# EdgeMask-HGNN: Learning to Sparsify Hypergraphs 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. However, existing HGNNs consider the input hypergraph as fixed during training, ignoring the fact that real-world hypergraphs may often contain noisy hyperedges or links that are irrelevant for the downstream task. This makes them prone to overfitting, poor generalizability, and degrades their effectiveness on heterophilic hypergraphs. To address these issues, we propose **EdgeMask-HGNN**, a supervised sparsification method that learns a discrete subhypergraph under an explicit budget constraint. EdgeMask-HGNN offers two distinct sparsification schemes: a fine-grained sparsification and a coarse-grained sparsification, both trained end-to-end using supervision from the downstream task. Extensive experiments on node classification benchmarks demonstrate that EdgeMask-HGNN is effective on heterophilic hypergraphs. On more homophilic datasets, its performance is often comparable to strong baselines. Beyond node classification, EdgeMask-HGNN also shows superior link prediction performance on existing link prediction benchmarks compared to full training and unsupervised sparsification baselines.

## 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). A wide range of learning problems arises in hypergraphs– from node classification (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 these tasks, demonstrating strong empirical performance.

However, real-world hypergraphs are often noisy and densely connected, making HGNNs prone to overfitting, leading to poor generalizability. Sparsifying the hypergraph offers an effective solution, for instance, dropping hyperedges uniformly at random (*random sparsification*), or sampling edges based on their degree in the hypergraph (Leskovec & Faloutsos, 2006). However, these methods may not preserve spectral properties and may remove nodes important for downstream tasks. *Spectral sparsifiers* sample edges based on hypergraph effective resistance to approximately preserve eigenvalues and eigenvectors of the original hypergraph (Soma & Yoshida, 2019). However, spectral sparsification may still fail to preserve task-relevant hyperedges, as it does not exploit label information. At present, existing hypergraph sparsification methods are largely task-agnostic and do not leverage supervision to identify task-relevant higher-order relations.

**Why task-aware hypergraph sparsification.** In many applications, hypergraph structures are constructed heuristically and inevitably contain task-irrelevant or even task-adversarial higher-order relations. Treating the input hypergraph as fixed implicitly assumes that all hyperedges are equally valid for the downstream task, which is rarely the case. Figure 1 shows that sparsifying the hypergraph can improve predictive performance over training on the full hypergraph and improve generalization. This motivates viewing hy-

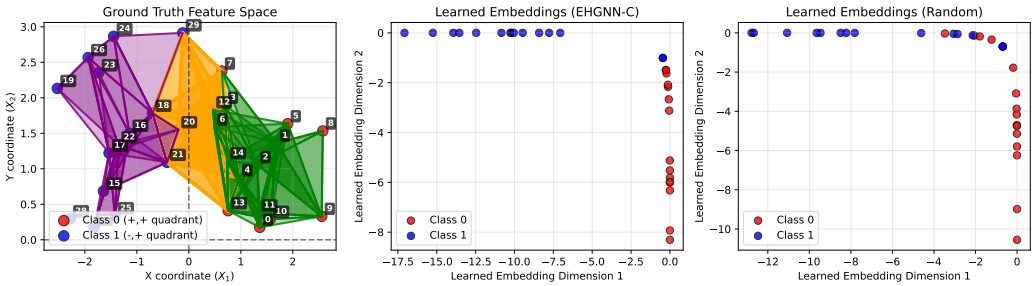



Figure 1: A synthetic 3-uniform hypergraph (#nodes=30, #edges=134) where (x,y)-coordinates are ground truth node features (left). 50% sparsified hypergraph node embeddings via EdgeMask-HGNN (middle) and via Random sparsification (right). The node embeddings learned by EdgeMask-HGNN are better separable than those of Random. Furthermore, Full training acc. = Random pruning acc.= 83.33% while EdgeMask-HGNN acc. = 100%, which suggest principled sparsification can improve HGNN generalization.

pergraph sparsification not merely as a heuristic compression technique but as a supervised subhypergraph selection problem, where a task-relevant subset of incidences or hyperedges is selected under a fixed budget. Building on this perspective, we formulate hypergraph sparsification as a supervised discrete mask learning problem subject to an explicitly specified global sparsity budget.

**Our contributions.** (I) We propose EdgeMask-HGNN– a budget-constrained, task-aware hypergraph sparsification method for HGNNs. EdgeMask-HGNN offers two distinct masking strategies: a fine-grained masking (incidence-level) and a coarse-grained masking (hyperedge-level), both trained end-to-end using supervision from the downstream task. The masks are constructed via weighted sampling without replacement, where sampling weights are either parameterized by learnable retention logits at the incidence/edge-level or learned by a feature-conditioned parameterized function. 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 shows that under the same sparsification budget, embeddings learned by EdgeMask-HGNN yield better class separation than the random sparsifier.

(II) We theoretically analyze the stability of the learned sparsifier. To be precise, we show that as the retention logits evolve across epochs, the expected number of changes in the sampled masks is bounded by the magnitude of logit updates. This result characterizes the stability of the sampling process.

(III) Empirical evaluation and analysis show that EdgeMask-HGNN is more effective than alternative sparsifiers such as random, degree-based, and spectral sparsifiers, as well as full hypergraph training on heterophilic hypergraphs. On more homophilic datasets, its performance is often comparable to strong baselines. EdgeMask-HGNN also yields superior performance on link prediction task. Finally, EdgeMask-HGNN is simple by design and easily adaptable to different HGNN backbones.

**Positioning.** Our method should be understood as a supervised sparsification approach that selects a task-relevant subhypergraph under an explicit budget constraint, rather than a general hypergraph structure learning framework. This distinction is important because our goal is not to reconstruct or augment the hypergraph, but to identify the most relevant subset of relations for the given learning task. This is particularly important in settings where (i) the observed hypergraph encodes real, interpretable relations (e.g., co-authorship, biological interactions), (ii) adding or hallucinating hyperedges may introduce spurious semantics, and (iii) computational constraints require controlling the number of incidences or edges explicitly.

## 2 Related Works

**Hypergraph Neural Networks (HGNNs).** There are broadly two families of HGNNs: *spectral* and *spatial* (Antelmi et al., 2023). The *spectral HGNNs* construct the hypergraph Laplacian matrix encoding the higher-order relationships, perform filtering operations in the spectral domain using eigenvalues and eigenvectors, and finally transform node features into node embeddings using these spectral filters. The *spatial*

*HGNNs* typically define two-stage message-passing and aggregation: node → hyperedge and hyperedge → node. Despite the apparent difference, the operations performed by certain spectral HGNNs can be interpreted as forms of message passing (Hayhoe et al., 2024). 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 focuses on learning by identifying and preserving only task-relevant hyperedges, making it a practical augmentation for such general-purpose designs.

**Existing Hypergraph sparsification methods.** 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. Recently, Kapralov et al. (2022) proved a significant result showing that it is possible to obtain an $\epsilon$-spectral sparsifier of linear size. These methods are unsupervised, as they do not incorporate node labels into their sparsification decisions. As a result, they may not be suitable for representation learning tasks that require preserving only task-relevant hyperedges.

Among task-aware methods, HSL (Cai et al., 2022) proposes to reconstruct and enhance the input hypergraph by jointly pruning and adding node-hyperedge connections under no global sparsity constraint. In contrast, EdgeMask-HGNN focuses on selecting a task-relevant *subhypergraph* from a given hypergraph under an *explicit global sparsity budget*. Our method does not augment connectivity, as in many practical settings, such as recommendation systems or scientific data analysis, modifying or hallucinating higher-order relations can be undesirable or difficult to operationalize. Consequently, EdgeMask-HGNN is designed to be complementary to HSL, prioritizing controllable sparsity and scalability over structural reconstruction.

**Graph structure learning and sparsification.** Several works study sparsification of neural network parameters, such as the unified lottery ticket hypothesis for GNNs (Chen et al., 2021; Hui et al., 2023; Liao et al., 2025) and graph gradual pruning (Liu et al., 2023a). These methods jointly prune GNN weights and graph edges to reduce parameter count and training cost, with the primary goal of model compression. Our work is complementary: we keep the HGNN parameters dense while learn a structural sparsifier that selectively retains a subset of node-hyperedge incidences. There have also been several notable recent works regarding degree-based spectral sparsfication (Liu et al., 2023b) and supervised graph sparsification (Zheng et al., 2020; Ye & Ji, 2021; Zhang et al., 2025). Unlike graph pruning methods that operate on pairwise edges, our method is 'higher-order' in the sense that it operates on hypergraph incidences.

While supervised sparsification and edge masking have been explored in GNNs Ye & Ji (2021), extending these ideas to hypergraphs is nontrivial due to the higher-order incidence structure. In particular, hyperedges induce node-to-edge bipartite incidence relations rather than node-to-node pairwise relations, giving rise to distinct sparsification granularities at the incidence and hyperedge levels. Our method explicitly models these two levels of sparsification, which have no direct analogue in standard graph settings.

## 3 Problem Statement

Let $\mathcal{H} \triangleq (V, E, \boldsymbol{X}) \equiv (\boldsymbol{H}, \boldsymbol{X})$ be a hypergraph, where $V$ is the set of nodes, $E$ is the set of hyperedges, $\boldsymbol{H} \in \{0,1\}^{|V|\times|E|}$ 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 $V$, i.e., $H_{v,e} = 1$ if $v \in e$, or 0 otherwise. Let $\mathcal{Y}$ denote the output space, $\mathcal{S}$ denote the set of supervised elements (e.g., nodes, hyperedges), and $\mathcal{S}_L \subseteq \mathcal{S}$ denote the labeled subset with labels $\boldsymbol{y}_L$. The goal of a supervised learning task on $\mathcal{H}$ is to learn a model $f_\theta : \mathcal{S} \to \mathcal{Y}$ that predicts labels for elements $s \in \mathcal{S}$ by minimizing an empirical task loss: $\theta^* = \arg\min_\theta \mathcal{L}_{task}(f_\theta(s; \boldsymbol{H}, \boldsymbol{X}), \boldsymbol{y}_L)$. This formulation is task-agnostic and applies to node classification, hyperedge classification, and link prediction tasks.

*Budget-constrained task-aware sparsification.* Let $t = \sum_{e\in E}|e|$ denote the total #incidence pairs before sparsification. As a hyperedge contains more than 2 nodes, the budget constraint can be applied in two forms: $k$, the #incidence pairs retained after sparsification, and $\kappa$, the #hyperedges retained after sparsification.

Given a budget $k$ (or $\kappa$), the goal is to construct a subhypergraph $\hat{\mathcal{H}} = (\tilde{\boldsymbol{H}}, \boldsymbol{X}) \subseteq \mathcal{H}$ satisfying the desired budget such that the downstream task loss is minimized. This can be formulated as the following optimization problem:

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

where $\tilde{\boldsymbol{H}}$ is the *masked* incidence matrix obtained from a learnable sparsifier. The downstream task loss can be any differentiable loss depending on the supervised task. For example, for classification tasks:

$$\mathcal{L}_{task} = \frac{1}{|\mathcal{S}_L|} \sum_{s\in\mathcal{S}_L} \ell(\boldsymbol{y}_s, \hat{\boldsymbol{y}}_s),$$

where $\hat{\boldsymbol{y}}_s = f_\theta(s; \tilde{\boldsymbol{H}}, \boldsymbol{X})$ denotes the prediction for the supervised element $s \in \mathcal{S}_L$.

## 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 prior hypergraph pruning methods that rely on fixed heuristics, EdgeMask-HGNN is trained end-to-end, allowing the model to jointly optimize the hypergraph structure and the HGNN parameters under a strict global sparsity budget constraint.

Sparsification can be applied at two structural levels of a hypergraph: (i) individual node–hyperedge incidences and (ii) entire hyperedges. Thus, two distinct masking schemes emerge: *Incidence-level masking* and *Edge-Level masking*. Incidence-level masking learns an incidence-level mask by sampling incidences based on incidence-level weights, whereas Edge-Level masking learns an edge-level mask by sampling edges based on edge-level weights. These weights are either parameterized by independent learnable logits or learned by a shared parameterized function. Finally, based on the task loss of the downstream HGNN, the gradients are backpropagated to compute the updated masks as well as the HGNN parameters. Figure 2 illustrates a schematic of EdgeMask-HGNN parameterized by independent learnable logits with a small example.

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

Fine-grained incidence-level masking focuses on learning a soft importance weight 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 logit $s_{v,e}$, which is converted to a non-negative weight via sigmoid transformation

$$w_{v,e} = \sigma(s_{v,e}) \tag{1}$$

where $\sigma$ is the sigmoid function.

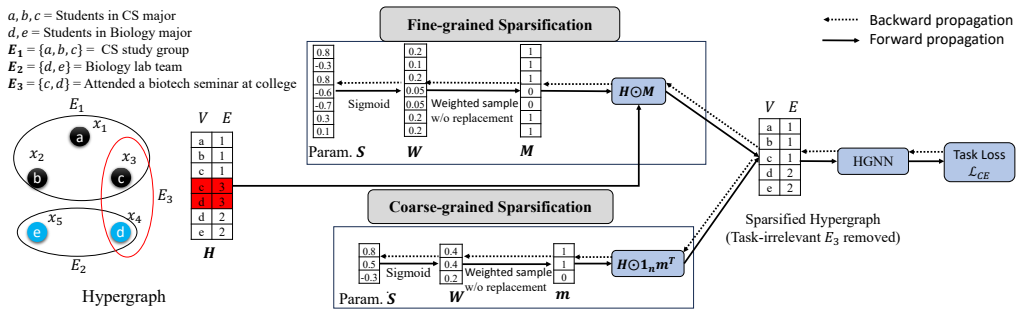



Figure 2: Schematic of EHGNN-F (top) and EHGNN-C (bottom) with a small example.

