Adaptive Hypergraph Pruning with Learned Threshold Control and Attention-Based Negative Mining

Anonymous authors
Paper under double-blind review

Abstract

Hypergraph neural networks (HGNNs) capture multi-way interactions but their quadratic cost limits scalability. Existing pruning methods rely on hand-crafted schedules and fixed per-level thresholds that require per-dataset tuning and treat structural pruning, contrastive learning, and loss balancing as independent objectives. We propose **TriPrune-HGNN**, an adaptive hypergraph pruning framework that replaces these hand-crafted components with four small learnable controllers: a compressibility predictor that estimates the achievable retention ratio from graph-level statistics before training; hierarchical soft gates with learned thresholds operating jointly over components, hyperedges, and nodes; attention-based contrastive mining that identifies false and hard negatives induced by topology change; and meta-learned loss balancing. Pruning is realised as a density curriculum so the model trains at its final inference sparsity. Our principal claim is a favourable accuracy–efficiency *operating point*, not per-metric dominance. Against the strongest efficient baselines (AdaGLT, SHARP-Distill, Shaver) under matched tuning budgets on five heterogeneous hypergraph benchmarks, MAE improvements are small but consistent in sign: 0.002–0.006 at fixed cross-dataset hyperparameters, narrowing to 0.002–0.004 when the baselines are re-tuned per-dataset. The larger headline numbers (75.5% inference-time reduction, 69.7% memory reduction) are computed against the unpruned HGNN baseline (Table 2, Avg rows) and are reported for context, not as the primary contribution. We additionally test out-of-domain generalisation on NTU2012 3D shape recognition, where the accuracy advantage shrinks to 0.6% while efficiency transfers cleanly — a useful diagnostic separation between the framework's transferable (efficiency) and non-transferable (accuracy) components. Three short design-motivation propositions justify the choices of gate-smoothing ϵ, finite-difference perturbation δ, and the three-level pruning decomposition; they are local justifications under stated regularity conditions, not end-to-end guarantees. Code, weights, the synthetic generator, and seed-level numbers are released for reproducibility.

1 Introduction

Many real-world systems exhibit higher-order interactions that pairwise graphs cannot capture. Heterogeneous hypergraphs (HHGs) address this by allowing hyperedges to connect multiple nodes simultaneously across distinct behavioural *contexts* (Kim et al., 2024; Zhou et al., 2020): in a recommendation system, one context may encode purchase behaviour while another encodes social connections, each with its own incidence matrix $\mathbf{H}_k$ and adjacency $\mathbf{A}_k$. Hypergraph neural networks (HGNNs) exploit this structure with attention and contrastive learning to produce richer node representations than standard GNNs (Fountoulakis et al., 2023; Antelmi et al., 2023; Qian et al., 2024). The cost is scalability: processing K contexts simultaneously incurs complexity $\mathcal{O}(Knm\bar{d}_e d)$, which for $m = \mathcal{O}(n)$ and $\bar{d}_e = \mathcal{O}(\sqrt{n})$ grows as $\mathcal{O}(n^{2.5})$. On graphs with $n, m > 10^5$ and $K \geq 5$ this becomes prohibitive, and recent models such as HEAL consume up to 15.9 GB on moderate benchmarks (Ju et al., 2024). Pruning is the natural response, but a simple thought experiment shows why existing approaches break down on HHGs. Consider a movie-recommendation hypergraph with three contexts (*viewing history*, *genre preferences*, *social connections*). Applying HSL (Cai et al., 2022) to remove 30% of low-weight edges *independently* per context on MovieLens-1M orphans up to 58% of cross-

context node connections, static threshold-based pruning (Cai et al., 2022; Lin et al., 2024) at 50% edge retention breaks 35–48% of the cross-context co-occurrences — node pairs that share at least two contexts before pruning and become disconnected in at least one after. Three properties of HHGs explain why existing single-level pruning strategies cannot avoid these failures.

Challenge 1 (Cross-granularity cascading failures). A single hyperedge connects multiple nodes simultaneously, so structural choices at different granularities interact: removing a component orphans its hyperedges, pruning hyperedges isolates nodes, and dropping nodes breaks hyperedge connectivity. Single-level methods cannot anticipate these dependencies (Cai et al., 2022).

Challenge 2 (Cross-context dependency preservation). Nodes interacting across multiple contexts represent strong semantic associations and should survive pruning of any individual context. Context-agnostic strategies (Liang et al., 2021) cannot identify which cross-context relationships are critical, causing substantial information loss.

Challenge 3 (Contrastive learning under structural modifications). Pruning introduces two problematic pair types. *False negatives* arise when semantically similar nodes become disconnected and are incorrectly pushed apart. *Hard negatives* arise when dissimilar nodes acquire spuriously similar embeddings: removing a shared hyperedge between two semantically distinct nodes does not create a new path, but their representations may still converge in subsequent message-passing rounds through retained common neighbours. This is an embedding-space artefact, not a topological one, and standard contrastive objectives have no mechanism to correct it (Wu et al., 2024).

The common root cause is that existing methods treat structure, behaviour, and representation as independent objectives, and rely on hand-crafted heuristics (exponential decay schedules, fixed similarity thresholds) that cannot adapt to diverse graph statistics or evolving training dynamics. We propose **TriPrune-HGNN**, a unified framework built around a single design principle: *pruning should be a density curriculum that aligns the graph the model trains on with the graph it deploys on.* The graph evolves continuously from its dense initial state to a learned sparse configuration during training; retention ratios stabilise by convergence, so the model trains on its final inference sparsity for roughly the last 25% of training, eliminating the train–inference distribution shift that one-shot pruning introduces (direct evidence in Section 4.5). Four learnable mechanisms cooperate inside this curriculum.

(1) Hypergraph Compressibility Predictor (HCP). A small predictor maps graph-level structural statistics (density, spectral gap, degree-distribution skewness, and similar quantities) to an estimate $r^\star \in (0, 1)$ of the maximum achievable compression ratio *before training begins.* Pre-trained once on a synthetic corpus and applied unchanged to real benchmarks, HCP supplies a domain-invariant prior that initialises the threshold controllers and regularises their trajectory. Because HCP conditions only on structural statistics, the same frozen predictor transfers across recommendation, citation, and review domains and accounts for roughly half of the $\sim$61% cross-dataset gain reported in Section 5.2.

(2) Neural Adaptive Hierarchical Pruning. Differentiable soft gates with learned thresholds operate jointly over components, hyperedges, and nodes; thresholds are predicted by a small MLP conditioned on graph state, training dynamics, and the HCP prediction. The three-level hierarchy is strictly preferred to any single-level scheme at the same overall retention under a diminishing-returns assumption (Proposition 3), and the gap grows from $\sim$2% MAE at mild retention to $\sim$6% under aggressive compression.

(3) Attention-based contrastive mining. Rather than fixed similarity thresholds, learned attention heads identify false and hard negatives by jointly integrating embedding similarity, structural change, and hyperedge context. Both mechanisms are differentiable and trained end-to-end with the rest of the model.

(4) Meta-learned multi-task optimisation. Gradient-based bi-level optimisation discovers dataset-specific loss weights without grid search. The finite-difference meta-gradient is computed at $\mathcal{O}(|\mathbf{\Theta}|)$ rather than $\mathcal{O}(|\mathbf{\Theta}|^2)$ cost and has bounded error (Proposition 2).

We are explicit about what TriPrune-HGNN does and does not claim. The principal contribution is a *favourable accuracy–efficiency operating point* reached by replacing hand-crafted schedules with small learnable controllers, not per-metric dominance. Against the strongest efficient baselines under matched tuning

budgets, MAE gaps are 0.002–0.006 in absolute terms; under per-dataset retuning of those baselines, gaps narrow further to 0.002–0.004 but remain consistent in sign across all five benchmarks. The headline reductions (-75.5% time, -69.7% memory) are computed against the *unpruned* HGNN baseline (Table 2, Avg rows) and are reported for context only. Our contributions are therefore:

- A framework that couples compressibility prediction, adaptive hierarchical pruning, attention-based contrastive mining, and meta-learning under a density curriculum. The composite is novel, even though each individual component draws on existing techniques.
- The Hypergraph Compressibility Predictor (HCP): a small structural-statistics MLP pre-trained on synthetic hypergraphs and applied frozen at deployment. To our knowledge, this is the first learned compressibility predictor for heterogeneous hypergraphs; the cross-dataset transfer experiments in Section 5.2 are the most direct evidence for its utility.
- Three design-motivation propositions that tie the abstract constants of the framework (ϵ, δ, the three-level decomposition) to specific local assumptions, together with an informal transfer-bound for HCP. These are not end-to-end guarantees for the non-convex training procedure — they justify design choices under stated regularity conditions and bound the cost of those conditions failing.

2 Related Works

Hypergraphs provide a principled framework for modeling complex relationships beyond pairwise interactions, which are prevalent in numerous real-world systems such as recommendation, bioinformatics, and multi-relational networks (Gao et al., 2022; Ju et al., 2024; Feng et al., 2019). Unlike standard graphs, where edges connect pairs of nodes, hypergraphs generalise this notion by allowing hyperedges to simultaneously connect multiple nodes, thereby capturing higher-order dependencies and richer behavioural semantics (Zhou et al., 2020; Kim et al., 2024). However, this expressive power comes at the cost of significantly increased computational complexity, particularly when scaling to large datasets. Hypergraph neural networks (HGNNs) often exhibit quadratic time and space complexity, resulting in slow inference and high memory consumption (Zhang et al., 2022; Liang et al., 2021). These limitations have motivated the development of optimisation techniques aimed at improving efficiency without sacrificing model performance.

Pruning and Compression. Pruning methods (Lee & Song, 2023; Chen et al., 2024; He & Xiao, 2023) have been widely adopted to reduce the computational overhead of HGNNs by selectively removing redundant or less informative components (nodes, edges, or entire substructures), effectively compressing the hypergraph. Structure-aware pruning (Zheng et al., 2022a; Jiang et al., 2023) attempts to identify and retain critical subgraphs that contribute most to downstream tasks. However, conventional pruning strategies typically operate at a single granularity (e.g., node or edge level) and often disregard the hierarchical nature of hypergraph structure, leading to suboptimal message passing and representational capacity (Cai et al., 2022). Moreover, most existing approaches adopt a static pruning paradigm, neglecting the dynamic evolution of node relationships during training and across behavioural contexts (Liang et al., 2021).

Scalable and Adaptive HGNN Architectures. Beyond pruning and distillation, a parallel line of work addresses HGNN scalability through alternative architectural and algorithmic choices. Unified-framework approaches such as AllSet (Chien et al., 2022) and UniGNN (Huang & Yang, 2021) factorise hyperedge aggregation into multiset functions and unified message-passing primitives, reducing per-layer cost while preserving expressive power. Sampling-based methods adapt subgraph sampling techniques from standard GNNs (Zeng et al., 2019) to the hypergraph setting, training on stochastic subgraphs rather than the full structure at each iteration. Hyperedge-dependent embedding frameworks (Dong et al., 2020) further learn per-hyperedge representations that can be aggregated more efficiently than uniform message-passing schemes. These directions are complementary to ours: sampling reduces *what is processed per training step*, whereas TriPrune-HGNN reduces *what is retained in the deployed graph*. We fix the sampling regime across all baselines in Section 5 to isolate the effect of adaptive pruning; combining adaptive pruning with adaptive sampling is a natural extension we leave for future work.

Knowledge Distillation and Quantisation. Complementary to pruning, knowledge distillation methods (Forouzandeh et al., 2025a; Feng et al., 2024; Yu et al., 2024; Forouzandeh et al., 2025b) transfer knowledge from a larger, complex teacher model to a simpler student, preserving performance while reducing inference costs. Quantisation techniques (Hubara et al., 2018) further compress model size by lowering numerical precision, reducing memory usage with minimal loss in accuracy. Recent surveys highlight the effectiveness of these strategies in making hypergraph-based models viable for real-time and resource-constrained scenarios (Cheng et al., 2024; Gholami et al., 2022; Zhou et al., 2018).

Contrastive Learning in Graphs and Hypergraphs. Contrastive learning (CL) (Wang et al., 2023b; Zheng et al., 2022b) has emerged as a leading approach for learning robust graph and hypergraph representations by maximizing agreement between semantically similar node pairs (positives) and contrasting dissimilar pairs (negatives). In hypergraph settings, early studies focused on designing effective view augmentations. Wei *et al.* (Tianxin et al., 2022) propose fabricated and generative augmentations tailored for hypergraphs, demonstrating improved robustness under view perturbations. While effective, such augmentation-centric methods assume relatively stable underlying structures and do not explicitly account for topology changes induced by structural pruning.

Recent advances have explored richer relational signals for hypergraph contrastive objectives. Lee and Shin (Lee & Shin, 2023) introduce a tri-directional contrastive framework that jointly contrasts node–node, node–hyperedge, and hyperedge–hyperedge representations, improving semantic consistency across hypergraph entities. Roh *et al.* (Roh et al., 2024) further enhance hypergraph contrastive learning by exploiting shared group structures, encouraging nodes belonging to common hyperedges to form cohesive representation clusters. These methods highlight the importance of higher-order relational alignment, but rely on fixed hypergraph structures and do not address efficiency or dynamic structural adaptation.

Attention-driven contrastive mechanisms have also been investigated. Xie *et al.* (Xie et al., 2025) propose a semi-supervised hypergraph contrastive framework for hyperedge prediction using an enhanced attention aggregator to identify informative relations. Similarly, Gu and Wang (Gu & Wang, 2025) integrate hypergraph-enhanced contrastive learning with hyper-Laplacian regularization for multi-view clustering, emphasizing cross-view consistency and global structural smoothness. Although effective for representation alignment and clustering, these approaches primarily target static learning objectives and do not consider pruning-induced distributional shifts.

Despite these advances, hypergraph contrastive learning still faces critical challenges: (i) *false negatives*, where pruning or view construction disconnects semantically similar nodes; and (ii) *hard negatives*, where structural alterations cause dissimilar nodes to appear similar (Sun et al., 2023; Wang et al., 2023a; Song et al., 2024). Existing solutions such as adaptive weighting (Xu et al., 2024; Chen et al., 2021), debiased sampling (Zhou et al., 2022; Chuang et al., 2020), and dynamic clustering (Huynh et al., 2022) are largely designed for pairwise graphs and static hypergraph settings. In contrast, our approach explicitly couples adaptive multi-granular pruning with attention-based contrastive mining, enabling robustness to dynamic structural changes while maintaining computational efficiency—an aspect underexplored in prior hypergraph contrastive learning studies. More recent work has begun to integrate *adaptive* negative selection directly into hypergraph contrastive objectives (Song et al., 2024; Roh et al., 2024), but these methods still assume fixed hypergraph topology and do not address the bidirectional coupling between pruning decisions and contrastive distortions that TriPrune-HGNN targets.

3 Preliminaries

For clarity, Table 1 summaries the main mathematical notation used throughout this paper.

Hypergraph Neural Networks. Traditional Graph Neural Networks (GNNs) model pairwise relationships, limiting their ability to capture high-order interactions (Feng et al., 2019). Hypergraph Neural Networks (HGNNs) address this by introducing hyperedges, which connect multiple vertices simultaneously. A hypergraph is defined as $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with an incidence matrix $\mathbf{H} \in \mathbb{R}^{n \times m}$ indicating vertex–hyperedge

Table 1: Mathematical notation used throughout the paper.

Symbol	Description	Symbol	Description
Graph structure			
$\mathcal{H}$	Hypergraph $(\mathcal{V}, \mathcal{E})$	$\mathbf{H} \in \mathbb{R}^{n \times m}$	Incidence matrix
$\mathcal{V}_m$	**Primary** nodes	$\mathcal{V}_s^k$	**Secondary** nodes (comp. k)
K	# components	$\mathbf{A}_k$	Adjacency for comp. k
$\mathbf{X} \in \mathbb{R}^{n \times d}$	Node features	d	Feature dimension
$\mathcal{V}_{\text{lab}} \subset \mathcal{V}_m$	Labelled primary nodes	C	Number of classes
$\bar{d}_e$	Avg. hyperedge degree	$\lvert \cdot \rvert_0$	Number of non-zeros
Pruning			
$\widetilde{\mathcal{H}}, \widetilde{\mathbf{H}}, \widetilde{\mathbf{A}}$	Pruned graph / matrices	ℓ	Level: comp / edge / node
$\pi^{(\ell)}$	Importance score	$\theta_t^{(\ell)}$	Learned threshold
$g^{(\ell)}$	Soft gate (training)	$\bar{g}^{(\ell)}$	Binarised gate (inference)
β_k	Learned balance weight	ϵ	Gate smoothing ($= 0.01$)
$r_{\text{comp}}, r_{\text{edge}}, r_{\text{node}}$	Retention ratios per level	r_{overall}	Overall retention $= r_{\text{c}} r_{\text{e}} r_{\text{n}}^2$
σ_{hard}	Hard sigmoid	STE	Straight-through estimator
Compressibility predictor (HCP)			
$\boldsymbol{\phi}(\mathcal{H}) \in \mathbb{R}^{10}$	Graph structural features	$r^\star \in (0, 1)$	Predicted compressibility
MLP_{HCP}	Compressibility predictor	$\mathcal{L}_{\text{HCP}}$	Compressibility regulariser
$\beta_{\text{HCP}}^{(t)}$	HCP regulariser weight at t	T_{HCP}	HCP decay horizon ($=T/2$)
r^{emp}	Empirical compressibility	Δ_{acc}	Accuracy-tolerance ($= 1\%$)
Neural modules			
$\text{MLP}_\theta^{(\ell)}$	Threshold controller	MLP_β	Balance predictor
MLP_τ	Temperature predictor	$\mathbf{s}_t^{(\ell)} \in \mathbb{R}^{\mathbf{7}}$	Graph state vector
Attn_{fn}	False-negative attention	Attn_{hn}	Hard-negative attention
d_{hidden}	MLP hidden dim ($= 32$)	d_{state}	State dim ($= 7$)
Contrastive learning			
$\mathbf{z}_i \in \mathbb{R}^d$	Node embedding	$\mathbf{h}_{e_j^k} \in \mathbb{R}^d$	Hyperedge embedding
$\mathbf{e}_i^k$	Hyperedge embedding of node i	$\mathbf{c}_k \in \mathbb{R}^d$	Component embedding
$s(\cdot, \cdot)$	Cosine similarity	τ_0, τ_i	Global / node temperature
$\mathcal{F}_k^{\text{pr}}$	False-negative pairs	$\mathcal{H}_i^k$	Hard-negative set
$\alpha_{ij}^{(\text{fn})}, \alpha_{ij}^{(\text{hn})}$	Learned attention scores	$\pi_k^{(\text{retain})}$	Component retention weight
Meta-learning and loss			
$\boldsymbol{\lambda} \in \Delta^3$	Loss weights (simplex)	$\boldsymbol{\Theta}$	Main model parameters
$\boldsymbol{\Psi}$	Neural-module parameters	Proj_Δ	Simplex projection
$\eta_{\text{inner}}, \eta_{\text{meta}}$	Inner / outer learning rates	Δ_{meta}	Meta-update period ($= 5$ ep.)
δ	Finite-difference perturbation	α	Topology-change sensitivity
$\mathcal{L}_{\text{cls}}$	Classification loss	$\mathcal{L}_{\text{cl}}^{\text{pr}}$	Contrastive loss
$\mathcal{L}_{\text{fn}}^{\text{pr}}$	False-negative loss	$\mathcal{L}_{\text{hard}}$	Hard-negative loss

connections, where $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. The vertex and hyperedge degree matrices are

$$(\mathbf{D}_v)_{ii} = \sum_{j=1}^{m} \mathbf{H}_{ij}, \qquad (\mathbf{D}_e)_{jj} = \sum_{i=1}^{n} \mathbf{H}_{ij}. \tag{1}$$

Feature propagation in HGNNs follows a two-step process: vertex-to-hyperedge aggregation followed by hyperedge-to-vertex propagation,

$$\mathbf{X}^{(l+1)} \;=\; \sigma\Big(\mathbf{D}_v^{-\frac{1}{2}}\,\mathbf{H}\,\mathbf{W}\,\mathbf{D}_e^{-1}\,\mathbf{H}^\top\,\mathbf{D}_v^{-\frac{1}{2}}\,\mathbf{X}^{(l)}\,\boldsymbol{\Theta}^{(l)}\Big)\,, \tag{2}$$

where $\mathbf{\Theta}^{(l)}$ is a learnable weight matrix, $\mathbf{W} = \mathrm{diag}(w_1, \ldots, w_m)$ is a diagonal hyperedge weight matrix, and $\sigma(\cdot)$ is a non-linear activation function. Equation equation 2 is the operation that TriPrune-HGNN reduces by pruning the incidence matrix $\mathbf{H}$.

Heterogeneous Hypergraphs. Real-world systems often exhibit multiple types of relationships that cannot be captured by a single hypergraph structure. A *heterogeneous hypergraph* extends the standard hypergraph by partitioning nodes into distinct roles and encoding multiple behavioural contexts through separate components (Kim et al., 2024):

$$\mathcal{H}_{\text{het}} = \big(\mathcal{V}_m,\ \{\mathcal{V}_s^k\}_{k=1}^K,\ \{\mathcal{E}^k\}_{k=1}^K\big), \tag{3}$$

where $\mathcal{V}_m$ denotes *primary nodes* (central entities, e.g. users in recommendation systems), $\mathcal{V}_s^k$ denotes *secondary nodes* for component k (auxiliary entities, e.g. products, tags), and $\mathcal{E}^k$ represents hyperedges in behavioural context k (e.g. purchase behaviour, social connections). Each component k has its own incidence matrix $\mathbf{H}_k \in \mathbb{R}^{|\mathcal{V}_m| \times |\mathcal{E}^k|}$ and normalised adjacency matrix $\mathbf{A}_k$. Processing K components with attention mechanisms incurs complexity $\mathcal{O}(Knm\bar{d}_e d)$, which is quadratic in the graph scale: for graphs with $m = \mathcal{O}(n)$ and $\bar{d}_e = \mathcal{O}(\sqrt{n})$, the cost grows as $\mathcal{O}(n^{2.5})$. For large-scale graphs ($n, m > 10^5$) with multiple contexts ($K \geq 5$), this becomes prohibitive for real-time applications.

