BATCH BEFORE YOU LIFT: SCALABLE TOPOLOGICAL DEEP LEARNING ON LARGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Topological Deep Learning extends graph-based learning to higher-order domains, such as hypergraphs, cellular, and simplicial complexes. These domains are typically constructed from patterns in an input graph through a process of graph lifting. Current methods operate only after this lifting phase, requiring the entire lifted domain to be materialized prior to training. On large and dense datasets like Reddit (233k nodes and 57.3M edges), this global materialization becomes a severe computational bottleneck, often rendering training infeasible. To address this limitation, we introduce Cluster-TNN, a domain-agnostic framework that avoids this bottleneck by lifting locally instead. After partitioning the input graph during preprocessing, at runtime Cluster-TNN dynamically samples groups of node clusters, reconstructs their induced subgraphs to form mini-batches, and applies the chosen lifting within each mini-batch. By retaining all edges among the sampled nodes, Cluster-TNN preserves higher-order structures spanning sampled clusters, producing topological mini-batches that existing Topological Neural Networks can process directly. Across evaluated domains, lifting procedures, and architectures, Cluster-TNN reduces peak GPU memory by an average of 83.2% while maintaining competitive predictive performance. Notably, such a reduction enables, to our knowledge, the first training of multiple different higher-order Topological Neural Networks on large datasets such as Reddit and OGBN Products. These results establish Cluster-TNN workflow as a general strategy for scaling Topological Deep Learning beyond the limits of global domain construction.

1 INTRODUCTION

Graphs are a dominant representation for relational learning: nodes encode entities and edges encode pairwise relations. Graph Neural Networks (GNNs) learn on such structures across diverse machine learning tasks (Kipf & Welling, 2017; Hamilton et al., 2017a; Gilmer et al., 2017; Bronstein et al., 2021; Corso et al., 2024). However, many systems involve multi-way interactions, such as coauthorship groups, molecular rings, or many-body couplings, that pairwise graphs cannot represent explicitly. Topological Deep Learning (TDL) extends relational learning to higher-order domains, such as hypergraphs, cellular, and simplicial complexes, on which Topological Neural Networks (TNNs) operate (Papamarkou et al., 2024; Hajij et al., 2023; Papillon et al., 2024). By explicitly representing higher-order relations, these domains provide complementary inductive biases for systems whose structure extends beyond pairwise connectivity.

However, scalability in higher-order domains remains a central challenge for TDL (Papamarkou et al. (2024), Open Problem 6). For graph-supported data, a *lifting* maps patterns in the input graph, such as neighbourhoods, cliques, or cycles, to explicit higher-order elements and their incidence relations. This lifting determines both which higher-order relations become available to the model and the size of the resulting computational domain. Because the number of lifted elements depends on graph connectivity rather than node count alone, the lifted domain can be orders of magnitude larger than the input graph. For example, the Reddit dataset has approximately 233k nodes and 57.3M edges, yet clique-complex lifting produces approximately 8.36B simplicial 2-cells (filled triangles). Enumerating these elements, their features, and their incidence relations may exceed available memory before training begins, making full-domain construction a scalability bottleneck.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