Let $\boldsymbol{w} := (w_{v,e})$ denote the vector of weights over all incidence pairs. To construct a sparsified hypergraph with exactly $k$ active incidences, we sample $k$ distinct incidence pairs without replacement according to these weights. This sampling procedure is equivalent to drawing the first $k$ elements of a permutation generated by a *Plackett–Luce (PL) distribution*(Luce et al., 1959) parameterized by $\mathbf{w}$. Specifically, if $(i_1, \ldots, i_k)$ denote the sampled ordered indices corresponding to incidence pairs, their probability under the PL model is

$$P(i_1, \ldots, i_k) = \prod_{t=1}^{k} \frac{w_{i_t}}{\sum_{j \notin \{i_1, \ldots, i_{t-1}\}} w_j}.$$

The resulting hard incidence mask $\hat{\mathbf{m}}$ is defined as

$$\hat{m}_{v,e} = \begin{cases} 1 & \text{if } (v,e) \text{ is selected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \sum_{v,e} \hat{m}_{v,e} = k.$$

Thus, the sparsifier samples exactly $k$ active incidences according to a *weighted sampling-without-replacement scheme*, which can be viewed as the top-$k$ marginal of a Plackett–Luce ranking distribution. Let $\hat{\boldsymbol{M}} \in \{0,1\}^{n \times m}$ denote the binary incidence mask whose entries are $\hat{m}_{v,e}$. The sparsified hypergraph is obtained using the *hard mask* $\hat{M}$ in the forward pass:

$$\tilde{\boldsymbol{H}} = \boldsymbol{H} \odot \hat{\boldsymbol{M}}, \tag{2}$$

while gradients are propagated through the surrogate mask $m$ defined by the straight-through estimator (Equation (3)).

**Differentiability.** The sampling step used to construct the incidence mask is discrete and therefore non-differentiable. To enable gradient-based optimization, we employ a masked straight-through estimator (STE) (Bengio et al., 2013). In the forward pass, a hard binary mask $\hat{m}_{v,e} \in \{0,1\}$ is produced by sampling $k$ incidences according to the weights $w_{v,e} = \sigma(s_{v,e})$. In the backward pass, we define

$$m_{v,e} = \hat{m}_{v,e} + \big(w_{v,e} - \text{stop_grad}(w_{v,e})\big)\hat{m}_{v,e}, \tag{3}$$

so that $\partial m_{v,e} / \partial w_{v,e} = \hat{m}_{v,e}$. Consequently, gradients are propagated only through $k$ incidences selected by the hard mask, while unselected incidences receive zero gradient during that iteration. By the chain rule, the gradient with respect to the logit satisfies

$$\frac{\partial \mathcal{L}}{\partial s_{v,e}} = \frac{\partial \mathcal{L}}{\partial \tilde{H}_{v,e}} H_{v,e}\, \hat{m}_{v,e}\, w_{v,e}(1 - w_{v,e}).$$

This estimator provides a practical surrogate for backpropagation through the discrete masking operation: the forward computation remains exactly sparse, while the backward pass uses the continuous weights $w_{v,e}$ to

propagate gradients to the scorer parameters. Although the estimator is biased relative to the true gradient of the discrete sampling process, it has been widely adopted in discrete neural architectures (Zheng et al., 2020) and works well empirically.

**Variant of EHGNN-F conditioned on node features.** EHGNN-F is feature-agnostic, each incidence pair $(v, e)$ is assigned an independent learnable parameter $s_{v,e}$. While flexible, this parameterization does not share information across incidence pairs and may generalize poorly when supervision is limited. To address this issue, feature-conditioned EHGNN-F variant **EHGNN-F (cond.)** uses a shared parameter *scorer* module that learns $s_{v,e}$ as functions of node features:

$$s_{v,e} = \texttt{MLP}(\boldsymbol{X}_v || \hat{\boldsymbol{X}}_e), \tag{4}$$

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.

### 4.2 Coarse-grained 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 logit parameter per edge. For instance, if the input hypergraph is $k$-uniform (every edge contains $k$ nodes), edge-level masking reduces the model parameters by $k$.

The learnable edge score parameter is converted to non-negative edge weights via sigmoid transformation:

$$w_e = \sigma(s_e) \tag{5}$$

To construct a sparsified hypergraph with exactly $\kappa$ edges, we sample $\kappa$ edges without replacement according to weights $\boldsymbol{w}_e$ via PL sampling procedure. The sampled hard edge mask is defined as

$$\hat{m}_e = \begin{cases} 1 & \text{if } e \text{ is selected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \sum_e \hat{m}_e = \kappa.$$

To backpropagate through this discrete edge selection, similar to EHGNN-F, we use a masked STE: $m_e = \hat{m}_e + (w_e - \text{stop_grad}(w_e))\hat{m}_e$. This allows gradients to propagate only through the selected edges at that epoch. The sparsified incidence matrix becomes:

$$\tilde{\boldsymbol{H}} = \boldsymbol{H} \odot \mathbf{1}_n \boldsymbol{m}^T, \tag{6}$$

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. This feature-agnostic formulation may limit supervision propagation in semi-supervised settings. To address this issue, feature-conditioned EHGNN-C (**EHGNN-C (cond.)**) learns a shared scorer as a parameter of node features using a permutation-invariant operator, such as mean pooling:

$$\hat{\boldsymbol{X}}_e = \frac{1}{|e|}\sum_{v\in e}\boldsymbol{X}_v, \quad s_e = \texttt{MLP}(\hat{\boldsymbol{X}}_e) \tag{7}$$

### 4.3 Issues with Feature Conditioned Variants and Low-Rank Reparameterization

The MLP-based scorer defined in Equations (4) and (7) is memory-intensive on large-scale hypergraphs. For instance, EHGNN-F (cond) defines incidence scores by evaluating $w_{v,e} = \sigma\left(\texttt{MLP}_\theta(\boldsymbol{X}_v || \hat{\boldsymbol{X}}_e)\right)$, where $\boldsymbol{X}_v \in \mathbb{R}^F$ is the feature of node $v$, $\hat{\boldsymbol{X}}_e$ is hyperedge representation pooled from $\{\boldsymbol{X}_v : v \in e\}$. When the number of incidences $t$ is large, evaluating $\texttt{MLP}_\theta$ across all incidences induces high activation memory. With a two-layer MLP scorer with hidden width $h$, EHGNN-F(cond) requires $\approx \Theta(tF + th)$ activation storage, since both the concatenated incidence input $[X_v \,\|\, \hat{X}_e] \in \mathbb{R}^{2F}$ and the hidden layer activations must be retained for backpropagation. To address this, we adopt a lightweight low-rank parameterization of EdgeMask-HGNN.

**Low-Rank feature-conditioned incidence scoring: EHGNN-F(cond,LR).** To obtain a scalable scorer while preserving the conditioning on node features, we replace $\texttt{MLP}_\theta$ with a rank-$r$ bilinear parameterization:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{Z}^\top,$$

where $\boldsymbol{U}$ and $\boldsymbol{Z}$ are the latent node factors and edge factors, respectively. The node factor for node $v$ and the edge factor for edge $e$ are constructed as

$$\boldsymbol{u}_v = \boldsymbol{W}_x^\top \boldsymbol{X}_v \in \mathbb{R}^r, \qquad \boldsymbol{z}_e \in \mathbb{R}^r, \tag{8}$$

where $\boldsymbol{W}_x \in \mathbb{R}^{F\times r}$ is a learnable projection matrix and $r << F$. Finally, the incidence score and weights are computed as

$$s_{v,e} = \langle \boldsymbol{u}_v, \boldsymbol{z}_e \rangle, \qquad w_{v,e} = \sigma(s_{v,e}). \tag{9}$$

This parameterization constrains the unnormalized score matrix $\boldsymbol{S}$ to satisfy $rank(\boldsymbol{S}) \leq r$. Unlike EHGNN-F(cond), we do not compute pooled hyperedge features, thereby avoiding the $\Theta(tF)$ cost of aggregating node features for each incidence.

In terms of expressivity, Equation 9 can be treated as a restricted low-rank parameterization of the incidence scoring function $\texttt{MLP}_\theta(.)$ used in EHGNN-F(cond), which trades expressive power for improved computational and memory efficiency. The score computation still depends on node features $\boldsymbol{x}_v$, each hyperedge retains a learnable embedding $\boldsymbol{z}_e$, and the masking and straight-through estimator remain unchanged. In terms of activation memory, EHGNN-F(cond,LR) stores $\Theta(|V|r + tr)$ activations, corresponding to storing projected node features ($\boldsymbol{u}_v$) and edge factors ($\boldsymbol{z}_e$). Hence, the scorer activation-memory is reduced by roughly $\frac{tF+th}{|V|r+tr} \approx F/r$ when $t >> |V|$ and $F >> h$.

**Low-rank feature-conditioned edge scoring: EHGNN-C(cond,LR).** Similar to EHGNN-F(cond,LR), we compute the latent hyperedge factor $\boldsymbol{z}_e$ and node factors $\boldsymbol{u}_v$ as in Equation 8. The projected node embeddings are then aggregated via mean-pooling to construct hyperedge embedding:

$$\boldsymbol{u}_e = \phi(\{\boldsymbol{u}_v \mid v \in e\}) \in \mathbb{R}^r. \tag{10}$$

Finally, instead of using $\texttt{MLP}$, we use a rank-$r$ bilinear form as a scorer:

$$s_e = \langle \boldsymbol{u}_e, \boldsymbol{z}_e \rangle, \qquad w_e = \sigma(s_e). \tag{11}$$

This produces hyperedge-level weights similar to original EHGNN-C (cond) but avoids the cost of evaluating an MLP on each hyperedge.

### 4.4 Training and Inference.

The sparsified 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 $\boldsymbol{w}$ for stability and reproducibility.

## 5 Theoretical Guarantees

Since the logits evolve gradually across epochs, we study how small changes in the logit vector affect the stochastic masks produced by PL sampling. The following theoretical results show that the expected number of mask flips is controlled by the magnitude of logit updates. The proofs are in the Appendix.

**Theorem 5.1** (Epoch-to-epoch stability of fine-grained masks)**.** *Let $t = \sum_{e\in E} |e|$ be the number of incidence pairs and $k \in \{1, \ldots, t\}$ be a fixed incidence pair budget. Let $\boldsymbol{s}^{(\tau)}, \boldsymbol{s}^{(\tau+1)} \in \mathbb{R}^t$ denote the logit vectors at two consecutive epochs of EHGNN-F, and incidence weights*

$$\boldsymbol{w}^{(\tau)} := \sigma(\boldsymbol{s}^{(\tau)}), \quad \boldsymbol{w}^{(\tau+1)} := \sigma(\boldsymbol{s}^{(\tau+1)}).$$

*Let $S^{(\tau)}$ and $S^{(\tau+1)}$ denote the $k$-subsets obtained by sampling from the Plackett–Luce distributions $\mathrm{PL}(\boldsymbol{w}^{(\tau)})$ and $\mathrm{PL}(\boldsymbol{w}^{(\tau+1)})$ respectively.*

*We assume that there exists $\alpha > 0$ such that for all epoch $\tau$ and all $i \in [t]$, $w^{(\tau)}(i) \geq \alpha$. Then the expected number of incidence mask flips between the two consecutive epochs satisfies*

$$\mathbb{E}\Big[|S^{(\tau+1)} \triangle S^{(\tau)}|\Big] \leq \frac{k}{2\alpha} \ln \frac{t}{t-k} \, \|\boldsymbol{s}^{(\tau+1)} - \boldsymbol{s}^{(\tau)}\|_1.$$

**Theorem 5.2** (Epoch-to-epoch stability of coarse-grained masks)**.** *Let $m = |E|$ be the number of hyperedges and $\kappa \in \{1, \ldots, m\}$ be a fixed edge budget. At iteration $\tau$, let the edge logits $\boldsymbol{s}_e^{(\tau)} \in \mathbb{R}^m$ and edge weights*

$$w_e^{(\tau)}(e) := \sigma(s_e^{(\tau)}(e)) \in (0,1), \qquad e \in [m].$$

*EHGNN-C samples a permutation $\Gamma^{(\tau)} \sim \mathrm{PL}(\boldsymbol{w}_e^{(\tau)})$ and selects the first $\kappa$ edges*

$$T^{(\tau)} := \{\Gamma_1^{(\tau)}, \ldots, \Gamma_\kappa^{(\tau)}\} \subseteq [m].$$

*We assume that there exists $\alpha_c > 0$ such that $w_e^{(\tau)}(e) \geq \alpha_c$ for all $\tau, e$. Then the expected number of edge mask flips between consecutive epochs satisfies*

$$\mathbb{E}\Big[|T^{(\tau+1)} \triangle T^{(\tau)}|\Big] \leq \frac{\kappa}{2\alpha_c} \ln \frac{m}{m-\kappa} \, \|\boldsymbol{s}_e^{(\tau+1)} - \boldsymbol{s}_e^{(\tau)}\|_1.$$

**Discussion.** (i) The above results imply that if the logit vector changes slowly across epochs (as is typical near convergence), the expected number of mask changes also becomes small. Thus, the stochastic sparsification procedure stabilizes during training. (ii) The assumption of PL weights being lower-bounded is mild in our setting, since the mask logits are produced by a learnable scoring function applied to node and hyperedge representations. When input features are bounded, and model parameters remain bounded during training, the resulting logits are also bounded. In particular, if $s^{(\tau)}(i) \geq -B$ for all $i$, then $w^{(\tau)}(i) = \sigma(s^{(\tau)}(i)) \geq \sigma(-B)$. (iii) Our results characterize the stability of the stochastic mask sampling process under bounded logit changes. Although this does not provide guarantees on optimization convergence, recovery of an optimal subhypergraph, or generalization performance, it does justify the empirical observation that the learned sparsifier stabilizes over training (see Appendix H).

## 6 Experiments

We evaluate EdgeMask-HGNN on semi-supervised node classification task in the transductive setting as well as on the link prediction task. On the node classification task, 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 512 GB RAM and a single NVIDIA Tesla V100 GPU with 32 GB memory. All algorithms run for 1000 epochs except Trivago (2000 epochs) with a retention rate of $k = 50\%$ ($\kappa = 50\%$).