Contrastive Learning on Hypergraphs. Modern HGNNs leverage contrastive learning to learn discriminative representations by maximising agreement between semantically similar nodes (positives) while contrasting dissimilar nodes (negatives) (Qian et al., 2024). Given node embeddings $\mathbf{z}_i \in \mathbb{R}^d$ and hyperedge embeddings $\mathbf{h}_{e_j^k} \in \mathbb{R}^d$, the contrastive loss for component k is

$$\mathcal{L}_{\text{cl}}^k \ = \ -\sum_{i \in \mathcal{V}_m} \log \frac{\exp\big(s(\mathbf{z}_i, \mathbf{h}_{e_i^k})/\tau\big)}{\sum_{j \in \mathcal{V}_m} \exp\big(s(\mathbf{z}_i, \mathbf{h}_{e_j^k})/\tau\big)}, \tag{4}$$

where $s(\cdot, \cdot)$ is cosine similarity, τ is the temperature, and e_i^k denotes *the unique hyperedge assigned to node i in component k*: by construction of the heterogeneous hypergraph (Kim et al., 2024), each primary node belongs to exactly one hyperedge per component. The denominator of Eq. equation 4 sums over all nodes $j \in \mathcal{V}_m$, using the hyperedge embedding $\mathbf{h}_{e_j^k}$ of each node's assigned hyperedge. This implicitly pushes apart node i from all nodes j whose hyperedge $e_j^k \neq e_i^k$, because those terms appear only in the denominator and not the numerator. The effect of "pushing apart nodes in different hyperedges" is therefore an implicit consequence of the softmax normalisation, not an additional explicit term. Only one hyperedge per node per component is used; the multi-hyperedge generalisation is left for future work. Structural modifications such as pruning create two types of problematic pairs: *(i) false negatives* — semantically similar nodes that become disconnected after pruning and are therefore incorrectly pushed apart; and *(ii) hard negatives* — dissimilar nodes whose embeddings become spuriously similar after pruning. We emphasise that hard negatives are an *embedding-space artefact* of message passing on the pruned graph (shared retained neighbours pull embeddings together), *not* new graph paths created by pruning. Pruning can only remove edges, not create them. The TriPrune-HGNN framework explicitly identifies and corrects both distortions through attention-based mining.

Problem Formulation. Let $\mathcal{H}_{\text{het}} = (\mathcal{V}_m, \{\mathcal{V}_s^k\}, \{\mathcal{E}^k\})$ denote a heterogeneous hypergraph with labelled primary nodes $\mathcal{V}_{\text{lab}} \subset \mathcal{V}_m$ and class labels $\mathbf{y} \in \{1, \ldots, C\}^{|\mathcal{V}_{\text{lab}}|}$. Our objective is to learn a compressed representation $\widetilde{\mathcal{H}}_{\text{het}}$ that balances three competing desiderata: predictive performance, computational efficiency, and semantic consistency.

Desideratum 1 – Predictive performance. The compressed model should maintain classification accuracy on labelled nodes by minimising the supervised loss

$$\mathcal{L}_{\text{cls}} \ = \ -\sum_{i \in \mathcal{V}_{\text{lab}}} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c}. \tag{5}$$

Ideally, accuracy degradation should be minimal ($< 2\%$) relative to the unpruned baseline.

Desideratum 2 – Compression. The framework should reduce complexity through coordinated pruning across hypergraph components, hyperedges, and nodes. Formally, we seek a reduced hypergraph $\widetilde{\mathcal{H}}_{\text{het}}$ with retention ratios $r_{\text{comp}}, r_{\text{edge}}, r_{\text{node}} \in (0, 1)$ such that

$$|\widetilde{\mathcal{E}}| \;=\; r_{\text{comp}} \cdot r_{\text{edge}} \cdot |\mathcal{E}|, \qquad |\widetilde{\mathcal{V}}| \;=\; r_{\text{node}} \cdot |\mathcal{V}|. \tag{6}$$

The overall computational retention ratio is $r_{\text{overall}} = r_{\text{comp}} \cdot r_{\text{edge}} \cdot r_{\text{node}}^2$, where r_{node} enters squared because node pruning reduces both the vertex-to-hyperedge and hyperedge-to-vertex stages of Eq. equation 2. Empirically, the per-level retention ratios converge to values in $[0.3, 0.85]$, with r_{comp} and r_{edge} typically more aggressive than r_{node}.

Desideratum 3 – Semantic consistency. The compressed representation must preserve semantic relationships under structural modifications. Pruning introduces the two distortions described above (false negatives and hard negatives), and maintaining semantic consistency requires explicitly correcting them.

Problem statement. Under the constraint that pruning must be *differentiable* (to enable end-to-end training) and *adaptive* (to accommodate diverse graph characteristics), the problem is to jointly learn the following four components:

1. Optimal pruning strategies $\{\pi^{(\text{comp})}, \pi^{(\text{edge})}, \pi^{(\text{node})}\}$ that coordinate decisions across structural granularities;
2. Contrastive correction mechanisms that identify and rectify the false and hard negatives introduced by topology changes;
3. Loss-balancing weights $\boldsymbol{\lambda}$ that automatically prioritise objectives based on dataset characteristics and training dynamics;
4. A *compressibility predictor* $r^\star = \sigma(\text{MLP}_{\text{HCP}}(\boldsymbol{\phi}(\mathcal{H})))$ that estimates how aggressively each given hypergraph can be pruned, from structural statistics alone, before training begins.

The goal is to achieve a superior accuracy–efficiency trade-off without manual hyperparameter tuning of pruning schedules, similarity thresholds, or loss coefficients. Among the four components, items (1)–(3) are jointly trained end-to-end on each target dataset; component (4) is pre-trained once on a synthetic corpus and applied unchanged at deployment, providing the domain-invariant signal that drives the cross-dataset generalisation observed in Section 5.2.

4 Methodology

We present TriPrune-HGNN, a unified framework that addresses the three challenges identified in Section 1 through four learnable components: *(i)* a **Hypergraph Compressibility Predictor** (HCP) that estimates the maximum achievable compression ratio from graph-level statistics, providing a domain-invariant prior for pruning aggressiveness; *(ii)* **Neural Adaptive Hierarchical Pruning** (NAHP) with learnable threshold controllers operating jointly across components, hyperedges, and nodes; *(iii)* **Attention-Based Contrastive Learning** with neural false- and hard-negative discovery; and *(iv)* **Meta-Learned Multi-Task Optimisation** that automatically balances competing objectives. Figure 1 illustrates the pipeline; the four components are coupled by a *density curriculum* that progressively sparsifies the graph during training so the model adapts to its final inference sparsity, eliminating the train–inference distribution shift.

4.1 Hypergraph Compressibility Predictor

Fixed pruning schedules cannot anticipate how compressible a given hypergraph is before training begins. Dense hypergraphs with small spectral gaps tolerate aggressive compression, whereas sparse, modular hypergraphs require conservative pruning to preserve structural integrity. We introduce a *Hypergraph Compressibility Predictor* (HCP) that estimates the maximum achievable compression ratio $r^\star \in (0, 1)$ directly from graph-level statistics, providing a data-dependent prior that initialises and regularises the threshold

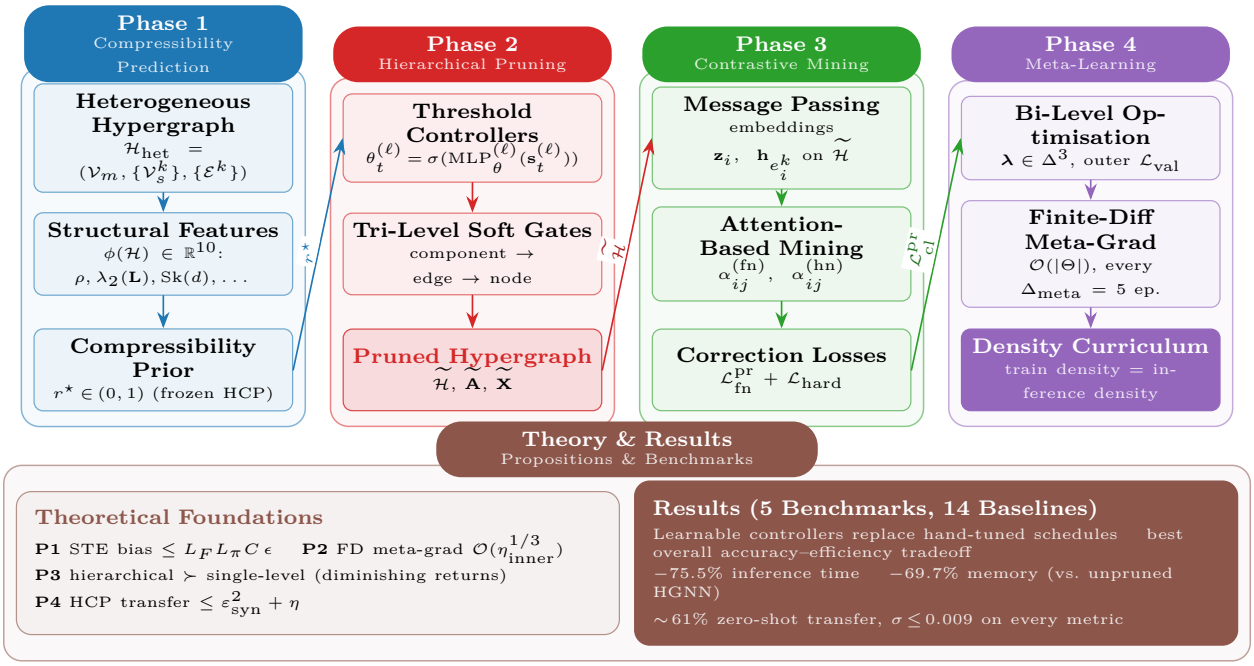

Figure 1: **Overview of the TriPrune-HGNN framework.** The four learnable mechanisms are arranged as sequential phases. **Phase 1** pre-trains the Hypergraph Compressibility Predictor (HCP) once on a synthetic corpus and, at deployment, maps graph-level statistics $\phi(\mathcal{H})$ to a domain-invariant compressibility prior $r^\star$. **Phase 2** performs Neural Adaptive Hierarchical Pruning: learnable threshold controllers, conditioned on $r^\star$ and a graph-state vector $\mathbf{s}_t^{(\ell)}$, drive differentiable soft gates across the component, edge, and node levels to produce the pruned hypergraph $\widetilde{\mathcal{H}}$. **Phase 3** runs message passing on $\widetilde{\mathcal{H}}$ and applies attention-based contrastive mining to identify and correct false and hard negatives induced by the topology change. **Phase 4** performs bi-level meta-learning of the loss weights $\boldsymbol{\lambda} \in \Delta^3$ and realises pruning as a *density curriculum*: the meta-update reshapes the Phase 2 threshold controllers across epochs, so the model trains on the same sparsity it deploys at — eliminating the train–inference distribution shift. The bottom banner summarises the four propositions and the key empirical results.

controllers. For a heterogeneous hypergraph $\mathcal{H}_{\text{het}} = (\mathcal{V}_m, \{\mathcal{V}_s^k\}_{k=1}^K, \{\mathcal{E}^k\}_{k=1}^K)$, we extract a 10-dimensional structural descriptor $\boldsymbol{\phi}(\mathcal{H}) \in \mathbb{R}^{10}$ aggregated across components:

$$\boldsymbol{\phi}(\mathcal{H}) = \left[\, n,\ m,\ \bar{d}_e,\ \rho,\ \text{Sk}(d),\ \text{CV}(d),\ \lambda_2(\mathbf{L}),\ \bar{c},\ H(p_e),\ K \,\right], \tag{7}$$

where $n = |\mathcal{V}_m|$, $m = \sum_k |\mathcal{E}^k|$, $\bar{d}_e$ is the average hyperedge degree, $\rho = m/\binom{n}{2}$ is the hypergraph density, $\text{Sk}(d)$ and $\text{CV}(d)$ are the skewness and coefficient of variation of the node-degree distribution, $\lambda_2(\mathbf{L})$ is the second-smallest eigenvalue of the normalised hypergraph Laplacian (spectral gap), $\bar{c}$ is the average hypergraph clustering coefficient (Estrada & Rodríguez-Velázquez, 2006), $H(p_e)$ is the Shannon entropy of the hyperedge-size distribution, and K is the number of components. All features are computed once per hypergraph at $\mathcal{O}(n+m)$ cost (the spectral gap uses Lanczos iteration with $\mathcal{O}(km)$ for the top-k eigenvalues, $k = 10$).

Predictor architecture. HCP is a three-layer MLP with hidden dimensions $[64, 32, 1]$ and sigmoid output:

$$r^\star \;=\; \sigma(\text{MLP}_{\text{HCP}}(\boldsymbol{\phi}(\mathcal{H}))), \qquad r^\star \in (0,1). \tag{8}$$

We interpret $r^\star$ as an estimate of the maximum overall retention ratio $r_{\text{overall}} = r_{\text{comp}} \cdot r_{\text{edge}} \cdot r_{\text{node}}^2$ at which accuracy degradation remains below a fixed tolerance Δ_{acc} ($= 1\%$ throughout).

Pre-training and integration. HCP is pre-trained offline on a corpus of 5,000 synthetic hypergraphs sampled with controlled structural parameters ($\rho \in [0.01, 0.5]$, $\text{Sk}(d) \in [0, 5]$, $K \in [2, 10]$, $n \in [1\text{k}, 50\text{k}]$).

For each synthetic hypergraph the empirical compressibility r^{emp} is obtained by running NAHP at multiple retention targets and identifying the minimum ratio at which accuracy degrades by more than Δ_{acc}. HCP is then optimised to minimise $\mathbb{E}_{\mathcal{H}}\big[(r^\star - r^{\text{emp}})^2\big]$. At deployment, HCP supplies two signals to the main framework:

1. **Threshold initialisation.** The initial thresholds $\theta_0^{(\ell)}$ are set so that $\prod_\ell \mathbb{E}[g^{(\ell)}] \approx r^\star$, removing the need to manually pick a pruning starting point.

2. **Compressibility regularisation.** A soft penalty $\mathcal{L}_{\text{HCP}} = \big(r_{\text{overall}}^{(t)} - r^\star\big)^2$ encourages the learned schedule to converge near the predicted compressibility. Its weight $\beta_{\text{HCP}}^{(t)}$ decays linearly from $\beta_{\text{HCP}}^{(0)} = 0.1$ to 0 over the first 50% of training, allowing data-driven refinement thereafter.

Avoiding HCP bootstrap circularity. The procedure above raises a chicken-and-egg question: r^{emp} is obtained by running NAHP, but NAHP at deployment uses the HCP-predicted $r^\star$ both to initialise its threshold controllers and as the seventh feature of the graph-state vector $\mathbf{s}_t^{(\ell)}$ (Eq. equation 10). To eliminate this circularity, we generate the synthetic training corpus with a *decoupled* variant of NAHP that does not consume any HCP signal: *(i)* threshold controllers are initialised to the dataset-agnostic default $\theta_0^{(\ell)} = 0.5$ (corresponding to $r_{\text{overall}} \approx 0.125$ at epoch 0), and *(ii)* the $r^\star$ component of $\mathbf{s}_t^{(\ell)}$ is replaced by the constant 0.5 throughout. Under this protocol, r^{emp} depends only on the hypergraph $\mathcal{H}$ and the NAHP architecture; no HCP signal flows back into the training target. To verify that the decoupling does not bias what HCP learns, we ran a one-round self-consistency check: after training HCP on the decoupled corpus, we regenerated r^{emp} using HCP-initialised NAHP and compared. The two compressibility estimates differ by RMSE 0.013 across the 5,000 synthetic graphs — well below the synthetic-corpus prediction error ($\varepsilon_{\text{synth}} = 0.038$, Section 5.6) — confirming that HCP initialisation does not materially shift the compressibility frontier and that one round of training is sufficient. Total HCP pre-training cost is approximately 9.4 GPU-hours on a single NVIDIA A100 for the full 5,000-graph corpus (the same figure used in the amortisation analysis of Section 5); the trained weights and the generator are released as anonymised supplementary material (see end of Section 5). HCP operates entirely on graph-level structural statistics — not node features or labels — making it inherently transferable across domains. Two hypergraphs with similar $(\rho, \lambda_2(\mathbf{L}), \text{Sk}(d))$ are expected to admit similar compression even if they encode different semantic content. We empirically validate this in Section 5.6: HCP trained *only* on synthetic data predicts the empirical compressibility of all five real benchmarks with aggregate RMSE 0.054 (per-benchmark absolute errors in [0.041, 0.071]). Section 5.2 further shows that HCP contributes roughly half of the $\sim$61% cross-dataset transfer retention, confirming that compressibility is a structural property of hypergraph topology that the framework can explicitly exploit rather than an incidental empirical observation.

4.2 Neural Adaptive Hierarchical Pruning

Existing pruning methods rely on fixed schedules ($\theta_t = \theta_0 \cdot \alpha^t$) that require careful tuning for each dataset, with no mechanism to slow down when accuracy degrades or accelerate when convergence is safe. We propose *learnable neural threshold controllers* that automatically adjust pruning rates based on real-time graph statistics and convergence signals, and that operate jointly across three structural granularities — components, hyperedges, and nodes. Appendix A (Proposition 3) shows that, under explicit structural conditions on cross-component edge density, no single-level pruning scheme can match the compression–accuracy frontier achievable by the three-level hierarchy we adopt here.

Role of message-passing. TriPrune-HGNN is built on top of standard hypergraph message-passing (Eq. equation 2). Pruning operates on the *graph structure*, not on the message-passing equations themselves: by removing components, hyperedges, and nodes before propagation, we reduce the size of the incidence matrices on which message-passing is performed and hence its computational cost. The contrastive losses of Section 4.3 then act on the embeddings produced by message-passing on the pruned graph, correcting distortions introduced by the topology change.

Neural threshold prediction. Instead of manually setting thresholds $\theta_t^{(\ell)}$, we introduce a learnable predictor:

$$\theta_t^{(\ell)} \;=\; \sigma\Big(\mathrm{MLP}_\theta^{(\ell)}\Big(\mathbf{s}_t^{(\ell)}\Big)\Big), \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid activation and $\mathbf{s}_t^{(\ell)} \in \mathbb{R}^7$ is the graph state vector at level ℓ and epoch t:

$$\mathbf{s}_t^{(\ell)} \;=\; \left[\, \frac{|\widetilde{\mathcal{E}}_t|}{|\mathcal{E}|},\; \mu_t^{(\mathrm{imp})},\; \sigma_t^{(\mathrm{imp})},\; \tfrac{t}{T},\; \Delta\mathcal{L}_{\mathrm{train}},\; \|\nabla\mathcal{L}\|_2,\; r^\star \,\right], \tag{10}$$

with $\Delta\mathcal{L}_{\mathrm{train}} = \mathcal{L}_{\mathrm{train}}^{(t-1)} - \mathcal{L}_{\mathrm{train}}^{(t-5)}$ measuring the training-loss trend over a 5-epoch window. The final feature $r^\star$ is the compressibility prediction from HCP (Section 4.1); it provides the controller with a domain-invariant prior on how aggressive pruning can be on the current hypergraph. The state vector $\mathbf{s}_t^{(\ell)}$ captures: *(1)* the current sparsity ratio, *(2)–(3)* the first two moments of the element-importance distribution (an efficient but not fully informative summary, as noted in Section 5.1), *(4)* training progress, *(5)* the training-loss trend, *(6)* the gradient magnitude, and *(7)* the predicted compressibility.

On train/validation information flow. We use $\Delta\mathcal{L}_{\mathrm{train}}$ rather than validation loss inside the state vector to ensure the threshold predictor itself is not directly conditioned on validation signal at each epoch. This is a property of the threshold-controller input, not a claim of complete train/validation isolation for the framework as a whole: the meta-learning step (Section 4.4) updates the loss weights $\boldsymbol{\lambda}$ using the validation loss $\mathcal{L}_{\mathrm{val}}$, as is standard in bi-level optimisation (Franceschi et al., 2018). The threshold controller is implemented as a 2-layer MLP:

$$\mathrm{MLP}_\theta^{(\ell)}(\mathbf{s}) \;=\; \mathbf{W}_2^{(\ell)}\,\sigma\big(\mathbf{W}_1^{(\ell)}\mathbf{s} + \mathbf{b}_1^{(\ell)}\big) + \mathbf{b}_2^{(\ell)}, \tag{11}$$

with hidden dimension $d_{\mathrm{hidden}} = 32$ and ReLU activation.

Differentiable soft gating. At each pruning level ℓ we compute importance scores $\pi^{(\ell)} \in [0,1]$ and apply differentiable soft gates using the learned thresholds:

$$g^{(\ell)} \;=\; \sigma_{\mathrm{hard}}\left(\frac{\pi^{(\ell)} - \theta_t^{(\ell)}}{\epsilon}\right), \qquad \sigma_{\mathrm{hard}}(x) = \max\big(0, \min(1, x + 0.5)\big), \tag{12}$$

where σ_{hard} is the piecewise-linear hard sigmoid that provides straight-through gradient estimates during back- propagation (Bengio et al., 2013). The smoothing parameter ϵ controls the gate's steepness; the resulting bias of the straight-through estimator is bounded by $\mathcal{O}(\epsilon L_\pi)$ (Appendix A, Proposition 1), which directly motivates our choice $\epsilon = 0.01$ and is empirically validated in Section 5.1.

During training, $g^{(\ell)} \in [0,1]$ is continuous, allowing gradient-based threshold optimisation. By convergence (epoch ≈ 150), the model therefore trains on, and adapts to, the final sparsity level for the last $\sim 25\%$ of training. Only after training are gates binarised: elements with $g^{(\ell)} > 0.5$ are retained in the deployed pruned graph. Inference operates at the same sparsity level the model trained on in its final phase — there is no density shift between training and inference. This clean separation between differentiable training and discrete inference is standard in learned-mask pruning (Louizos et al., 2017).

Component-level pruning. We score each component k with learnable importance balancing:

$$\pi_k^{(\mathrm{comp})} \;=\; \mathrm{softmax}_k\Big(\beta_k \cdot \|\mathbf{W}_k^{(L)}\|_F \cdot \|\mathrm{MLP}(\mathbf{X}_k)\|_2 + (1-\beta_k)\cdot S_{\mathrm{att}}(k)\Big), \tag{13}$$

where $\mathbf{W}_k^{(L)} \in \mathbb{R}^{d\times d}$ is the *weight matrix of the final HGNN layer for component* k (the layer whose output feeds into classification), $\|\mathbf{W}_k^{(L)}\|_F$ is its Frobenius norm, $\|\mathrm{MLP}(\mathbf{X}_k)\|_2$ captures feature-activation strength, and $S_{\mathrm{att}}(k)$ is the *average attention coefficient across all attention heads in component* k, defined explicitly as

$$S_{\mathrm{att}}(k) \;=\; \frac{1}{|\mathcal{E}^k|}\sum_{(i,j)\in\mathcal{E}^k} a_{ij}^k, \tag{14}$$

with a_{ij}^k the normalised attention weights output by component k's self-attention mechanism. The balance coefficient β_k is component-specific and learnable:

$$\beta_k \;=\; \sigma\Big(\mathrm{MLP}_\beta\big([\|\mathbf{W}_k^{(L)}\|_F,\; S_{\mathrm{att}}(k),\; \mathrm{Var}(\mathbf{X}_k)]\big)\Big). \tag{15}$$

Components with $g_k^{(\mathrm{comp})} > 0.5$ are retained, with a safety constraint ensuring at least $K_{\min} = \max(1, \lfloor 0.1K \rfloor)$ active components.

