P} = \sigma(\boldsymbol{S}) \in [0,1]^{n\times m}$, where each entry $p_{v,e} = \sigma(s_{v,e})$ is a sigmoid-transformed logit.*

*If the gradient of the loss with respect to each $p_{v,e}$ maintains a fixed sign and the logits are updated via gradient descent, then for any small $\epsilon > 0$, the time to convergence (either $p_{v,e} < \epsilon$ or $p_{v,e} > 1-\epsilon$) follows*

$$\tau \geq \mathcal{O}\left(\frac{1}{\epsilon}\right)$$

*Proof.* Each probability $p_{v,e} \in (0,1)$ is parameterized as a sigmoid function:

$$p_{v,e} = \sigma(s_{v,e}) = \frac{1}{1+e^{-s_{v,e}}}$$

For simplicity, we ignore the normalisation of the scores. Its derivative with respect to the logit is:

$$\frac{dp_{v,e}}{ds_{v,e}} = p_{v,e}(1-p_{v,e})$$

This gradient is positive and bounded by $\frac{1}{4}$, ensuring smooth and monotonic updates.

Using chain rule, the gradient descent update on the logit becomes

$$\frac{ds_{v,e}}{d\tau} = -\eta \cdot \frac{\partial \mathcal{L}_{\text{task}}}{\partial p_{v,e}} \cdot \frac{dp_{v,e}}{ds_{v,e}} = -\eta \cdot g_{v,e} \cdot p_{v,e}(1-p_{v,e}),$$

where

$$g_{v,e} := \text{sign}\left(\frac{\partial \mathcal{L}_{\text{task}}}{\partial p_{v,e}}\right).$$

is assumed fixed throughout training. There are two cases to consider.

**(I) $\mathbf{g_{v,e} > 0}$.** If $g_{v,e} > 0$, the gradient descent step decreases $s_{v,e}$, pushing $p_{v,e} \to 0$, which implies $(1-p_{v,e}) \to 1$. Hence we are in the low-probability regime, where we can approximate $p_{v,e}(1-p_{v,e}) \approx p_{v,e}$. Thus,

$$\begin{aligned}
\frac{dp_{v,e}}{d\tau} &= \frac{dp_{v,e}}{ds_{v,e}} \cdot \frac{ds_{v,e}}{d\tau} \\
&= p_{v,e}(1-p_{v,e}) \cdot (-\eta g_{v,e} p_{v,e}(1-p_{v,e})) \\
&= -\eta g_{v,e} \left(p_{v,e}(1-p_{v,e})\right)^2 \\
&\approx -\eta g_{v,e} p_{v,e}^2
\end{aligned}$$

This is a first-order linear differential equation. Solving this differential equation yields

$$p_{v,e} \approx \frac{1}{\eta g_{v,e} \tau}.$$

To reach a desired threshold $p_{v,e} \leq \epsilon$, we solve:

$$\tau \geq \frac{1}{\eta g_{v,e} \epsilon} \approx \mathcal{O}(1/\epsilon).$$

**(II) $\mathbf{g_{v,e} < 0}$.** If $g_{v,e} < 0$, the gradient descent step increases $s_{v,e}$, pushing $p_{v,e} \to 1$, which implies $(1-p_{v,e}) \to 0$. In this high-probability regime, we can approximate $p_{v,e}(1-p_{v,e}) \approx 1-p_{v,e}$. Thus,

$$\begin{aligned}
\frac{d(1-p_{v,e})}{d\tau} &= -\frac{dp_{v,e}}{d\tau} \\
&\approx \eta |g_{v,e}| (1-p_{v,e})^2
\end{aligned}$$

Solving this differential equation yields

$$1 - p_{v,e} \approx \frac{1}{\eta g_{v,e} \tau}$$

To reach a desired threshold $1 - p_{v,e} \leq \epsilon \Rightarrow p_{v,e} \geq 1 - \epsilon$, we require

$$\tau \geq \frac{1}{\eta g_{v,e} \epsilon} \approx \mathcal{O}(1/\epsilon)$$

Thus, convergence to an $\epsilon$-confidence threshold is in $\mathcal{O}(1/\epsilon)$. □