Table 1: Dataset statistics (Blue text = heterophillic dataset as per the criteria of Edge homophily $\leq 0.5$).

| | Cora | Citeseer | Pubmed | Cora-CA | DBLP-CA | 20News | Mushroom | NTU2012 | ModelNet40 | Yelp | House | Walmart | Actor | Twitch | Pokec | Trivago |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert V\rvert$ | 2708 | 3312 | 19717 | 2708 | 41302 | 16242 | 8124 | 2012 | 12311 | 50758 | 1290 | 88860 | 16255 | 16812 | 14998 | 172738 |
| $\lvert E\rvert$ | 1579 | 1079 | 7963 | 1072 | 22363 | 100 | 298 | 2012 | 12311 | 679302 | 341 | 69906 | 10164 | 2627 | 2406 | 233202 |
| # feature | 1433 | 3703 | 500 | 1433 | 1425 | 100 | 22 | 100 | 100 | 1862 | 100 | 100 | 50 | 7 | 65 | 300 |
| # class | 7 | 6 | 3 | 7 | 6 | 4 | 2 | 67 | 40 | 9 | 2 | 11 | 3 | 2 | 2 | 160 |
| t | 4786 | 3453 | 34629 | 4585 | 99561 | 65451 | 40620 | 10060 | 61555 | 4523594 | 11843 | 460630 | 53372 | 16356 | 5502 | 726861 |
| Avg edge size ($\bar{e}_{avg}$) | 3.03 | 3.20 | 4.35 | 4.28 | 4.45 | 654.51 | 136.31 | 5.00 | 5.00 | 6.66 | 34.73 | 6.59 | 5.25 | 6.23 | 2.29 | 3.12 |
| Density ($\bar{e}_{avg}/\lvert V\rvert$) | 0.00112 | 0.00097 | 0.00022 | 0.00158 | 0.00011 | 0.04030 | 0.01678 | 0.00249 | 0.00041 | 0.00013 | 0.02692 | 0.00007 | 0.00032 | 0.00037 | 0.00015 | 0.00002 |
| Edge homophily ($\mathcal{H}_e$) | 0.75 | 0.68 | 0.78 | 0.78 | 0.87 | 0.61 | 0.94 | 0.79 | 0.87 | 0.29 | 0.49 | 0.60 | 0.46 | 0.49 | 0.45 | 0.98 |
| Node homophily ($\mathcal{H}_n$) | 0.76 | 0.68 | 0.76 | 0.76 | 0.86 | 0.53 | 0.91 | 0.77 | 0.85 | 0.24 | 0.51 | 0.50 | 0.48 | 0.50 | 0.45 | 0.98 |

**Datasets.** The hypergraph datasets and their statistics are presented in Table 1. It includes recently proposed heterophilic benchmarks: Actor, Twitch, and 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 without any sparsification, ii) *Edgedeg*: $\kappa$ edges are sampled proportional to edge degree $deg(e) = |e|$. iii) *Random*: drops a incidence pair $(u, e)$ 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 compute effective resistance as $R(e) := \sum_{v \in e} \left(L^{\dagger}\right)_{vv}$, where $L^{\dagger}$ indicates the Moore-Penrose pseudoinverse of the hypergraph Laplacian (Zhou et al., 2006), and (v) Hypergraph Structure Learning, HSL (Cai et al., 2022), which learns a modified hypergraph via both edge reweighting and augmentation. For fair comparison under a sparsification setting, we disable the augmentation component and retain only the learned edge reweighting mechanism, so that all methods operate on the same underlying hypergraph without introducing new edges. Discussion on computational complexity, effectiveness on heterophilic hypergraphs, experiments with other HGNN backbones, scalability, model parameters, link prediction, ablation studies, and empirical evaluation of mask stability can be found in the Appendix. Our source codes: `https://anonymous.4open.science/r/ehgnn-4E4D/`.

### 6.1 Experimental evaluation

**I. Effectiveness on node classification task.** Table 2 highlights the effectiveness of EdgeMask-HGNN.

*Supervised vs Unsupervised sparsifiers:* On most datasets, EHGNN variants consistently outperform unsupervised sparsification baselines. 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 due to the computation of the large matrix pseudoinverse. HSL tends to perform well on datasets with relatively clean and homophilic structure. On heterophilic hypergraphs, its reliance on global reconstruction retains task-irrelevant relations, leading to suboptimal performance.

*Full training vs Sparsification:* EdgeMask-HGNNs outperform full training on several benchmarks, such as ModelNet40, Actor, Citeseer, House, Pokec, and Walmart. This shows that pruning irrelevant pairs enhances

Table 2: Accuracy ($\pm$ std) across different datasets for each algorithm at sparsity = 50%. OOM=Out-of-Memory. **Bold** and Underline text indicates the best and 2nd best result per dataset respectively. Avg. Rank denotes the average ranking across all datasets, where a smaller rank indicates better performance.

| Algorithms | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | Citeseer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| Full | 81.04 ± 0.05 | 94.72 ± 0.05 | 98.73 ± 0.37 | 88.03 ± 0.26 | 64.54 ± 0.03 | 68.19 ± 0.39 | 81.48 ± 0.36 | 90.77 ± 0.07 |
| Random | 76.85 ± 0.30 | 95.91 ± 0.33 | 97.60 ± 0.58 | 86.92 ± 0.36 | 77.85 ± 0.26 | 67.32 ± 1.27 | 75.92 ± 0.53 | 86.26 ± 0.32 |
| Edgedeg | 80.14 ± 0.22 | 95.18 ± 0.23 | 98.02 ± 0.40 | 86.32 ± 0.78 | 76.59 ± 0.18 | 67.27 ± 0.31 | **80.09 ± 0.70** | 88.76 ± 0.27 |
| Spectral | 80.19 ± 0.07 | 94.80 ± 0.07 | 98.06 ± 0.13 | 85.88 ± 0.31 | 75.22 ± 0.03 | 67.05 ± 0.59 | 79.03 ± 0.21 | OOM |
| HSL | **81.68 ± 0.09** | 79.93 ± 2.99 | **99.75 ± 0.00** | 80.58 ± 0.41 | 63.85 ± 0.76 | 63.37 ± 1.06 | 74.84 ± 2.10 | **90.31 ± 0.39** |
| EHGNN-F | 77.18 ± 0.10 | **96.32 ± 0.12** | 97.24 ± 0.21 | 87.48 ± 0.47 | **78.01 ± 0.11** | 66.76 ± 0.45 | 77.49 ± 0.49 | 87.01 ± 0.14 |
| EHGNN-C | 79.08 ± 0.22 | 95.73 ± 0.06 | 93.59 ± 0.50 | 87.20 ± 0.48 | 76.72 ± 0.10 | **68.60 ± 0.59** | 77.02 ± 0.32 | 85.28 ± 0.07 |
| EHGNN-F (cond) | 80.16 ± 0.19 | 95.76 ± 0.10 | 98.28 ± 0.30 | 88.11 ± 0.09 | 77.18 ± 0.33 | 67.71 ± 0.52 | 76.48 ± 0.46 | 86.45 ± 0.10 |
| EHGNN-C (cond) | 78.11 ± 0.98 | 95.01 ± 0.08 | 97.97 ± 0.33 | 87.12 ± 0.60 | 77.01 ± 0.33 | 68.50 ± 0.61 | 76.16 ± 0.56 | 86.29 ± 0.05 |
| EHGNN-F (cond,LR) | 76.94 ± 0.39 | 95.79 ± 0.12 | 98.23 ± 0.39 | 86.00 ± 0.78 | 77.98 ± 0.22 | 67.68 ± 0.41 | 77.02 ± 0.58 | 87.43 ± 0.13 |
| EHGNN-C (cond,LR) | 79.51 ± 0.55 | 95.47 ± 0.17 | 96.57 ± 0.30 | **88.11 ± 0.36** | 76.73 ± 0.42 | 67.97 ± 0.44 | 77.73 ± 0.63 | 86.73 ± 0.55 |

| Algorithms | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Trivago | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|
| Full | 78.23 ± 0.31 | 73.87 ± 0.28 | 58.36 ± 0.04 | 86.50 ± 0.05 | 51.22 ± 0.05 | 95.36 ± 0.01 | 33.60 ± 0.48 | 61.39 ± 1.15 | |
| Random | 73.65 ± 1.31 | 84.52 ± 0.79 | 59.00 ± 0.26 | 86.72 ± 0.24 | 50.92 ± 0.46 | **96.97 ± 0.16** | 32.77 ± 0.28 | 47.14 ± 1.52 | 5.31 |
| Edgedeg | 74.62 ± 1.42 | 80.31 ± 1.13 | 58.71 ± 0.14 | 86.72 ± 0.09 | 50.86 ± 0.38 | 95.40 ± 0.05 | 32.36 ± 0.57 | 53.59 ± 0.72 | 5.19 |
| Spectral | 75.92 ± 0.36 | 75.29 ± 0.14 | 58.47 ± 0.03 | 86.47 ± 0.01 | 51.39 ± 0.05 | OOM | OOM | OOM | 5.92 |
| HSL | 70.56 ± 0.37 | 54.28 ± 1.89 | 57.81 ± 0.71 | 77.51 ± 0.80 | 51.15 ± 0.99 | 62.57 ± 0.16 | OOM | **68.17 ± 2.16** | 7.13 |
| EHGNN-F | 75.72 ± 0.22 | 86.13 ± 0.51 | 58.53 ± 0.03 | 86.75 ± 0.06 | 50.84 ± 0.04 | 96.90 ± 0.02 | **33.36 ± 0.26** | 45.56 ± 1.56 | 4.38 |
| EHGNN-C | 75.60 ± 0.28 | 75.85 ± 0.31 | **59.15 ± 0.05** | 86.56 ± 0.07 | **51.44 ± 0.08** | 95.89 ± 0.08 | 31.98 ± 0.32 | 37.78 ± 1.42 | 5.31 |
| EHGNN-F (cond) | 74.71 ± 0.44 | **100.00 ± 0.00** | 59.14 ± 0.08 | 86.10 ± 0.07 | 50.58 ± 0.08 | 96.39 ± 1.41 | OOM | 40.08 ± 0.29 | 4.73 |
| EHGNN-C (cond) | 75.07 ± 0.48 | 73.37 ± 0.62 | 58.90 ± 0.31 | 86.43 ± 0.13 | 50.32 ± 0.23 | 95.20 ± 0.35 | OOM | 46.03 ± 2.26 | 6.53 |
| EHGNN-F (cond,LR) | 74.33 ± 0.65 | 91.83 ± 4.99 | 58.77 ± 0.18 | **86.97 ± 0.04** | 50.72 ± 0.10 | 96.65 ± 0.22 | 32.78 ± 0.37 | 51.45 ± 1.14 | 4.56 |
| EHGNN-C (cond,LR) | **76.13 ± 0.60** | 79.88 ± 0.31 | 58.93 ± 0.06 | 86.52 ± 0.05 | 51.26 ± 0.20 | 95.47 ± 0.53 | 33.03 ± 0.21 | 40.28 ± 0.89 | 4.56 |

Table 3: Correlation between accuracy gap Acc.(EdgeMask-HGNN) - Acc.(Full) and homophily ratio computed across all datasets.

| Metric | EHGNN-C | | EHGNN-F | |
|---|---|---|---|---|
| | Pearson | Spearman | Pearson | Spearman |
| $\mathcal{H}_e$ | -0.465 | **-0.511** | **-0.429** | **-0.437** |
| $\mathcal{H}_n$ | **-0.477** | **-0.540** | **-0.427** | **-0.455** |

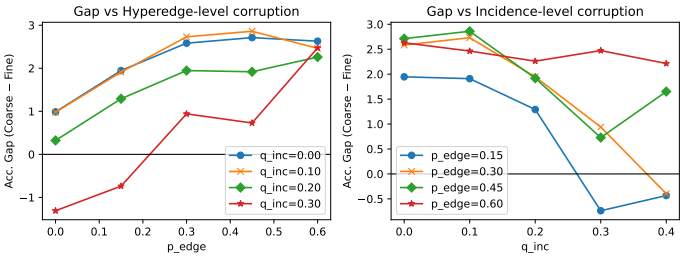



Figure 3: Classification performance on corrupted Cora.

generalization by improving the signal-to-noise ratio during message passing. On datasets such as 20news, Cora, Cora-CA, DBLP-CA, Mushroom, and Trivago Full training performs better.

*On heterophilic dataset:* We have also analyzed the performance gain of our method relative to full training by examining the homophily ratios across various hypergraph datasets. Table 3 suggests that the proposed models generally perform better than Full training, particularly in low homophily (high heterophily) regimes than in high homophily regimes. For instance, the Spearman correlation coefficient between the accuracy improvement of EHGNN-C over Full training vs edge homophily is $-0.51$, and that for EHGNN-F is $-0.44$. Figure 7 (Appendix) illustrates the relation between homophily ratio and accuracy gain EdgeMask-HGNN offers over full training.

*Coarse-grained vs Fine-grained Sparsification:* Although on the real-world datasets, generally we found fine-grained sparsification to perform better than coarse-grained sparsification, it is important to analyze under what conditions coarse-grained sparsification is more preferable than fine-grained or vice versa. To that end, we corrupted the hyperedges in Cora in two distinct yet controlled ways. Each hyperedge is assigned a target adversarial class based on the majority labels in that edge, and corrupted incidences are replaced by nodes drawn from this class. (i) *Coarse/hyperedge-level corruption:* For each edge, with probability $p_{\text{edge}}$, all its nodes are replaced by nodes from its adversarial class. (ii) *fine/incidence-level corruption*: For each edge, with probability $1 - p_{\text{edge}}$, $q_{\text{inc}}$ proportion of its nodes are replaced by nodes from its adversarial class.