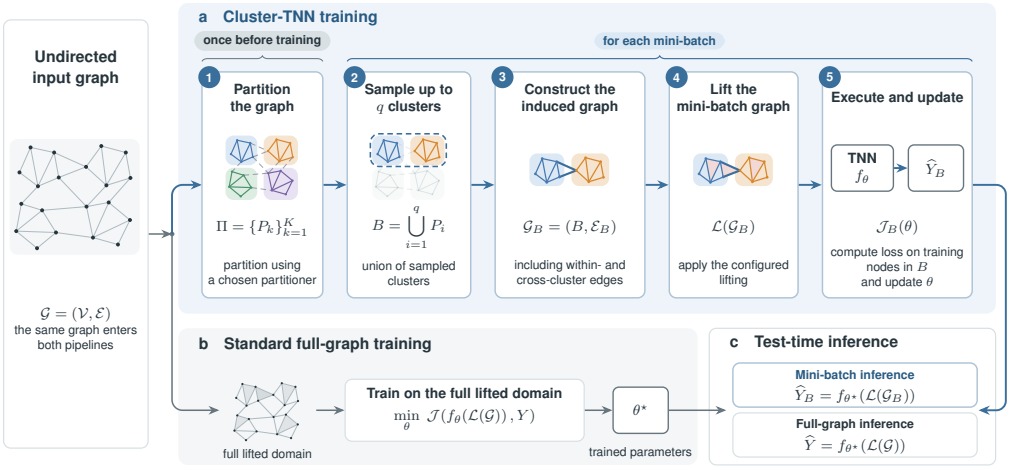


Figure 1: **Comparison of Cluster-TNN training (ours) and standard full-graph TDL training.** Both workflows start from the same featured undirected graph. **a** Cluster-TNN partitions the graph once per run and at each training step samples up to q clusters, reconstructs their induced subgraph, and performs lifting and model execution only on the resulting mini-batch. **b** Standard full-graph training instead constructs and processes the complete lifted domain throughout training. **c** Trained models support both mini-batch and full-graph inference.

Existing mini-batching strategies do not fully address this bottleneck because they typically intervene only after the higher-order domain has been constructed. Sampling or partitioning an already lifted hypergraph, simplicial complex, or cell complex can reduce the memory required by individual forward and backward passes, but it would still assume that the complete set of higher-order elements and their connectivity can be enumerated or materialized beforehand. Consequently, mini-batched model execution may remain infeasible when the preceding lifting operation exceeds the available computational resources. Moreover, batching directly on a lifted representation generally requires procedures tailored to its particular topological domain, limiting their reuse across different liftings and TNN architectures.

On the other hand, performing mini-batching before lifting could address both limitations: it would restrict lifting to the data required for the current optimization step and perform batching on the graph shared by the different higher-order constructions. However, this change in execution order introduces a structural challenge: if graph partitions are lifted independently, edges crossing partition boundaries are omitted. Consequently, higher-order structures supported by nodes in different partitions, including neighborhood hyperedges, cliques, and cycles, may disappear from the training data. Independently lifting small graph fragments can therefore reduce computational cost at the expense of systematically altering the higher-order domain observed by the model.

We address this gap by introducing Cluster-TNN, a *batch-before-lift* strategy that circumvents full topology materialization while retaining higher-order structural information with minimal structural modification. As illustrated in Figure 1, Cluster-TNN partitions the graph once into disjoint node clusters and samples a group of clusters at each optimization step. The proposed workflow then reconstructs the subgraph induced by the combined nodes of these clusters, and retains every original edge among its nodes, also including edges between clusters. Lastly, Cluster-TNN applies the requested lifting only to this reconstructed graph mini-batch and passes the resulting higher-order domain directly to a TNN.

Induced-subgraph reconstruction is central to the approach: unlike lifting each sampled cluster independently, it restores the edges connecting jointly sampled clusters. Under lifting compatibility (Sec. 3.3), a higher-order structure can therefore be recovered whenever all of its supporting nodes occur in the same batch. By recombining clusters across epochs, Cluster-TNN provides repeated opportunities to observe higher-order structures spanning different clusters.

Contributions. The main features and contributions of our work include:

- **Batch-before-lift strategy.** We introduce Cluster-TNN, a general strategy for scaling TDL beyond the limits imposed by global topology materialization. The same batching mechanism supports different graph-to-higher-order liftings and existing TNN architectures, which we evaluate on Reddit and OGBN Products.
- **Cross-cluster structural recovery of Cluster-TNN.** We characterize the recovery of higher-order structures under repeated cluster recombination (Sec. 3.3), and complement the analysis with empirical measurements of observable hyperedges, cellular 2-cells, and simplicial 2-cells (Appendix C.13).
- **Scalability-computation trade-off via extensive evaluation.** We empirically quantify the trade-off introduced by batch-before-lift in terms of full-graph preprocessing feasibility and cost, peak CPU and GPU memory, training time and predictive performance. In particular, across 21 matched model-dataset comparisons, Cluster-TNN reduces peak GPU memory in every setting by 83.2% on average, while maintaining competitive predictive performance.
- **Reusable open-source integration.** We implement Cluster-TNN in TopoBench (Telyatnikov et al., 2025a), reusing existing lifting and models across different higher-order domains without requiring architecture-specific modifications. Code is available in our anonymous repository.