## B Computational Complexity

We compare the per-layer computational complexity of full HGNNs with our fine-grained (EHGNN-F) and coarse-grained (EHGNN-C) sparsification variants. Let $n, m, t, d$, and $k$ denote the #nodes, #hyperedges, #node-hyperedge incidence pairs in $\boldsymbol{H}$, feature dimensions, and #node-hyperedge incidence pairs in $\tilde{\boldsymbol{H}}$ respectively. Table 6 shows that EHGNN-F reduces activation and computation overhead via incidence-level sparsification, but incurs a higher parameter cost. EHGNN-C reduces parameter overhead by scoring hyperedges via a shared MLP over pooled node features, sacrificing fine-grained control for scalability. Both variants are more scalable than full HGNN.

Let, $n = |V|$ denotes the number of nodes, $m = |E|$ denotes the number of hyperedges, $t = \sum_{e \in E} |e|$ denotes the number of incidence pairs (nonzeros in $\boldsymbol{H} \in \{0,1\}^{n \times m}$), $d$ denotes the input feature dimensionality, $k$ denotes the number of retained node–edge pairs after sparsification, $h$ denotes hidden layer size, and $\kappa$ denotes the number of hyperedges retained after sparsification, meaning, $\kappa \approx m.k/t$.

**Full HGNN.** Node-to-edge and edge-to-node aggregations per layer is typically done via sparse matrix–dense matrix multiplication (SpMM). In particular, node-to-edge aggregation is done via $\boldsymbol{H}^T \boldsymbol{X} \in \mathbb{R}^{m \times d}$ to construct hyperedge embedding, while edge-to-node aggregation is done via $\boldsymbol{H}(\boldsymbol{H}^T \boldsymbol{X}) \in \mathbb{R}^{n \times d}$ to construct embedding of the nodes in the next layer. Both aggregations has a time complexity $\mathcal{O}(td)$. The space complexity to hold the intermediate results is $\mathcal{O}(td)$. This is because there are $t$ non-zero entries in $\boldsymbol{H}$, for each we need to conduct $d$ elementwise multiply-add operations. There is no additional parameter overhead involved in Full HGNN training.

**Fine-grained masking (EHGNN-F).** *Time complexity:* EHGNN-F incurs computational cost in two main stages: sparsification and message passing. During sparsification, the model computes sigmoid scores for all $t$ incidence logits in $\mathcal{O}(t)$ time. To select the top-$k$ incidences, it uses top-$k$ Categorical sampling, costing $\mathcal{O}(t \log k)$. Once the top-$k$ mask is applied, message passing is executed over the reduced incidence matrix containing $k$ incidence pairs. Each such pair involves computations over feature vectors of dimension $d$, resulting in a total message passing cost of $\mathcal{O}(kd)$. Summing these components, the overall time complexity per forward pass is: $\mathcal{O}(kd + t \log k)$.

*Space complexity:* The space complexity of EHGNN-F consists of both the memory required for storing scores $s_{v,e}$ and intermediate activations stored during the forward pass. First, the model learns a scalar mask logit $s_{v,e}$ for every incidence pair (v,e), totaling $\mathcal{O}(t)$ persistent memory. During forward propagation, all $t$ scores are passed through a sigmoid and retained in memory for use in the straight-through estimator, contributing an additional $\mathcal{O}(t)$ temporary memory. The model then

Table 6: Time and space (activation) complexity, and parameter overhead comparison. Here $m = |E|$ is the original # hyperedges, $t = \sum_{e \in E} |e|$ is the original incidence size, $k \approx t.\kappa/m$ indicates the reduced incidence size after sparsification, and $\kappa$ indicates the #hyperedges after sparsification.

| Method | Time | Space/Activation | Param. overhead |
|---|---|---|---|
| Full HGNN | $\mathcal{O}(td)$ | $\mathcal{O}(td)$ | None |
| EHGNN-F (Fine) | $\mathcal{O}(t \log k + kd)$ | $\mathcal{O}(t + kd)$ | $\mathcal{O}(t)$ |
| EHGNN-F w/ cond. (Fine) | $\mathcal{O}(t \log k + kd + td)$ | $\mathcal{O}(t + kd)$ | $\mathcal{O}(d^2)$ |
| EHGNN-C (Coarse) | $\mathcal{O}(td + m \log \kappa + kd)$ | $\mathcal{O}(m + kd)$ | $\mathcal{O}(m)$ |
| EHGNN-C w/ cond. (Coarse) | $\mathcal{O}(td + md^2 + m \log \kappa + kd)$ | $\mathcal{O}(m + kd)$ | $\mathcal{O}(d^2)$ |