Figure 3 reveals a clear dependence of model performance on the granularity of corruption. (i) When $q_{\text{inc}}$ is small ($< 0.3$), increasing $p_{\text{edge}}$ leads to a monotonic increase in the performance gap in favor of coarse-grained sparsification, as entire hyperedges become uninformative and are best removed in their entirety. (ii)

Table 4: Performance comparison between Hyper-SAGNN (Baseline) and various sparsification variants at sparsity=50%.

| **Model** | **WordNet** (user, relation, tail) #V = (40504, 18, 40551) | | | **Drug** (user, drug, reaction) #V = (12, 1076, 6398) | | | **MovieLens** (user, movie, tag) #V = (2113, 5908, 9079) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Acc** | **AUC** | **AUPR** | **Acc** | **AUC** | **AUPR** | **Acc** | **AUC** | **AUPR** |
| Hyper-SAGNN | 88.35 | 87.92 | 68.62 | 93.93 | 95.72 | 89.00 | 90.62 | 90.93 | 75.39 |
| Random | 93.24 | 95.68 | 87.05 | 96.49 | 98.64 | 95.72 | 95.11 | 97.43 | 92.12 |
| EdgeDeg | 93.47 | 96.05 | 87.96 | 96.57 | 98.62 | 95.69 | 95.05 | 97.24 | 91.84 |
| Spectral | 91.29 | 92.02 | 79.75 | 96.80 | 98.72 | 95.97 | 94.08 | 96.16 | 88.85 |
| EHGNN-F | 96.30 | 97.02 | 92.59 | 96.72 | 97.57 | 94.73 | 96.47 | 97.20 | 93.42 |
| EHGNN-C | 93.21 | 95.80 | 87.03 | 96.58 | 98.66 | 95.74 | 95.11 | 97.24 | 91.84 |
| EHGNN-F(cond) | 94.81 | 96.84 | 90.92 | 97.16 | 99.01 | 96.66 | 96.62 | 98.59 | 95.54 |
| EHGNN-C(cond) | 93.76 | 96.49 | 88.81 | 96.70 | 98.81 | 96.14 | 95.59 | 97.81 | 93.16 |
| EHGNN-F(cond,LR) | **98.78** | **99.64** | **99.05** | **98.33** | **99.70** | **98.77** | **98.32** | **99.66** | **98.46** |
| EHGNN-C(cond,LR) | 91.84 | 93.34 | 82.07 | 95.87 | 97.74 | 93.89 | 93.89 | 95.61 | 88.39 |

Conversely, when $p_{\text{edge}}$ is small, increasing $q_{\text{inc}}$ causes the gap to decrease and eventually become negative or close to 0. This suggests that fine-grained masking, which selectively removes corrupted incidences, is more beneficial when incidence-level corruption is more predominant and edge-level corruption is small. However, when $p_{\text{edge}}$ is large, many hyperedges are already fully corrupted, and coarse masking remains advantageous regardless of $q_{\text{inc}}$.

**II. Effectiveness on link prediction task.** We also evaluate the applicability of EdgeMask-HGNN to link prediction by adapting it to **Hyper-SAGNN** (Zhang et al., 2020), a self-attention based architecture for hyperedge prediction. In the inductive setting, the model is trained on $\mathcal{E}_{train} \subset \mathcal{E}$ and evaluated on unseen hyperedges from $\mathcal{E}_{val}$ and $\mathcal{E}_{test}$. Training examples consist of positive edges $e \in \mathcal{E}_{train}$ and negative examples generated by corrupting one position of a positive hyperedge with another node of the same type.

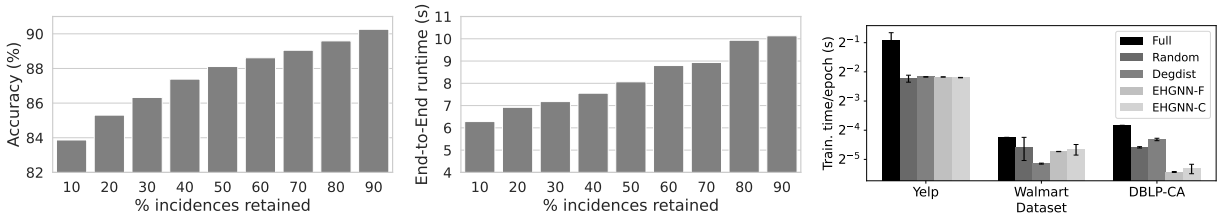
Hyper-SAGNN computes node embeddings $\boldsymbol{h}_v^{(0)} = \text{Embedding}(v)$ and applies self-attention within each hyperedge to yield an edge representation $\boldsymbol{z}_e$, which is later used for link prediction. However, Hyper-SAGNN processes each hyperedge independently without exploiting global hypergraph structure. In order to enable global hypergraph-structure aware link prediction, we introduce a learnable sparsification module $\texttt{EdgeMask-HGNN}_\Phi()$ with parameters $\Phi$, which learns a binary selection variable $m_e$ that determines whether an incident hyperedge contributes to the contextualized node representations. During training, we use a straight-through estimator to enable gradient-based optimization of this discrete selection. We construct contextualized node embeddings as follows:

$$\boldsymbol{h}_v^{(1)} = \boldsymbol{h}_v^{(0)} + \sum_{e: v \in e} \boldsymbol{m}_e \boldsymbol{g}_e,$$

where $\boldsymbol{g}_e = \frac{1}{|e|} \sum_{u \in e} \boldsymbol{W} \boldsymbol{h}_u^{(0)}$ is an edge message and $\boldsymbol{W}$ is a learnable parameter. The updated embeddings $\boldsymbol{h}_v^{(1)}$ are then fed into Hyper-SAGNN so that the predictions for a candidate hyperedge can incorporate information from other hyperedges sharing nodes. The contextualized representations enabled by sparsifiers help control the impact of noise on message passing. A detailed discussion is presented in Appendix E.

Experiments using variants of EdgeMask-HGNN, as presented in Table 4, show large improvements in accuracy, AUC, and AUPR at 50% sparsification. This indicates that the learned sparsifier effectively removes noisy hyperedges and improves representation quality. Compared to unsupervised sparsifiers, fine-grained sparsifier variants, in particular, EHGNN-F(cond,LR) yield superior performance. Overall, these results demonstrate that EdgeMask-HGNN can extend Hyper-SAGNN by introducing task-aware hypergraph sparsification, thus enabling structurally informed inductive link prediction.

**III. Accuracy/Runtime vs sparsity trade-off.** We analyze the trade-off between accuracy and sparsity in Figure 4a, and the impact of sparsity on end-to-end runtime (Training + Evaluation) in Figure 4b. On the x-axis, we plot % incidences retained, which is 100%-sparsity. As more incidences are retained, accuracy increases at the cost of higher runtime. Conversely, the more we sparsify, the lower the accuracy and runtime. This phenomenon mirrors that observed in graph sparsification (Das et al., 2025).




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

Figure 4: (a-b) Impact of sparsity on EHGNN-F's performance (DBLP-CA dataset) (c) Training time comparison of various sparsifiers.

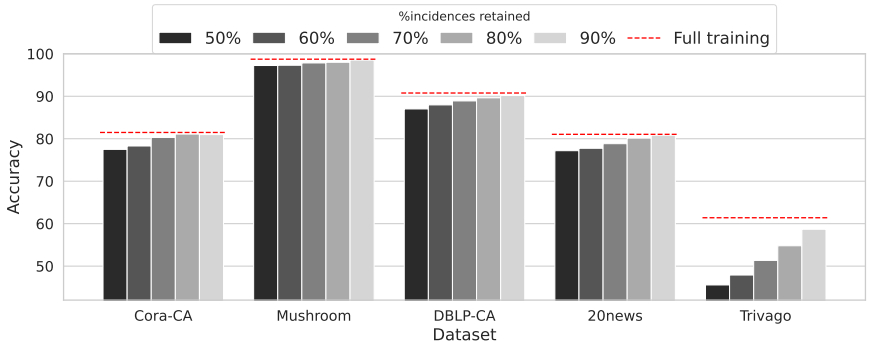



Figure 5: Impact of %retained incidences on EHGNN-F for homophilic hypergraphs.

**IV. Runtime efficiency.** Figure 4c compares the average training time per epoch across several large-scale datasets. On Yelp, Walmart, and DBLP-CA, the Full HGNN consistently incurs the highest, while both EHGNN-F and EHGNN-C achieve substantially lower runtimes. This aligns with the fact that incidence-level sparsification reduces the message-passing workload, which dominates the cost on large hypergraphs.

**V. Ablation study: Impact of sparsity on homophilic hypergraphs.** In table 2, we observe that on datasets 20news, Cora-CA, DBLP-CA, Mushroom, and Trivago, Full training outperforms EdgeMask-HGNN. These hypergraphs have edge homophily ratio 0.53, 0.76, 0.86, 0.91, and 0.98, respectively. Here, we investigate whether increasing the budget (% incidences retained) improves the performance of EHGNN-F on these highly homophilic datasets.

Figure 5 shows the result with varying % of incidence retained by EHGNN-F. We observe a near-monotonic increase in accuracy as more incidences are retained. This suggests that highly homophilic hypergraphs have a high sensitivity to sparsification. For instance, even at a high retention level (0.9), EHGNN-F does not fully match the performance of Full training (red dashed lines), especially on datasets such as Trivago, where the performance gap remains significant. This indicates that, in strongly homophilic regimes, sparsification may discard weak but collectively important connections that contribute to label propagation.

This is a key limitation of sparsification in homophilic settings. Although moderate pruning yields competitive performance, a large fraction of incidences need to be retained to yield performance equivalent to full-training. This contrasts with heterophilic scenarios, where sparsification can be more beneficial by removing noisy or misleading connections.

## 7 Conclusion, Limitations and Future works

We introduced EdgeMask-HGNN, a task-aware sparsification framework that prunes preserves and often improves the predictive performance of HGNNs on downstream tasks by pruning task-irrelevant edges. We proposed two learnable sparsification 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 the mask stability of the learned sparsifiers.

Extensive experiments across diverse and challenging node classification benchmarks suggest EdgeMask-HGNN is generally effective, while most effective on heterophilic hypergraphs. In particular, the fine-grained variants not only improve accuracy over full hypergraph training in many cases, but also achieve slightly smaller training time on large hypergraphs. Finally, EdgeMask-HGNN is adaptable to other downstream tasks, such as link prediction, offering superior performance over the baselines.

*Limitations.* A key limitation of EdgeMask-HGNN is that it relies on supervised signals and may not generalize as effectively in limited-label settings or fully unsupervised settings (e.g., node clustering). The additional computational overhead introduced by feature-conditioned variants also causes scalability challenges on large-scale hypergraphs. In the future, we plan to address these limitations.

## 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.

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.

Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *IJCAI*, pp. 1923–1929, 2022.

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

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.

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.

Mikhail Hayhoe, Hans Matthew Riess, Michael M Zavlanos, Victor Preciado, and Alejandro Ribeiro. Transferable hypergraph neural networks via spectral similarity. In *Learning on Graphs Conference*, pp. 18–1. PMLR, 2024.

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.

Bo Hui, Da Yan, Xiaolong Ma, and Wei-Shinn Ku. Rethinking graph lottery tickets: Graph sparsity matters. In *The Eleventh International Conference on Learning Representations*, 2023.

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.

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.

Ningyi Liao, Zihao Yu, Ruixiao Zeng, and Siqiang Luo. Unifews: You need fewer operations for efficient graph neural networks. In *International Conference on Machine Learning*, pp. 37587–37609. PMLR, 2025.

Chuang Liu, Xueqi Ma, Yibing Zhan, Liang Ding, Dapeng Tao, Bo Du, Wenbin Hu, and Danilo P Mandic. Comprehensive graph gradual pruning for sparse training in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):14903–14917, 2023a.

Zirui Liu, Kaixiong Zhou, Zhimeng Jiang, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. Dspar: An embarrassingly simple strategy for efficient gnn training and inference via degree-based sparsification. *Transactions on Machine Learning Research*, 2023b.

R Duncan Luce et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.

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.

Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

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.

Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. Graph sparsification via mixture of graphs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=7ANDviElAo`.

R Zhang, Y Zou, and J Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations (ICLR)*, 2020.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International conference on machine learning*, pp. 11458–11468. PMLR, 2020.

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 analyze the stability of the stochastic masks produced by EHGNN-F and EHGNN-C during training. In particular, we study how changes in the logit parameters affect the sampled incidence sets (for EHGNN-F) and sampled edges (for EHGNN-C) between consecutive epochs.

### A.1 Stability of Fine-grained Stochastic Masks

We analyze the stochastic fine mask used in EHGNN-F during training, where $k$ incidences are sampled without replacement from $t$ candidate incidences using probabilities proportional to weights $w_{v,e} := \sigma(s_{v,e})$.

Let $t = \sum_{e \in E} |e|$ denote the number of incidence pairs and $k$ denotes the number of retained incidence pairs after sparsification. At iteration $\tau$, let the logits $\boldsymbol{s}^{(\tau)} \in \mathbb{R}^t$ and the weights

$$w^{(\tau)}(i) := \sigma(s^{(\tau)}(i)) \in (0,1), \quad i \in [t].$$

For a fixed $k \in \{1, \ldots, t\}$, sampling $k$ incidence pairs without replacement with probabilities proportional to weights is equivalent to:

1. Sampling a permutation $\Pi^{(\tau)} \sim \mathrm{PL}(\boldsymbol{w}^{(\tau)})$ under the Plackett–Luce (PL) distribution (Luce et al., 1959).
2. Then selecting the first k indices

$$S^{(\tau)} := \{\Pi_1^{(\tau)}, \ldots, \Pi_k^{(\tau)}\} \subseteq [t], \quad |S^{(\tau)}| = k.$$

The selected set $S^{(\tau)}$ uniquely determines the incidence mask produced by EHGNN-F. We measure stability of fine-grained stochastic masks via the symmetric difference:

$$|S^{(\tau+1)} \triangle S^{(\tau)}|.$$

This equals the number of incidences whose inclusion state changes between epochs.

**Assumption A.1** (Uniformly lower-bounded weights)**.** There exists $\alpha > 0$ such that for all $\tau$ and all $i \in [t]$,

$$w^{(\tau)}(i) \geq \alpha.$$

This assumption holds, for instance, if logits are uniformly bounded. In particular, since $\sigma(x) > 0$, a uniform lower bound $s^{(\tau)}(i) \geq -B$ implies the following uniform weight lower bound

$$w^{(\tau)}(i) \geq \sigma(-B) := \alpha.$$

First, we formalize how perturbations in the weights $w^{(\tau)}$ at epoch $\tau$ affect the distribution of the first $k$ positions of the Plackett–Luce (PL) permutation.