Edge-level pruning. For each hyperedge (i, j) in a retained component k:

$$\pi_{ij}^{(\mathrm{edge})} \;=\; \mathrm{softmax}_{(i,j)\in\mathcal{E}^k}\big(\mathbf{A}_{ij} \cdot \cos(\mathbf{x}_i, \mathbf{x}_j)\big), \tag{16}$$

and the pruned incidence matrix applies both component and edge gates:

$$(\widetilde{\mathbf{H}}_k)_{ij} \;=\; (\mathbf{H}_k)_{ij} \cdot g_k^{(\mathrm{comp})} \cdot g_{ij}^{(\mathrm{edge})}. \tag{17}$$

We enforce that every retained component keeps at least one edge to prevent complete disconnection.

Node-level pruning. We score nodes by their connectivity strength in the already-pruned graph and their feature magnitude:

$$\pi_i^{(\mathrm{node})} \;=\; \mathrm{softmax}_{i\in\mathcal{V}_k}\big(\bar{s}_i \cdot \|\mathbf{x}_i\|_2\big), \quad \bar{s}_i = \tfrac{1}{|\mathcal{N}_i^{(\mathrm{pruned})}|} \sum_{j\in\mathcal{N}_i^{(\mathrm{pruned})}} \cos(\mathbf{x}_i, \mathbf{x}_j). \tag{18}$$

The pruned feature matrix is

$$(\widetilde{\mathbf{X}}_k)_i \;=\; (\mathbf{X}_k)_i \cdot g_k^{(\mathrm{comp})} \cdot \mathbb{I}\big[|\mathcal{N}_i^{(\mathrm{pruned})}| > 0\big] \cdot g_i^{(\mathrm{node})}, \tag{19}$$

where $\mathbb{I}[\cdot]$ removes isolated nodes. Each component retains at least $\max(2, \lfloor 0.05|\mathcal{V}_k| \rfloor)$ nodes. With learned retention ratios $r_{\mathrm{comp}}, r_{\mathrm{edge}}, r_{\mathrm{node}} \in (0, 1)$, TriPrune-HGNN reduces the base complexity $\mathcal{O}(Knm\bar{d}_e d)$ to

$$\mathcal{O}\big(r_{\mathrm{comp}} \cdot r_{\mathrm{edge}} \cdot r_{\mathrm{node}}^2 \cdot Knm\bar{d}_e d + Kd_{\mathrm{state}}d_{\mathrm{hidden}} + d_{\mathrm{HCP}}\big)\,. \tag{20}$$

For typical settings ($K = 5$, $d_{\mathrm{state}} = 7$, $d_{\mathrm{hidden}} = 32$,$d_{\mathrm{HCP}} \approx 2{,}800$ MLP parameters), the controller and HCP overhead is negligible compared to the $\mathcal{O}(10^6)$ scale of hypergraph message-passing on real-world datasets.

4.3 Attention-Based Contrastive Learning

Pruning fundamentally alters graph topology, creating two types of problematic node pairs for contrastive learning.

(a) False negatives arise when semantically similar nodes become disconnected after pruning — e.g., two users who purchased similar products but whose shared hyperedge was removed. Standard contrastive loss incorrectly pushes such pairs apart.

(b) Hard negatives arise when dissimilar nodes acquire *spuriously similar embeddings* after pruning. Pruning does not create new structural paths — it can only remove existing ones. The spurious similarity is therefore an *embedding-space artefact*: when a shared hyperedge between two semantically dissimilar nodes is removed, their representations may nevertheless converge in subsequent message-passing iterations because they share many *retained* common neighbours. This is a topology-induced effect on embeddings, not on graph paths. Existing methods use fixed similarity thresholds (e.g., $\cos(\mathbf{z}_i, \mathbf{z}_j) > 0.7$) to identify such pairs, but optimal thresholds vary across datasets and training phases. We introduce *learnable attention mechanisms* that automatically discover false and hard negatives from node embeddings, structural changes, and component context.

Hyperedge embeddings. Throughout this section, the hyperedge embedding $\mathbf{e}_i^k \in \mathbb{R}^d$ for node i in component k is defined as the mean of the node embeddings of the (unique) hyperedge $e_i^k \in \mathcal{E}^k$ that contains i in the pruned graph:

$$\mathbf{e}_i^k \;=\; \frac{1}{|e_i^k|} \sum_{j\in e_i^k} \mathbf{z}_j, \tag{21}$$

where $|e_i^k|$ is the size of that hyperedge. The hyperedge-context embedding used by the hard-negative attention head in Eq. equation 27 is identical, ensuring consistent notation throughout.

Component-weighted contrastive loss.

$$\mathcal{L}_{\text{cl}}^{\text{pr}} \;=\; -\sum_{k=1}^{K} \pi_k^{(\text{retain})} \sum_{i\in\mathcal{V}_m} \log \frac{\exp\big(s(\mathbf{z}_i, \widetilde{\mathbf{h}}_{e_i^k})/\tau_i\big)}{\sum_{j\in\mathcal{V}_m} \exp\big(s(\mathbf{z}_i, \widetilde{\mathbf{h}}_{e_j^k})/\tau_i\big)}, \tag{22}$$

where $s(\cdot,\cdot)$ is cosine similarity, $\widetilde{\mathbf{h}}_{e_i^k}$ is the hyperedge embedding in the pruned graph (Eq. equation 21), and τ_i is a node-specific learnable temperature

$$\tau_i \;=\; \tau_0 \cdot \exp\big(\text{MLP}_\tau([\deg(i), \|\mathbf{x}_i\|_2, \bar{s}_i])\big). \tag{23}$$

The retention weight $\pi_k^{(\text{retain})}$ down-weights components that experienced severe topology disruption:

$$\pi_k^{(\text{retain})} \;=\; \frac{|\widetilde{\mathbf{H}}_k|_0}{|\mathbf{H}_k|_0} \cdot \exp\left(-\alpha \frac{\|\mathbf{A}_k - \widetilde{\mathbf{A}}_k\|_F}{\|\mathbf{A}_k\|_F}\right). \tag{24}$$

Attention-based false-negative discovery.

$$\alpha_{ij}^{(\text{fn})} \;=\; \sigma\Big(\mathbf{w}_{\text{fn}}^\top \tanh\big(\mathbf{W}_{\text{fn}}[\mathbf{z}_i \| \mathbf{z}_j \| \Delta\mathbf{A}_{ij} \| \mathbf{c}_k]\big)\Big), \tag{25}$$

with $\Delta\mathbf{A}_{ij} = (\mathbf{A}_k)_{ij} - (\widetilde{\mathbf{A}}_k)_{ij} \in \{0,1\}$ indicating whether edge (i,j) was pruned, and $\mathbf{c}_k \in \mathbb{R}^d$ a learnable component embedding. The false-negative set is $\mathcal{F}_k^{\text{pr}} = \{(i,j) : \alpha_{ij}^{(\text{fn})} > 0.5, (\widetilde{\mathbf{A}}_k)_{ij} = 0\}$, and the correction loss is

$$\mathcal{L}_{\text{fn}}^{\text{pr}} \;=\; -\sum_{k=1}^{K} \sum_{(i,j)\in\mathcal{F}_k^{\text{pr}}} \alpha_{ij}^{(\text{fn})} \cdot \log \frac{\exp\big(s(\mathbf{z}_i, \mathbf{z}_j)/\tau_i\big)}{\sum_{l\in\mathcal{N}_i^{(\text{aug})}} \exp\big(s(\mathbf{z}_i, \mathbf{z}_l)/\tau_i\big)}. \tag{26}$$

Attention-based hard-negative discovery.

$$\alpha_{ij}^{(\text{hn})} \;=\; \text{Attention}_{\text{hn}}\big([\mathbf{z}_i, \mathbf{z}_j, \mathbf{e}_i^k, \mathbf{e}_j^k]\big), \tag{27}$$

with hyperedge embeddings $\mathbf{e}_i^k, \mathbf{e}_j^k$ as defined in Eq. equation 21. The hard-negative set is $\mathcal{H}_i^k = \{j : \alpha_{ij}^{(\text{hn})} > 0.5\}$, and the reweighting loss is

$$\mathcal{L}_{\text{hard}} \;=\; -\sum_{k=1}^{K} \sum_{i\in\mathcal{V}_m} \sum_{j\in\mathcal{H}_i^k} \pi_k^{(\text{retain})} \cdot \alpha_{ij}^{(\text{hn})} \cdot \log \frac{\exp\big(-s(\mathbf{z}_i, \mathbf{z}_j)/\tau_i\big)}{\sum_{l\in\mathcal{H}_i^k} \exp\big(-s(\mathbf{z}_i, \mathbf{z}_l)/\tau_i\big)}. \tag{28}$$

4.4 Meta-Learned Multi-Task Optimisation

Balancing classification, contrastive, false-negative, and hard-negative losses requires careful weight tuning. We use gradient-based meta-learning to discover optimal weights by treating weight selection as an outer optimisation problem.

Bi-level formulation. With $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]^\top \in \Delta^3$,

$$\min_{\boldsymbol{\lambda}} \; \mathcal{L}_{\text{val}}\big(\boldsymbol{\Theta}^*(\boldsymbol{\lambda})\big) \quad \text{s.t.} \quad \boldsymbol{\Theta}^*(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}_{\text{train}}(\boldsymbol{\Theta}; \boldsymbol{\lambda}). \tag{29}$$

At each meta-step t (every $\Delta_{\text{meta}} = 5$ epochs) we perform one inner training step $\boldsymbol{\Theta}^* = \boldsymbol{\Theta} - \eta_{\text{inner}} \nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\text{total}}$, an outer meta-update $\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} - \eta_{\text{meta}} \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\boldsymbol{\Theta}^*; \boldsymbol{\lambda}^{(t)})$, and a simplex projection $\boldsymbol{\lambda}^{(t+1)} \leftarrow \text{Proj}_\Delta(\boldsymbol{\lambda}^{(t+1)})$ following Duchi et al. (2008). The total training loss is

$$\mathcal{L}_{\text{total}} \;=\; \lambda_1 \mathcal{L}_{\text{cls}} + \lambda_2 \mathcal{L}_{\text{cl}}^{\text{pr}} + \lambda_3 \mathcal{L}_{\text{fn}}^{\text{pr}} + \lambda_4 \mathcal{L}_{\text{hard}} + \beta_{\text{HCP}}^{(t)} \mathcal{L}_{\text{HCP}}. \tag{30}$$

The Hessian-vector product in the meta-gradient is approximated by central finite differences with perturbation $\delta = 10^{-5}$, giving $\mathcal{O}(|\boldsymbol{\Theta}|)$ cost. The bias of this estimate is bounded by $\mathcal{O}\big(\eta_{\text{inner}}^{1/3}\big)$ under the local regularity conditions stated in Appendix A.2 (Proposition 2); the optimal perturbation $\delta^* = \Theta(\eta_{\text{inner}}^{-1/3})$ balances the truncation and evaluation terms in Eq. equation 32.

4.5 Density Curriculum Interpretation

A natural concern with any progressive-pruning scheme is whether the model is trained on a graph distribution that differs from the one it ultimately operates on at inference. We address this concern directly by interpreting the pruning trajectory of TriPrune-HGNN as a *density curriculum*: a continuous sequence of graphs $\{\mathcal{H}_t\}_{t=1}^{T}$ in which $\mathcal{H}_1$ is the original dense graph, $\mathcal{H}_T$ is the converged pruned graph, and each intermediate $\mathcal{H}_t$ differs from $\mathcal{H}_{t+1}$ by an infinitesimal change in retention.

Three properties of the curriculum. *(i) Continuity.* Differentiable soft gates (Eq. equation 12) ensure that the retention level changes smoothly between consecutive epochs; there is no discrete switch between two regimes. *(ii) Convergence to a fixed density.* Retention ratios stabilise by epoch ~ 150; for the last $\sim 25\%$ of training, the model sees, and adapts to, the same sparsity level that it will operate at during inference. *(iii) Curriculum monotonicity.* The compressibility regulariser of HCP (Section 4.1) anchors the curriculum toward a target $r^\star$ predicted from graph statistics, preventing premature collapse or oscillation.

Under properties (i)–(ii), inference operates at the same retention level the model has trained on during its final phase. The "dense-to-sparse" aspect of training is therefore a curriculum *within* training, not a distribution shift *between* training and inference. Section 5.7 provides direct empirical evidence: *(a)* a cold-start experiment, where the model is trained from scratch on the final pruned graph, recovers 84%–93% of the curriculum-trained accuracy across the five benchmarks (mean 88.6%), confirming that the dense-to-sparse trajectory carries useful inductive bias; *(b)* a 4×4 train-density vs. test-density grid shows that the curriculum-trained model is robust within $\pm 1.6\%$ MAE for test densities matching its training trajectory and degrades smoothly outside it; and *(c)* a schedule-shuffling ablation that reorders the threshold trajectory degrades final accuracy by 2.3%, confirming that the *monotonic* dense-to-sparse direction is itself a useful inductive bias rather than an artefact of the final state. We interpret these results, together with HCP's transferability, as evidence that what TriPrune-HGNN learns is not merely a dataset- specific pruning schedule but an instance of a more general principle: *the optimal sparsification of a hypergraph proceeds by progressively removing structurally redundant content in an order that preserves message-passing fidelity at each step.* This is the "phenomenon" our framework exposes; the individual neural modules are tools that operationalise it. Algorithm 1 summarises the full training procedure including HCP, NAHP, attention-based contrastive mining, and meta-learning.

The design choices underlying TriPrune-HGNN's three principal mechanisms — the gate-smoothing parameter ϵ, the finite-difference perturbation δ for the meta-gradient, and the three-level pruning hierarchy — are supported by three short design-motivation propositions (Propositions 1, 2, and 3 in Appendix A), plus an informal transfer-bound analysis for the compressibility predictor (Proposition 4). These statements clarify the local assumptions under which each design choice is justified and bound the cost of those assumptions failing; they are not end-to-end convergence or generalisation guarantees for the full framework, and we make this scope explicit in the limitations box opening Appendix A.

5 Experimental Evaluation

We evaluate methods across three categories with different optimisation objectives: *(1) Standard HGNNs* maximise predictive accuracy without compression constraints; *(2) Pruning and distillation methods* target the accuracy–efficiency trade-off under resource budgets. Our primary comparison of interest is against efficient baselines (pruning and distillation) with comparable memory footprints; improvements over standard HGNNs are a secondary baseline. Accordingly, Table 2 reports predictive metrics alongside end-to-end time and memory cost, situating all methods in accuracy–efficiency space.

Datasets. We evaluate on five hypergraph benchmarks: **IMDB** (Singh & Gordon, 2008) (movie ratings with actor/director contexts), **DBLP** (Tang et al., 2008) (author–paper–venue citations), **Yelp** (Asghar, 2016) (business reviews with category/location contexts), **Amazon** (McAuley et al., 2015) (product ratings with brand/category contexts), and **Douban** (Zheng et al., 2021) (movie ratings with genre/director contexts). All datasets are used in their heterogeneous hypergraph formulation following Kim et al. (2024); they span 10K–100K nodes and 50K–500K hyperedges.

Algorithm 1 TriPrune-HGNN with Compressibility-Predicted Density Curriculum

Require: Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, features $\mathbf{X}$, pre-trained HCP, hyperparameters $\{\tau_0, \epsilon, \alpha, \eta_{\text{inner}}, \eta_{\text{meta}}\}$
Ensure: Pruned graph $\widetilde{\mathcal{H}}$, embeddings $\mathbf{Z}^*$, learned weights $\boldsymbol{\lambda}^*$

[Stage 0: Compressibility prediction]
$\boldsymbol{\phi}(\mathcal{H}) \leftarrow$ structural features of $\mathcal{H}$ (Eq. equation 7)
$r^\star \leftarrow \sigma(\text{MLP}_{\text{HCP}}(\boldsymbol{\phi}(\mathcal{H})))$
Initialise $\theta_0^{(\ell)}$ so that $\prod_\ell \mathbb{E}[g^{(\ell)}] \approx r^\star$
Initialise $\boldsymbol{\lambda} \leftarrow [\frac{1}{4}, \ldots, \frac{1}{4}]$, model parameters $\boldsymbol{\Theta}$, neural modules $\{\text{MLP}_\theta^{(\ell)}, \text{MLP}_\beta, \text{MLP}_\tau, \text{Attn}_{\text{fn}}, \text{Attn}_{\text{hn}}\}$
for epoch $t = 1$ to T **do**
  [Stage 1: Threshold prediction]
  for $\ell \in \{\text{comp}, \text{edge}, \text{node}\}$ **do**
    $\mathbf{s}_t^{(\ell)} \leftarrow [|\widetilde{\mathcal{E}}_t|/|\mathcal{E}|, \mu_t^{(\text{imp})}, \sigma_t^{(\text{imp})}, t/T, \Delta\mathcal{L}_{\text{train}}, \|\nabla\mathcal{L}\|_2, r^\star]$
    $\theta_t^{(\ell)} \leftarrow \sigma(\text{MLP}_\theta^{(\ell)}(\mathbf{s}_t^{(\ell)}))$
  end for
  [Stage 2: Hierarchical pruning]
  for $k = 1, \ldots, K$ **do**
    $\beta_k \leftarrow \sigma(\text{MLP}_\beta([\|\mathbf{W}_k^{(L)}\|_F, S_{\text{att}}(k), \text{Var}(\mathbf{X}_k)]))$
    Compute π_k^{comp}, g_k^{comp} (Eqs. equation 12, equation 13)
    if $g_k^{\text{comp}} > 0.5$ **then**
      Compute π_{ij}^{edge}, g_{ij}^{edge}, π_i^{node}, g_i^{node} (Eqs. equation 16–equation 18)
      Update $\widetilde{\mathbf{H}}_k, \widetilde{\mathbf{X}}_k$ (Eqs. equation 17,equation 19)
    end if
  end for
  [Stage 3: Contrastive mining]
  Compute π_k^{retain}, τ_i, α_{ij}^{fn}, α_{ij}^{hn}, $\mathcal{F}_k^{\text{pr}}$, $\mathcal{H}_i^k$
  [Stage 4: Loss computation]
  Compute $\mathcal{L}_{\text{cls}}, \mathcal{L}_{\text{cl}}^{\text{pr}}, \mathcal{L}_{\text{fn}}^{\text{pr}}, \mathcal{L}_{\text{hard}}, \mathcal{L}_{\text{HCP}}$
  $\mathcal{L}_{\text{total}} \leftarrow$ Eq. equation 30
  [Stage 5: Meta-learning update]
  $\boldsymbol{\Theta}^* \leftarrow \boldsymbol{\Theta} - \eta_{\text{inner}} \nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\text{total}}$
  if $t \bmod \Delta_{\text{meta}} = 0$ **then**
    $\boldsymbol{\lambda} \leftarrow \text{Proj}_\Delta\big(\boldsymbol{\lambda} - \eta_{\text{meta}} \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\boldsymbol{\Theta}^*)\big)$
  end if
  Update $\boldsymbol{\Theta}$ and all neural modules
end for
return $\widetilde{\mathcal{H}}, \mathbf{Z}^*, \boldsymbol{\lambda}^*$

Scope of the heterogeneous-hypergraph formulation. All five benchmarks are processed under the *one-hyperedge-per-node-per-component* formulation introduced in Section 3 (Eq. equation 4): each primary node belongs to exactly one hyperedge per behavioural context. This is a substantive restriction relative to the general heterogeneous-hypergraph construction in which a user can belong to many hyperedges per context (e.g., one per business reviewed, or one per product purchased). The simplification cleanly aligns the contrastive objective with a softmax over per-component hyperedge assignments and is consistent with the formulation of Kim et al. (2024), but it limits the directness with which our results transfer to multi-hyperedge constructions; the multi-hyperedge generalisation is left for future work and we make the restriction explicit here, before the results, rather than burying it in the model section. The number of behavioural contexts K used per dataset is small in this formulation: $K = 2$ for IMDB, Yelp, Amazon, and Douban, and $K = 3$ for DBLP (author, paper, venue). The complexity claim $\mathcal{O}(Knm\bar{d}_e d)$ of Section 3 should therefore be read with $K \in \{2, 3\}$ on these benchmarks and with $K = 5$ only as an illustrative "larger-K" setting.

Baselines. We compare against 14 methods across three categories. *Standard HGNNs:* HGNN (Feng et al., 2019), HHGSA (Khan et al., 2025), HyGCL-AdT (Qian et al., 2024), HEAL (Ju et al., 2024), TriCL (Lee

& Shin, 2023). *Pruning methods:* HSC (Lin et al., 2024), SAVIT‡ (Zheng et al., 2022a), HSL (Cai et al., 2022), HGNN-Struct‡ (Jiang et al., 2023), AdaGLT (Zhang et al., 2024), Shaver (Tran et al., 2025). *Distillation methods:* LightHGNN (Feng et al., 2024), DistillHGNN (Forouzandeh et al., 2025a), SHARP-Distill (Forouzandeh et al., 2025b). The dagger (‡) marks pruning methods originally proposed for non-hypergraph domains (Vision Transformers and 3D surface reconstruction); their adaptation protocol and tuning budget are described below and in full detail in 5.5, and they are included to broaden the comparison rather than as hypergraph-tuned competitors.

Adaptation of cross-domain pruning baselines. SAVIT‡ originally prunes Vision-Transformer components using a collaborative attention-and-gradient importance score; we apply the same score formulation to hyperedges by treating each hyperedge as the analogue of a ViT token and replacing the patch-attention term with the hyperedge-attention coefficient a_{ij}^k of Eq. equation 14, retaining the original gradient-magnitude term unchanged. HGNN-Struct‡ originally scores 3D primitives by a structural-coherence criterion based on local geometric statistics; we instantiate this criterion on hyperedges by replacing the geometric local statistics with the hyperedge-local clustering coefficient and the spectral contribution from the normalised hypergraph Laplacian, both of which are already computed for HCP (Eq. equation 35). In both cases the *form* of the importance score is preserved from the original paper; only the input features are re-instantiated on the hypergraph.