2 Related work

Higher-order domains, liftings, and TDL software. Evaluating TNNs on graph datasets requires constructing the higher-order domain on which the model operates, making the lifting itself part of the model evaluation. TopoX (Hajij et al., 2024) provides topological data structures and neural operators, while TopoBench (Telyatnikov et al., 2025a) integrates datasets, liftings, and models within a common benchmarking framework. This standardization supports comparisons across hypergraph, cellular, and simplicial architectures. However, Telyatnikov et al. (2025a) report out-of-memory errors during clique and cycle lifting on medium-sized graphs with fewer than $25\,000$ nodes. Full-domain materialization can therefore prevent model execution even at this scale.

Scalability methods for higher-order learning. Existing approaches reduce model-execution costs. FastHGNN (Lu & Ng, 2024) and Telyatnikov et al. (2025b) subsample preconstructed hypergraphs, while Topo-MLP (Ramamurthy et al., 2023) accelerates simplicial inference by removing explicit message passing. TopoMamba (Montagna et al., 2025) introduces architecture-specific neighbourhood batching using node-to-cell incidence relations. SaNN (Gurugubelli & Chepuri, 2024) precomputes simplicial feature aggregations. HOPSE (Bernárdez et al., 2025) precomputes positional and structural encodings through Hasse graph decompositions for message-passing-free learning. These methods operate on constructed higher-order domains, leaving a complementary objective: limiting domain construction across lifting families while retaining existing TNNs.

Graph mini-batching as the methodological foundation. GNN methods restrict computation through neighbourhood sampling with GraphSAGE (Hamilton et al., 2017b), layer-wise importance sampling with FastGCN (Chen et al., 2018), or subgraph sampling with GraphSAINT (Zeng et al., 2020). Cluster-GCN (Chiang et al., 2019) partitions the graph's nodes into disjoint clusters and forms mini-batches from one or more clusters, retaining all edges among the selected nodes, including those connecting different clusters. For graph-supported TDL, preserving this connectivity has an additional significance: missing edges can prevent higher-order structures from being constructed, even when all their supporting nodes are present.

3 Methods

Cluster-TNN moves mini-batching before higher-order domain construction. At each training step, we group up to q graph clusters, reconstruct their induced subgraph, and apply the selected lifting only to that subgraph. Retaining all original edges among the sampled nodes provides the connectivity needed to construct higher-order structures spanning jointly sampled clusters. A TNN for the selected higher-order domain then processes the lifted mini-batch. This execution order makes mini-batching control both domain construction and model execution across higher-order domains, without materializing the full lifted domain.

3.1 MINI-BATCH CONSTRUCTION AND LIFTING

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, F_{\mathcal{V}}, F_{\mathcal{E}})$ denote a graph with node set $\mathcal{V}$, edge set $\mathcal{E}$, and corresponding feature maps $F_{\mathcal{V}}$ and $F_{\mathcal{E}}$. Following the multi-cluster batching scheme of Cluster-GCN (Chiang et al., 2019), we first partition $\mathcal{V}$ into K disjoint clusters $\Pi = \{P_1, \ldots, P_K\}$, with $\mathcal{V} = \bigcup_{k=1}^{K} P_k$. We compute this partition using METIS (Karypis & Kumar, 1998) or random partitioning and keep it fixed throughout the run (Figure 1a, step 1).At the beginning of each training epoch, Cluster-TNN randomly groups clusters containing at least one training node, referred to as *active* clusters, into mini-batches of at most q clusters (see step 2). Validation and testing similarly use only clusters containing validation and test nodes, respectively. For a group Z, the mini-batch node set is $B = \bigcup_{k \in Z} P_k$, from which we extract the induced subgraph $\mathcal{G}_B = (B, \mathcal{E}_B)$ with $\mathcal{E}_B = \{(u, v) \in \mathcal{E} : u, v \in B\}$. Node features, labels, split masks, and edge features are restricted accordingly. Thus, $\mathcal{G}_B$ preserves all edges among co-sampled clusters, including inter-cluster edges (step 3). The selected lifting $\mathcal{L}$ maps $\mathcal{G}_B$ to higher-order domains required by the TNN, constructing higher-order elements and their features from the connectivity of $\mathcal{G}_B$ (step 4, Appendix A.2.1). The TNN then processes $\mathcal{L}(\mathcal{G}_B)$ through the same interface used for full-domain execution (step 5), enabling a unified workflow across hypergraph, cellular, and simplicial domains. This approach contrasts with the standard workflow, where TNNs operate on the entire lifted object $\mathcal{L}(\mathcal{G})$ as illustrated in Figure 1b.