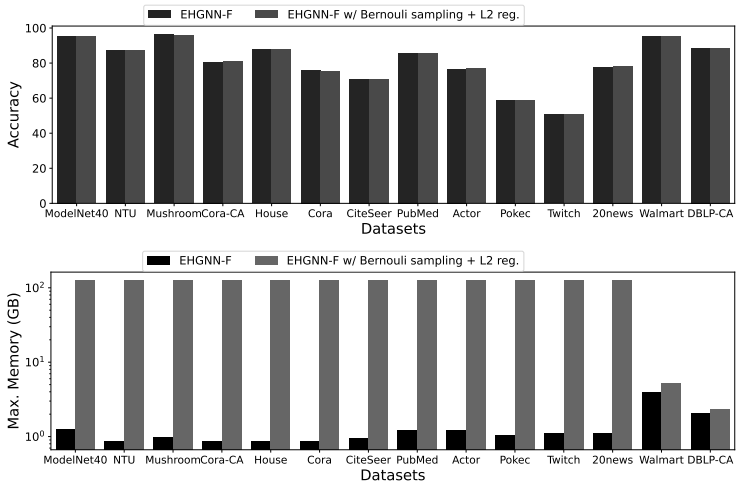



Figure 4: Ablation study of hard mask sampling from soft probabilities.

selects the top-$k$ incidences and performs message passing only over this pruned subset, requiring storage of $\mathcal{O}(kd)$ for the selected node or edge features. Thus, the total space complexity is: $\mathcal{O}(t + kd)$. This becomes prohibitive in large, dense hypergraphs where $t \gg n, m$; however, it is still more efficient than Full HGNN's space complexity $\mathcal{O}(td)$.

*Parameter overhead:* The parameter overhead in EHGNN-F originates from its fine-grained masks. For each incidence pair, the model maintains a dedicated scalar parameter. This leads to a total parameter count of: $\mathcal{O}(t)$.

**Coarse-grained masking (EHGNN-C w/ cond.).** *Time Complexity:* The runtime overhead of EHGNN-C consists of edge scoring and sparse message passing. First, node features are aggregated to hyperedges via $\boldsymbol{H}^T\boldsymbol{X}$, requiring $\mathcal{O}(td)$ elementwise multiply-add operations. Then, the edge-level MLP scores each of the $m$ hyperedges in $\mathcal{O}(md^2)$ time if a single-layer MLP is used. Selecting the top-$\kappa$ edges costs $\mathcal{O}(m \log \kappa)$. Finally, sparse message passing occurs over $k$ incidence pairs costs $\mathcal{O}(kd)$. Thus, the total time complexity per forward pass is: $\mathcal{O}(td + md^2 + m \log \kappa + kd)$. This reflects an efficient sparsification process, in particular for dense hypergraphs where the number of hyperedges $m \ll t$.

*Space Complexity:* EHGNN-C requires memory primarily for intermediate activations and edge-level mask scores. During message passing, only the top-$\kappa$ hyperedges are retained which contribute to roughly $k$ incidence pairs that participate in message passing. Thus, message-passing activations are stored for $k$ node–edge pairs, each of feature size $d$, totaling $\mathcal{O}(kd)$. Additionally, the model stores $m$ scalar scores ($s_e$) for edge selection, which adds $\mathcal{O}(m)$ space. The total space complexity during forward propagation is: $\mathcal{O}(m + kd)$. This is significantly better than EHGNN-F's space complexity in dense hypergraphs where $m \ll t$.

*Parameter overhead:* The parameter overhead of EHGNN-C comes from its coarse-grained scoring MLP, which is applied once per hyperedge. For a single shared MLP (across all edges), the number of parameters is $\mathcal{O}(d^2)$, assuming one hidden layer. More importantly, the parameter count is independent of the number of node-edge incidences $t$. This makes EHGNN-C far more parameter-efficient than EHGNN-F, especially for large-scale hypergraphs with millions of incidences but only thousands of hyperedges.