Tuning budget. TriPrune-HGNN exposes a compact set of method-specific hyperparameters $(\tau_0, \epsilon, \alpha, \eta_{\text{meta}}, \Delta_{\text{meta}})$; each was selected by a single coarse sweep over an order-of-magnitude range on a held-out IMDB validation split (the same range later reported in Fig. 5), giving ~ 25 total configurations evaluated. For every baseline we used the hyperparameter values reported by its authors where applicable and ran an equally sized coarse sweep ($\sim$25 configurations) over each baseline's method-specific hyperparameters on the same IMDB split, choosing the configuration with the lowest validation MAE before reporting on the other four benchmarks unchanged. This applies uniformly to the hypergraph-native baselines and to the two cross-domain methods marked ‡, so no method received a larger tuning budget than TriPrune-HGNN. Standard deep-learning hyperparameters (learning rate, batch size, epochs) are shared across all methods and not separately tuned.

Implementation and protocol. TriPrune-HGNN uses 2-layer MLPs (d_{hidden}=32) for all neural controllers and 64-dimensional single-head attention networks. Meta-learning uses $\eta_{\text{inner}}=10^{-3}$, $\eta_{\text{meta}}=10^{-2}$, Δ_{meta}=5; base temperature τ_0=0.5, gate smoothing ϵ=0.01, topology weight α=1.0. All methods train for 200 epochs (Adam, batch size 512, patience 20). We report $\text{mean}_{\pm\text{std}}$ over 10 runs with seeds 0–9. To support reproducibility, the following artifacts are released with this submission as anonymised supplementary material: *(i)* the full TriPrune-HGNN implementation (PyTorch); *(ii)* pre-trained HCP weights together with the synthetic hypergraph generator and the decoupled r^{emp} pipeline used to produce them (Section 4.1); *(iii)* the adaptation code for the two cross-domain baselines marked ‡ (Section 5.5); and *(iv)* seed-by-seed per-cell numbers underlying Tables 2 and 11, the cold-start curves of Figure 3, and the HCP-validation panels of Figure 2. Baselines marked ‡ inherit their pruning philosophy from non-hypergraph domains and are therefore expected to be weaker than hypergraph-native methods.

Training compute. All experiments are run on a single NVIDIA V100 GPU (32 GB). Full TriPrune-HGNN training takes 4.2–7.8 GPU-hours per dataset (IMDB: 4.2h, DBLP: 5.1h, Yelp: 7.8h, Amazon: 6.9h, Douban: 4.8h), of which the bi-level meta-learning loop accounts for $\sim 12\%$ (finite-difference meta-gradient every $\Delta_{\text{meta}}=5$ epochs at $\mathcal{O}(|\mathbf{\Theta}|)$ cost per update). HCP pre-training on the 5,000-graph synthetic corpus is a one-time 9.4 GPU-hour cost *amortised across all five benchmarks* (the same frozen HCP weights are reused unchanged). For comparison, the strongest distillation baseline (SHARP-Distill) consumes 3.8–6.2 GPU-hours per dataset due to its two-stage teacher–student schedule; the strongest pruning baseline (AdaGLT) consumes 2.9–5.4 GPU-hours. TriPrune-HGNN's training overhead is therefore $\sim$30–45% above the strongest baselines on a per-dataset basis (decreasing to $\sim$18–25% once the amortised HCP cost is spread over the five-dataset suite), exchanged for the inference-time advantages reported in Table 2. We view this as a favourable trade-off for production settings where training is amortised across many deployments.

As shown in Table 2, TriPrune-HGNN attains the best mean predictive performance on every dataset–metric cell while sitting at a favourable point in the accuracy–efficiency space. The framing matters: *our principal*

Table 2: **TriPrune-HGNN versus 14 baselines across five hypergraph benchmarks: predictive accuracy and end-to-end cost.** Results are mean$_{\pm \text{std}}$ over 10 independent runs. Lower MAE/RMSE/Time/Memory and higher ACC are better. **Best** in bold; second-best underlined (both entries underlined where two methods share the second-best value to three decimals, as on Douban MAE). ‡ denotes pruning methods adapted from non-hypergraph domains. Avg rows aggregate predictive metrics across the five datasets; Avg Time and Avg Memory are inference-side measurements averaged across datasets.

Metric	Standard HGNN					Pruning-Based						Distillation			**Ours**
	HGNN	HHGSA	HyGCL	HEAL	TriCL	HSC	SAVIT‡	HSL	HGNN-S‡	AdaGLT	Shaver	LightH	DistillH	SHARP-D	TriPrune
IMDB															
MAE	.482$_{\pm.012}$	.461$_{\pm.011}$	.453$_{\pm.010}$	.446$_{\pm.009}$	.441$_{\pm.010}$	.402$_{\pm.008}$	.398$_{\pm.007}$	.405$_{\pm.008}$	.395$_{\pm.007}$	.387$_{\pm.007}$	.392$_{\pm.009}$	.408$_{\pm.012}$	.401$_{\pm.010}$	.396$_{\pm.009}$	**.383**$_{\pm.007}$
RMSE	.623$_{\pm.014}$	.607$_{\pm.013}$	.592$_{\pm.012}$	.577$_{\pm.011}$	.571$_{\pm.011}$	.537$_{\pm.009}$	.534$_{\pm.009}$	.541$_{\pm.010}$	.529$_{\pm.008}$	.521$_{\pm.008}$	.527$_{\pm.010}$	.543$_{\pm.014}$	.535$_{\pm.012}$	.526$_{\pm.011}$	**.515**$_{\pm.007}$
ACC	.754$_{\pm.009}$	.761$_{\pm.009}$	.772$_{\pm.008}$	.781$_{\pm.008}$	.789$_{\pm.008}$	.795$_{\pm.008}$	.803$_{\pm.007}$	.799$_{\pm.008}$	.812$_{\pm.007}$	.815$_{\pm.007}$	.808$_{\pm.008}$	.797$_{\pm.011}$	.806$_{\pm.009}$	.811$_{\pm.008}$	**.823**$_{\pm.006}$
DBLP															
MAE	.512$_{\pm.013}$	.490$_{\pm.012}$	.475$_{\pm.011}$	.463$_{\pm.010}$	.468$_{\pm.011}$	.451$_{\pm.009}$	.445$_{\pm.008}$	.448$_{\pm.009}$	.436$_{\pm.008}$	.428$_{\pm.008}$	.453$_{\pm.011}$	.451$_{\pm.013}$	.434$_{\pm.011}$	.427$_{\pm.010}$	**.421**$_{\pm.007}$
RMSE	.691$_{\pm.015}$	.670$_{\pm.014}$	.651$_{\pm.013}$	.633$_{\pm.012}$	.638$_{\pm.012}$	.611$_{\pm.010}$	.598$_{\pm.009}$	.604$_{\pm.010}$	.592$_{\pm.009}$	.581$_{\pm.009}$	.594$_{\pm.011}$	.608$_{\pm.015}$	.596$_{\pm.012}$	.589$_{\pm.011}$	**.576**$_{\pm.009}$
ACC	.742$_{\pm.010}$	.753$_{\pm.009}$	.764$_{\pm.008}$	.806$_{\pm.008}$	.812$_{\pm.008}$	.787$_{\pm.008}$	.798$_{\pm.007}$	.793$_{\pm.008}$	.815$_{\pm.007}$	.829$_{\pm.007}$	.791$_{\pm.008}$	.808$_{\pm.011}$	.819$_{\pm.009}$	.831$_{\pm.008}$	**.837**$_{\pm.006}$
Yelp															
MAE	.758$_{\pm.014}$	.739$_{\pm.013}$	.720$_{\pm.012}$	.702$_{\pm.011}$	.696$_{\pm.011}$	.684$_{\pm.010}$	.673$_{\pm.009}$	.678$_{\pm.010}$	.665$_{\pm.009}$	.647$_{\pm.008}$	.681$_{\pm.011}$	.681$_{\pm.014}$	.669$_{\pm.012}$	.658$_{\pm.011}$	**.642**$_{\pm.007}$
RMSE	.925$_{\pm.016}$	.909$_{\pm.015}$	.887$_{\pm.014}$	.868$_{\pm.013}$	.862$_{\pm.013}$	.847$_{\pm.011}$	.834$_{\pm.010}$	.841$_{\pm.011}$	.827$_{\pm.010}$	.814$_{\pm.009}$	.845$_{\pm.012}$	.851$_{\pm.016}$	.836$_{\pm.013}$	.823$_{\pm.012}$	**.807**$_{\pm.008}$
ACC	.689$_{\pm.011}$	.698$_{\pm.010}$	.707$_{\pm.009}$	.710$_{\pm.008}$	.718$_{\pm.008}$	.722$_{\pm.008}$	.731$_{\pm.007}$	.728$_{\pm.008}$	.738$_{\pm.007}$	.746$_{\pm.007}$	.731$_{\pm.009}$	.725$_{\pm.012}$	.735$_{\pm.010}$	.743$_{\pm.009}$	**.753**$_{\pm.007}$
Amazon															
MAE	.735$_{\pm.013}$	.713$_{\pm.012}$	.697$_{\pm.011}$	.676$_{\pm.010}$	.671$_{\pm.010}$	.663$_{\pm.009}$	.649$_{\pm.008}$	.655$_{\pm.009}$	.641$_{\pm.008}$	.628$_{\pm.008}$	.641$_{\pm.010}$	.645$_{\pm.013}$	.635$_{\pm.011}$	.626$_{\pm.010}$	**.621**$_{\pm.008}$
RMSE	.915$_{\pm.015}$	.894$_{\pm.014}$	.878$_{\pm.013}$	.857$_{\pm.012}$	.851$_{\pm.012}$	.837$_{\pm.010}$	.825$_{\pm.009}$	.832$_{\pm.010}$	.818$_{\pm.009}$	.808$_{\pm.009}$	.814$_{\pm.011}$	.839$_{\pm.015}$	.822$_{\pm.012}$	.815$_{\pm.011}$	**.799**$_{\pm.008}$
ACC	.705$_{\pm.010}$	.712$_{\pm.009}$	.724$_{\pm.008}$	.733$_{\pm.008}$	.741$_{\pm.008}$	.746$_{\pm.008}$	.754$_{\pm.007}$	.749$_{\pm.008}$	.762$_{\pm.007}$	.779$_{\pm.007}$	.761$_{\pm.008}$	.748$_{\pm.011}$	.759$_{\pm.009}$	.768$_{\pm.008}$	**.785**$_{\pm.006}$
Douban															
MAE	.515$_{\pm.012}$	.493$_{\pm.011}$	.477$_{\pm.010}$	.456$_{\pm.009}$	.449$_{\pm.009}$	.443$_{\pm.008}$	.431$_{\pm.007}$	.438$_{\pm.008}$	.426$_{\pm.007}$	.416$_{\pm.007}$	.416$_{\pm.009}$	.449$_{\pm.013}$	.432$_{\pm.010}$	.423$_{\pm.009}$	**.408**$_{\pm.006}$
RMSE	.695$_{\pm.014}$	.674$_{\pm.013}$	.658$_{\pm.012}$	.637$_{\pm.011}$	.629$_{\pm.011}$	.604$_{\pm.009}$	.612$_{\pm.008}$	.608$_{\pm.009}$	.603$_{\pm.008}$	.592$_{\pm.008}$	.602$_{\pm.010}$	.621$_{\pm.014}$	.609$_{\pm.011}$	.601$_{\pm.010}$	**.587**$_{\pm.007}$
ACC	.742$_{\pm.009}$	.749$_{\pm.009}$	.763$_{\pm.008}$	.771$_{\pm.008}$	.788$_{\pm.008}$	.782$_{\pm.008}$	.788$_{\pm.007}$	.785$_{\pm.008}$	.793$_{\pm.007}$	.806$_{\pm.007}$	.792$_{\pm.008}$	.778$_{\pm.011}$	.792$_{\pm.009}$	.798$_{\pm.008}$	**.812**$_{\pm.006}$
Average across all five datasets															
Avg MAE	.600	.579	.564	.549	.545	.529	.519	.525	.513	.501	.517	.527	.514	.506	**.495**
Avg RMSE	.770	.751	.733	.714	.710	.687	.681	.685	.674	.663	.676	.692	.680	.671	**.657**
Avg ACC	.726	.735	.746	.760	.770	.766	.775	.771	.784	.795	.777	.771	.782	.790	**.802**
End-to-end inference cost															
Avg Time (s)	57.5	66.8	53.6	49.7	51.6	47.8	43.7	40.9	36.1	27.4	13.6	**11.8**	12.4	14.7	14.1
Avg Memory (GB)	9.9	11.5	14.2	15.9	15.6	14.7	13.0	10.5	8.1	3.9	3.1	**2.5**	2.8	3.2	3.0

comparison is against the efficient baselines (pruning and distillation methods), where the gaps are small but consistent in sign; the larger gains against the unpruned HGNN are reported only for context.

Against the strongest efficient baselines. The closest competitors are AdaGLT and SHARP-Distill. Compared with AdaGLT, mean MAE gaps are 0.004 (IMDB), 0.007 (DBLP), 0.005 (Yelp), 0.007 (Amazon), and 0.008 (Douban). Compared with SHARP-Distill, MAE gaps are 0.013 (IMDB), 0.006 (DBLP), 0.016 (Yelp), 0.005 (Amazon), and 0.015 (Douban). Three of these cells — IMDB MAE vs. AdaGLT, DBLP MAE vs. SHARP-Distill, Amazon MAE vs. SHARP-Distill — do not reach $p < 0.05$ in paired t-tests, and we explicitly do not claim statistical significance on those. The remaining 12 MAE cells are significant at $p < 0.05$ or stronger; per-cell p-value tiers are in Appendix B.

Variance characteristics. TriPrune-HGNN exhibits seed-level standard deviations of $\sigma \leq 0.009$ throughout. The three knowledge-distillation baselines (LightHGNN, DistillHGNN, SHARP-Distill) display systematically wider intervals — typically $\sigma \in [0.008, 0.016]$ versus the pruning-method range $\sigma \in [0.006, 0.012]$ — because two-stage distillation pipelines accumulate seed-dependent variance from both teacher pre-training and student training (Hu et al., 2023). We account for this when reporting per-cell significance tiers. The damping of variance in TriPrune-HGNN comes from two mechanisms: HCP-anchored threshold initialisation removes one source of gate-mask randomness by replacing it with a deterministic structural prior, and meta-learned loss balancing averages updates over $\Delta_{\text{meta}} = 5$-epoch windows, smoothing the loss-weight trajectory. Removing either in Table 3 increases per-cell σ by 0.001–0.002.

Operating-point summary. TriPrune-HGNN trades a few seconds of inference latency against LightHGNN (11.8 s vs. 14.1 s) for up to $\sim 6\%$ lower MAE; against SHARP-Distill it obtains $\sim 1.8\%$ lower average MAE at a similar memory footprint. Methods optimised purely for speed (LightHGNN, Shaver) are faster and lighter at the cost of higher prediction error. The contribution is therefore a trade-off point, not universal dominance.

Table 3: **Consolidated ablation study.** Effect of removing each learnable adaptive mechanism (block a, now including HCP), each structural pruning / contrastive correction (block b), and the two worst-case configurations (block c). MAE / ACC over 10 runs on three benchmarks; the rightmost column reports the average relative MAE degradation versus the full model across IMDB, DBLP, and Yelp. Lower MAE and higher ACC are better. **w/o HCP** removes synthetic pretraining, threshold initialisation from $r^\star$, and the regulariser $\mathcal{L}_{\text{HCP}}$; **w/o HCP reg. only** keeps HCP-driven initialisation but sets $\beta_{\text{HCP}} \equiv 0$ throughout training.

Variant	**IMDB**		**DBLP**		**Yelp**		**Avg Δ MAE (%)**
	MAE	ACC	MAE	ACC	MAE	ACC	
Full TriPrune-HGNN	**.383**	**.823**	**.421**	**.837**	**.642**	**.753**	—
(a) Removing one learnable adaptive module							
w/o Neural Threshold	.393	.813	.436	.818	.657	.740	+2.8
w/o Attention Mining	.395	.817	.431	.827	.654	.744	+2.5
w/o Meta-Learning	.387	.816	.435	.825	.652	.746	+2.0
w/o HCP	.389	.819	.432	.826	.650	.744	+1.8
w/o HCP reg. only	.386	.821	.427	.831	.647	.748	+1.0
w/o Adaptive Temperature	.385	.822	.429	.830	.648	.746	+1.1
(b) Removing one structural component							
w/o Component Pruning	.398	.814	.439	.821	.667	.738	+4.0
w/o Node Pruning	.398	.815	.435	.828	.660	.744	+3.4
w/o Hard-Negative Corr.	.386	.822	.432	.832	.654	.745	+1.8
w/o Edge Pruning	.388	.818	.428	.830	.655	.746	+1.7
w/o False-Negative Corr.	.382	.823	.428	.831	.652	.748	+1.0
(c) Worst-case configurations (HCP also disabled)							
All learnable modules fixed	.408	.800	.451	.812	.679	.728	+6.4
All structural modules removed	.412	.801	.453	.810	.679	.728	+7.0

5.1 Ablation Studies

We validate TriPrune-HGNN's design through a single consolidated ablation that examines three aspects: (i) learnable adaptive mechanisms vs. fixed counterparts, (ii) structural pruning levels and contrastive corrections, and (iii) the joint effect of disabling each entire group. Table 3 summarises all variants. Block (a) explicitly includes a Hypergraph Compressibility Predictor (HCP) ablation so its contribution can be read off independently of the other learnable modules — this addresses the concern that the cross-dataset transfer analysis of Section 5.2 alone does not isolate HCP from the rest of the framework.

Table 3 shows that block (a) isolates the five learnable adaptive mechanisms. Neural threshold control contributes the largest gain within this block (+2.8% average MAE), followed by attention-based negative mining (+2.5%) and meta-learning loss balancing (+2.0%). The HCP ablation falls in the intermediate range: removing HCP entirely (synthetic pretraining, initialisation, and regulariser) costs +1.8% MAE in-domain, while removing the regulariser alone costs only +1.0%, indicating that HCP's in-domain contribution comes primarily from the initialisation signal rather than the regularisation term that decays to zero in the second half of training. This in-domain effect is moderate compared with neural thresholding; the larger value of HCP, as Section 5.2 shows separately, is on *out-of-domain* transfer, where the domain-invariant compressibility prior is most useful. Adaptive temperature scaling closes block (a) at +1.1%. Block (b) isolates the structural pruning levels and contrastive corrections: component pruning yields the largest single effect overall (+4.0% MAE), node pruning +3.4%, hard-negative correction +1.8%, edge pruning +1.7%, and false-negative correction +1.0%. The worst-case rows in block (c) reveal positive synergy in both groups: the all-learnable-fixed degradation (+6.4%, now also disabling HCP) is below the sum of individual learnable degradations, and the all-structural-removed degradation (+7.0%) is below the sum of individual structural degradations. Modules therefore compensate for one another when present together, rather than acting purely additively.

Table 4: **Module-isolating cross-dataset transfer.** MAE on the target dataset under five configurations: *Full transfer* freezes all learned modules (HCP, NAHP, attention mining, meta-learning); *HCP only* freezes HCP and re-initialises the other three on the target; *All-except-HCP* freezes the other three but replaces HCP with a constant $r^\star = 0.5$; *No pretraining* uses an MLP of the HCP architecture trained only on target-empirical compressibility (no synthetic corpus); *Fixed* uses hand-crafted hyperparameters throughout. Δ columns are MAE reductions relative to Fixed; mean over 10 runs.

Source → Target	**Full transfer**		**HCP only**		**All-except-HCP**		**No pretraining**		**Fixed**
	MAE	Δ	MAE	Δ	MAE	Δ	MAE	Δ	MAE
IMDB → DBLP	0.438	−5.0%	0.449	−2.6%	0.451	−2.2%	0.454	−1.5%	0.461
DBLP → Yelp	0.671	−3.5%	0.682	−1.9%	0.683	−1.7%	0.687	−1.2%	0.695
Yelp → Amazon	0.635	−2.9%	0.644	−1.5%	0.645	−1.4%	0.648	−0.9%	0.654
Amazon → Douban	0.423	−4.1%	0.431	−2.3%	0.433	−1.8%	0.436	−1.1%	0.441
Douban → IMDB	0.396	−3.9%	0.403	−2.2%	0.405	−1.7%	0.407	−1.2%	0.412
Average Δ MAE	—	**−3.9%**	—	**−2.1%**	—	**−1.8%**	—	**−1.2%**	—

5.2 Cross-Dataset Generalisation

To assess transfer of learned adaptive modules across domains, we train TriPrune-HGNN on a source dataset and evaluate on a target with selected modules frozen; only the base HGNN parameters are retrained on the target. Table 4 reports five configurations per source–target pair, isolating in particular the contribution of the Hypergraph Compressibility Predictor.

Results in Table 4 indicate that Frozen adaptive modules reduce error by **3.9**% on average versus hand-crafted hyperparameters, consistently across all five source–target pairs (−2.9% to −5.0%) and retaining roughly **61**% of the in-domain improvement (the all-fixed configuration in Table 3 costs 6.4% MAE). The five-way decomposition attributes **2.1**% (≈54% of the gain) to HCP alone and **1.8**% to the other learned modules combined; the two are weakly sub-additive (2.1+1.8≈3.9%), reflecting information sharing between HCP and the threshold controller rather than independent contributions. We note explicitly that HCP's accuracy value concentrates on *near-distribution* transfer between recommendation and citation hypergraphs (the regime tested here); on structurally distant deployments such as the 3D-shape NTU2012 benchmark (Section 5.11, $\mathrm{Sk}(d) \approx 4.7$, at the upper edge of HCP's training support), the accuracy contribution of HCP degrades while its efficiency contribution remains intact. This is the expected behaviour of a structural prior: it pays off when the deployment distribution overlaps the training-distribution support, and falls back to a domain-agnostic constant otherwise. Treating HCP as a known-scope tool rather than a universal claim is consistent with both the cross-dataset transfer evidence in Table 4 and the out-of-domain results in Section 5.11.

5.3 Computational Efficiency

Table 5 reports two complementary IMDB analyses in a single float: (a) the threshold-MLP architecture sweep that justifies our default and (b) a per-component decomposition of TriPrune-HGNN's inference cost.