Algorithm 1 Cluster-TNN

Require: Featured graph $\mathcal{G}$, node labels Y, lifting $\mathcal{L}$, number of clusters K, maximum clusters per mini-batch q
Ensure: Trained parameters θ
1: Partition $\mathcal{V}$ into K clusters
2: **for** each training step **do**
3: Take at most q active clusters
4: Form mini-batch node set B
5: Construct induced subgraph $\mathcal{G}_B$
6: Lift the mini-batch: $\mathcal{L}(\mathcal{G}_B)$
7: Compute $\mathcal{J}_B(\theta)$
8: Take an optimizer step on $\mathcal{J}_B(\theta)$
9: **end for**

Algorithm 1 summarizes Cluster-TNN training protocol. The training objective and inference protocols are detailed in Sec. 3.2. The workflow applies when the selected graph-to-domain lifting can be evaluated on an induced graph and the downstream TNN can process the variable-size lifted domains through the same interface used for the full-domain execution. For node classification, the model must also produce predictions for the target nodes contained in each mini-batch. Architectures that require operators or state defined on the complete lifted domain require additional adaptation, whereas datasets provided natively as higher-order domains require batching directly in that domain rather than the graph-first construction considered here (see Appendix B.7).

3.2 TRAINING AND INFERENCE

Let Y denote the labels available for the training nodes $\mathcal{V}_{\text{train}} \subseteq \mathcal{V}$, and let f_θ denote the model with parameters θ. For each mini-batch, we evaluate the node-classification loss on training nodes contained in B, given by

$$\mathcal{J}_B(\theta) = \frac{1}{|B \cap \mathcal{V}_{\text{train}}|} \sum_{v \in B \cap \mathcal{V}_{\text{train}}} \ell\left(f_\theta\left(\mathcal{L}(\mathcal{G}_B)\right)_v, Y_v\right),$$

where $\ell(\cdot, \cdot)$ denotes the cross-entropy loss between the model predictions and the ground truth. Since the loader uses only active train clusters, each mini-batch contains supervised nodes. Each lifted mini-batch produces one optimization step, regardless of the number of supervised nodes.

At inference, the model can operate on one cluster grouping, average predictions across several independent groupings, or, when memory permits, materialize the full lifted graph. We refer to these modes as *single-pass mini-batch*, *ensemble mini-batch*, and *full-graph* inference, respectively. The primary partitioned results use ensemble mini-batch inference with ten independent cluster recombinations, whereas the inference ablation compares all three modes (Sec. 4.4).

3.3 THEORETICAL GUARANTEES

Each mini-batch is lifted from the subgraph induced by its sampled clusters (Sec. 3.1). A structure from the full lifted graph cannot appear unless all its supporting nodes are included. This section establishes which higher-order structures Cluster-TNN can recover without full-domain lifting and quantifies how repeated recombination of all partition clusters increases their coverage.

A structure can be recovered only if its supporting clusters fit within one mini-batch. Let S^* denote the finite set of structures produced by lifting the full graph. For a structure $s \in S^*$, its *cluster span* c_s counts the clusters containing its supporting nodes. For example, a triangle whose vertices lie in three different clusters requires all three clusters to appear in the same mini-batch. We assume *lifting compatibility*: a full-graph structure appears in a lifted mini-batch exactly when that mini-batch contains all its supporting nodes (Definition 7). Under this assumption, structures with $c_s \leq q$ can be recovered, whereas structures with $c_s > q$ cannot appear in a mini-batch containing q clusters. We call the former structures *q-observable* and define $S^*_q = \{s \in S^* : c_s \leq q\}$.