## C Parameter settings of EHGNN-C and EHGNN-F

Recall that EHGNN-C (cond.) and EHGNN-F (cond.) passes the node features of the nodes in a hyperedge to an MLP (shared with other hyperedges) to compute edge-level scores for each hyperedge. Table 7 reports the #neurons in the hidden layer of the MLP on various datasets. The reported setting produced the best accuracy on the test set.

Table 7: Parameter settings of EHGNN-C (cond.) and EHGNN-F (cond.)

| Dataset | # Hidden layers of MLP |
|---|---|
| 20news | 8 |
| Actor | 16 |
| CiteSeer | 16 |
| Cora | 32 |
| Cora-CA | 8 |
| DBLP-CA | 16 |
| House | 8 |
| ModelNet40 | 16 |
| Mushroom | 32 |
| NTU | 16 |
| Pokec | 32 |
| PubMed | 32 |
| Twitch | 16 |
| Walmart | 32 |
| Yelp | 16 |

Table 8: Comparison (accuracy $\pm$ std) of EHGNN-F and its variant without Probabilistic Normalization across datasets.

| **Model** | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | Citeseer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| EHGNN-F w/o Norm. | **77.96 ± 0.10** | **95.28 ± 0.05** | 96.30 ± 0.49 | 87.28 ± 0.42 | **76.82 ± 0.21** | 70.87 ± 0.72 | 80.71 ± 0.43 | **88.44 ± 0.08** |
| EHGNN-F | 77.94 ± 0.11 | 95.24 ± 0.07 | **96.35 ± 0.41** | **87.40 ± 0.18** | 76.68 ± 0.14 | **70.94 ± 0.37** | **80.77 ± 0.46** | 88.37 ± 0.07 |
| | Cora | House | Pokec | Pubmed | Twitch | Walmart | Yelp | |
| EHGNN-F w/o Norm. | **75.92 ± 0.43** | 87.80 ± 1.19 | 58.76 ± 0.03 | 85.57 ± 0.07 | 50.71 ± 0.07 | 95.16 ± 0.02 | **30.92 ± 0.14** | |
| EHGNN-F | 75.42 ± 0.57 | **87.93 ± 0.22** | **58.82 ± 0.09** | **85.59 ± 0.03** | **50.72 ± 0.11** | **95.17 ± 0.02** | 30.57 ± 0.46 | |

# D Ablation Studies

We analyze the various design choices for EHGNN-F. To that end, we conduct two studies: (a) whether it is worth normalizing $\sigma(s_{v,e})$ during the computation of probabilities $p_{v,e}$, and (b) whether the Bernoulli sampling performs better than Top-$k$ Categorical sampling.

(a) In Table 8, we observe that in most of the datasets (9 out of 15), normalization slightly helps improve the performance. As normalization is a constant-time operation, we decided to do so in EHGNN-F for an additional boost in performance. (b) In Figure 4, we find that the Bernoulli sampling, along with enforcing the budget constraint via L2 regularization, does not boost accuracy, but rather consumes more memory. Thus, instead, we opted for Top-$k$ Categorical sampling in EHGNN-F.

# E Additional Experiments

## E.1 Effectiveness on heterophilic hypergraphs.

Li et al. (2025) proposed a synthetic dataset containing hypergraphs with various homophily ratios. We compare EHGNN-F and EHGNN-C with this dataset to understand the effectiveness of EdgeMask-HGNN on heterophilic hypergraphs. The results are presented in Figure 5.

On heterophilic hypergraphs (e.g., homophily ratio = 0.3), where connected nodes often have dissimilar labels, EHGNN-F slightly outperforms EHGNN-C. This suggests that a more flexible fine-grained masking may better preserve diverse cross-class connections, which are important for information propagation in heterophilic settings. In contrast, EHGNN-C may prematurely prune such informative edges due to its more selective sparsification criteria. The performance of EHGNN-C becomes slightly better than EHGNN-F as hypergraphs become more homophilic (e.g., homophily ratio = 0.9).