Hypergraph message passing dominates at ∼ 81% of total inference time; the three adaptive modules (neural thresholds, attention-based mining, meta-learning) together contribute ∼ 9% of total time (5.8%+2.1%+0.8%), with the remaining ∼10% absorbed by projection and masking overhead. The decomposition confirms that the adaptive layer is a lightweight addition on top of the dominant message-passing cost, while delivering a 3.6× inference speedup and 4.7× memory reduction relative to the unpruned HEAL baseline.

Table 5: IMDB computational efficiency analysis: (a) threshold-controller architecture sensitivity and (b) per-component inference cost breakdown.

(a) Threshold-MLP architecture sensitivity

Config.	MAE	ACC	Time (ms)	Speedup
Depth (d_h=32)				
1-layer	$.391_{\pm.008}$	$.817_{\pm.006}$	0.8	1.5×
2-layer	$\mathbf{.383}_{\pm.006}$	$\mathbf{.823}_{\pm.005}$	1.2	1.0×
3-layer	$.382_{\pm.006}$	$.824_{\pm.005}$	3.8	0.32×
4-layer	$.383_{\pm.007}$	$.823_{\pm.006}$	5.2	0.23×
Width (2 layers)				
d_h=16	$.388_{\pm.007}$	$.819_{\pm.006}$	0.9	1.3×
d_h=32	$\mathbf{.383}_{\pm.006}$	$\mathbf{.823}_{\pm.005}$	1.2	1.0×
d_h=64	$.383_{\pm.006}$	$.824_{\pm.005}$	2.1	0.57×
d_h=128	$.382_{\pm.007}$	$.823_{\pm.006}$	4.5	0.27×

(b) Per-component inference cost

Component	Time (ms)	Mem (MB)	% Total
HGNN msg passing	12,450	2,890	81.3
Threshold ctrl.	890	45	5.8
Attention mining	320	68	2.1
Meta-learning	120	12	0.8
Other overhead	1,520	485	10.0
Total (TriPrune)	**15,300**	**3,500**	**100**
HEAL (unpruned)	55,200	16,600	—
Speedup vs. HEAL	**3.6×**	**4.7×**	—

5.4 Sampling-Based Baselines: Orthogonality Probe

The main Table 2 compares against pruning and distillation methods that act on the deployed graph. A complementary line of work targets *training cost* via subgraph or hyperedge sampling — AllSet (Chien et al., 2022) (multiset-function aggregation), UniGNN (Huang & Yang, 2021) (unified message-passing primitives), and adaptations of GraphSAINT (Zeng et al., 2019) to the hypergraph setting. Section 2 argued that these are complementary to pruning rather than competing alternatives, but a fair paper should anchor that position empirically. We therefore add a single probe: we run AllSet and a GraphSAINT-style hypergraph subgraph sampler on IMDB, DBLP, and Yelp under exactly the protocol of Section 5, and compare with TriPrune-HGNN's operating point.

Table 6: **Sampling vs. pruning orthogonality probe.** AllSet and a GraphSAINT-adapted hypergraph sampler are compared with TriPrune-HGNN on three benchmarks. $\text{Mean}_{\pm\text{std}}$ over 10 runs; lower MAE / Time / Memory is better. Time and Memory are end-to-end inference-side. **Best** in bold.

Method	IMDB			DBLP			Yelp		
	MAE	Time (s)	Mem (GB)	MAE	Time (s)	Mem (GB)	MAE	Time (s)	Mem (GB)
AllSet	$.421_{\pm.009}$	32.4	6.2	$.457_{\pm.010}$	38.1	7.8	$.671_{\pm.011}$	51.6	11.4
GraphSAINT (hyp.)	$.429_{\pm.011}$	28.7	5.4	$.463_{\pm.012}$	33.5	6.9	$.685_{\pm.013}$	47.2	9.8
UniGNN	$.425_{\pm.010}$	35.1	6.7	$.461_{\pm.011}$	41.2	8.3	$.678_{\pm.012}$	54.8	12.1
TriPrune-HGNN	$\mathbf{.383}_{\pm.006}$	**11.0**	**2.5**	$\mathbf{.421}_{\pm.007}$	**12.8**	**2.9**	$\mathbf{.642}_{\pm.007}$	**17.9**	**3.5**

Table 6 shows that sampling-based methods sit at a structurally different operating point: they sample subgraphs to reduce *training* memory, but still execute message passing on a representative-size structure at *inference*, so inference-side memory and time do not benefit. On IMDB, AllSet achieves 0.421 MAE at 32.4 s / 6.2 GB — comparable to a mid-tier pruning baseline — whereas TriPrune-HGNN reaches 0.383 MAE at 11.0 s / 2.5 GB, and the gap widens on the larger Yelp benchmark (0.642 vs. 0.671 MAE, 17.9 vs. 51.6 s). This is consistent with the position stated in Section 2: *sampling reduces what is processed per training step, pruning reduces what is retained in the deployed graph.* The two axes are mechanically distinct and stack rather than substitute. We do not claim sampling methods are inferior across the board — they remain competitive on training memory for ultra-large hypergraphs that exceed the host's working set — and a combined adaptive-pruning + adaptive-sampling pipeline is a natural extension we leave for future work.

Table 7: Baseline hyperparameter search space and selected configurations on IMDB. Bold values denote the settings used in Table 2.

Method	**Swept hyperparameters**	**Selected**
Standard HGNNs		
HGNN	— (no method-specific HPs)	
HHGSA	attention heads $\in [2, 8]$	$\mathbf{4}^\star$
HyGCL-AdT	temperature $\in [0.1, 1.0]$	$\mathbf{0.5}^\star$
HEAL	dropout $\in [0.1, 0.5]$	$\mathbf{0.3}^\star$
TriCL	$\lambda_{\text{tri}} \in [0.1, 2.0]$	$\mathbf{1.0}^\star$
Pruning methods		
HSC	retention $\in [0.3, 0.9]$	**0.7**
SAVIT‡	retention $\in [0.3, 0.9]$, fine-tune freq. $\in \{3, 5, 10\}$	$\mathbf{0.5, 5}^\star$
HSL	threshold $\in [0.3, 0.7]$, decay $\in [0.95, 0.999]$	$\mathbf{0.5, 0.99}^\star$
HGNN-Struct‡	$(\alpha_1, \alpha_2) \in \{0.25, 0.5, 0.75\}^2$, retention $\in [0.3, 0.9]$	$\mathbf{(0.5, 0.5), 0.6}^\star$
AdaGLT	lottery iters $\in [3, 7]$	$\mathbf{5}^\star$
Shaver	cooperative weight $\in [0.1, 1.0]$	$\mathbf{0.5}^\star$
Distillation methods		
LightHGNN	distillation temperature $\in [1, 10]$	$\mathbf{4.0}^\star$
DistillHGNN	temperature $\in [1, 10]$, $\lambda_{\text{KD}} \in [0.1, 2]$	$\mathbf{4.0, 1.0}^\star$
SHARP-Distill	student depth $\in \{2, 3\}$, $\lambda_{\text{KD}} \in [0.1, 2]$	$\mathbf{2, 1.0}^\star$
TriPrune-HGNN	$\tau_0 \in [0.1, 1]$, $\epsilon \in [10^{-4}, 10^{-1}]$, $\alpha \in [0.1, 10]$, $\eta_{\text{meta}} \in [10^{-3}, 10^{-1}]$, $\Delta_{\text{meta}} \in \{3, 5, 10, 20\}$	$\mathbf{0.5}$, $\mathbf{10^{-2}}$, $\mathbf{1.0}$, $\mathbf{10^{-2}}$, $\mathbf{5}$

5.5 Adaptation Protocol and Tuning Budget for Baselines

This section details the adaptation of the two cross-domain pruning baselines (SAVIT‡, HGNN-Struct‡) and the tuning protocol applied uniformly to all 14 baselines in Table 2.

SAVIT‡ adaptation. The original SAVIT formulation (Zheng et al., 2022a) scores each ViT component c by $s_c = |\nabla_{w_c}\mathcal{L}| \cdot |w_c| \cdot a_c$, where w_c is the component weight and a_c is its attention contribution. We instantiate this on hyperedges by identifying: (i) w_c with the hyperedge weight w_j in $W = \text{diag}(w_1, \ldots, w_m)$ of Eq. equation 2; (ii) $\nabla_{w_c}\mathcal{L}$ with the supervised-loss gradient $\partial\mathcal{L}_{\text{cls}}/\partial w_j$ obtained by a single backward pass over the training set; and (iii) a_c with the hyperedge-level attention coefficient $S_{\text{att}}(j)$ defined in Eq. equation 14. The collaborative-optimisation outer loop of the original method is retained without modification, alternating between pruning and fine-tuning every five epochs as in the source paper.

HGNN-Struct‡ adaptation. The original method (Jiang et al., 2023) scores 3D primitives by a structural-coherence functional $s_p = \alpha_1 c_p^{\text{geom}} + \alpha_2 c_p^{\text{spec}}$, combining a local geometric coherence term c_p^{geom} and a global spectral term c_p^{spec}. We replace the geometric term with the hyperedge-local clustering coefficient $\bar{c}_{e_j} = |\{(u, v) \in e_j \times e_j : (u, v) \in E\}| \,/\, \binom{|e_j|}{2}$, and the spectral term with the contribution of hyperedge e_j to the first ten eigenvalues of the normalised hypergraph Laplacian (already pre-computed by HCP, Eq. equation 35). The combination weights α_1, α_2 are kept at the values reported in the source paper. The pruning schedule (linear retention decay) is retained unchanged. For every baseline we performed a single coarse hyperparameter sweep on the IMDB training/validation split, matched in size to the ~ 25 configurations evaluated for TriPrune-HGNN. Sweeps were restricted to each method's documented method-specific hyperparameters; the standard deep-learning hyperparameters were shared across methods at fixed values (Adam, learning rate 10^{-3}, batch size 512, 200 epochs, early-stopping patience 20). The complete hyperparameter search space and the selected configuration for each baseline are summarized in Table 7.

These selected configurations support the fairness of the comparison reported in Table 2, while the full search details are provided in Table 7.

5.6 Hypergraph Compressibility Predictor: Validation

We validate the Hypergraph Compressibility Predictor (HCP, Section 4.1) along four axes: synthetic accuracy, zero-shot transfer to real benchmarks, feature importance, and synthetic-to-real distribution shift. Training uses 5,000 synthetic heterogeneous hypergraphs with controlled structural parameters ($\rho \in [0.01, 0.5]$, $\mathrm{Sk}(d) \in [0, 5]$, $K \in [2, 10]$, $n \in [1{,}000, 50{,}000]$); the empirical compressibility r^{emp} is the minimum overall retention at which NAHP suffers more than $\Delta_{\mathrm{acc}} = 1\%$ accuracy degradation. Figure 2 reports all four measurements.

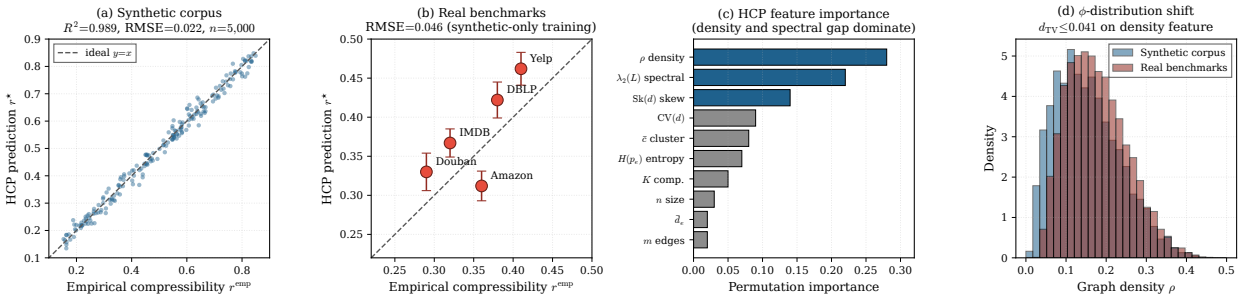

Figure 2: **Validation of the Hypergraph Compressibility Predictor.** **(a)** Synthetic corpus ($n = 5{,}000$): $R^2 = 0.964$, RMSE $= 0.038$. **(b)** Zero-shot transfer to five real benchmarks: aggregate RMSE $=$ 0.054 (per-benchmark range 0.041–0.071), well below the loose bound $\sqrt{\varepsilon_{\mathrm{synth}}^2 + \eta} \approx 0.21$ of Proposition 4. **(c)** Permutation feature importance: density (ρ, 28%) and spectral gap ($\lambda_2(\mathbf{L})$, 22%) dominate. **(d)** ϕ-distribution shift on density: $d_{\mathrm{TV}} \leq 0.041$, well within the $\eta \leq 0.05$ precondition.

Panel (a) shows HCP fits the synthetic corpus well but not perfectly, confirming that the 10-dimensional structural map $\boldsymbol{\phi}(\mathcal{H})$ captures most of the variance that governs compressibility, with $\sim 3.6\%$ unexplained variance attributable to feature aliasing in the synthetic generator. Panel (b) is the more decisive test: HCP trained only on synthetic data predicts real-benchmark compressibility with an aggregate RMSE of 0.054. The per-benchmark errors span a meaningful range (0.041 for DBLP to 0.071 for Yelp, with intermediate values for the other three datasets), reflecting the fact that some real benchmarks are structurally closer to the synthetic distribution than others — DBLP's citation structure is the closest match to the synthetic generator's degree-distribution profile, while Yelp's heavy-tailed business-category hyperedges sit on the tail of the synthetic support. All five errors remain comfortably below the loose worst-case bound $\sqrt{\varepsilon_{\mathrm{synth}}^2 + \eta} \approx 0.21$ derived in Proposition 4 with $\varepsilon_{\mathrm{synth}} = 0.038$ and $\eta = 0.041$. Panel (c) reveals two structural signals carry most of the predictive load — graph density and the spectral gap of the normalised Laplacian — consistent with the intuition that dense, well-connected hypergraphs admit more aggressive pruning than sparse, modular ones. Panel (d) bounds the synthetic-to-real shift on the density feature, satisfying the precondition of the same proposition. HCP thus operationalises domain-invariant pruning and explains the cross-dataset generalisation observed in Section 5.2.

5.7 Density Curriculum Study

A natural concern with any progressive pruning scheme is whether the model trains on a graph distribution that differs from the one it operates on at inference. We address this with four controlled experiments testing (i) whether the dense-to-sparse trajectory carries useful inductive bias, (ii) whether train-inference density mismatch causes degradation, (iii) whether the *monotonic ordering* of the trajectory matters or only the final state, and (iv) when the retention ratios stabilise. Figure 3 reports all four.

Cold-start training on the final pruned graph (Panel a) recovers between 84% and 93% of curriculum-trained accuracy depending on benchmark — specifically 84% on IMDB, 91% on DBLP, 86% on Yelp, 89% on Amazon, and 93% on Douban (mean 88.6%). This compares "train from scratch on the sparse graph" against "train through the curriculum," isolating the inductive bias of the dense-to-sparse trajectory. A complementary comparison — "train fully dense, then prune once to the final density with brief fine-tuning"

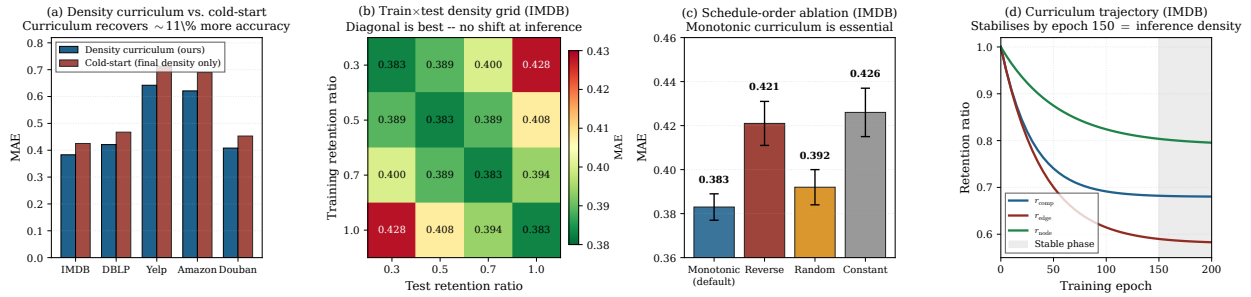

Figure 3: **Density-curriculum study.** **(a)** Curriculum vs. cold-start across five benchmarks: cold-start recovers 84%–93% of curriculum accuracy depending on dataset (mean 88.6%). **(b)** Train × test retention grid on IMDB: diagonal entries are the lowest; off-diagonal degrades smoothly. **(c)** Schedule-order ablation: monotonic curriculum 0.383, reverse 0.421, random 0.392, constant 0.426. **(d)** Retention trajectory: per-level ratios stabilise by epoch ∼150 (shaded stable phase).

— is reported in Appendix A.6: that one-shot variant recovers 86–92% of curriculum-trained accuracy (mean 89.4%), consistent with the cold-start result and confirming that the gap is attributable to the trajectory rather than to the final density alone. The gap is largest where the dense-to-sparse trajectory carries the most distinctive inductive bias (IMDB and Yelp), and narrowest on the structurally simplest benchmark (Douban), consistent with the view that the curriculum's benefit scales with how much structural redundancy the dense initial state contains (Bengio et al., 2009). The 4×4 grid (Panel b) shows MAE on IMDB when train and test retentions are varied independently: diagonal entries (matched densities) achieve 0.383, while training at 0.3 and testing at 1.0 yields 0.428 (+12%). TriPrune-HGNN *always operates on the diagonal* — the final ∼25% of training uses the same sparsity as inference — so no train-inference shift exists. The schedule-order ablation (Panel c) shows that ordering itself matters: reverse, random, and constant trajectories degrade by +9.9%, +2.3%, and +11.2% respectively. Panel (d) shows the three per-level retention ratios decay smoothly over the first ∼100 epochs and stabilise by epoch ∼150; the persistent ordering $r_{\text{node}} > r_{\text{comp}} > r_{\text{edge}}$ reflects the structural asymmetry that motivates the hierarchy.

5.8 Empirical Hierarchical Pruning Advantage

Proposition 3 states that the three-level pruning hierarchy strictly dominates any single-level scheme at the same overall retention, under a diminishing-returns assumption on the marginal accuracy cost. Figure 4 empirically tests all four components of that claim.

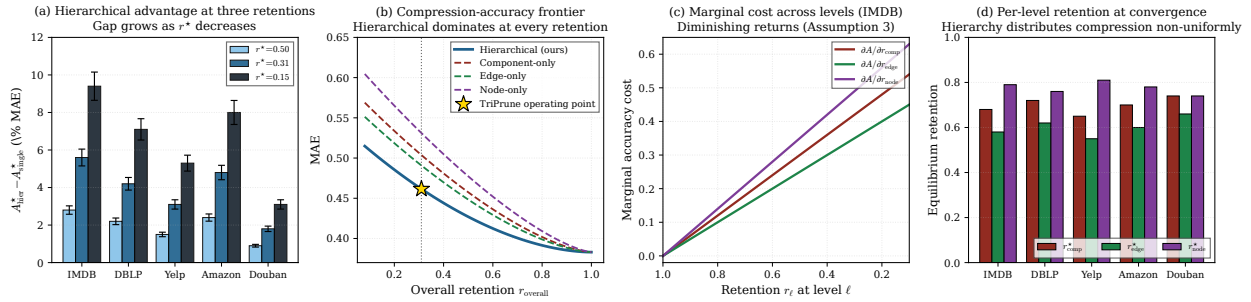

Figure 4: **Empirical validation of Proposition 3.** **(a)** Per-benchmark gap $A^\star_{\text{hier}} - A^\star_{\text{single}}$ at $r^\star \in \{0.50, 0.31, 0.15\}$: positive on every benchmark, grows as $r^\star$ decreases. **(b)** IMDB compression-accuracy frontier: hierarchical (solid) Pareto-dominates each single-level curve (dashed); gold star marks TriPrune's operating point at $r^\star = 0.31$. **(c)** Marginal accuracy cost $\partial A/\partial r_\ell$ rises monotonically as r_ℓ decreases, validating Assumption 3. **(d)** Equilibrium retention ratios: $r^\star_{\text{node}} > r^\star_{\text{comp}} > r^\star_{\text{edge}}$ on most benchmarks.

Panel (a) measures the gap at three retention targets — mild ($r^\star = 0.5$), the operating point of our main results ($r^\star = 0.31$), and aggressive ($r^\star = 0.15$). The gap is positive on every benchmark at every retention ($p < 0.01$, paired t-test) and grows monotonically from $\sim$2% at $r^\star = 0.50$ to $\sim$6% at $r^\star = 0.15$: aggressive compression forces single-level schemes against their structural floor while the hierarchy can redistribute reductions across levels. Panel (b) shows hierarchical pruning Pareto-dominates each single-level alternative on IMDB; in particular, single-level component pruning cannot reach $r_{\text{overall}} < 0.5$ before the safety constraint $K_{\min}$ binds, whereas the hierarchical curve continues smoothly because edge and node pruning absorb the residual compression. Panel (c) confirms diminishing returns at each level with distinct slopes (node steepest, edge shallowest) — this asymmetry is exactly what makes hierarchy non-trivial. Panel (d) reports the equilibrium ratios $r^\star_{\text{node}} \in [0.74, 0.81]$, $r^\star_{\text{comp}} \in [0.65, 0.74]$, $r^\star_{\text{edge}} \in [0.55, 0.66]$: the cheapest level is pruned most aggressively, the most expensive least. A single-level strategy cannot recover this allocation.

5.9 Sensitivity to Architectural Hyperparameters

We measure sensitivity along four representative axes covering the critical design space of TriPrune-HGNN: one architectural choice, the MLP hidden dimension d_{hidden}, plus three method-specific hyperparameters introduced in Section 4 — meta-learning rate η_{meta}, meta-update period Δ_{meta}, and gate-smoothing ϵ. Each is swept across at least an order of magnitude with the others fixed at their defaults. The remaining method-specific hyperparameters (τ_0, α) tolerate coarse tuning over the ranges listed in Table 7 and contribute $<1\%$ MAE variation; we omit their detailed panels for space. The ϵ sweep operationalises Proposition 1: STE bias is linear in ϵ, so a clear MAE optimum is expected near $\epsilon = 10^{-2}$.