**Lemma A.2** (Single-step conditional stability of PL weights)**.** *Let $R \subseteq [t]$ with $|R| = r$. Let us define categorical distributions on $R$:*

$$u(i) = \frac{w(i)}{\sum_{j \in R} w(j)}, \quad v(i) = \frac{w'(i)}{\sum_{j \in R} w'(j)}.$$

*If $w(i), w'(i) \geq \alpha$ for all $i$, then the total variation distance*

$$\mathrm{TV}(u, v) \leq \frac{1}{r\alpha} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

*Proof.* Let

$$P_R := \sum_{j \in R} w(j), \quad Q_R := \sum_{j \in R} w'(j).$$

Since $w(j) \geq \alpha$, we have

$$P_R \geq r\alpha, \quad Q_R \geq r\alpha.$$

By definition of TV,

$$\mathrm{TV}(u, v) = \frac{1}{2} \sum_{i \in R} \left| \frac{w(i)}{P_R} - \frac{w'(i)}{Q_R} \right|.$$

Insert and subtract $\frac{w(i)}{Q_R}$:

$$\left| \frac{w(i)}{P_R} - \frac{w'(i)}{Q_R} \right| \leq \left| \frac{w(i)}{P_R} - \frac{w(i)}{Q_R} \right| + \left| \frac{w(i) - w'(i)}{Q_R} \right|.$$

Summing over $i \in R$ yields

$$\sum_{i \in R} \left| \frac{w(i)}{P_R} - \frac{w'(i)}{Q_R} \right| \leq \frac{|Q_R - P_R|}{Q_R} + \frac{1}{Q_R} \sum_{i \in R} |w(i) - w'(i)|.$$

Since

$$|Q_R - P_R| = \left| \sum_{j \in R} (w'(j) - w(j)) \right| \leq \|\boldsymbol{w} - \boldsymbol{w}'\|_1,$$

and $Q_R \geq r\alpha$, we obtain

$$\mathrm{TV}(u, v) \leq \frac{1}{r\alpha} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

□

**Lemma A.3** (Total variation bound for the first $k$ positions of the Plackett–Luce permutation)**.** *Let $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^t$ satisfy $w(i), w'(i) \geq \alpha > 0 \quad \forall i \in [t]$. Let $\Pi^{(w)}$ and $\Pi^{(w')}$ be permutations sampled from the Plackett–Luce distributions with base weights $w$ and $w'$ respectively. Let us denote the first $k$ positions by $\Pi^{(w)}_{1:k}, \quad \Pi^{(w')}_{1:k}$. Then*

$$\mathrm{TV}\big(\Pi^{(w)}_{1:k}, \Pi^{(w')}_{1:k}\big) \leq \frac{1}{\alpha} \ln \frac{t}{t-k} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

*Proof.* The PL process samples sequentially.

At step $r$, given previously selected elements $\Pi_{1:r-1}$, the next element is drawn from the remaining set $R_r$ of size $|R_r| = t - r + 1$. At step $r$ of the Plackett–Luce process, let us denote

$$\mathrm{TV}_r := \mathrm{TV}\big(u_{R_r}, v_{R_r}\big),$$

where $u_{R_r}$ and $v_{R_r}$ are the conditional categorical distributions restricted to $R_r$.

By Lemma A.2, for any realization of the remaining set $R_r$

$$\mathrm{TV}_r \leq \frac{1}{(t - r + 1)\alpha} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

We now construct a maximal coupling of the two PL processes, coupling the draws at each step.

Let $E_r$ be the event that the two processes differ for the first time at step $r$. Using maximal coupling at each step of the sequential PL sampling process, the probability that the two permutations first disagree at step

$r$, conditioned on agreement up to step $r-1$, is at most the total variation distance between the conditional distributions at that step:

$$\mathbb{P}(E_r \mid \text{agreement up to } r-1) \leq \mathrm{TV}_r,$$

Therefore, by a union bound over the first $k$ steps,

$$\mathbb{P}\big(\Pi_{1:k}^{(w)} \neq \Pi_{1:k}^{(w')}\big) \leq \sum_{r=1}^{k} \mathrm{TV}_r = \sum_{r=1}^{k} \frac{1}{(t-r+1)\alpha} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

Since

$$\sum_{r=1}^{k} \frac{1}{t-r+1} = H_t - H_{t-k},$$

we obtain

$$\mathbb{P}\big(\Pi_{1:k}^{(w)} \neq \Pi_{1:k}^{(w')}\big) \leq \frac{H_t - H_{t-k}}{\alpha} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

Here $H_t := \sum_{i=1}^{t} \frac{1}{i}$. Since total variation distance equals the minimal disagreement probability under coupling,

$$\mathrm{TV}\big(\Pi_{1:k}^{(w)}, \Pi_{1:k}^{(w')}\big) \leq \frac{H_t - H_{t-k}}{\alpha} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

As $H_t - H_{t-k} \leq \ln \frac{t}{t-k}$, we have

$$\mathrm{TV}\big(\Pi_{1:k}^{(w)}, \Pi_{1:k}^{(w')}\big) \leq \frac{1}{\alpha} \ln \frac{t}{t-k} \|\boldsymbol{w} - \boldsymbol{w}'\|_1.$$

□

We now state and prove a data processing property of total variation distance, which states that total variation distance cannot increase under deterministic mappings. We will apply this property of total variation distance to pass from permutation prefixes to subsets.

**Lemma A.4** (Data processing inequality for TV)**.** *Let $X$ and $Y$ be random variables with distributions $\mu$ and $\nu$ respectively. Let $f$ be any deterministic measurable function. Then the distributions of $f(X)$ and $f(Y)$ satisfy*

$$\mathrm{TV}\big(\mathcal{L}(f(X)),\, \mathcal{L}(f(Y))\big) \leq \mathrm{TV}(\mu, \nu).$$

*Proof.* Let $\mu_f := \mathcal{L}(f(X))$ and $\nu_f := \mathcal{L}(f(Y))$ denote the distributions of the transformed random variables $f(X)$ and $f(Y)$ respectively. Recall that the total variation distance between two probability measures $\mu$ and $\nu$ is defined as

$$\mathrm{TV}(\mu, \nu) = \sup_A |\mu(A) - \nu(A)|.$$

Let $B$ be any measurable subset of the codomain of $f$. Then the events $\{f(X) \in B\}$ and $\{X \in f^{-1}(B)\}$ are identical. Therefore

$$\mu_f(B) = \mathbb{P}(f(X) \in B) = \mu(f^{-1}(B)), \qquad \nu_f(B) = \mathbb{P}(f(Y) \in B) = \nu(f^{-1}(B)).$$

Hence

$$|\mu_f(B) - \nu_f(B)| = |\mu(f^{-1}(B)) - \nu(f^{-1}(B))|.$$

Since $f^{-1}(B)$ is a measurable subset of the domain,

$$|\mu(f^{-1}(B)) - \nu(f^{-1}(B))| \leq \sup_A |\mu(A) - \nu(A)| = \mathrm{TV}(\mu, \nu).$$

Taking the supremum over all measurable sets $B$ yields

$$\mathrm{TV}(\mu_f, \nu_f) = \sup_B |\mu_f(B) - \nu_f(B)| \leq \mathrm{TV}(\mu, \nu).$$

□

In our setting, the mapping $f$ sends the prefix permutation $\Pi_{1:k}$ to the selected subset $S$. Hence, the total variation distance between the distributions of $S$ is bounded by that of the prefixes:

$$\mathrm{TV}(S(\boldsymbol{w}), S(\boldsymbol{w}')) \leq \mathrm{TV}(\Pi_{1:k}^{(w)}, \Pi_{1:k}^{(w')}).$$

We now bound the expected number of incidence flips between two consecutive training iterations.

**Theorem 5.1** (Epoch-to-epoch stability of fine-grained masks)**.** *Let $s^{(\tau)}, s^{(\tau+1)} \in \mathbb{R}^t$ denote the logit vectors at two consecutive epochs, and let*

$$\boldsymbol{w}^{(\tau)} := \sigma(\boldsymbol{s}^{(\tau)}), \qquad \boldsymbol{w}^{(\tau+1)} := \sigma(\boldsymbol{s}^{(\tau+1)}).$$

*Let $S^{(\tau)}$ and $S^{(\tau+1)}$ denote the $k$-subsets obtained by sampling from the Plackett–Luce distributions $\mathrm{PL}(\boldsymbol{w}^{(\tau)})$ and $\mathrm{PL}(\boldsymbol{w}^{(\tau+1)})$ respectively.*

*Under Assumption A.1, the expected number of incidence mask flips between the two consecutive epochs satisfies*

$$\mathbb{E}\Big[|S^{(\tau+1)} \triangle S^{(\tau)}|\Big] \leq \frac{k}{2\alpha} \ln \frac{t}{t-k} \, \|\boldsymbol{s}^{(\tau+1)} - \boldsymbol{s}^{(\tau)}\|_1.$$

*Proof.* Since $|S^{(\tau)}| = |S^{(\tau+1)}| = k$, we have

$$|S^{(\tau+1)} \triangle S^{(\tau)}| \leq 2k \cdot \mathbf{1}\{S^{(\tau+1)} \neq S^{(\tau)}\}.$$

Taking expectations gives

$$\mathbb{E}\left[|S^{(\tau+1)} \triangle S^{(\tau)}|\right] \leq 2k \, \mathbb{P}(S^{(\tau+1)} \neq S^{(\tau)}).$$

Under maximal coupling,

$$\mathbb{P}(S^{(\tau+1)} \neq S^{(\tau)}) = \mathrm{TV}\big(S(\boldsymbol{w}^{(\tau+1)}), S(\boldsymbol{w}^{(\tau)})\big).$$

By Lemmas A.4 and A.3,

$$\mathrm{TV}\big(S(\boldsymbol{w}^{(\tau+1)}), S(\boldsymbol{w}^{(\tau)})\big) \leq \mathrm{TV}(\Pi_{1:k}^{(w^{(\tau+1)})}, \Pi_{1:k}^{(w^{(\tau)})}) \leq \frac{1}{\alpha} \ln \frac{t}{t-k} \|\boldsymbol{w}^{(\tau+1)} - \boldsymbol{w}^{(\tau)}\|_1.$$

Thus

$$\mathbb{E}\left[|S^{(\tau+1)} \triangle S^{(\tau)}|\right] \leq 2k \cdot \frac{1}{\alpha} \ln \frac{t}{t-k} \|\boldsymbol{w}^{(\tau+1)} - \boldsymbol{w}^{(\tau)}\|_1.$$

Since $\sigma'(x) = \sigma(x)(1 - \sigma(x)) \leq \frac{1}{4}$, the sigmoid map is 1/4-Lipschitz. Hence

$$\|\boldsymbol{w}^{(\tau+1)} - \boldsymbol{w}^{(\tau)}\|_1 \leq \frac{1}{4} \|\boldsymbol{s}^{(\tau+1)} - \boldsymbol{s}^{(\tau)}\|_1.$$

Substituting yields

$$\mathbb{E}\Big[|S^{(\tau+1)} \triangle S^{(\tau)}|\Big] \leq \frac{k}{2\alpha} \ln \frac{t}{t-k} \, \|\boldsymbol{s}^{(\tau+1)} - \boldsymbol{s}^{(\tau)}\|_1.$$

□

Thus, the expected number of incidence flips between epochs is controlled by the change in the logit vector. We now show that an analogous stability guarantee holds for the coarse-grained mask used in EHGNN-C.

### A.2 Stability of Coarse-grained Stochastic Masks

**Theorem 5.2** (Epoch-to-epoch stability of coarse-grained masks)**.** *Let $m = |E|$ be the number of hyperedges and fix an edge budget $\kappa \in \{1, \dots, m\}$. At iteration $\tau$, let edge logits $s_e^{(\tau)} \in \mathbb{R}^m$ and edge weights*

$$w_e^{(\tau)}(e) := \sigma(s_e^{(\tau)}(e)) \in (0,1), \qquad e \in [m].$$

*EHGNN-C samples a permutation $\Gamma^{(\tau)} \sim \mathrm{PL}(\boldsymbol{w}_e^{(\tau)})$ and selects the first $\kappa$ edges*

$$T^{(\tau)} := \{\Gamma_1^{(\tau)}, \dots, \Gamma_\kappa^{(\tau)}\} \subseteq [m].$$

*Assume there exists $\alpha_c > 0$ such that $w_e^{(\tau)}(e) \geq \alpha_c$ for all $\tau, e$. Then the expected number of edge-mask flips between consecutive epochs satisfies*

$$\mathbb{E}\Big[|T^{(\tau+1)} \triangle T^{(\tau)}|\Big] \leq \frac{\kappa}{2\alpha_c} \ln \frac{m}{m-\kappa} \, \|\boldsymbol{s}_e^{(\tau+1)} - \boldsymbol{s}_e^{(\tau)}\|_1.$$

*Proof.* The result follows from the same argument as Theorem 5.1 for the fine-grained mask. In EHGNN-C the stochastic mask is generated by sampling a $\kappa$-subset of hyperedges from a Plackett–Luce distribution with weights $\boldsymbol{w}_e^{(\tau)} = \sigma(\boldsymbol{s}_e^{(\tau)})$.

Applying the total variation bound for the first $\kappa$ positions of the Plackett–Luce permutation (Lemma A.3) together with the data processing property (Lemma A.4) yields

$$\mathbb{P}\Big(T^{(\tau+1)} \neq T^{(\tau)}\Big) \leq \frac{1}{\alpha_c} \ln \frac{m}{m-\kappa} \|\boldsymbol{w}_e^{(\tau+1)} - \boldsymbol{w}_e^{(\tau)}\|_1.$$

Since $|T^{(\tau)}| = |T^{(\tau+1)}| = \kappa$, we have

$$|T^{(\tau+1)} \triangle T^{(\tau)}| \leq 2\kappa \, \mathbf{1}\{T^{(\tau+1)} \neq T^{(\tau)}\}.$$

Taking expectations and using the fact that the sigmoid is 1/4-Lipschitz gives

$$\|\boldsymbol{w}_e^{(\tau+1)} - \boldsymbol{w}_e^{(\tau)}\|_1 \leq \frac{1}{4} \|\boldsymbol{s}_e^{(\tau+1)} - \boldsymbol{s}_e^{(\tau)}\|_1,$$

which yields the stated bound. □

## 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 $n, m, t, F$, 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. In addition, Let $h$ denotes hidden layer size of MLP, and $\kappa$ denotes the number of hyperedges retained after sparsification, meaning, $\kappa \approx m.k/t$, where $t = \sum_{e \in E} |e|$. Table 5 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.

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

**Fine-grained EdgeMask-HGNN variants.** *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 PL 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 $F$, resulting in a total message passing cost of $\mathcal{O}(kF)$. Summing these components, the overall time complexity per forward pass is: $\mathcal{O}(t \log k + kF)$.

In EHGNN-F(cond), computing incidence scores requires first aggregating node features via $\boldsymbol{H}^T\boldsymbol{X}$ to obtain hyperedge embeddings $\hat{X}_e$, concatenating with node features $\boldsymbol{X}_v$, and then evaluating a shared scorer on each original incidence pair $(v, e)$. Aggregation costs $\Theta(tF)$, Concatenation costs $\Theta(2tF)$, and evaluating the MLP scorer with hidden width $h$ for all incidences costs $\Theta(tFh)$. Since $h$ is a constant, the overall cost of computing scores $s_{v,e}$ is $\mathcal{O}(tF)$. Together with the PL sampling cost and message passing cost, the overall time complexity of EHGNN-F(cond) is $\mathcal{O}(t \log k + kF + tF)$.