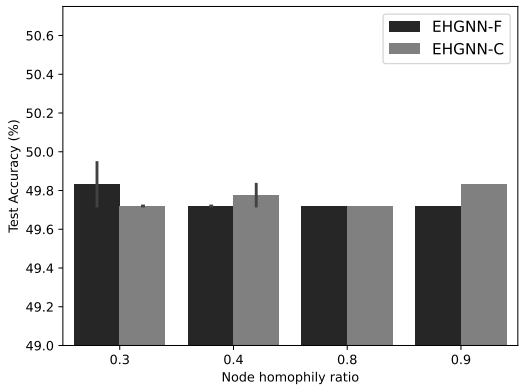



Figure 5: Performance of EdgeMask-HGNN on hypergraphs with various node homophily ratios.

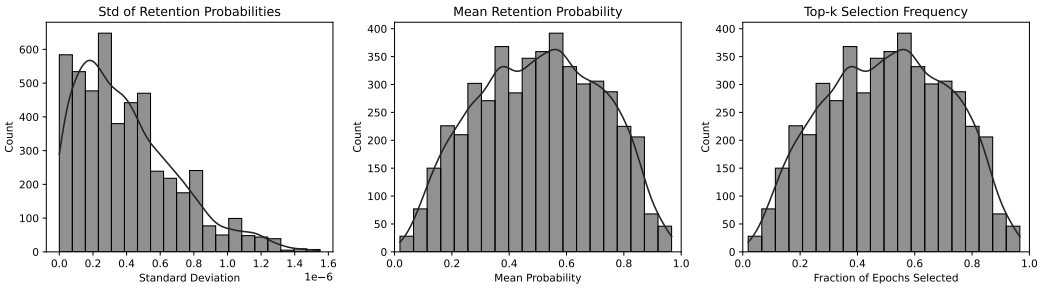



Figure 6: Analysing the retention probabilities of EHGNN-F on Cora (sparsity = 1%)

### E.2 Convergence of retention probabilities.

We analyze the retention probabilities $p_{v,e}$ in EHGNN-F to better understand their convergence behavior. Figure 6 presents the results regarding (a) the standard deviation of $p_{v,e}^{(t)}$ across epochs $t$ (left plot), (b) the mean $p_{v,e}^{(t)}$ across epochs $t$. (middle plot), and (c) the proportion of epochs where the pairs $(v, e)$ were selected to be in the top-$k$ (right plot).

(a) From the left plot, we observe that the vast majority of $(v, e)$ pairs have low standard deviation, indicating that their retention probabilities remain stable across training epochs. This demonstrates strong convergence of the learned mask, as the model consistently assigns very similar probabilities across epochs.

(b) The mean probabilities are evenly distributed across the full $[0, 1]$ range, forming a near-uniform bell shape. This suggests the model learns a broad spectrum of importance scores, effectively distinguishing task-relevant vs. irrelevant incidence pairs. This reflects the model's ability to differentiate between task-relevant and irrelevant pairs, reinforcing the value of the learned masking.

(c) The selection frequencies also follow a wide and symmetric distribution. Some edges are consistently selected, appearing in the top-$k$ mask in most epochs. These are likely critical incidence pairs, showing high agreement between learned probabilities and sampled mask selections. While others are rarely selected, reinforcing that the model has converged to a sparse and discriminative selection pattern.

## F Model Parameter Sizes

We report the model parameter sizes in Table 9 for a complete understanding of the parameter overhead, which was discussed earlier theoretically. Recall that EHGNN-C and EHGNN-F have parameter overhead of $\mathcal{O}(d^2)$ and $\mathcal{O}(t)$, respectively, where $d$ represents the dimension of node

Table 9: Number of model parameters (integers) across datasets for each algorithm at sparsity = 50%. OOM=Out-of-Memory.