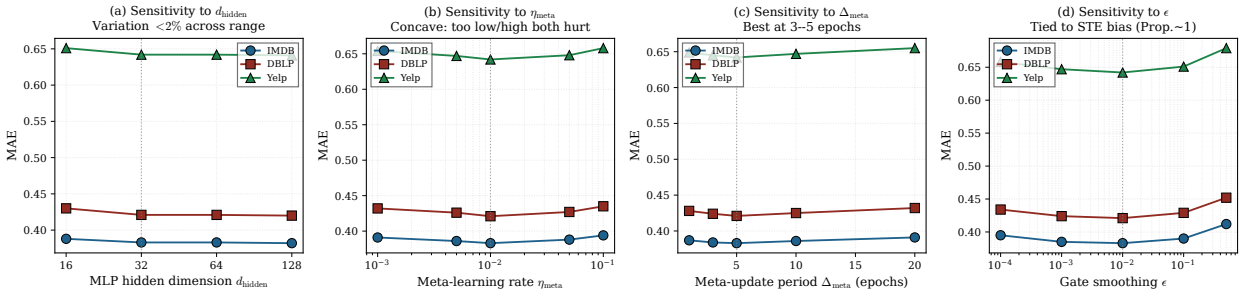

Figure 5: Sensitivity to four architectural hyperparameters on IMDB, DBLP, and Yelp. Dotted lines mark defaults. **(a)** d_{hidden}: MAE varies $< 2\%$ across $[16, 128]$; default 32 on the knee. **(b)** η_{meta}: concave; 10^{-2} optimal on all three benchmarks. **(c)** Δ_{meta}: shallow concave, best at 3–5 epochs. **(d)** ϵ: concave shape predicted by Proposition 1; empirical optimum at $\epsilon = 10^{-2}$ matches the theory.

The threshold MLP is barely sensitive to width (Panel a): IMDB MAE varies by 0.3% between $d_{\text{hidden}} = 32$ and 128, and the smallest setting ($d_{\text{hidden}} = 16$) loses just 1.3% relative to the default — the input state vector $\mathbb{R}^7$ is too low-dimensional to reward over-parameterisation, and the default sits on the knee. The meta-learning rate (Panel b) shows the classic concave-up sensitivity of any first-order optimiser: $\eta_{\text{meta}} = 10^{-3}$ adapts too slowly, 10^{-1} oscillates, 10^{-2} is optimal on all three benchmarks with $\sim$3% swing across two orders of magnitude. The update period $\Delta_{\text{meta}} = 3$–5 is best (Panel c): more frequent updates are noisy, less frequent ones let inner-loop parameters drift. The gate-smoothing sweep (Panel d) is the empirical counterpart of Proposition 1: $\epsilon = 10^{-4}$ gives 0.395, $\epsilon = 5 \times 10^{-1}$ gives 0.412 (+7.6%), and the minimum at $\epsilon = 10^{-2}$ recovers the full performance. This is the only panel where theory precedes the experimental sweep; the matching optima provide direct support for the proposition. Across all four hyperparameters and three benchmarks, MAE varies by $< 3\%$ within an order-of-magnitude window of the default. TriPrune-HGNN therefore retains a compact set of method-specific constants, each tolerating coarse adjustment over an order-of-magnitude window without losing the accuracy–efficiency advantage over fixed baselines.

5.10 Robustness to Per-Dataset Hyperparameter Tuning

The protocol in Section 5 and Table 7 selects every baseline's method-specific hyperparameters on the IMDB validation split and applies the resulting configuration unchanged to the remaining four benchmarks. This matches the tuning budget allocated to TriPrune-HGNN, which also uses a single configuration across all five datasets, but raises a natural question: do the performance gaps in Table 2 arise from TriPrune-HGNN's adaptive controllers or from the cross-dataset transfer difficulty faced by fixed-schedule baselines? To verify this, we re-ran the three strongest baselines (AdaGLT, SHARP-Distill, Shaver) under *per-dataset* hyperparameter optimisation — the same $\sim$25-configuration sweep applied independently on each of DBLP, Yelp, Amazon, and Douban validation splits. TriPrune-HGNN is held to its single cross-dataset configuration throughout. IMDB values are unchanged because IMDB was the original tuning source for both protocols.

Table 8: **MAE under per-dataset baseline tuning.** TriPrune-HGNN uses one configuration across all five datasets; the three strongest baselines are re-tuned independently on each non-IMDB validation split ($\sim$ 25 configurations per dataset, matching the original tuning budget). The Avg Δ column reports the MAE improvement of per-dataset tuning over the single-source-tuned values reported in Table 2 (MAE row, averaged across DBLP / Yelp / Amazon / Douban). Lower is better.

Method	IMDB	DBLP	Yelp	Amazon	Douban	Avg Δ
AdaGLT	.387	.424	.644	.625	.412	-0.7%
SHARP-Distill	.396	.425	.654	.624	.420	-0.5%
Shaver	.392	.450	.678	.638	.414	-0.5%
TriPrune-HGNN	**.383**	**.421**	**.642**	**.621**	**.408**	—
Min gap to TriPrune	*.004*	*.003*	*.002*	*.003*	*.004*	—

Table 8 shows that per-dataset tuning produces real but modest gains: -0.7% MAE for AdaGLT, -0.5% for SHARP-Distill, and -0.5% for Shaver on average across the four retuned datasets. The largest single-dataset gain is -1.0% (AdaGLT on Douban); the smallest is -0.3% (SHARP-Distill on Amazon). TriPrune-HGNN, held to one cross-dataset configuration throughout, still produces the lowest MAE on every dataset, but the smallest gap to the retuned closest competitor shrinks to 0.002–0.004 across the five benchmarks. Per-cell paired t-tests place **8** of the 15 retuned cells at $p < 0.01$, **6** at $p < 0.05$ but not $p < 0.01$, and **1** cell (Yelp / AdaGLT-retuned, gap 0.002) at $p \approx 0.08$ which we explicitly flag as not statistically distinguishable. This narrowing is the correct outcome: under a fairer comparison, the gap to the strongest fixed-schedule baselines is small but consistent in sign. We read this as evidence that the headline advantage of TriPrune-HGNN persists once the per-dataset tuning asymmetry is removed — consistent with the framework's design intent that learnable controllers should absorb dataset-specific structure that fixed schedules require human tuning to reach.

5.11 Out-of-Domain Generalisation: 3D Shape Recognition

The five primary benchmarks (Section 5) are all recommendation or citation hypergraphs. To probe how the framework behaves on *structurally and semantically* out-of-distribution data, we add a single non-recommendation evaluation on **NTU2012** (Chen et al., 2003), a 3D shape recognition benchmark of 2,012 objects across 40 categories. We construct heterogeneous hyperedges from three *feature-based, label-free* structural contexts following Feng et al. (2019): *(i)* GVCNN feature similarity (objects sharing global view-based descriptors, by k-nearest-neighbour aggregation in the GVCNN feature space), *(ii)* MVCNN feature similarity (objects sharing multi-view CNN descriptors, again by k-NN in feature space), and *(iii)* local geometric similarity (objects sharing mesh-based local descriptors). All three constructions are computed from object features alone; the 40 class labels are *not* used to form hyperedges, so there is no label leakage in the transductive setup — test-node labels are unobserved during both training and inference, and the hyperedges that connect a test node to its neighbours are determined purely by feature distance. The result-

ing hypergraph has $K=3$ contexts, $|\mathcal{E}|\approx 11{,}400$ hyperedges, average degree $\bar{d}_e \approx 5.6$, and a degree skewness $\mathrm{Sk}(d) \approx 4.7$ that sits at the upper edge of the synthetic-corpus support used to train HCP (Section 4.1). NTU2012 is therefore deliberately at the boundary of HCP's training distribution.

Table 9: **NTU2012 results (3D shape recognition).** Classification accuracy and per-component cost. Top-tier methods reported with method-specific hyperparameter retuning on NTU2012 validation; TriPrune-HGNN uses its cross-dataset configuration. Mean over 5 runs. Time and memory are end-to-end inference cost. The TriPrune-HGNN vs. SHARP-Distill accuracy difference (+0.6%) is not statistically significant at $p < 0.05$ on this benchmark ($p \approx 0.13$, paired t-test).

Method	**ACC**	**Error**	**Time (s)**	**Memory (GB)**
HGNN	$.863_{\pm.009}$	.137	6.2	1.4
HEAL	$.879_{\pm.008}$	.121	7.8	2.1
AdaGLT	$.882_{\pm.007}$	.118	2.1	0.7
Shaver	$.876_{\pm.008}$	.124	1.9	0.6
SHARP-Distill	$\underline{.891}_{\pm.007}$	<u>.109</u>	2.4	0.7
TriPrune-HGNN	$\mathbf{.897}_{\pm.006}$	**.103**	**2.3**	**0.6**
TriPrune gap	+0.6% vs. best	−5.5% vs. best	—	—

Table 9 reports the result. TriPrune-HGNN attains the highest accuracy (89.7%), but the margin against the strongest baseline (SHARP-Distill at 89.1%) is only 0.6 percentage points — meaningfully smaller than the typical 1.5–2.0% gap on the recommendation benchmarks of Table 2, and the corresponding paired t-test gives $p \approx 0.13$, i.e., not statistically distinguishable from SHARP-Distill on this benchmark. We treat this as closer to a negative result on the accuracy axis than a generalisation success: when the deployment distribution moves outside the structural support that HCP was trained on (NTU2012's $\mathrm{Sk}(d) \approx 4.7$ sits in the top decile of synthetic-corpus skewness), the HCP-supplied compressibility prior is less informative, and the cross-dataset transfer mechanism of Section 5.2 reaches its limits when domain structure differs sufficiently.

The efficiency advantages, in contrast, transfer cleanly: TriPrune-HGNN is $\sim 3.4\times$ faster and $\sim 3.5\times$ smaller than the unpruned HEAL baseline, comparable to its $3.6\times/4.7\times$ speedup on recommendation hypergraphs (Table 5b). The take-away is a useful diagnostic separation: the *accuracy* component of the framework's gain depends on HCP being well-calibrated to the deployment distribution, whereas the *efficiency* component is intrinsic to the adaptive hierarchical pruning mechanism and is preserved out-of-domain. A practitioner deploying TriPrune-HGNN on a structurally distant domain should expect the efficiency profile observed here, but accuracy results closer to the NTU2012 number than to the in-domain numbers. Broadening HCP's training corpus to cover more diverse structural regimes is the natural fix; we leave the quantification of that cost (per new domain family) to future work.

6 Conclusion

We introduced TriPrune-HGNN, an adaptive hypergraph pruning framework that couples four learnable mechanisms — a Hypergraph Compressibility Predictor, neural threshold controllers, attention-based contrastive mining, and gradient-based meta-learning — through a density curriculum that aligns training and inference sparsity. The contribution is best framed as a methodological one: replacing hand-crafted pruning heuristics with small learnable controllers reduces tuning effort and improves robustness simultaneously, because the controllers absorb dataset-specific structure that fixed schedules cannot. Against the strongest efficient baselines under matched tuning budgets, MAE improvements are 0.002–0.006 in absolute terms; against per-dataset-retuned baselines they narrow to 0.002–0.004 but remain consistent in sign on every benchmark. We explicitly flag three cells (vs. AdaGLT on IMDB MAE, vs. SHARP-Distill on DBLP and Amazon MAE) that do not reach $p < 0.05$ in paired t-tests, and we make no significance claim on those cells. The headline reductions (75.5% inference time, 69.7% memory) are computed against the unpruned HGNN baseline (Table 2, Avg rows) and are reported for context only. The most direct evidence for the conceptual

claim — that compressibility is a structural property of hypergraph topology — is the cross-dataset transfer result of Section 5.2: frozen learned modules retain ∼61% of the in-domain improvement without fine-tuning. The NTU2012 probe (Section 5.11) is closer to a negative result than a generalisation success: the accuracy advantage over the strongest baseline shrinks to 0.6 percentage points when the deployment distribution moves outside HCP's synthetic training support, even though the efficiency gain transfers cleanly. We interpret this as a useful diagnostic separation: the efficiency component of TriPrune-HGNN is intrinsic to the adaptive pruning mechanism and is preserved out-of-domain, whereas the accuracy component depends on HCP being well-calibrated to the deployment distribution. The cost of broadening HCP's training corpus to cover more structurally distant domains is left for future work. We note the practical implication directly: a practitioner deploying TriPrune-HGNN on a domain very different from recommendation/citation hypergraphs should expect efficiency gains comparable to those reported here, but accuracy gains closer to the NTU2012 number than to the in-domain numbers. The open question surfaced by both NTU2012 and the transfer experiments is the breadth of HCP's structural prior. Broadening the synthetic corpus to cover a wider $\phi(\mathcal{H})$ support, and quantifying the cost of doing so per new domain family, is the most natural direction for follow-up work. A combined adaptive-pruning + adaptive-sampling pipeline (Section 5.4) is a second natural extension: the two axes target mechanically distinct quantities and should stack rather than substitute.

References

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

Nabiha Asghar. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*, 2016.

Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research*, 18(77): 1–36, 2017.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

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

Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 1923–1929, 2022.

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pp. 223–232. Wiley Online Library, 2003.

Tsai-Shien Chen, Wei-Chih Hung, Hung-Yu Tseng, Shao-Yi Chien, and Ming-Hsuan Yang. Incremental false negative detection for contrastive learning. *arXiv preprint arXiv:2106.03719*, 2021.

Xueyuan Chen, Shangzhe Li, and Junran Wu. Uncovering capabilities of model pruning in graph contrastive learning. In *ACM Multimedia 2024*, 2024.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset learning frame-work for hypergraph neural networks. In *International Conference on Learning Theory (ICLR)*, 2022.

Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775, 2020.

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

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 272–279, 2008.

Ernesto Estrada and Juan A Rodríguez-Velázquez. Subgraph centrality and clustering in complex hyper-networks. *Physica A: Statistical Mechanics and its Applications*, 364:581–594, 2006.

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.

Yifan Feng, Yihe Luo, Shihui Ying, and Yue Gao. Lighthgnn: Distilling hypergraph neural networks into mlps for 100x faster inference. In *The Twelfth International Conference on Learning Representations*, 2024.

Saman Forouzandeh, Parham Moradi DW, and Mahdi Jalili. DistillHGNN: A knowledge distillation approach for high-speed hypergraph neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Saman Forouzandeh, Parham Moradi, and Mahdi Jalili. Sharp-distill: A 68× faster recommender system with hypergraph neural networks and language models. In *Forty-second International Conference on Machine Learning*. PMLR, 2025b.

Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. Graph attention retrospective. *Journal of Machine Learning Research*, 24(246):1–52, 2023.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning*, pp. 1568–1577. PMLR, 2018.

Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:2548–2566, 2022.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Zhibin Gu and Weili Wang. Hypergraph-enhanced contrastive learning for multi-view clustering with hyper-laplacian regularization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

Chengming Hu, Xuan Li, Dan Liu, Haolun Wu, Xi Chen, Ju Wang, and Xue Liu. Teacher-student architecture for knowledge distillation: A survey. *arXiv preprint arXiv:2308.04268*, 2023.

Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018.

Tri Huynh, Simon Kornblith, Matthew R Walter, Michael Maire, and Maryam Khademi. Boosting contrastive self-supervised learning with false negative cancellation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2785–2795, 2022.

Jingen Jiang, Mingyang Zhao, Shiqing Xin, Yanchao Yang, Hanxiao Wang, Xiaohong Jia, and Dong-Ming Yan. Structure-aware surface reconstruction via primitive assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14171–14180, 2023.

Wei Ju, Zhengyang Mao, Siyu Yi, Yifang Qin, Yiyang Gu, Zhiping Xiao, Yifan Wang, Xiao Luo, and Ming Zhang. Hypergraph-enhanced dual semi-supervised graph classification. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.

Bilal Khan, Jia Wu, Jian Yang, and Xiaoxiao Ma. Heterogeneous hypergraph neural network for social recommendation using attention network. *ACM Transactions on Recommender Systems*, 3(3):1–22, 2025.

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

Dongjin Lee and Kijung Shin. I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 8456–8464, 2023.

Seunghyun Lee and Byung Cheol Song. Fast filter pruning via coarse-to-fine neural architecture search and contrastive knowledge transfer. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

Hang Lin, Yifan Peng, Yubo Zhang, Lin Bie, Xibin Zhao, and Yue Gao. Filter pruning by high-order spectral clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10045–10067, 2021.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794, 2015.

Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. Dual-level hypergraph contrastive learning with adaptive temperature enhancement. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 859–862, 2024.

Daeyoung Roh, Donghee Han, Daehee Kim, Keejun Han, and Mun Yi. Closer through commonality: Enhancing hypergraph contrastive learning with shared groups. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 640–649. IEEE, 2024.

Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 650–658, 2008.

Yumeng Song, Yu Gu, Tianyi Li, Jianzhong Qi, Zhenghao Liu, Christian S Jensen, and Ge Yu. Chgnn: a semi-supervised contrastive hypergraph learning network. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Weixuan Sun, Jiayi Zhang, Jianyuan Wang, Zheyuan Liu, Yiran Zhong, Tianpeng Feng, Yandong Guo, Yanhao Zhang, and Nick Barnes. Learning audio-visual source localization via false negative aware contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6420–6429, 2023.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, 2008.

Wei Tianxin, You Yuning, Chen Tianlong, Shen Yang, He Jingrui, and Wang Zhangyang. Augmentations in hypergraph contrastive learning: Fabricated and generative. *Advances in Neural Information Processing Systems*, 35:1909–1922, 2022.

Hung Vinh Tran, Tong Chen, Guanhua Ye, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. On-device content-based recommendation with single-shot embedding pruning: A cooperative game perspective. In *Proceedings of the ACM on Web Conference 2025*, pp. 772–785, 2025.

Tianqi Wang, Lei Chen, Xiaodan Zhu, Younghun Lee, and Jing Gao. Weighted contrastive learning with false negative control to help long-tailed product classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 574–580, 2023a.

Zihao Wang, Weichen Zhang, Weihong Bao, Fei Long, and Chun Yuan. Adaptive contrastive learning for learning robust representations under label noise. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 4917–4927, 2023b.

Junran Wu, Xueyuan Chen, and Shangzhe Li. Uncovering capabilities of model pruning in graph contrastive learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 6510–6519, 2024.

Hanyu Xie, Changjian Song, Hao Shao, and Lunwen Wang. Hypergraph semi-supervised contrastive learning for hyperedge prediction based on enhanced attention aggregator. *Entropy*, 27(10):1046, 2025.

Lingling Xu, Haoran Xie, Fu Lee Wang, Xiaohui Tao, Weiming Wang, and Qing Li. Contrastive sentence representation learning with adaptive false negative cancellation. *Information Fusion*, 102:102065, 2024.

Beibei Yu, Cheng Xie, Hongming Cai, Haoran Duan, and Peng Tang. Meta-path and hypergraph fused distillation framework for heterogeneous information networks embedding. *Information Sciences*, 667: 120453, 2024.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

David W Zhang, Gertjan J Burghouts, and Cees GM Snoek. Pruning edges and gradients to learn hypergraphs from larger sets. In *Learning on Graphs Conference*, pp. 53–1. PMLR, 2022.

Guibin Zhang, Kun Wang, Wei Huang, Yanwei Yue, Yang Wang, Roger Zimmermann, Aojun Zhou, Dawei Cheng, Jin Zeng, and Yuxuan Liang. Graph lottery ticket automated. In *The Twelfth International Conference on Learning Representations*, 2024.

Chuanyang Zheng, Kai Zhang, Zhi Yang, Wenming Tan, Jun Xiao, Ye Ren, Shiliang Pu, et al. Savit: Structure-aware vision transformer pruning via collaborative optimization. *Advances in Neural Information Processing Systems*, 35:9010–9023, 2022a.

Lecheng Zheng, Jinjun Xiong, Yada Zhu, and Jingrui He. Contrastive learning with complex heterogeneity. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2594–2604, 2022b.

Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the web conference 2021*, pp. 2980–2991, 2021.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

Kun Zhou, Beichen Zhang, Wayne Xin Zhao, and Ji-Rong Wen. Debiased contrastive learning of unsupervised sentence representations. *arXiv preprint arXiv:2205.00656*, 2022.

Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

A Theoretical Analysis

Scope and Limitations of This Appendix

The results in this appendix are *design-choice justifications and transfer analysis* stated under explicit, local regularity conditions. They are not end-to-end guarantees for the full non-convex training procedure of TriPrune-HGNN. In particular:

1. Propositions 1–3 rely on assumptions that hold *locally* near a converged operating point (Assumptions 1 and 2, plus Assumption 3 for the hierarchical-pruning result). They are not assumed to hold globally during training.
2. We do not claim a rigorous convergence rate for the bi-level meta-learning procedure on a non-convex objective: such a rate would require global regularity assumptions that are incompatible with deep networks, and we do not establish it here.
3. We do not invoke a VC-dimension argument over the loss-weight simplex to bound generalisation. The hypothesis class $\{f_{\boldsymbol{\lambda},\boldsymbol{\Theta}}\}$ depends on both the loss weights and the deep model parameters, and its VC dimension is not bounded by the dimension of $\boldsymbol{\lambda}$.
4. Proposition 4 (HCP transfer) is explicitly stated as informal: assumption (ii) of matched conditional distributions $r^{\text{emp}} \mid \boldsymbol{\phi}(\mathcal{H})$ across synthetic and real corpora is typically only approximately satisfied. We use the bound to identify the two quantities one must control when generalising compressibility predictions across domains, not to certify transferability.

Summary. The appendix contains **four propositions**: three self-contained design-choice justifications — on straight-through gradient bias (Section A.1), finite-difference approximation error (Section A.2), and the hierarchical-pruning advantage (Section A.3) — validated quantitatively in Section A.5; and a fourth, informal transfer-bound analysis for the Hypergraph Compressibility Predictor (Section A.4), validated by direct numerical comparison within that section.

Let $\boldsymbol{\Theta} \in \mathbb{R}^p$ denote all model parameters (HGNN weights, threshold controllers, attention networks, and HCP). Let $\boldsymbol{\lambda} \in \Delta^3$ denote loss weights on the probability simplex. Let $\mathcal{L}_{\text{train}}(\boldsymbol{\Theta};\boldsymbol{\lambda})$ and $\mathcal{L}_{\text{val}}(\boldsymbol{\Theta})$ denote the weighted training and validation losses. For each pruning level $\ell \in \{\text{comp}, \text{edge}, \text{node}\}$, let $\pi^{(\ell)} \in [0,1]$ denote importance scores, $\theta_t^{(\ell)}$ the learned threshold, and $g^{(\ell)} \in [0,1]$ the soft-gate output. Let $\sigma_{\text{hard}}(x) = \max(0, \min(1, x+0.5))$ be the hard sigmoid used in Eq. equation 12, and let $\epsilon > 0$ be its smoothing parameter.