In EHGNN-F(cond,LR), the node factor construction costs $\mathcal{O}(nFr)$, since it involves multiplying matrices of dimensions $n \times F$ and $F \times r$. Incidence scorer, which computes $r$-dimensional dot product costs $\Theta(tr)$. The top-$k$ selection cost and message passing costs remain the same as EHGNN-F. This yields total runtime complexity of $\mathcal{O}(nFr + t \log k + kF + tr)$.

*Space/Activation complexity:* The space complexity of EHGNN-F consists of both the memory required for storing scores $s_{v,e}$ and intermediate activations stored during HGNN message passing. 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 selects the top-$k$ incidences and performs message passing only over this pruned subset, requiring storage of $\mathcal{O}(kF)$ for the selected node or edge features. Thus, the total space complexity is: $\mathcal{O}(t + kF)$. This becomes prohibitive in large, dense hypergraphs where $t \gg n, m$; however, it is still more efficient than Full HGNN's space complexity $\mathcal{O}(tF)$.

In EHGNN-F(cond), the activation cost remains $\mathcal{O}(t + kF)$, because the pooled embeddings $\hat{\boldsymbol{X}}_e$ are intermediate tensors that do not persist once the scorers are computed. Similarly, in EHGNN-F(cond,LR) the activation cost remains $\mathcal{O}(t + kF)$.

*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: $\mathcal{O}(t)$.

However, in EHGNN-F(cond), the parameter overhead originates from the MLP. A 2-layer MLP with $h$ hidden layers costs $\mathcal{O}(hF)$. Assuming $h$ as constant, the overall parameter overhead of EHGNN-F(cond) is $\mathcal{O}(F)$.

Table 5: Time and space (activation) complexity, and parameter overhead comparison. Here $F$ = feature dimension, $m = |E|$ 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. Low-rank variants use rank $r \ll F$.

| **Method** | **Time** | **Space/Activation** | **Param. overhead** |
|---|---|---|---|
| Full HGNN | $\mathcal{O}(tF)$ | $\mathcal{O}(tF)$ | None |
| EHGNN-F | $\mathcal{O}(t \log k + kF)$ | $\mathcal{O}(t + kF)$ | $\mathcal{O}(t)$ |
| EHGNN-F (cond) | $\mathcal{O}(t \log k + kF + tF)$ | $\mathcal{O}(t + kF)$ | $\mathcal{O}(F)$ |
| EHGNN-F (cond, LR) | $\mathcal{O}(nFr + t \log k + kF + tr)$ | $\mathcal{O}(t + kF)$ | $\mathcal{O}(Fr + mr)$ |
| EHGNN-C | $\mathcal{O}(m \log \kappa + kF)$ | $\mathcal{O}(m + kF)$ | $\mathcal{O}(m)$ |
| EHGNN-C (cond) | $\mathcal{O}(tF + m \log \kappa + kF + mF)$ | $\mathcal{O}(m + kF)$ | $\mathcal{O}(F)$ |
| EHGNN-C (cond, LR) | $\mathcal{O}(nFr + m \log \kappa + kF + tr + mr)$ | $\mathcal{O}(m + kF)$ | $\mathcal{O}(Fr + mr)$ |

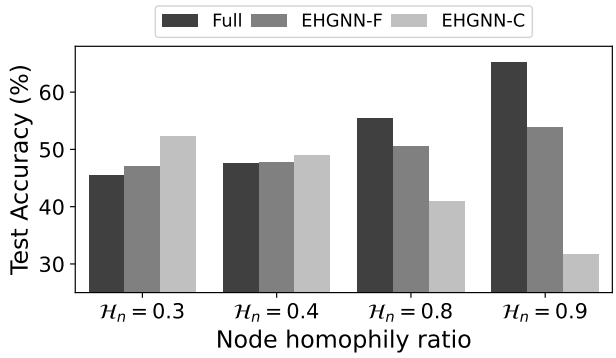



Figure 6: Performance of EdgeMask-HGNN on synthetic hypergraphs having the same # nodes, but different node homophily ratios.

In EHGNN-F(cond,LR), the parameter overhead comes from node feature projection matrix $\boldsymbol{W}_x \in \mathbb{R}^{F \times r}$ and hyperedge factor $Z \in \mathbb{R}^{m \times r}$, incurring a total overhead of $\mathcal{O}(Fr + mr)$.

**Coarse-grained EdgeMask-HGNN variants.** We present their complexity in Table 5.

# C Additional Experiments

## C.1 Effectiveness on heterophilic hypergraphs.

Li et al. (2025) proposed a synthetic dataset containing hypergraphs with various homophily ratios. We evaluate Full trianing, EHGNN-F and EHGNN-C on this dataset to understand the effectiveness of EdgeMask-HGNN on synthetically generated hypergraphs with controlled homophily ratio. The results with 50% sparsification are presented in Figure 6.

On highly heterophilic hypergraphs (e.g., homophily ratio = 0.3 and 0.4), where connected nodes often have dissimilar labels, EHGNN-C and EHGNN-F outperforms Full training. This suggests that sparsification prunes cross-class connections that are harmful for label propagation in highly heterophilic settings. However, as the hypergraph becomes more homophillic, sparsification hurts the performance.

The performance of EHGNN-F becomes better than that of EHGNN-C as hypergraphs become more homophilic (e.g., homophily ratio = 0.8 and 0.9). This suggests that across homophily ratios, EHGNN-F better maintains its performance by selectively pruning incidences that adversely affect the downstream node classification accuracy.

Furthermore, we have analyzed their performance gain relative to Full training by examining the homophily ratios across various hypergraph datasets. Figure 7 indicates a negative correlation, which suggests that

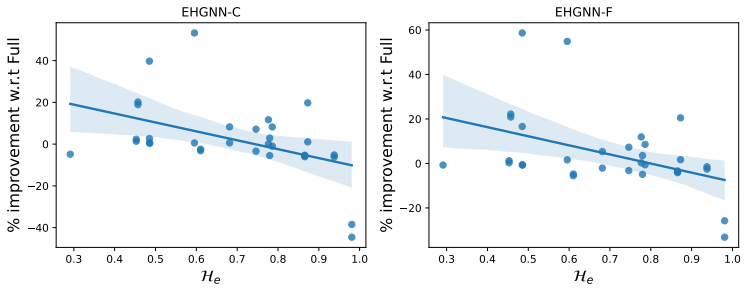



Figure 7: Correlation between Edge homophily vs performance gain of EHGNN-C and EHGNN-F. EdgeMask-HGNN variants offer more accuracy gain near low homophily/high heterophily regimes.

Table 6: Comparing various state-of-the-art HGNNs and their EHGNN-F enhanced counterparts (sparsity = 50%).

| Models | 20news | Actor | Citeseer | Cora | Cora-CA | DBLP-CA | House | ModelNet40 |
|---|---|---|---|---|---|---|---|---|
| AllSetTrans. | 82.23 ± 0.26 | 67.24 ± 0.54 | **67.37 ± 0.92** | **76.43 ± 0.45** | **81.71 ± 1.23** | **91.11 ± 0.06** | **100.00 ± 0.00** | **97.94 ± 0.15** |
| AllSetTrans.+EHGNN-F | **82.30 ± 0.15** | **85.41 ± 0.29** | 66.59 ± 0.93 | 73.77 ± 1.13 | 77.28 ± 1.59 | 88.48 ± 0.21 | 98.33 ± 2.35 | 97.71 ± 0.08 |
| ED-GNN | **82.36 ± 0.16** | 67.73 ± 0.10 | **69.42 ± 0.50** | **78.14 ± 0.50** | **82.36 ± 0.65** | **91.44 ± 0.09** | 93.31 ± 4.74 | 97.36 ± 0.12 |
| ED-GNN+EHGNN-F | 80.66 ± 0.08 | **83.10 ± 0.07** | 69.20 ± 0.76 | 75.45 ± 0.78 | 78.97 ± 0.85 | 89.66 ± 0.18 | **97.89 ± 1.08** | **97.63 ± 0.08** |
| HyperUFG | **81.74 ± 0.24** | 75.94 ± 1.73 | 69.90 ± 0.62 | **74.12 ± 0.46** | **74.24 ± 0.50** | 86.53 ± 0.08 | **100.00 ± 0.00** | 91.35 ± 0.10 |
| HyperUFG+EHGNN-F | 81.71 ± 0.21 | **80.27 ± 0.89** | **69.93 ± 0.48** | 74.03 ± 0.32 | 74.21 ± 0.31 | **86.66 ± 0.06** | **100.00 ± 0.00** | **91.82 ± 0.21** |
| | **Mushroom** | **NTU** | **Pokec** | **PubMed** | **Twitch** | **Walmart** | **Yelp** | |
| AllSetTrans. | **100.00 ± 0.00** | 88.23 ± 0.43 | 59.25 ± 0.35 | 87.90 ± 0.42 | **50.85 ± 0.27** | 99.79 ± 0.04 | OOM | |
| AllSetTrans.+EHGNN-F | 99.98 ± 0.04 | **89.03 ± 0.43** | **59.64 ± 0.63** | **87.91 ± 0.30** | 50.52 ± 0.05 | **99.98 ± 0.01** | **31.01 ± 0.58** | |
| ED-GNN | **99.82 ± 0.07** | **89.38 ± 0.41** | 58.94 ± 0.67 | **88.02 ± 0.25** | 50.14 ± 0.04 | 99.34 ± 0.17 | 30.49 ± 1.69 | |
| ED-GNN+EHGNN-F | 99.28 ± 0.14 | 89.15 ± 0.46 | **59.00 ± 0.44** | 87.98 ± 0.21 | **50.20 ± 0.14** | **99.69 ± 0.08** | **31.51 ± 0.42** | |
| HyperUFG | 89.31 ± 3.51 | 69.54 ± 0.76 | 55.78 ± 1.01 | 86.22 ± 0.50 | 49.36 ± 0.21 | **100.00 ± 0.00** | 29.17 ± 0.94 | |
| HyperUFG+EHGNN-F | **89.39 ± 4.85** | **71.13 ± 0.82** | **56.03 ± 0.83** | **86.30 ± 0.39** | **49.50 ± 0.41** | **100.00 ± 0.00** | **29.54 ± 0.71** | |

the proposed models perform better in low homophily (high heterophily) regimes than in high homophily regimes. For instance, on EHGNN-C, the Spearman correlation coefficient is $-0.51$, and on EHGNN-F it is $-0.44$.

### C.2 Adaptability to existing HGNN backbones.

EdgeMask-HGNN is model-agnostic and easily adaptable to different hypergraph architectures. To demonstrate this, we have employed EHGNN-F into existing architectures such as AllSetTransformer (Chien et al., 2021), ED-HNN (Wang et al., 2023), and a heterophily-specific architecture– HyperUFG (Li et al., 2025). The performance comparison is presented in Table 6. 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 the existing HGNNs' representational power.

### C.3 Exact vs. Approximate effective resistance sparsification.

In addition to the exact effective-resistance-based sparsifier, which requires a dense pseudoinverse of the hypergraph Laplacian, we implemented an approximate spectral sparsifier using a Hutchinson-type random projection estimator for $\mathrm{diag}(L^+)$. Specifically, we sample vectors $g$ from Rademacher distribution, solve $(L+\lambda I)z = g$ via Conjugate Gradient, and approximate $\mathrm{diag}(L^+) \approx \mathbb{E}[z \odot g]$. These approximate resistances are then aggregated per hyperedge as in the exact method. This follows the standard line of random-projection-based spectral sparsification and avoids forming the pseudoinverse explicitly.

In practice, this approximation yields some loss in accuracy compared to full training while being significantly more scalable. Table 7 shows that on DBLP-CA it reduces memory consumption than exact sparsification, while enables training on Yelp and Walmart.

### C.4 Scalability

Table 8 reports peak GPU memory across methods at the same sparsity budget.

Table 7: Comparison (Accuracy, peak memory) between exact and approximate effective resistance based sparsification.

| | DBLP-CA | Yelp | Walmart |
|---|---|---|---|
| **Spectral (exact Effective resistance)** | OOM | OOM | OOM |
| **Spectral (approximate Effective resistance)** | 88.79, 1.15 GB | 32.88, 16.02 GB | 94.76, 2.23 GB |

Table 8: Mean peak GPU memory usage (in GB) during training. **Bold**=best, ↓=%reduction w.r.t. Full.