| Algorithms | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | Citeseer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| Full | 53764 | 72745 | 12802 | 86083 | 27651 | 1899526 | 737799 | 733190 |
| Random | 53764 | 72745 | 12802 | 86083 | 27651 | 1899526 | 737799 | 733190 |
| Degdist | 53764 | 72745 | 12802 | 86083 | 27651 | 1899526 | 737799 | 733190 |
| Spectral | 53764 | 72745 | 12802 | 86083 | 27651 | 1899526 | 737799 | OOM |
| EHGNN-F | 119215 | 134300 | 53422 | 96143 | 81023 | 1902979 | 742384 | 832751 |
| EHGNN-F (cond.) | 60229 | 79210 | 14275 | 92548 | 30916 | 2136583 | 829576 | 824455 |
| EHGNN-C | 119215 | 134300 | 53422 | 96143 | 81023 | 1902979 | 742384 | 832751 |
| EHGNN-C (cond.) | 57029 | 76010 | 13571 | 89348 | 29316 | 2018087 | 783720 | 778855 |
| | Cora | House | Pokec | Pubmed | Twitch | Walmart | Yelp | |
| Full | 737799 | 2562 | 34818 | 258051 | 5122 | 11787 | 958473 | |
| Random | 737799 | 2562 | 34818 | 258051 | 5122 | 11787 | 958473 | |
| Degdist | 737799 | 2562 | 34818 | 258051 | 5122 | 11787 | 958473 | |
| Spectral | 737799 | 2562 | 34818 | 258051 | 5122 | OOM | OOM | |
| EHGNN-F | 742585 | 14405 | 40320 | 292680 | 21478 | 472417 | 5482067 | |
| EHGNN-F (cond.) | 829576 | 2755 | 39043 | 290116 | 5635 | 12556 | 1077706 | |
| EHGNN-C | 742585 | 14405 | 40320 | 292680 | 21478 | 472417 | 5482067 | |
| EHGNN-C (cond.) | 783720 | 2691 | 36963 | 274116 | 5411 | 12204 | 1018122 | |

features and $t = \sum_{e \in E} |e|$ is the number of node–hyperedge pairs. Note that $d$ does not depend on $|V|$ or $|E|$ and it's a constant, but $t$ does. Hence, we observe that EHGNN-C has a smaller parameter overhead than EHGNN-F in general.

EHGNN-F has a higher parameter overhead than Full, Random, Degdist, and Spectral due to the fact that it needs to learn the mask conditioned on the supervision signal from the downstream task, requiring additional parameters. While EHGNN-F introduces a higher parameter overhead, its memory usage ($\mathcal{O}(t + kd)$) is dominated by sparsified message passing layers. The substantial reduction in active node–hyperedge incidences ($k \ll t$) leads to a much smaller activation footprint. Thus, despite having more parameters, EHGNN-F consumes roughly equal and oftentimes less memory than the Full training (see Table 3).

## G Evaluating EdgeMask-HGNN on Node Clustering task

We adopt an unsupervised autoencoder framework, where the encoder learns node embeddings $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$ from the hypergraph. The decoder reconstructs the incidence structure $\boldsymbol{H}$ from the embeddings. The encoder-decoder are trained based on reconstruction loss (e.g. Binary cross-entropy loss). Finally, we run a standard clustering algorithm (e.g., $k$-means) on the learned embedding $\boldsymbol{Z}$.

**Encoder.** Given the hypergraph $\mathcal{H} = (\boldsymbol{X}, \boldsymbol{H})$, an HGNN (Feng et al., 2019) encoder $f_\theta$ produces node embeddings:

$$\boldsymbol{Z} = f_\theta(\boldsymbol{X}, \boldsymbol{H}) \in \mathbb{R}^{n \times d}.$$

The encoder may also include a sparsification mask (EHGNN-F or EHGNN-C), in which case the effective incidence is:

$$\tilde{\boldsymbol{H}} = \boldsymbol{H} \odot \boldsymbol{M}, \quad \boldsymbol{M} \in \{0, 1\}^{n \times m},$$

where $\boldsymbol{M}$ is the learned binary mask as discussed in section 4.

**Decoder.** The decoder reconstructs the incidence matrix $\boldsymbol{H}$ from $\boldsymbol{Z}$. Each hyperedge $e \in E$ is represented by aggregating its node embeddings:

$$\boldsymbol{h}_e = \frac{1}{|e|} \sum_{v \in e} \boldsymbol{Z}_v, \quad \boldsymbol{h}_e \in \mathbb{R}^d.$$

Table 10: Node clustering Accuracy (± std) across datasets. Bold denotes the best non-*Full* result per dataset. Avg. Rank computed over datasets where all non-*Full* algorithms report accuracies (10 datasets: actor, citeseer, cora, house-committees, ModelNet40, Mushroom, NTU2012, pokec, pubmed, twitch).