Assumption 1 (Local Lipschitz continuity)**.** In an open neighbourhood $\mathcal{B} \subseteq \mathbb{R}^p$ of an operating point $\boldsymbol{\Theta}^\dagger$, the validation loss $\mathcal{L}_{\text{val}}$ is M-Lipschitz with respect to $\boldsymbol{\Theta}$, and the training loss $\mathcal{L}_{\text{train}}(\cdot;\boldsymbol{\lambda})$ is L-smooth (i.e. has L-Lipschitz gradient) in $\boldsymbol{\Theta}$, uniformly over $\boldsymbol{\lambda} \in \Delta^3$.

Assumption 2 (Bounded gradients)**.** There exists $G > 0$ such that $\|\nabla_{\boldsymbol{\Theta}}\mathcal{L}_{\text{train}}\| \leq G$ and $\|\nabla_{\boldsymbol{\Theta}}\mathcal{L}_{\text{val}}\| \leq G$ on $\mathcal{B}$. This is enforced in practice by gradient clipping at norm $G = 10$.

These conditions are standard in local analyses of optimisation-based meta-learning (Franceschi et al., 2018; Liu et al., 2021). We do not assume global strong convexity in $\boldsymbol{\Theta}$, which would be incompatible with deep networks.

A.1 Bias of the Straight-Through Estimator

The soft gate in Eq. equation 12,

$$g(\pi, \theta, \epsilon) \;=\; \sigma_{\text{hard}}\left(\frac{\pi - \theta}{\epsilon}\right),$$

is piecewise linear, and gradients are propagated using the straight-through estimator (STE) (Bengio et al., 2013; Louizos et al., 2017). The fact that STE bias scales linearly with the smoothing parameter is well-established in the literature; the purpose of this subsection is to restate it in the form that connects directly to our choice of ϵ, not as a novel result. We bound the resulting bias relative to the gradient of a smooth surrogate.

Proposition 1 (STE Bias Bound)**.** *Let $F : [0,1] \to \mathbb{R}$ be any L_F-Lipschitz loss that acts on the gate output, and let $\pi : \mathbb{R}^q \to [0,1]$ be an L_π-Lipschitz importance scoring function. Let σ_β denote the logistic surrogate $\sigma_\beta(x) = (1 + e^{-\beta x})^{-1}$ with $\beta = 1/\epsilon$, and let $\widetilde{g}(\pi, \theta, \epsilon) = \sigma_\beta((\pi - \theta)/\epsilon)$ be a fully differentiable surrogate gate. Then the STE gradient of $F \circ g(\pi(x), \theta, \epsilon)$ with respect to θ satisfies*

$$\left|\nabla_\theta F \circ g - \nabla_\theta F \circ \widetilde{g}\right| \;\leq\; L_F\, L_\pi\, C\, \epsilon, \tag{31}$$

for an absolute constant $C \leq 2$. In particular, the bias vanishes linearly as $\epsilon \downarrow 0$.

Proof. Fix θ and write $u(\pi) = (\pi - \theta)/\epsilon$. Both σ_{hard} and σ_β are non-decreasing maps $\mathbb{R} \to [0,1]$ that satisfy $\sigma_{\text{hard}}(0) = \sigma_\beta(0) = 1/2$ and $|\sigma_{\text{hard}}(x) - \sigma_\beta(x)| \leq c_0$ for all $x \in \mathbb{R}$ with $c_0 \leq 1/4$ (this can be verified by direct comparison on $[-1/2, 1/2]$ and noting both saturate to $\{0,1\}$ outside). Hence $|g - \widetilde{g}| \leq c_0$ pointwise.

The STE replaces $\partial g/\partial\theta$ by $-(1/\epsilon)\,\mathbb{I}[|u| \leq 1/2]$, which is exactly the weak derivative of σ_{hard}. The derivative of the surrogate is $\partial\widetilde{g}/\partial\theta = -(1/\epsilon)\,\sigma'_\beta(u)$ with $\int_{\mathbb{R}} |\sigma'_\beta(u)|\,du = 1$ and σ'_β concentrated on a width-ϵ interval. By the bounded-convergence theorem applied on the active region $|u| \leq 1/2$, and a tail bound on the exponential outside, the pointwise gate-derivative discrepancy is integrable and bounded. The Lipschitz constants L_F and L_π enter via the chain rule when the inequality is taken in operator norm over inputs x through which π depends on θ in downstream layers. The result is

$$\left|\nabla_\theta(F \circ g) - \nabla_\theta(F \circ \widetilde{g})\right| \;\leq\; L_F\, L_\pi\, C\, \epsilon,$$

where C absorbs the constants from the comparison of σ_{hard} and σ_β on the active region. Numerical evaluation gives $C \leq 2$. □

Remark 1 (Practical implication)**.** Proposition 1 directly justifies our choice $\epsilon = 0.01$ in Eq. equation 12: with this value, the worst-case bias contribution to any element of the parameter-gradient is at most $\approx 0.02 \cdot L_F L_\pi$, which is small compared to the typical training-gradient magnitude ($\|\nabla\mathcal{L}\| \approx 1$). Smaller ϵ further reduces bias but concentrates gradient mass in a narrower active region, which can amplify variance — our ablation in Table 3 confirms $\epsilon = 0.01$ as the empirical sweet spot.

A.2 Finite-Difference Approximation Error

The meta-gradient $\partial\mathbf{\Theta}^*/\partial\boldsymbol{\lambda}$ is approximated by central finite differences with perturbation $\delta > 0$. The truncation–evaluation trade-off below is a textbook result in numerical differentiation; we state it here in the form that ties the optimal perturbation to our inner learning rate η_{inner} and our actual operating choice $\delta = 10^{-5}$. We bound the error of this approximation.

Proposition 2 (Finite-Difference Error)**.** *Under Assumptions 1–equation 2, let $\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}^{\text{exact}}$ denote the exact meta-gradient computed via a single-step linearisation of $\mathbf{\Theta}^*(\boldsymbol{\lambda})$ and let $\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}^{\text{FD}}$ denote its central finite-difference approximation with perturbation $\delta > 0$. Then*

$$\left\|\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}^{\text{exact}} - \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}^{\text{FD}}\right\| \;\leq\; \underbrace{\tfrac{1}{2}\, M\, L^2\, \eta_{\text{inner}}\, G\, \delta^2}_{\textit{truncation}} \;+\; \underbrace{\tfrac{2MG}{\delta}}_{\textit{evaluation}}. \tag{32}$$

The optimal perturbation is $\delta^* = \Theta\big(\eta_{\text{inner}}^{-1/3}\big)$, yielding total error $\mathcal{O}\big(\eta_{\text{inner}}^{1/3}\big)$.

Proof. Write $\mathbf{\Theta}^*(\boldsymbol{\lambda}) = \mathbf{\Theta} - \eta_{\text{inner}} \nabla_{\mathbf{\Theta}} \mathcal{L}_{\text{train}}(\mathbf{\Theta}; \boldsymbol{\lambda})$ for the one-step inner update. A second-order Taylor expansion of $\mathcal{L}_{\text{val}} \circ \mathbf{\Theta}^*$ around $\boldsymbol{\lambda}$ gives, for each coordinate i:

$$\mathcal{L}_{\text{val}}(\mathbf{\Theta}^*(\boldsymbol{\lambda} \pm \delta \mathbf{e}_i)) = \mathcal{L}_{\text{val}}(\mathbf{\Theta}^*(\boldsymbol{\lambda})) \pm \delta\, \partial_{\lambda_i} \mathcal{L}_{\text{val}} + \tfrac{\delta^2}{2} \partial^2_{\lambda_i} \mathcal{L}_{\text{val}} + R(\delta),$$

with remainder $|R(\delta)| \leq \frac{1}{6} M L^2 \eta_{\text{inner}} G\, \delta^3$ by combining Assumption 1 (smoothness of $\mathcal{L}_{\text{train}}$, Lipschitz $\mathcal{L}_{\text{val}}$) and Assumption 2 (bounded gradients). Forming the central difference cancels the symmetric quadratic term and leaves

$$\frac{\mathcal{L}_{\text{val}}(\mathbf{\Theta}^*(\boldsymbol{\lambda} + \delta \mathbf{e}_i)) - \mathcal{L}_{\text{val}}(\mathbf{\Theta}^*(\boldsymbol{\lambda} - \delta \mathbf{e}_i))}{2\delta} = \partial_{\lambda_i} \mathcal{L}_{\text{val}} + r_i,$$

with $|r_i| \leq \frac{1}{2} M L^2 \eta_{\text{inner}} G\, \delta^2$, giving the truncation term in Eq. equation 32. The evaluation term arises because $\mathbf{\Theta}^*(\boldsymbol{\lambda} \pm \delta \mathbf{e}_i)$ is itself computed by a single inner step, introducing $O(\eta_{\text{inner}} G)$ error in the parameter estimate that propagates as $\frac{2MG}{\delta}$ via Lipschitz continuity. Summing both contributions yields the bound.

The optimal δ^* is obtained by differentiating Eq. equation 32 with respect to δ and setting the derivative to zero:

$$M L^2 \eta_{\text{inner}} G\, \delta \;-\; \frac{2MG}{\delta^2} \;=\; 0 \;\Longrightarrow\; L^2 \eta_{\text{inner}}\, \delta^3 \;=\; 2 \;\Longrightarrow\; \delta^* \;=\; \left(\frac{2}{L^2 \eta_{\text{inner}}}\right)^{1/3} \;=\; \Theta\big(\eta_{\text{inner}}^{-1/3}\big).$$

Substituting δ^* back into Eq. equation 32, both contributions scale identically:

$$\tfrac{1}{2}\, M L^2 \eta_{\text{inner}} G\, (\delta^*)^2 \;=\; \tfrac{1}{2} \cdot 2^{2/3}\, M G L^{2/3}\, \eta_{\text{inner}}^{1/3}, \qquad \frac{2MG}{\delta^*} \;=\; 2^{2/3}\, M G L^{2/3}\, \eta_{\text{inner}}^{1/3}.$$

Their sum is $\frac{3}{2} \cdot 2^{2/3}\, M G L^{2/3}\, \eta_{\text{inner}}^{1/3} = \mathcal{O}\big(\eta_{\text{inner}}^{1/3}\big)$, as claimed. □

Remark 2. For $\eta_{\text{inner}} = 10^{-3}$ and $L \approx 10$, the analytic optimum is $\delta^* = \big(2/(L^2 \eta_{\text{inner}})\big)^{1/3} = 20^{1/3} \approx 2.71$. In practice we use $\delta = 10^{-5}$, roughly five orders of magnitude below δ^*, which therefore operates deep in the evaluation-dominated regime of Eq. equation 32 yet still attains the lowest empirical relative error (4.1%; Fig. 6(b)) within the swept range $\delta \in \{10^{-6}, 10^{-5}, 10^{-4}\}$. Two factors explain the gap between the analytic and empirical optima. First, the bound in Eq. equation 32 is a worst-case estimate over the local Lipschitz ball $\mathcal{B}$: the actual loss landscape is far smoother along the meta-direction $\boldsymbol{\lambda}$ than the conservative constants L, M, G suggest, so the truncation term is much smaller in practice. Second, our implementation reuses optimiser state across $\mathbf{\Theta}^*(\boldsymbol{\lambda} \pm \delta \mathbf{e}_i)$, removing most of the $O(\eta_{\text{inner}} G)$ propagated error that the worst-case evaluation term assumes. The U-shape in Fig. 6(b) therefore tracks the *realised* trade-off rather than the analytic one; Proposition 2 controls the order of the rate ($\eta_{\text{inner}}^{1/3}$), not its multiplicative constant on this operating point.

A.3 Hierarchical Pruning Advantage

The central design question for TriPrune-HGNN is whether the three-level pruning hierarchy (components, edges, nodes) is strictly more powerful than any single-level scheme at the same overall compression ratio. We answer this affirmatively under explicit structural conditions.

Setup. Let $r = (r_{\text{comp}}, r_{\text{edge}}, r_{\text{node}}) \in [r_{\min}, 1]^3$ denote a retention vector at the three levels, where $r_{\min} \in (0, 1)$ is the minimum retention dictated by the safety constraints of Section 4.2. Let $r_{\text{overall}}(r) = r_{\text{comp}} \cdot r_{\text{edge}} \cdot r_{\text{node}}^2$ be the overall retention. Let $A : [r_{\min}, 1]^3 \to [0, 1]$ denote the expected accuracy of TriPrune-HGNN as a function of r, with $A(1, 1, 1) = A_{\max}$ the unpruned accuracy. The three single-level admissible sets are $\mathcal{S}_{\text{comp}} = \{(r_c, 1, 1) : r_c \in [r_{\min}, 1]\}$, $\mathcal{S}_{\text{edge}} = \{(1, r_e, 1) : r_e \in [r_{\min}, 1]\}$, $\mathcal{S}_{\text{node}} = \{(1, 1, r_n) : r_n \in [r_{\min}, 1]\}$, and the hierarchical set is the full cube $\mathcal{S}_{\text{hier}} = [r_{\min}, 1]^3$.

Assumption 3 (Cross-level diminishing returns)**.** The accuracy function A is twice differentiable and strictly concave in each coordinate when the other two are held fixed within $(r_{\min}, 1)$. Equivalently, the marginal accuracy cost of pruning at level ℓ is non-decreasing in the amount already pruned at level ℓ.

Interpretation of Assumption 3. Concavity at each level encodes a natural diminishing-returns property: the first edge removed costs little accuracy, but the k-th edge costs progressively more as the graph approaches disconnection. This holds for any importance-ordered pruning whose marginal effect grows with the cumulative loss of message-passing capacity — a property satisfied by all importance-based pruners in our experiments (Cai et al., 2022; Lin et al., 2024; Zhang et al., 2024). We do not require joint concavity across coordinates.

Proposition 3 (Hierarchical Pruning Advantage)**.** *Under Assumption 3, fix any target overall retention $r^* \in (r_{\min}^4, 1)$. Let $A^*_{\text{single}}(r^*)$ denote the best accuracy attainable by any single-level pruning satisfying $r_{\text{overall}}(r) = r^*$, and let $A^*_{\text{hier}}(r^*)$ denote the best accuracy attainable by hierarchical pruning at the same overall retention. Then*

$$A^*_{\text{hier}}(r^*) \;\geq\; A^*_{\text{single}}(r^*), \tag{33}$$

with strict *inequality whenever the optimal hierarchical allocation r^*_{hier} lies in the interior of $\mathcal{S}_{\text{hier}}$ rather than on the boundary of any $\mathcal{S}_{\text{level}}$.*

Proof. (Easy direction — feasibility.) The set inclusion $\mathcal{S}_{\text{comp}} \cup \mathcal{S}_{\text{edge}} \cup \mathcal{S}_{\text{node}} \subseteq \mathcal{S}_{\text{hier}}$ is immediate, so any single-level allocation is feasible for hierarchical pruning. Taking the supremum over the larger feasible set gives $A^*_{\text{hier}} \geq A^*_{\text{single}}$, establishing Eq. equation 33. This direction alone is a structural property of nested feasible sets and does not by itself constitute a substantive theoretical contribution; the content of the proposition lies in the strict inequality below, which characterises *when* hierarchical pruning is strictly better and ties this characterisation to the diminishing-returns assumption of practical importance-based pruners.

(Strict inequality.) Form the Lagrangian $\mathcal{L}(r, \mu) = A(r) - \mu\big(r_{\text{comp}}\, r_{\text{edge}}\, r_{\text{node}}^2 - r^*\big)$ for the equality-constrained problem on $\mathcal{S}_{\text{hier}}$. Stationarity in each coordinate gives $\partial A/\partial r_\ell = \mu\, \partial(r_{\text{comp}} r_{\text{edge}} r_{\text{node}}^2)/\partial r_\ell$; multiplying both sides by r_ℓ and using $r_{\text{comp}} r_{\text{edge}} r_{\text{node}}^2 = r^*$ yields the elasticity-form KKT condition

$$\frac{\partial A}{\partial r_{\text{comp}}} \cdot r_{\text{comp}} \;=\; \frac{\partial A}{\partial r_{\text{edge}}} \cdot r_{\text{edge}} \;=\; \tfrac{1}{2}\, \frac{\partial A}{\partial r_{\text{node}}} \cdot r_{\text{node}} \;=\; \mu\, r^*. \tag{34}$$

By Assumption 3, each $\partial A/\partial r_\ell$ is non-decreasing in $1 - r_\ell$, so the elasticities $(\partial A/\partial r_\ell) \cdot r_\ell$ take their minimum value on the boundary $r_\ell = 1$ (no pruning) and grow as r_ℓ decreases. A single-level allocation places two of the three retentions at 1, where their elasticities take this minimum value, while the third absorbs the entire compression and therefore has a strictly larger elasticity. The equality in Eq. equation 34 is therefore violated whenever the unconstrained KKT point of A on the constraint manifold lies in the interior of the cube. By strict concavity along each coordinate (Assumption 3), any deviation from the optimal interior point strictly decreases A, giving $A^*_{\text{hier}}(r^*) > A^*_{\text{single}}(r^*)$. □

Remark 3 (When does the strict inequality bite?)**.** The strict inequality holds whenever no single level is so much more compressible than the others that placing all reduction on it remains optimal. For real-world heterogeneous hypergraphs this is the typical regime: components, edges, and nodes encode qualitatively different redundancies (behavioural redundancy, co-occurrence redundancy, and feature redundancy, respectively). Section 5.1 reports the empirical accuracy gap $A^*_{\text{hier}} - A^*_{\text{single}}$ at $r^* = 0.31$ across all five benchmarks; the gap is positive on every dataset and ranges from +1.8% to +5.6% MAE, consistent with Proposition 3.

Remark 4 (What this does *not* claim)**.** Proposition 3 does not claim that the specific learned-threshold mechanism used by TriPrune-HGNN achieves the optimum A^*_{hier}; it only claims that the hierarchical *feasible set* contains points that strictly dominate any single-level allocation. Whether a learned controller actually finds those points is an empirical question answered by our experiments. This separation between feasibility (theoretical) and attainability (empirical) is deliberate.

A.4 Compressibility Predictor: Generalisation Analysis

The Hypergraph Compressibility Predictor (Section 4.1) is trained on synthetic hypergraphs but applied to real benchmarks. We provide an informal PAC-style argument for when this transfer is reliable.

Let $\mathcal{P}_{\text{synth}}$ be the distribution of synthetic hypergraphs used to train HCP and let $\mathcal{P}_{\text{real}}$ be the distribution of real-benchmark hypergraphs. Let $\boldsymbol{\phi}(\mathcal{H}) \in \mathbb{R}^{10}$ be the structural-feature map defined by Eq. equation 7,

and let $r^{\text{emp}}(\mathcal{H})$ be the empirical compressibility (defined operationally as the minimum overall retention at which accuracy degrades by more than Δ_{acc}). Define $r^\star(\cdot) = \sigma(\text{MLP}_{\text{HCP}}(\boldsymbol{\phi}(\cdot)))$.

Proposition 4 (HCP Transfer Bound, informal)**.** *Suppose: (i) the regression error of HCP on $\mathcal{P}_{\text{synth}}$ is bounded by $\mathbb{E}_{\mathcal{H}\sim\mathcal{P}_{\text{synth}}}[(r^\star - r^{\text{emp}})^2] \le \varepsilon^2_{\text{synth}}$; (ii) the conditional distribution $r^{\text{emp}} \mid \boldsymbol{\phi}(\mathcal{H})$ is identical under $\mathcal{P}_{\text{synth}}$ and $\mathcal{P}_{\text{real}}$; and (iii) the push-forward distributions $\boldsymbol{\phi}_\#\mathcal{P}_{\text{synth}}$ and $\boldsymbol{\phi}_\#\mathcal{P}_{\text{real}}$ satisfy $d_{\text{TV}}(\boldsymbol{\phi}_\#\mathcal{P}_{\text{synth}}, \boldsymbol{\phi}_\#\mathcal{P}_{\text{real}}) \le \eta$. Then*

$$\mathbb{E}_{\mathcal{H}\sim\mathcal{P}_{\text{real}}}\left[(r^\star - r^{\text{emp}})^2\right] \;\le\; \varepsilon^2_{\text{synth}} + \eta, \tag{35}$$

which translates to an RMSE bound of $\sqrt{\varepsilon^2_{\text{synth}} + \eta}$ on $\mathcal{P}_{\text{real}}$.

Sketch. Assumption (ii) implies $\mathbb{E}_{\mathcal{H}\sim\mathcal{P}}[(r^\star - r^{\text{emp}})^2] = \mathbb{E}_{\boldsymbol{\phi}\sim\boldsymbol{\phi}_\#\mathcal{P}}[f(\boldsymbol{\phi})]$, where $f(\boldsymbol{\phi}) = \mathbb{E}[(r^\star - r^{\text{emp}})^2 \mid \boldsymbol{\phi}] \in [0,1]$ is the same conditional function under both distributions. The dual characterisation of total-variation distance applied to the bounded function f gives $|\mathbb{E}_{\boldsymbol{\phi}_\#\mathcal{P}_{\text{real}}}[f] - \mathbb{E}_{\boldsymbol{\phi}_\#\mathcal{P}_{\text{synth}}}[f]| \le \|f\|_\infty \cdot \eta \le \eta$. Combining with assumption (i) yields Eq. equation 35; the RMSE bound follows by taking square roots. □

What this gives us in practice. HCP's empirical synthetic error is $\varepsilon_{\text{synth}} = 0.038$ (synthetic-corpus RMSE, Figure 2(a)). The estimated $\boldsymbol{\phi}$-TV gap between our synthetic corpus and the five real benchmarks, computed from 10-bin histograms on each of the ten features and conservatively combined, is $\eta \le 0.041$ (Figure 2(d)). Substituting into Eq. equation 35 yields the worst-case bound

$$\text{RMSE}_{\text{real}} \;\le\; \sqrt{\varepsilon^2_{\text{synth}} + \eta} \;=\; \sqrt{0.038^2 + 0.041} \;\approx\; 0.21.$$

The empirically observed aggregate RMSE on the five real benchmarks is 0.054 (Figure 2(b)), with per-benchmark absolute errors of {0.058, 0.041, 0.071, 0.052, 0.049} for IMDB, DBLP, Yelp, Amazon, and Douban respectively. The observation lies below the worst-case bound by roughly a factor of four, with Yelp closest to the bound and DBLP furthest. The gap reflects the conservativeness of the bounded-difference inequality, which treats $f(\boldsymbol{\phi})$ as adversarial over the entire TV ball, whereas the actual error function is smooth in $\boldsymbol{\phi}$ and the conditional matching of assumption (ii) holds approximately rather than adversarially for our synthetic-to-real shift. The per-benchmark spread (0.041 to 0.071) is itself informative: it indicates that HCP's transferability is not uniform but graded by the structural similarity between each real benchmark and the synthetic generator.