| Algorithms | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | Citeseer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| Full | 0.44 | 0.41 | 0.26 | 0.08 | 0.31 | 0.18 | 0.10 | 1.29 |
| Random | **0.31 (29% ↓)** | **0.28 (31% ↓)** | **0.18 (31% ↓)** | **0.06 (24% ↓)** | **0.29 (6% ↓)** | **0.17 (5% ↓)** | **0.09 (11% ↓)** | **1.06 (18% ↓)** |
| Edgedeg | 0.36 (18% ↓) | 0.29 (29% ↓) | 0.23 (10% ↓) | 0.06 (24% ↓) | 0.32 (2% ↑) | 0.18 (3% ↓) | 0.10 (5% ↓) | 1.17 (10% ↓) |
| Spectral | 0.39 (12% ↓) | 0.29 (29% ↓) | 0.24 (6% ↓) | 0.06 (24% ↓) | 0.32 (4% ↑) | 0.18 (3% ↓) | 0.10 (5% ↓) | OOM |
| HSL | 1.15 (161% ↑) | 1.08 (161% ↑) | 0.70 (175% ↑) | 0.23 (178% ↑) | 0.82 (164% ↑) | 0.30 (65% ↑) | 0.21 (105% ↑) | 2.65 (104% ↑) |
| EHGNN-F | 0.41 (7% ↓) | 0.35 (14% ↓) | 0.24 (6% ↓) | 0.08 (9% ↓) | 0.34 (10% ↑) | 0.18 (4% ↓) | 0.09 (8% ↓) | 1.12 (14% ↓) |
| EHGNN-C | 0.46 (5% ↑) | 0.37 (12% ↓) | 0.28 (11% ↑) | 0.08 (9% ↓) | 0.36 (14% ↑) | 0.18 (4% ↓) | 0.09 (8% ↓) | 1.18 (9% ↓) |
| EHGNN-F(cond) | 0.47 (6% ↑) | 0.41 (1% ↓) | 0.25 (2% ↓) | 0.08 (1% ↑) | 0.37 (18% ↑) | 0.35 (88% ↑) | 0.16 (60% ↑) | 2.71 (109% ↑) |
| EHGNN-C(cond) | 0.47 (8% ↑) | 0.37 (10% ↓) | 0.27 (5% ↑) | 0.08 (8% ↓) | 0.35 (13% ↑) | 0.20 (9% ↑) | 0.10 (3% ↓) | 1.25 (3% ↓) |
| EHGNN-F (cond,LR) | 0.41 (7% ↓) | 0.35 (14% ↓) | 0.24 (6% ↓) | 0.08 (9% ↓) | 0.34 (10% ↑) | 0.18 (3% ↓) | 0.10 (7% ↓) | 1.13 (13% ↓) |
| EHGNN-C (cond,LR) | 0.45 (2% ↑) | 0.37 (11% ↓) | 0.25 (1% ↓) | 0.08 (9% ↓) | 0.36 (14% ↑) | 0.18 (4% ↓) | 0.09 (9% ↓) | 1.18 (9% ↓) |
| | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Trivago |
| Full | 0.10 | 0.08 | 0.19 | 0.43 | 0.25 | 2.81 | 20.02 | 6.41 |
| Random | **0.09 (11% ↓)** | **0.05 (31% ↓)** | **0.18 (8% ↓)** | **0.36 (18% ↓)** | **0.21 (13% ↓)** | **1.86 (34% ↓)** | **11.14 (44% ↓)** | **4.69 (27% ↓)** |
| Edgedeg | 0.10 (8% ↓) | 0.06 (21% ↓) | 0.19 (5% ↓) | 0.39 (10% ↓) | 0.22 (10% ↓) | 2.23 (20% ↓) | 15.92 (20% ↓) | 5.25 (18% ↓) |
| Spectral | 0.10 (9% ↓) | 0.06 (16% ↓) | 0.19 (4% ↓) | 0.39 (9% ↓) | 0.23 (7% ↓) | OOM | OOM | OOM |
| HSL | 0.33 (211% ↑) | 0.23 (211% ↑) | 0.37 (90% ↑) | 0.92 (113% ↑) | 0.52 (112% ↑) | 7.39 (163% ↑) | OOM | 13.35 (108% ↑) |
| EHGNN-F | 0.09 (9% ↓) | 0.07 (5% ↓) | 0.18 (8% ↓) | 0.38 (13% ↓) | 0.22 (9% ↓) | 2.44 (13% ↓) | 18.45 (8% ↓) | 6.15 (4% ↓) |
| EHGNN-C | 0.10 (9% ↓) | 0.07 (1% ↓) | 0.18 (5% ↓) | 0.39 (11% ↓) | 0.22 (12% ↓) | 2.49 (11% ↓) | 18.71 (7% ↓) | 6.67 (4% ↑) |
| EHGNN-F(cond) | 0.17 (61% ↑) | 0.07 (3% ↓) | 0.18 (6% ↓) | 0.51 (18% ↑) | 0.23 (8% ↓) | 2.53 (10% ↓) | OOM | 4.74 (26% ↓) |
| EHGNN-C(cond) | 0.10 (2% ↓) | 0.08 (5% ↑) | 0.19 (5% ↓) | 0.40 (7% ↓) | 0.22 (12% ↓) | 2.58 (8% ↓) | OOM | 7.07 (10% ↑) |
| EHGNN-F (cond,LR) | 0.10 (9% ↓) | 0.07 (4% ↓) | 0.18 (7% ↓) | 0.38 (11% ↓) | 0.23 (9% ↓) | 2.46 (12% ↓) | 18.64 (7% ↓) | 6.23 (3% ↓) |
| EHGNN-C (cond,LR) | 0.10 (8% ↓) | 0.07 (6% ↓) | 0.18 (5% ↓) | 0.38 (12% ↓) | 0.22 (12% ↓) | 2.42 (14% ↓) | 18.83 (6% ↓) | 6.68 (4% ↑) |

*On small-scale datasets (e.g., NTU2012, Cora-CA):* On small datasets, all methods exhibit comparable peak memory since the full incidence structure already fits comfortably in GPU memory. In this regime, sparsification yields only marginal gains, and differences across methods are small (typically within ∼0.02–0.05 GB). Notably, HSL incurs substantially higher memory overhead (e.g., 0.23 GB vs. 0.08 GB on NTU2012).

*On moderately large hypergraphs (e.g., DBLP-CA, Walmart):* As the incidence count increases, clear differences emerge. On DBLP-CA and Walmart, activation memory begins to dominate, and sparsification becomes critical. Random and degree-based pruning achieve the largest memory reductions. In contrast, EHGNN-F achieves consistent memory reductions compared to Full (e.g., 1.12 GB on DBLP-CA, 2.44 GB on Walmart) while preserving strong predictive performance. Finally, HSL shows substantially higher memory usage (e.g., 2.65 GB on DBLP-CA and 7.39 GB on Walmart), indicating poor scalability.

*On large-scale hypergraphs (Yelp, Trivago):* At large scale, memory behavior becomes more heterogeneous. On Yelp, Full already requires 20.02 GB, and most learnable sparsifiers provide only modest reductions (e.g., EHGNN-F: 18.45 GB, EHGNN-C: 18.71 GB). Feature-conditioned variants introduce additional overhead due to per-incidence feature aggregation and scoring, leading to increased memory or even OOM (EHGNN-F(cond), EHGNN-C(cond)). In contrast, simpler sparsifiers (Random) achieve the largest reductions (down to 11.14 GB), though at a significant accuracy cost. On Trivago, which is large but less extreme, EHGNN variants consistently reduce memory relative to Full (e.g., 6.15 GB vs. 6.41 GB), with feature-conditioned variants remaining competitive unless additional overhead dominates.

Overall, peak GPU memory is primarily governed by hypergraph scale, particularly the total incidence count $t$. Unsupervised sparsifiers, such as random and degree-based sparsifiers, have the lowest memory footprint, since they do not need to perform additional computation during training and, in fact, they benefit from reduced #incidences during message passing. However, as observed in Table 2, their performance is often superseded by EdgeMask-HGNN variants. EHGNN-F provides a favorable trade-off, achieving consistent memory reductions while maintaining strong performance. EHGNN-C offers similar benefits, but is more sensitive to hyperedge structure.

## D 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 $\mathcal{O}(m)$ and $\mathcal{O}(t)$, respectively, where $m = \#\text{edges}$ and $t = \sum_{e \in E} |e|$ is the number of node–hyperedge pairs.

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 |
| Edgedeg | 53764 | 72745 | 12802 | 86083 | 27651 | 1899526 | 737799 | 733190 |
| Spectral | 53764 | 72745 | 12802 | 86083 | 27651 | 1899526 | 737799 | OOM |
| HSL | 88718 | 91123 | 78448 | 92813 | 82153 | 557238 | 262203 | 261098 |
| EHGNN-F | 119215 | 134300 | 53422 | 96143 | 81023 | 1902979 | 742384 | 832751 |
| EHGNN-C | 53864 | 85056 | 13100 | 88095 | 37815 | 1900605 | 738871 | 755553 |
| EHGNN-F (cond) | 60229 | 79210 | 14275 | 92548 | 30916 | 2136583 | 829576 | 824455 |
| EHGNN-C (cond) | 57029 | 76010 | 13571 | 89348 | 29316 | 2018087 | 783720 | 778855 |
| EHGNN-F (cond, LR) | 54564 | 122389 | 14082 | 94531 | 68507 | 1918654 | 747819 | 828342 |
| EHGNN-C (cond, LR) | 54564 | 122389 | 14082 | 94531 | 68507 | 1918654 | 747819 | 828342 |
| | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Trivago |
| Full | 737799 | 2562 | 34818 | 258051 | 5122 | 11787 | 958473 | 498848 |
| Random | 737799 | 2562 | 34818 | 258051 | 5122 | 11787 | 958473 | 498848 |
| Edgedeg | 737799 | 2562 | 34818 | 258051 | 5122 | 11787 | 958473 | 498848 |
| Spectral | 737799 | 2562 | 34818 | 258051 | 5122 | OOM | OOM | OOM |
| HSL | 651963 | 75848 | 84038 | 140653 | 76498 | 77603 | OOM | 124858 |
| EHGNN-F | 742585 | 14405 | 40320 | 292680 | 21478 | 472417 | 5482067 | 1225709 |
| EHGNN-C | 739378 | 2903 | 37224 | 266014 | 7749 | 81693 | 1637775 | 732050 |
| EHGNN-F (cond) | 829576 | 2755 | 39043 | 290116 | 5635 | 12556 | OOM | 255457 |
| EHGNN-C (cond) | 783720 | 2691 | 36963 | 274116 | 5411 | 12204 | OOM | 508513 |
| EHGNN-F (cond, LR) | 749847 | 3934 | 44702 | 291903 | 15658 | 291455 | 3683129 | 1432856 |
| EHGNN-C (cond, LR) | 749847 | 3934 | 44702 | 291903 | 15658 | 291455 | 3683129 | 1432856 |

Since $m << t$ in most of the datasets, 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 ($\mathcal{O}(t + kF)$) 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 generally consumes less memory than the Full training (see in earlier Table 8).

## E EdgeMask-HGNN on Inductive Link Prediction Task

While the main experiments evaluate EdgeMask-HGNN on node classification task, we also investigate its adaptability on link prediction task. To that end, we adapt **Hyper-SAGNN** (Zhang et al., 2020), a self-attention-based architecture designed for this task.

**Link prediction task.** Let $\mathcal{V}$ denote the set of nodes and let $\mathcal{E} = \{e_1, \ldots, e_m\}$ be the set of observed hyperedges, where each hyperedge $e \subseteq \mathcal{V}$ represents a higher-order relation among multiple nodes. The goal of hyperedge prediction is to learn a scoring function

$$f_\theta : 2^{\mathcal{V}} \to [0, 1],$$

that estimates the likelihood that a candidate node set forms a valid hyperedge. In this paper, we focus on a simpler variant of link prediction where all edges have the same cardinality. Thus, in this paper, the scoring function we intend to learn is the following:

$$f_\theta : [\mathcal{V}]^k \to [0, 1], \tag{12}$$

where $[\mathcal{V}]^k$ indicates the set of all k-sets of $\mathcal{V}$. Formally, given a candidate hyperedge $e = \{v_1, \ldots, v_k\}$, the model predicts

$$\hat{y}_e = f_\theta(e), \tag{13}$$

where $\hat{y}_e$ is the probability that $e$ corresponds to a valid higher-order relation.

**Inductive setting.** In the inductive link prediction setting, the model is trained using a subset of observed hyperedges $\mathcal{E}_{train} \subset \mathcal{E}$, while validation and test hyperedges are drawn from $\mathcal{E}_{val}$ and $\mathcal{E}_{test}$. Importantly, node embeddings must generalize to previously unseen hyperedges composed of nodes from $\mathcal{V}$.

To train the model, we construct a binary classification dataset consisting of positive and negative hyperedges. Positive examples correspond to observed hyperedges $e \in \mathcal{E}_{train}$, while negative examples are generated by corrupting one position of a positive hyperedge with another node of the same type and rejecting corrupted tuples that already appear in the observed hyperedge set. Each example is assigned a label

$$y_e = \begin{cases} 1 & e \in \mathcal{E}_{train}, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

The model parameters are optimized using the binary cross-entropy loss

$$L = \frac{1}{|\mathcal{B}|} \sum_{e \in \mathcal{B}} \ell(\hat{y}_e, y_e), \tag{15}$$

where $\mathcal{B}$ denotes a minibatch of candidate hyperedges.

**Hyper-SAGNN backbone.** Given a candidate hyperedge (tuple) $e = (v_1, \ldots, v_k)$, each node $v_i$ is first mapped to an initial embedding $\boldsymbol{h}_{v_i}^{(0)}$. These embeddings are then processed through two parallel transformations.

First, a position-wise feed-forward network produces the *static embedding*

$$\boldsymbol{h}_{v_i}^{stat} = \tanh(W_s^\top \boldsymbol{h}_{v_i}^{(0)}), \tag{16}$$

which depends only on the individual node $v_i$ and is independent of the other nodes in the hyperedge.

Second, a multi-head self-attention layer is applied over the set $\{\boldsymbol{h}_{v_1}^{(0)}, \ldots, \boldsymbol{h}_{v_k}^{(0)}\}$ to produce the *dynamic embedding*

$$\boldsymbol{h}_{v_i}^{dyn} = \text{Attention}(\boldsymbol{h}_{v_i}^{(0)}, \{\boldsymbol{h}_{v_j}^{(0)}\}_{j=1}^{k}), \tag{17}$$

which captures interactions among all nodes within the hyperedge.

Thus, $\boldsymbol{h}_{v_i}^{stat}$ represents node-specific information that is invariant across hyperedges, while $\boldsymbol{h}_{v_i}^{dyn}$ encodes context-dependent information that varies with the composition of the hyperedge.

Finally, the resulting edge representation is computed as

$$\boldsymbol{z}_e = \frac{1}{|e|} \sum_{v_i \in e} (\boldsymbol{h}_{v_i}^{dyn} - \boldsymbol{h}_{v_i}^{stat})^2, \tag{18}$$

which is passed to a classification head to predict the validity of the hyperedge. Hyper-SAGNN treats each hyperedge independently; thus, the predictions depend only on the nodes contained in the target hyperedge.

**EdgeMask-HGNN adaptation** To incorporate EdgeMask-HGNN variants, we introduce a learnable sparsification module that assigns an importance score $\boldsymbol{m}_e \in [0, 1]$ to each hyperedge. Unlike the baseline Hyper-SAGNN model, these scores are used to construct contextualized node embeddings prior to the edge scoring stage.