Cumulative recovery under cluster recombination. A structure need not appear in every epoch to be observed over the course of training. Independent cluster recombinations provide repeated opportunities for its supporting clusters to occur together. The following result quantifies both its finite-epoch recovery probability and the limiting recovery of the observable structure set. Consider a finite graph with a fixed partition into K clusters. At each epoch, independently draw a uniform random permutation of all K clusters and divide it into groups of q clusters. Assume $q \mid K$ and lifting compatibility. For a full-graph structure s spanning c_s clusters, its per-epoch recovery probability is $p_s = \binom{q-1}{c_s-1} \Big/ \binom{K-1}{c_s-1}$ when $c_s \leq q$, and $p_s = 0$ otherwise. After T epochs, the probability that s has been observed at least once is $1 - (1 - p_s)^T$. Consequently, the probability that every structure in S^*_q has been observed approaches one as $T \to \infty$. If $q \geq \max_{s \in S^*} c_s$, this gives cumulative recovery of the global structure set S^*.

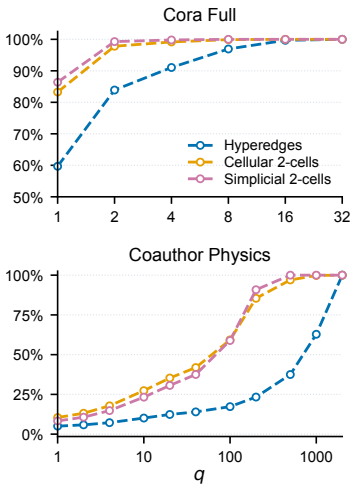


Figure 2: Structural support recovery as a function of q. Cora Full uses $K = 32$ and Coauthor Physics $K = 2\,000$. Curves show means over ten independent reshuffling runs. See Appendix C.13.1 for more details.

When split-specific batching excludes inactive clusters, structures involving nodes from those clusters cannot be recovered (Appendix C). The per-epoch probability and cumulative-recovery proofs are given in Appendix C.2-C.4. Recovery is cumulative: different structures may be observed across mini-batches and epochs, without materializing the complete lifted domain. For models that exchange messages between incident cells, structural recovery makes the corresponding message paths available. See Appendix C.9-C.11 for more details.

Larger mini-batches expand what can be recovered, while additional epochs provide repeated opportunities. A structure spanning more than q clusters cannot be recovered regardless of the number of epochs. Increasing q can make more structures observable and increase their recovery probabilities. Figure 2 reports the fraction of reference structures whose supporting nodes are jointly sampled within 200 epochs. At $q = 4$, this fraction exceeds 99% for both cellular and simplicial 2-cells on Cora Full. On Coauthor Physics at $q = 100$, the corresponding fractions are only 59.3% for cellular and 58.9% for simplicial supports, although each individual reference 2-cell could fit within a mini-batch of this size. Fitting within a mini-batch therefore does not ensure that the required clusters are sampled together. Complete neighborhood hyperedges reach only 62.8% recovery even at $q = 1\,000$. Their slower recovery is consistent with larger cluster spans: triangles and retained cycles involve three and at most nine nodes, respectively, whereas a neighborhood hyperedge includes its center and all its neighbors. These larger supports can span more clusters, reducing their joint-sampling probability. See Appendix C.3 and C.6 for more details.

Repeated recombination motivates ensemble inference. A single cluster grouping can omit higher-order structures that involve a target node, even when those structures are observable. Independent recombinations provide further opportunities to include their supporting nodes. This motivates averaging predictions from the same trained model over several independently sampled cluster groupings, without materializing the full lifted domain (Sec. 3.2). Ensemble inference achieves a higher mean test metric than single-pass inference in all 21 reported model-dataset comparisons (Table 2). These results support using additional mini-batch inference passes to improve predictive performance while retaining the memory benefits of processing one lifted mini-batch at a time.