| Algorithms | actor | citeseer | cora | house-committees | ModelNet40 | Mushroom | NTU2012 | pokec | pubmed | twitch | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Full | 0.41 ± 0.02 | 0.41 ± 0.02 | 0.41 ± 0.03 | 0.67 ± 0.00 | 0.93 ± 0.00 | 0.84 ± 0.05 | 0.69 ± 0.01 | 0.51 ± 0.01 | 0.52 ± 0.08 | 0.51 ± 0.00 | |
| EHGNN-F | 0.41 ± 0.03 | 0.40 ± 0.05 | **0.42 ± 0.04** | **0.61 ± 0.03** | **0.92 ± 0.02** | 0.69 ± 0.07 | **0.67 ± 0.02** | **0.51 ± 0.00** | 0.53 ± 0.07 | 0.51 ± 0.00 | 2.00 |
| EHGNN-C | **0.43 ± 0.03** | 0.41 ± 0.04 | 0.41 ± 0.02 | 0.58 ± 0.05 | 0.90 ± 0.02 | **0.75 ± 0.10** | 0.67 ± 0.02 | 0.51 ± 0.00 | 0.52 ± 0.08 | 0.51 ± 0.00 | 2.60 |
| EHGNN-F (cond.) | 0.41 ± 0.03 | 0.39 ± 0.04 | 0.40 ± 0.04 | 0.59 ± 0.02 | 0.91 ± 0.01 | 0.70 ± 0.08 | 0.66 ± 0.01 | 0.51 ± 0.01 | **0.53 ± 0.07** | **0.51 ± 0.00** | 2.90 |
| EHGNN-C (cond.) | 0.42 ± 0.03 | **0.41 ± 0.04** | 0.41 ± 0.02 | 0.61 ± 0.05 | 0.91 ± 0.01 | 0.75 ± 0.10 | 0.65 ± 0.01 | 0.51 ± 0.00 | 0.53 ± 0.07 | 0.51 ± 0.00 | 2.50 |

Table 11: Node clustering NMI (± std) across datasets. Bold denotes the best non-*Full* result per dataset. Avg. Rank computed over datasets where all non-*Full* algorithms report accuracies (10 datasets: actor, citeseer, cora, house-committees, ModelNet40, Mushroom, NTU2012, pokec, pubmed, twitch).

| Algorithms | actor | citeseer | cora | house-committees | ModelNet40 | Mushroom | NTU2012 | pokec | pubmed | twitch | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Full | 0.00 ± 0.00 | 0.20 ± 0.03 | 0.24 ± 0.02 | 0.22 ± 0.00 | 0.92 ± 0.00 | 0.41 ± 0.11 | 0.82 ± 0.00 | 0.00 ± 0.00 | 0.12 ± 0.06 | 0.00 ± 0.00 | |
| EHGNN-F | 0.00 ± 0.00 | 0.18 ± 0.04 | 0.22 ± 0.03 | 0.14 ± 0.04 | 0.92 ± 0.00 | 0.14 ± 0.07 | 0.80 ± 0.00 | 0.00 ± 0.00 | 0.13 ± 0.06 | 0.00 ± 0.00 | 2.30 |
| EHGNN-C | 0.00 ± 0.00 | 0.18 ± 0.03 | 0.23 ± 0.03 | 0.10 ± 0.07 | 0.91 ± 0.00 | 0.24 ± 0.12 | 0.80 ± 0.00 | 0.00 ± 0.00 | 0.12 ± 0.07 | 0.00 ± 0.00 | 2.50 |
| EHGNN-F (cond.) | 0.00 ± 0.00 | 0.17 ± 0.04 | 0.22 ± 0.04 | 0.12 ± 0.03 | 0.92 ± 0.00 | 0.15 ± 0.08 | 0.80 ± 0.01 | 0.00 ± 0.00 | 0.14 ± 0.06 | 0.00 ± 0.00 | 2.90 |
| EHGNN-C (cond.) | 0.00 ± 0.00 | 0.19 ± 0.03 | 0.23 ± 0.03 | 0.14 ± 0.08 | 0.91 ± 0.00 | 0.24 ± 0.12 | 0.80 ± 0.00 | 0.00 ± 0.00 | 0.13 ± 0.06 | 0.00 ± 0.00 | 2.30 |