For each hyperedge $e$, we compute an edge-level message

$$\boldsymbol{g}_e = \frac{1}{|e|} \sum_{u \in e} W h_u^{(0)}, \tag{19}$$

Table 10: Performance comparison between Hyper-SAGNN (Baseline) and various sparsification variants at sparsity=50%.

| **Model** | **WordNet** (user, relation, tail) #V = (40504, 18, 40551) | | | **Drug** (user, drug, reaction) #V = (12, 1076, 6398) | | | **MovieLens** (user, movie, tag) #V = (2113, 5908, 9079) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Acc** | **AUC** | **AUPR** | **Acc** | **AUC** | **AUPR** | **Acc** | **AUC** | **AUPR** |
| Hyper-SAGNN | 88.35 | 87.92 | 68.62 | 93.93 | 95.72 | 89.00 | 90.62 | 90.93 | 75.39 |
| Random | 93.24 | 95.68 | 87.05 | 96.49 | 98.64 | 95.72 | 95.11 | 97.43 | 92.12 |
| EdgeDeg | 93.47 | 96.05 | 87.96 | 96.57 | 98.62 | 95.69 | 95.05 | 97.24 | 91.84 |
| Spectral | 91.29 | 92.02 | 79.75 | 96.80 | 98.72 | 95.97 | 94.08 | 96.16 | 88.85 |
| EHGNN-F | 96.30 | 97.02 | 92.59 | 96.72 | 97.57 | 94.73 | 96.47 | 97.20 | 93.42 |
| EHGNN-C | 93.21 | 95.80 | 87.03 | 96.58 | 98.66 | 95.74 | 95.11 | 97.24 | 91.84 |
| EHGNN-F(cond) | 94.81 | 96.84 | 90.92 | 97.16 | 99.01 | 96.66 | 96.62 | 98.59 | 95.54 |
| EHGNN-C(cond) | 93.76 | 96.49 | 88.81 | 96.70 | 98.81 | 96.14 | 95.59 | 97.81 | 93.16 |
| EHGNN-F(cond,LR) | **98.78** | **99.64** | **99.05** | **98.33** | **99.70** | **98.77** | **98.32** | **99.66** | **98.46** |
| EHGNN-C(cond,LR) | 91.84 | 93.34 | 82.07 | 95.87 | 97.74 | 93.89 | 93.89 | 95.61 | 88.39 |

and update node embeddings using masked hypergraph aggregation

$$\boldsymbol{h}_v^{(1)} = \boldsymbol{h}_v^{(0)} + \sum_{e: v \in e} \boldsymbol{m}_e \boldsymbol{g}_e. \tag{20}$$

For a node $v$, its contextualized embeddings $\boldsymbol{h}_v^{(1)}$ are then used as inputs to Hyper-SAGNN:

$$\hat{y}_e = f_\theta(\{\boldsymbol{h}_v^{(1)} : v \in e\}), \tag{21}$$

where $f_\theta$ denotes the Hyper-SAGNN encoder.

This modification introduces a lightweight form of structural reasoning into Hyper-SAGNN: predictions for an edge now depend on other hyperedges sharing nodes through the masked aggregation step. Next, we discuss the computation of mask $\boldsymbol{m}_e$.

**Mask computation and Training objective.** The mask is obtained from the underlying EdgeMask-HGNN sparsifier. We first discuss the case of coarse-grained sparsifier. Let the model parameters of the sparsifier be $\Phi$. Then the global edge mask $\boldsymbol{m}_e$ is obtained as

$$\boldsymbol{m}_e = \texttt{EdgeMask-HGNN}_\Phi(\mathcal{B}, \boldsymbol{X} = \boldsymbol{X}_\mathcal{B}), \tag{22}$$

where $\boldsymbol{X}_\mathcal{B} = (\boldsymbol{h}_v^{(0)})_{v \in \mathcal{B}}$ are the node features of the nodes in batch $\mathcal{B}$. The model is trained end-to-end using the loss $L$ defined in Equation (15), where the gradients propagate through both the Hyper-SAGNN encoder and the sparsification module.

When the sparsifier is fine-grained, we obtain $\boldsymbol{m}_e$ from aggregating learned incidence masks $\boldsymbol{m}_{v,e}$:

$$\boldsymbol{m}_{v,e} = \texttt{EdgeMask-HGNN}_\Phi(\mathcal{B}, \boldsymbol{X} = \boldsymbol{X}_\mathcal{B}), \quad \boldsymbol{m}_e = \frac{1}{|e|} \sum_{v \in e} \boldsymbol{m}_{v,e} \tag{23}$$

**Experimental results and discussion.** Table 10 summarizes the performance of the baseline Hyper-SAGNN model and the EHGNN-enhanced variant across three datasets at 50% sparsity.

The results demonstrate that EdgeMask-HGNN can significantly enhance Hyper-SAGNN in inductive link prediction tasks. For instance, on WordNet, EHGNN-F(cond,LR) increases AUC from 87.92 to 99.64, while AUPR improves from 68.62 to 99.05. Similarly, on the Drug dataset, AUC increases from 95.72 to 99.70.

Table 11: Statistical test by conducting paired (related) samples t-tests: Full vs. EHGNN-C and Full vs. EHGNN-F. Here $* = p < 0.05, ** = p < 0.01$ and $- = p > 0.05$ indicates no statistical significance. Edgemask variants are run with 50% sparsification.

| | | EHGNN-C | | | | EHGNN-F | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | **Full** | **Mean Acc.** | **Δ** | **p-value** | **Sig.** | **Mean Acc.** | **Δ** | **p-value** | **Sig.** | **Better Method** |
| 20news | 81.04 | 79.08 | -1.96 | 2.58e-05 | ** | 77.18 | -3.86 | 3.42e-07 | ** | Full |
| Actor | 64.54 | 76.72 | **+12.19** | 2.16e-09 | ** | 78.01 | **+13.47** | 1.33e-09 | ** | EHGNN-F |
| Citeseer | 68.19 | 68.60 | **+0.41** | 1.46e-01 | - | 66.76 | -1.43 | 1.27e-02 | * | EHGNN-C |
| Cora | 78.23 | 75.60 | -2.63 | 3.38e-04 | ** | 75.72 | -2.51 | 8.48e-05 | ** | Full |
| Cora-CA | 81.48 | 77.02 | -4.46 | 1.92e-05 | ** | 77.49 | -3.99 | 4.43e-05 | ** | Full |
| DBLP-CA | 90.77 | 85.28 | -5.49 | 5.14e-08 | ** | 87.01 | -3.75 | 7.30e-07 | ** | Full |
| House | 73.87 | 75.85 | **+1.98** | 7.03e-04 | ** | 86.13 | **+12.26** | 5.15e-07 | ** | EHGNN-F |
| ModelNet40 | 94.72 | 95.73 | **+1.01** | 1.68e-05 | ** | 96.32 | **+1.60** | 1.74e-05 | ** | EHGNN-F |
| Mushroom | 98.73 | 93.59 | -5.14 | 3.20e-05 | ** | 97.24 | -1.49 | 4.57e-05 | ** | Full |
| NTU2012 | 88.03 | 87.20 | -0.83 | 8.15e-03 | ** | 87.48 | -0.56 | 7.29e-02 | - | Full |
| Pokec | 58.36 | 59.15 | **+0.79** | 4.37e-07 | ** | 58.53 | **+0.17** | 1.40e-03 | ** | EHGNN-C |
| PubMed | 86.50 | 86.56 | **+0.06** | 2.31e-01 | - | 86.75 | **+0.25** | 7.02e-03 | ** | EHGNN-F |
| Twitch | 51.22 | 51.44 | **+0.22** | 9.97e-03 | ** | 50.84 | -0.38 | 1.10e-05 | ** | EHGNN-C |
| Trivago | 61.39 | 37.78 | -23.61 | 2.16e-05 | ** | 45.56 | -15.83 | 5.33e-05 | ** | Full |
| Walmart | 95.36 | 95.89 | **+0.53** | 7.30e-05 | ** | 96.90 | **+1.55** | 9.77e-09 | ** | EHGNN-F |
| Yelp | 33.60 | 31.98 | -1.62 | 1.14e-03 | ** | 33.36 | -0.24 | 2.89e-01 | - | Full |

This indicates that the learned sparsifier effectively removes noisy hyperedges and improves representation quality. Furthermore, compared to unsupervised sparsifiers, fine-grained sparsifier variants generally yield superior performance.

**Effect of sparsification on link prediction.** The key benefit of the proposed adaptation is that it introduces structural context into a model that originally treats hyperedges independently. As a result, predictions for a candidate hyperedge can incorporate information from other related hyperedges through the masked aggregation step. This capability is particularly beneficial in datasets with noisy relational structures. Overall, these experiments demonstrate that EdgeMask-HGNN can extend the capabilities of Hyper-SAGNN by enabling task-aware hypergraph sparsification for inductive link prediction.

## F Statistical Significance Tests

In order to evaluate whether the proposed EdgeMask variants are meaningfully superior to full hypergraph training baseline, we conduct paired (related) samples t-tests comparing Full vs. EHGNN-C and Full vs. EHGNN-F across all datasets in Table 11. Here, $* = p < 0.05$, $** = p < 0.01$, and $- = p > 0.05$ indicate statistical significance levels. All Edgemask-HGNN variants operate under a 50% sparsification budget.

First, sparsification does not universally degrade performance. On many datasets, particularly Actor, House, Walmart, Pokec, and Twitch, we observe that both EHGNN-C and EHGNN-F significantly outperform the full hypergraph model, despite using only 50% of the hyperedges. This indicates that dense hypergraphs often contain redundant or noisy high-order relations, and that principled sparsification can improve generalization by acting as a structural regularizer.

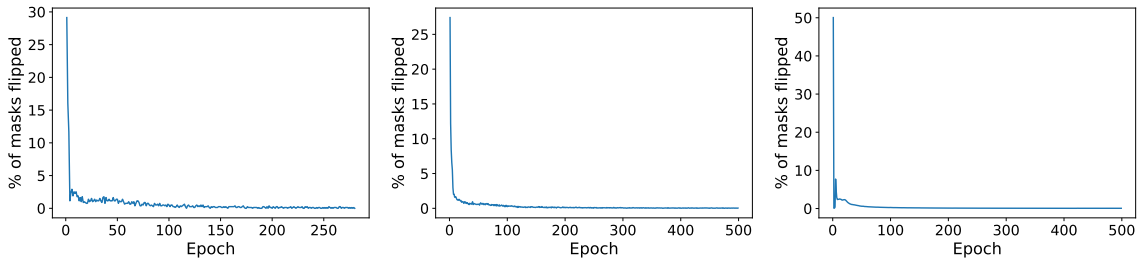
Second, datasets with strong homophily, such as Cora, Cora-CA, CiteSeer, and DBLP-CA, show significant performance drops after sparsification. Here, full connectivity provides a useful neighborhood signal, and removing hyperedges disrupts information flow.

## G Ablation Studies

**Choice of sampler.** Table 12 compares the default weighted sampling without replacement strategy used in EHGNN-F against two alternatives: deterministic top-$k$ based on the scores $\boldsymbol{s}$ and independent Bernoulli sampling with $\ell_2$ regularization. The three variants are generally competitive, but EHGNN-F achieves the most consistent performance across datasets, supporting our choice of budget-constrained stochastic sampling.

Table 12: Comparison (accuracy ± std) of EHGNN-F, deterministic top-k EHGNN-F, and EHGNN-F w/ Bernoulli sampling + L2 reg. across datasets.

| **Model** | 20news | ModelNet40 | Mushroom | NTU2012 | Actor | Citeseer | Cora-CA | DBLP-CA |
|---|---|---|---|---|---|---|---|---|
| EHGNN-F w/ deterministic top-k | 76.64 ± 0.05 | 96.23 ± 0.03 | **97.34 ± 0.33** | 86.08 ± 0.34 | 77.69 ± 0.05 | **68.91 ± 0.22** | 75.81 ± 0.51 | 86.03 ± 0.12 |
| EHGNN-F w/ Bern. sampling + $\ell_2$ | 77.04 ± 0.09 | 96.23 ± 0.11 | **97.34 ± 0.34** | 87.00 ± 0.67 | **77.99 ± 0.12** | 67.03 ± 0.53 | 77.02 ± 0.34 | 86.88 ± 0.10 |
| EHGNN-F | **77.18 ± 0.10** | **96.32 ± 0.11** | 97.24 ± 0.21 | **87.44 ± 0.36** | 77.90 ± 0.05 | 66.76 ± 0.45 | **77.61 ± 0.34** | **87.00 ± 0.10** |
| | Cora | House | Pokec | PubMed | Twitch | Walmart | Yelp | Trivago |
| EHGNN-F w/ deterministic top-k | **76.37 ± 0.18** | **86.01 ± 0.26** | **58.61 ± 0.04** | 86.66 ± 0.06 | 50.73 ± 0.03 | 96.92 ± 0.01 | 33.13 ± 0.16 | 45.22 ± 1.65 |
| EHGNN-F w/ Bern. sampling + $\ell_2$ | 75.66 ± 0.69 | 84.40 ± 1.09 | 58.52 ± 0.09 | **86.91 ± 0.13** | **50.94 ± 0.09** | 96.74 ± 0.08 | **33.33 ± 0.24** | 44.18 ± 1.90 |
| EHGNN-F | 75.72 ± 0.22 | 85.82 ± 0.96 | 58.56 ± 0.03 | 86.76 ± 0.05 | 50.82 ± 0.05 | **96.92 ± 0.02** | 33.29 ± 0.22 | **46.75 ± 0.71** |




(a) Cora (b) Actor (c) Yelp

Figure 8: % of Learned mask flipped per epoch during training of EHGNN-C with 50% pruning rate.

## H Empirical Convergence and Stability of Learned Masks

A central concern in learnable sparsification is whether the discrete masks $\hat{m}_{v,e}$ converge to a stable selection of hyperedges or incidences. Since the forward pass uses a hard selection (top-$k$) but the backpropagation uses a straight-through estimator (STE), instability of these surrogate gradients could lead to erratic mask behavior.

In order to verify that the learned masks truly converge during training, we track the mask flips, meaning, hyperedges whose hard mask value changes between consecutive epochs for EHGNN-C:

$$\% \text{ Flip}^{(t)} = \frac{100}{|E|} \sum_{e \in E} \mathbf{1}\big[\hat{m}_e^{(t)} \neq \hat{m}_e^{(t-1)}\big],$$

where $\hat{m}_e^{(t)} \in \{0, 1\}$ denotes whether hyperedge $e$ is kept by the sparsifier at epoch $t$. If the learned sparsifier is unstable, we expect frequent flips throughout training (a higher %Flip).

Figures 8a, 8b, and 8c show the %Flip for three datasets (Cora, Actor, and Yelp), where we observe that %Flip rapidly decreases after the initial epochs and stabilizes to very low values ($< 1\%$). This demonstrates that the learned discrete mask converges to a stable sparsification pattern, with only small local perturbations as training progresses.