4 EVALUATION

We evaluate whether moving mini-batching before lifting enables TNN training at larger scales and examine its computational costs, predictive performance, and structural recovery. Our experiments address three questions:

Q1: Scalability and computational trade-offs. Can Cluster-TNN enable training of topological models when full-domain construction or model execution exceeds available resources, and how does it affect preprocessing time, peak CPU/GPU memory, training time, and end-to-end runtime?

Q2: Predictive performance. How does Cluster-TNN compare with full-graph execution in predictive performance on the matched benchmarks, and how do higher-order models trained with Cluster-TNN compare with the GCN baseline on Reddit and OGBN Products?

Q3: Structural recovery and design choices. How do mini-batch size and repeated cluster recombination influence structural recovery, and how do partitioning, cross-cluster edge retention, and inference strategy affect predictive performance?

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate Cluster-TNN on six node-classification benchmarks spanning diverse application domains, and covering three experimental regimes:

- *Controlled full-domain* regime includes Questions (Platonov et al., 2023) (48 921 nodes), Amazon Ratings (Platonov et al., 2023) (24 492), and Cora Full (Bojchevski & Günnemann, 2018) (19 793), characterized by low mean degree (6.28-7.60) and limited lifting expansion ($\leq$ 2.60 cellular and $\leq$ 4.52 simplicial 2-cells/node).
- *Intermediate-density* regime is represented by Coauthor Physics (Shchur et al., 2019) (34,493 nodes; mean degree 14.38; 13.58 simplicial 2-cells/node).
- *Large-scale* regime comprises Reddit (Hamilton et al., 2017b) (232 965 nodes) and OGBN Products (Hu et al., 2020) (2 449 029 nodes), both containing approximately 60M edges. These datasets expose complementary scaling regimes: Reddit combines high connectivity (mean degree 491.99) with substantial structural expansion (174.20 cellular and 35 886.67 simplicial 2-cells/node), whereas OGBN Products primarily stresses node capacity at moderate expansion (mean degree 50.52; 13.98 cellular and 253.60 simplicial 2-cells/node).

Models. We compare a graph-based GCN model (Kipf & Welling, 2017) with representative higher-order models from three domains: hypergraph models EDHNN (Wang et al., 2023) and UniGNN (Huang & Yang, 2021); cellular models CWN (Bodnar et al., 2021) and TopoTune (Papillon et al., 2025); and simplicial models SCN (Ebli et al., 2020) and SCCNN (Yang & Isufi, 2023).

Setup. For all Cluster-TNN experiments, we use the same mini-batch lifting workflow across models, removing differences from model-specific batching implementations. We tune the full-graph and partitioned training settings separately and use the same generated random node splits across models and settings (Appendix E). For each configuration, we train the model with five random seeds and report test performance as mean $\pm$ standard deviation. The controlled full-domain regime enables direct comparison between full-graph and Cluster-TNN training. For the intermediate-density and large-scale regimes, full-domain execution of at least one higher-order TNN exceeds the available resources. Therefore, we evaluate those regimes using Cluster-TNN training. Appendix B provides a detailed analysis of the resulting resource requirements, including contributions from structural expansion, feature storage, and model-required operators.

4.2 COMPUTATIONAL EFFICIENCY

To address Q1, we primarily evaluate the memory savings enabled by Cluster-TNN execution, while also reporting its effects on preprocessing cost and total runtime. In the controlled full-domain regime Cluster-TNN reduces run-wide peak GPU memory in all 21 comparisons with full-graph execution, with a mean reduction of 83.2% and a median reduction of 86.7%. Comparison-level reductions range from 46.4% to 98.7%, based on the reported means across five seeds (Figure 3 and Table 7). Furthermore, Figure 5 separates CPU preprocessing peaks from GPU training and evaluation peaks. As an example, for CWN on Cora Full, Cluster-TNN reduces mean peak GPU memory from 24.1 to 3.7 GB (Table 7), while total runtime increases from 942.8 to 1 059.8 s (Table 6).