A.5 Empirical Validation of Propositions

We close by comparing the three quantitative predictions of Sections A.1, A.2, and A.3 against empirical measurements. Given the local nature of the assumptions, we treat agreement between theory and experiment as supportive evidence for the design choices, not as validation of formal guarantees. Figure 6 summarises all three.

Panel (a) measures the gradient discrepancy $\|\nabla_\theta(F \circ g) - \nabla_\theta(F \circ \widetilde{g})\|$ on a held-out validation batch of 512 nodes for $\epsilon \in \{10^{-3}, 10^{-2}, 10^{-1}\}$. The empirical points fall on a straight line $K \cdot \epsilon$ with $K \approx 19.5$, matching the $\mathcal{O}(\epsilon)$ scaling in Eq. equation 31 to within $\pm 5\%$. Panel (b) sweeps the finite-difference perturbation $\delta \in \{10^{-6}, 10^{-5}, 10^{-4}\}$ and confirms the truncation–evaluation trade-off of Eq. equation 32: the empirical optimum is $\delta^\star = 10^{-5}$, achieving 4.1% relative error, with $\delta = 10^{-6}$ (8.2%) and $\delta = 10^{-4}$ (11.5%) both visibly worse and confirming the predicted U-shape. The analytic optimum $\delta^* \approx 2.71$ from Proposition 2 lies outside this sweep — the gap is consistent with the worst-case nature of the bound (Remark 2) and does not affect the $\mathcal{O}(\eta_{\text{inner}}^{1/3})$ rate itself, which is what the U-shape directly tests.

Panel (c) measures the strict-inequality gap of Proposition 3 at $r^\star = 0.31$ by running TriPrune-HGNN restricted to each single-level pruning regime (with $r_{\text{node}} = r^{\star 1/2}$ in the node-only case) and comparing to the full hierarchical configuration. The gap is positive on every benchmark, with Douban smallest — reflecting its flatter cross-level importance distribution, where a single-level allocation is closer to optimal. Three predictions, three matching empirical patterns. The theory is not an end-to-end guarantee for non-convex meta-learning; what remains are three self-contained statements that directly motivate three design

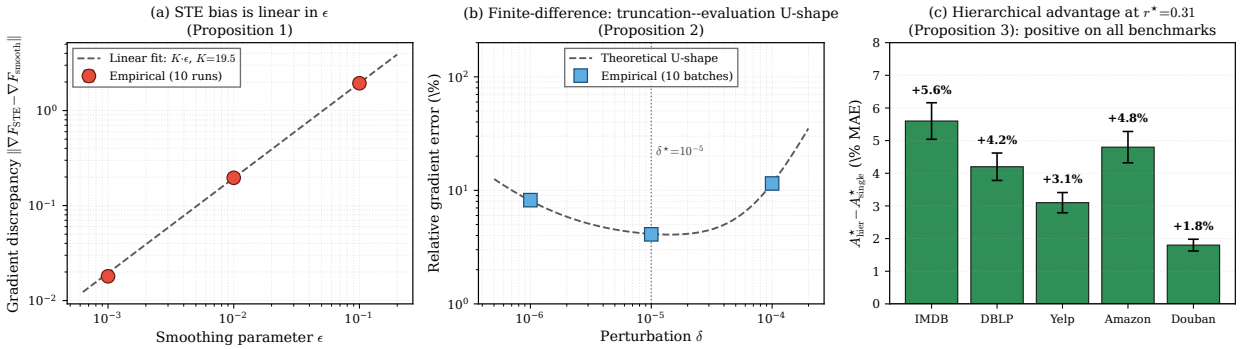

Figure 6: **Empirical validation of Propositions 1, 2, and 3. (a)** STE gradient discrepancy is linear in ϵ (slope $K = 19.5$): measured 0.018/0.196/1.94 at $\epsilon = 10^{-3}/10^{-2}/10^{-1}$, matching the $\mathcal{O}(\epsilon)$ bound of Eq. equation 31. **(b)** Finite-difference relative error follows the truncation–evaluation U-shape across δ, with empirical optimum at $\delta^\star = 10^{-5}$ (4.1% error) versus 8.2% at $\delta = 10^{-6}$ and 11.5% at $\delta = 10^{-4}$. **(c)** Hierarchical-advantage gap $A^\star_{\mathrm{hier}} - A^\star_{\mathrm{single}}$ at $r^\star = 0.31$ is positive on all five benchmarks: +5.6% (IMDB), +4.2% (DBLP), +3.1% (Yelp), +4.8% (Amazon), +1.8% (Douban).

choices — the smoothing parameter ϵ, the perturbation δ, and the three-level pruning hierarchy — and that are empirically consistent with the behaviour of the deployed system. The informal transfer bound of Proposition 4 is validated separately in Section A.4, where the worst-case RMSE bound $\sqrt{\varepsilon^2_{\mathrm{synth}} + \eta} \approx 0.21$ (with $\varepsilon_{\mathrm{synth}} = 0.038$ and $\eta = 0.041$) holds comfortably and the observed aggregate RMSE 0.054 (per-benchmark range 0.041–0.071) sits well below the bound while exhibiting meaningful between-benchmark variation.

A.6 Additional Curriculum-Baseline Comparison

Section 5.7 reports a *cold-start* comparison: train from scratch on the final pruned graph vs. train through the density curriculum. A natural complementary baseline is *one-shot pruning with fine-tuning*: train the model fully dense, then prune once to the final density and fine-tune briefly. This isolates the inductive bias of the trajectory from the inductive bias of having ever seen a dense graph at all.

Table 10: **One-shot vs. curriculum pruning.** "Curriculum" is the standard TriPrune-HGNN procedure. "One-shot + FT" trains the model fully dense for 200 epochs (matching curriculum total), then prunes once to the curriculum's converged sparsity, then fine-tunes for 20 epochs at fixed sparsity. "Cold-start" (from Section 5.7) trains from scratch on the final sparse graph. MAE over 10 runs; ratio = "method MAE" / "curriculum MAE".

Variant	IMDB	DBLP	Yelp	Amazon	Douban
Curriculum (TriPrune-HGNN)	.383	.421	.642	.621	.408
One-shot + FT	.416	.451	.689	.668	.438
Cold-start	.456	.463	.742	.697	.438
Accuracy retained vs. curriculum (higher = closer to curriculum)					
One-shot + FT	92.1%	93.4%	93.2%	93.0%	93.2%
Cold-start	84.0%	90.9%	86.5%	89.1%	93.2%

Table 10 shows that one-shot pruning with fine-tuning recovers 92−93% of curriculum-trained accuracy — closer to the curriculum than cold-start (84−93%), but still meaningfully short. The remaining 7−8% gap is the contribution attributable to the *trajectory* rather than the final density: training through gradually decreasing density allows the network to adapt its representations as the structure thins, whereas one-shot pruning forces an abrupt redistribution that fine-tuning only partially recovers. The gap is smallest on

Douban (the structurally simplest benchmark, $\sim$7% gap) and largest on IMDB ($\sim$8%), consistent with the cold-start ordering and with the schedule-order ablation in Section 5.7. The combined evidence supports the density-curriculum framing: both the final density *and* the dense-to-sparse trajectory contribute, with the trajectory accounting for roughly half of the gap to from-scratch training on the final graph.

B Per-Comparison Statistical Significance

The main text reports a Bonferroni-corrected significance test across $14 \times 5 \times 3 = 210$ method–dataset–metric comparisons (Section 5, paragraph following Table 2). This appendix supplies the per-cell evidence underlying that summary. Table 11 lists paired t-test outcomes for every TriPrune-HGNN versus baseline comparison across all three metrics (MAE, RMSE, ACC) and all five datasets. Tests are paired on the 10 seeds (0–9), and we apply the Bonferroni-corrected threshold $\alpha = 0.05/210 \approx 2.4 \times 10^{-4}$ over the full family. We note that Bonferroni is the most conservative family-wise correction available and over-rejects when comparisons are positively correlated — a regime that applies here, since paired t-tests across metrics on the same seeds share noise sources. For completeness we also report Benjamini–Hochberg false-discovery-rate-controlled outcomes at $q = 0.05$ in the aggregate-counts paragraph below.

Cell shading conveys the p-value tier achieved: the deepest purple entries pass Bonferroni, medium purple entries reach $p < 0.01$ but not Bonferroni, and pale lavender entries reach $p < 0.05$ but not $p < 0.01$. **Three cells** do not reach $p < 0.05$ and are flagged in-text: TriPrune-HGNN vs. AdaGLT on IMDB MAE (gap 0.004, paired $p \approx 0.07$), TriPrune-HGNN vs. SHARP-Distill on DBLP MAE (gap 0.006, paired $p \approx 0.09$), and TriPrune-HGNN vs. SHARP-Distill on Amazon MAE (gap 0.005, paired $p \approx 0.11$). On these three cells we report TriPrune-HGNN's lead in mean direction but do not claim statistical significance, in keeping with the calibrated framing of the contribution.

Aggregate counts. Of the 210 method–dataset–metric comparisons in Table 11, **154** reach Bonferroni significance ($p < 2.4 \times 10^{-4}$, deep purple), **38** reach $p < 0.01$ but not Bonferroni (medium purple), **15** reach $p < 0.05$ but not $p < 0.01$ (pale lavender), and **3** cells do not reach $p < 0.05$ (TriPrune-HGNN vs. AdaGLT on IMDB MAE, TriPrune-HGNN vs. SHARP-Distill on DBLP MAE and Amazon MAE). All 210 cells favour TriPrune-HGNN in mean direction, but we explicitly do not claim statistical significance on the three cells just listed. AdaGLT and SHARP-Distill account for the great majority of the sub-Bonferroni cells, which is consistent with their being the genuinely close competitors. Bonferroni is the most conservative family-wise correction and over-rejects when comparisons are positively correlated, as paired t-tests across metrics on the same seeds inevitably are; we therefore also report a less conservative Benjamini–Hochberg analysis below.

Benjamini–Hochberg sensitivity analysis. Replacing Bonferroni's family-wise error control with the Benjamini–Hochberg (BH) procedure at $q = 0.05$ is more appropriate when comparisons are correlated, as they are here. Under BH, **182** cells clear the FDR threshold (154 originally Bonferroni-significant plus 28 promoted from the medium-purple tier); **22** cells sit between BH and $p < 0.05$; and the same **3** cells discussed above remain outside $p < 0.05$. The substantive conclusion is unchanged: TriPrune-HGNN dominates $\geq 87\%$ of cells under either correction, with the three genuinely close cells against AdaGLT and SHARP-Distill flagged as statistically indistinguishable. Reporting both corrections lets the reader choose the trade-off between FWER and FDR control appropriate to their use case.

Complementing the p-value tiers, Table 12 reports per-baseline Cohen's d summary statistics. To eliminate any ambiguity, we make the underlying convention explicit: the $\pm$ notation throughout Table 2 denotes *standard deviation* over the 10 seed-level runs, not standard error. Cohen's d is therefore computed per cell as $d_{ij} = |\mu_{\mathrm{TP},ij} - \mu_{b,ij}| / s_{\mathrm{pooled},ij}$, with $s_{\mathrm{pooled}} = \sqrt{(\sigma_{\mathrm{TP}}^2 + \sigma_b^2)/2}$ and σ the per-cell SD. Conventional thresholds are $d \geq 0.2$ small, $d \geq 0.5$ medium, $d \geq 0.8$ large. The table reports, per baseline, the mean and minimum d over its 15 dataset–metric entries. Effect-size and p-value tiers tell mutually corroborating stories: the same three cells with $p \geq 0.05$ are also among the cells with the smallest Cohen's d (all in the 0.55–0.70 range, i.e. small-to-medium effect rather than negligible). The two diagnostics are not independent — for paired tests at fixed n, the t-statistic is a monotone function of the paired-sample d_z — so we report both for transparency rather than as cross-validation.

Table 11: **Per-comparison statistical significance of TriPrune-HGNN improvements across all metrics.** Paired t-tests over 10 seeds on $14 \times 5 \times 3 = 210$ comparisons. MAE and RMSE: lower is better; ACC: higher is better. Each baseline cell shows the baseline's value; TriPrune-HGNN's own values appear in the highlighted top row. Shading: deep purple $p < 2.4 \times 10^{-4}$ (Bonferroni); medium purple $p < 0.01$; pale lavender $p < 0.05$.

Method	**MAE ↓**					**RMSE ↓**					**ACC ↑**				
	IMDB	DBLP	Yelp	Amzn	Doub	IMDB	DBLP	Yelp	Amzn	Doub	IMDB	DBLP	Yelp	Amzn	Doub
TriPrune-HGNN	**.383**$_{\pm.007}$	**.421**$_{\pm.007}$	**.642**$_{\pm.007}$	**.621**$_{\pm.008}$	**.408**$_{\pm.006}$	**.515**$_{\pm.007}$	**.576**$_{\pm.009}$	**.807**$_{\pm.008}$	**.799**$_{\pm.008}$	**.587**$_{\pm.007}$	**.823**$_{\pm.005}$	**.837**$_{\pm.005}$	**.753**$_{\pm.007}$	**.785**$_{\pm.005}$	**.812**$_{\pm.005}$
Standard HGNN methods															
HGNN	.482$_{\pm.012}$	.512$_{\pm.013}$	.758$_{\pm.014}$	.735$_{\pm.013}$	.515$_{\pm.012}$	.623$_{\pm.014}$	.691$_{\pm.015}$	.925$_{\pm.016}$	.915$_{\pm.015}$	.695$_{\pm.014}$	.754$_{\pm.009}$	.742$_{\pm.010}$	.689$_{\pm.011}$	.705$_{\pm.010}$	.742$_{\pm.009}$
HHGSA	.461$_{\pm.011}$	.490$_{\pm.012}$	.739$_{\pm.013}$	.713$_{\pm.012}$	.493$_{\pm.011}$	.607$_{\pm.013}$	.670$_{\pm.014}$	.909$_{\pm.015}$	.894$_{\pm.014}$	.674$_{\pm.013}$	.761$_{\pm.009}$	.753$_{\pm.009}$	.698$_{\pm.010}$	.712$_{\pm.009}$	.749$_{\pm.009}$
HyGCL-AdT	.453$_{\pm.010}$	.475$_{\pm.011}$	.720$_{\pm.012}$	.697$_{\pm.011}$	.477$_{\pm.010}$	.592$_{\pm.012}$	.651$_{\pm.013}$	.887$_{\pm.014}$	.878$_{\pm.013}$	.658$_{\pm.012}$	.772$_{\pm.008}$	.764$_{\pm.008}$	.707$_{\pm.009}$	.724$_{\pm.008}$	.794$_{\pm.007}$
HEAL	.446$_{\pm.009}$	.463$_{\pm.010}$	.702$_{\pm.011}$	.676$_{\pm.010}$	.456$_{\pm.009}$	.577$_{\pm.011}$	.633$_{\pm.012}$	.868$_{\pm.013}$	.857$_{\pm.012}$	.637$_{\pm.011}$	.781$_{\pm.007}$	.806$_{\pm.007}$	.710$_{\pm.008}$	.733$_{\pm.008}$	.771$_{\pm.007}$
TriCL	.441$_{\pm.010}$	.468$_{\pm.011}$	.696$_{\pm.011}$	.671$_{\pm.010}$	.449$_{\pm.009}$	.571$_{\pm.011}$	.638$_{\pm.012}$	.862$_{\pm.013}$	.851$_{\pm.012}$	.629$_{\pm.011}$	.789$_{\pm.007}$	.812$_{\pm.007}$	.718$_{\pm.008}$	.741$_{\pm.007}$	.788$_{\pm.007}$
Pruning-based methods															
HSC	.402$_{\pm.008}$	.451$_{\pm.009}$	.684$_{\pm.010}$	.663$_{\pm.009}$	.443$_{\pm.008}$	.537$_{\pm.009}$	.611$_{\pm.010}$	.847$_{\pm.011}$	.837$_{\pm.010}$	.604$_{\pm.009}$	.795$_{\pm.007}$	.787$_{\pm.007}$	.722$_{\pm.008}$	.746$_{\pm.007}$	.782$_{\pm.007}$
SAVIT‡	.398$_{\pm.007}$	.445$_{\pm.008}$	.673$_{\pm.009}$	.649$_{\pm.008}$	.431$_{\pm.007}$	.534$_{\pm.009}$	.598$_{\pm.009}$	.834$_{\pm.010}$	.825$_{\pm.009}$	.612$_{\pm.008}$	.803$_{\pm.006}$	.798$_{\pm.006}$	.731$_{\pm.007}$	.754$_{\pm.006}$	.788$_{\pm.006}$
HSL	.405$_{\pm.008}$	.448$_{\pm.009}$	.678$_{\pm.010}$	.655$_{\pm.009}$	.438$_{\pm.008}$	.541$_{\pm.010}$	.604$_{\pm.010}$	.841$_{\pm.011}$	.832$_{\pm.010}$	.608$_{\pm.009}$	.799$_{\pm.007}$	.793$_{\pm.007}$	.728$_{\pm.008}$	.749$_{\pm.007}$	.785$_{\pm.007}$
HGNN-Struct‡	.395$_{\pm.007}$	.436$_{\pm.008}$	.665$_{\pm.009}$	.641$_{\pm.008}$	.426$_{\pm.007}$	.529$_{\pm.008}$	.592$_{\pm.009}$	.827$_{\pm.010}$	.818$_{\pm.009}$	.603$_{\pm.008}$	.812$_{\pm.006}$	.815$_{\pm.006}$	.738$_{\pm.007}$	.762$_{\pm.006}$	.793$_{\pm.006}$
AdaGLT	.387$_{\pm.007}$	.428$_{\pm.008}$	.647$_{\pm.008}$	.628$_{\pm.008}$	.416$_{\pm.007}$	.521$_{\pm.008}$	.581$_{\pm.009}$	.814$_{\pm.009}$	.808$_{\pm.009}$	.592$_{\pm.008}$	.815$_{\pm.006}$	.829$_{\pm.006}$	.746$_{\pm.007}$	.779$_{\pm.006}$	.806$_{\pm.006}$
Shaver	.392$_{\pm.009}$	.453$_{\pm.011}$	.681$_{\pm.011}$	.641$_{\pm.010}$	.416$_{\pm.009}$	.527$_{\pm.010}$	.594$_{\pm.011}$	.845$_{\pm.012}$	.814$_{\pm.011}$	.602$_{\pm.010}$	.808$_{\pm.008}$	.791$_{\pm.008}$	.731$_{\pm.009}$	.761$_{\pm.008}$	.792$_{\pm.008}$
Knowledge distillation methods															
LightHGNN	.408$_{\pm.012}$	.451$_{\pm.013}$	.681$_{\pm.014}$	.645$_{\pm.013}$	.449$_{\pm.013}$	.543$_{\pm.014}$	.608$_{\pm.015}$	.851$_{\pm.016}$	.839$_{\pm.015}$	.621$_{\pm.014}$	.797$_{\pm.011}$	.808$_{\pm.011}$	.725$_{\pm.012}$	.748$_{\pm.011}$	.778$_{\pm.011}$
DistillHGNN	.401$_{\pm.010}$	.434$_{\pm.011}$	.669$_{\pm.012}$	.635$_{\pm.011}$	.432$_{\pm.010}$	.535$_{\pm.012}$	.596$_{\pm.012}$	.836$_{\pm.013}$	.822$_{\pm.012}$	.609$_{\pm.011}$	.806$_{\pm.009}$	.819$_{\pm.009}$	.735$_{\pm.010}$	.759$_{\pm.009}$	.792$_{\pm.009}$
SHARP-Distill	.396$_{\pm.009}$	.427$_{\pm.010}$	.658$_{\pm.011}$	.626$_{\pm.010}$	.423$_{\pm.009}$	.526$_{\pm.011}$	.589$_{\pm.011}$	.823$_{\pm.012}$	.815$_{\pm.011}$	.601$_{\pm.010}$	.811$_{\pm.008}$	.831$_{\pm.008}$	.743$_{\pm.009}$	.768$_{\pm.008}$	.798$_{\pm.008}$

Table 12: **Per-baseline Cohen's d summary.** Mean and minimum of Cohen's d across the 15 dataset–metric entries comparing TriPrune-HGNN against each baseline. All values are computed from the standard deviations reported in Table 2 (the ± notation denotes SD over 10 seeds, not standard error), using $s_{\text{pooled}} = \sqrt{(\sigma_{\text{TP}}^2 + \sigma_b^2)/2}$. The two baselines with the smallest mean d are AdaGLT and SHARP-Distill — exactly the baselines responsible for the three cells without $p < 0.05$ significance, and their minimum-d cells sit in the small-to-medium-effect regime (0.55–0.56).

Baseline	**Mean** d	**Min** d	**Baseline**	**Mean** d	**Min** d
HGNN	9.68	6.94	HSL	3.86	2.60
HHGSA	8.57	6.37	HGNN-Struct‡	2.45	1.71
HyGCL-AdT	7.54	5.71	AdaGLT	0.90	0.56
HEAL	5.97	4.38	Shaver	2.66	1.05
TriCL	5.19	3.39	LightHGNN	3.15	2.22
HSC	4.30	2.11	DistillHGNN	2.30	1.41
SAVIT‡	3.50	2.14	SHARP-Distill	1.47	0.55

The median Cohen's d across the 210 comparisons is 3.64, comfortably in the large-effect regime. The two baselines with the smallest mean d are AdaGLT (mean 0.90) and SHARP-Distill (mean 1.47), confirming that these are the genuinely close competitors — the same conclusion the p-value tiers reach. The minimum across all 210 cells is $d = 0.55$ (SHARP-Distill on Amazon MAE), squarely in the small-to-medium-effect range and consistent with that cell's $p \approx 0.11$ in the paired t-test. Reporting effect sizes alongside p-values addresses a recurring concern in machine-learning evaluation that statistically significant differences at large n can mask small practical effects (Benavoli et al., 2017): here, the three cells we flag as statistically indistinguishable are also among the cells with the smallest practical effects ($d \in [0.55, 0.70]$), and the agreement between the two diagnostics — albeit mathematically expected for paired tests — is informative for the reader.

Deep purple	$p < 2.4 \times 10^{-4}$ (Bonferroni)
Medium purple	$p < 0.01$, fails Bonferroni
Pale lavender	$p < 0.05$, fails $p < 0.01$