Table 12: Node clustering ARI (± std) across datasets. Bold denotes the best non-*Full* result per dataset. Avg. Rank computed over datasets where all non-*Full* algorithms report accuracies (10 datasets: actor, citeseer, cora, house-committees, ModelNet40, Mushroom, NTU2012, pokec, pubmed, twitch).

| Algorithms | actor | citeseer | cora | house-committees | ModelNet40 | Mushroom | NTU2012 | pokec | pubmed | twitch | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Full | 0.01 ± 0.00 | 0.16 ± 0.02 | 0.18 ± 0.03 | 0.12 ± 0.00 | 0.90 ± 0.00 | 0.48 ± 0.13 | 0.65 ± 0.01 | 0.00 ± 0.00 | 0.11 ± 0.06 | 0.00 ± 0.00 | |
| EHGNN-F | 0.01 ± 0.00 | 0.13 ± 0.03 | 0.16 ± 0.03 | 0.05 ± 0.02 | 0.91 ± 0.01 | 0.16 ± 0.10 | 0.62 ± 0.03 | 0.00 ± 0.00 | 0.13 ± 0.07 | 0.00 ± 0.00 | 2.10 |
| EHGNN-C | 0.01 ± 0.00 | 0.14 ± 0.03 | 0.16 ± 0.04 | 0.04 ± 0.05 | 0.88 ± 0.02 | 0.29 ± 0.16 | 0.63 ± 0.04 | 0.00 ± 0.00 | 0.11 ± 0.08 | 0.00 ± 0.00 | 2.60 |
| EHGNN-F (cond.) | 0.01 ± 0.00 | 0.12 ± 0.03 | 0.15 ± 0.03 | 0.04 ± 0.02 | 0.90 ± 0.01 | 0.18 ± 0.12 | 0.61 ± 0.02 | 0.00 ± 0.00 | 0.13 ± 0.07 | 0.00 ± 0.00 | 2.80 |
| EHGNN-C (cond.) | 0.01 ± 0.00 | 0.14 ± 0.03 | 0.16 ± 0.03 | 0.06 ± 0.05 | 0.90 ± 0.01 | 0.29 ± 0.16 | 0.60 ± 0.02 | 0.00 ± 0.00 | 0.12 ± 0.07 | 0.00 ± 0.00 | 2.50 |

The probability that node $v$ belongs to hyperedge $e$ is modeled as:

$$\hat{\boldsymbol{H}}_{v,e} = \sigma\left(\langle \boldsymbol{W}_n \boldsymbol{Z}_v, \boldsymbol{W}_e \boldsymbol{h}_e \rangle\right),$$

where $W_n, W_e \in \mathbb{R}^{d\times d}$ are learnable projections of the decoder, $\langle \cdot, \cdot \rangle$ denotes the dot product, and $\sigma$ is the sigmoid activation function.

**Reconstruction Loss.** We sample the same number of positive incidences $(v, e)$ where $H_{v,e} = 1$, and negatives $(v, e)$ where $H_{v,e} = 0$. The reconstruction loss is binary cross-entropy loss:

$$\mathcal{L}_{\text{recon}} = -\sum_{(v,e)\in\Omega^+} \log \hat{H}_{v,e} - \sum_{(v,e)\in\Omega^-} \log(1 - \hat{H}_{v,e}),$$

where $\Omega^+ = \{(v, e) : H_{v,e} = 1\}$ and $\Omega^-$ is a set of sampled negatives.

The encoder parameters $\theta$, and decoder projections $(W_n, W_e)$ are optimized jointly to minimize $\mathcal{L}_{\text{recon}}$.

**Observations.** We observe that all algorithms (including Full) perform poorly on heterophilic hypergraph benchmarks (Actor, Pokec, twitch). We believe the reason is the following: on heterophilic hypergraphs node neighbors often have different labels which renders laplacian-style averaging over incident nodes under homophily assumption ineffective. As there are currently no studies on the node-clustering performance of HGNNs on heterophilic hypergraphs, addressing this issue not only under full training but also sparsified training setting would be an interesting future work.