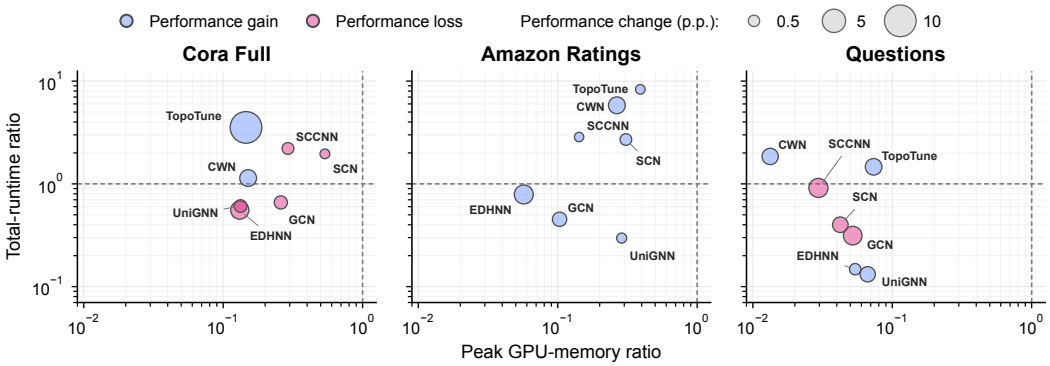


Figure 3: Cluster-TNN reduces peak GPU memory across all model-dataset pairs. Each circle represents one model, positioned by the mean ratios of Cluster-TNN to full-graph peak GPU memory and total runtime across five seeds. Moving left indicates memory savings, whereas moving down indicates shorter runtime. The lower-left region marks improvements in both measures. Marker size represents the absolute test-performance difference in percentage points, and color distinguishes performance gains from losses. Dashed lines mark parity with full-graph execution.

Memory savings are particularly important for computationally expensive topological configurations, where full-domain preprocessing can itself exceed practical memory budgets. On Reddit, storing full-graph simplicial features alone is estimated to require 20.3 TB (Table 4). Full-graph preprocessing for TopoTune reaches 480.1 GB peak CPU memory, compared with 11.9 GB for Cluster-TNN using $K = 7\,000$ partitions (Table 5), with K selected through hyperparameter optimization (Table 13). The associated preprocessing time reduces from 6 965.2 to 496.3 s.

At fixed hidden width of 512 hidden channels, peak GPU memory grows substantially more slowly with network depth under Cluster-TNN for both TopoTune and SCCNN (Figure 4). At depth 16, their mean peaks remain below 20 GB, compared with approximately 69–80 GB for full-graph execution, showing that the memory advantage extends to deeper architectures.

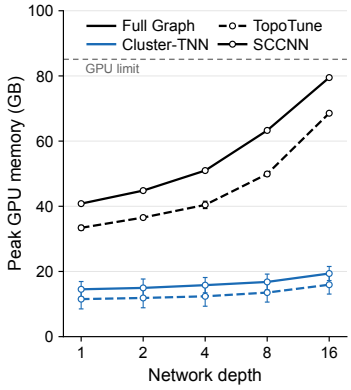


Figure 4: GPU memory scaling on Cora Full with 512 hidden channels for $K = 64$ and $q = 8$. Peak memory is reported as mean $\pm$ standard deviation across five seeds. Network depth counts TopoTune blocks and SCCNN layers.

Cluster-TNN also reduces preprocessing cost in most cases. Across all 21 comparisons, preprocessing time decreases in 16, increases in four, and remains unchanged in one. Among 16 reductions, the mean is 53.9% and median 53.1%, including the graph-partitioning cost (Table 6). Thus, the memory reduction is not solely achieved by shifting cost to preprocessing. Partitioned execution also reaches the best validation checkpoint in fewer epochs in 19 of 21 comparisons (Table 9). For EDHNN on Questions, the mean selected-checkpoint epoch decreases from 211 to 42, while mean test AUROC increases from 75.73 to 76.23.

Total runtime effects are mixed, as Cluster-TNN reduces runtime in 11 of 21 comparisons while increases it in 10, with the mean increase driven by a few substantial slowdowns (Table 6). GCN and all hypergraph runs are faster, all cellular runs slower, and simplicial runs faster only on Questions. Thus, while Cluster-TNN consistently reduces memory usage across model-dataset pairs, its runtime depends on the selected TNN architecture.

4.3 Feasibility and Predictive Performance

Table 1 addresses the feasibility aspect of Q1 and the predictive-performance comparison in Q2. It compares Cluster-TNN and full-graph training to assess whether Cluster-TNN preserves predictive performance while enabling larger-scale training where full-graph execution is not possible.

Table 1: Test performance under full-graph and Cluster-TNN settings. We report AUROC for Questions and accuracy for all other datasets. Cluster-TNN results use ensemble mini-batch inference with ten cluster recombinations, contrary to full-graph inference. Values are mean ± sample standard deviation across five seeds. Boldface indicates best-performing model for each setting.

	Cora Full		Amazon Ratings		Questions		Coauthor Physics	Reddit	OGBN Products
Model	**Full-graph**	**Cluster-TNN**	**Full-graph**	**Cluster-TNN**	**Full-graph**	**Cluster-TNN**	**Cluster-TNN**	**Cluster-TNN**	**Cluster-TNN**
GCN	70.18 ± 0.42	69.19 ± 0.66	48.29 ± 1.31	49.62 ± 0.96	76.69 ± 1.18	73.83 ± 2.01	$\mathbf{96.68 \pm 0.08}$	$\mathbf{92.62 \pm 0.10}$	85.81 ± 0.04
EDHNN	69.74 ± 0.89	67.06 ± 1.19	48.41 ± 1.18	$\mathbf{51.41 \pm 0.72}$	75.73 ± 0.92	$\mathbf{76.23 \pm 1.05}$	96.31 ± 0.32	92.17 ± 0.73	86.51 ± 0.25
UniGNN	68.44 ± 0.79	67.61 ± 0.56	47.05 ± 1.76	47.24 ± 0.70	69.44 ± 1.68	71.09 ± 0.98	94.97 ± 0.37	91.28 ± 0.16	85.87 ± 0.07
CWN	60.50 ± 0.60	62.66 ± 1.11	44.07 ± 0.64	46.25 ± 0.67	68.21 ± 3.48	70.20 ± 0.88	95.80 ± 0.24	91.29 ± 0.18	83.93 ± 0.06
TopoTune	46.40 ± 0.47	56.45 ± 1.42	43.18 ± 1.03	43.27 ± 0.68	71.53 ± 1.73	73.61 ± 1.88	95.22 ± 0.20	89.55 ± 0.09	83.79 ± 0.55
SCN	70.45 ± 0.35	$\mathbf{70.32 \pm 0.46}$	50.08 ± 0.36	50.65 ± 0.64	74.77 ± 1.34	73.04 ± 1.84	96.22 ± 0.22	91.80 ± 0.13	87.03 ± 0.05
SCCNN	$\mathbf{70.81 \pm 0.58}$	70.20 ± 0.29	$\mathbf{51.20 \pm 1.10}$	51.24 ± 0.79	$\mathbf{77.14 \pm 0.99}$	74.10 ± 1.85	96.00 ± 0.25	92.45 ± 0.25	$\mathbf{88.63 \pm 0.05}$

Across 21 model-dataset pairs for which results are available under both training settings, Cluster-TNN training achieves a higher reported mean of the corresponding test metric in 13 comparisons, while full-graph training performs better in the remaining eight. On Cora Full, partitioned training yields a slightly better result for CWN and a notably better result for TopoTune compared with full-graph training. On Amazon Ratings, partitioned training achieves better results for all seven evaluated models, with particularly pronounced differences for EDHNN and CWN. Similarly, on Questions, partitioned training yields better performance for four of the evaluated models. Notably, across all three datasets, the best-performing model is one of the higher-order TNNs under both full-graph and partitioned training. These results highlight two observations. First, they demonstrate the potential benefits of higher-order TNNs over conventional GNN models on these datasets. Second, they show that Cluster-TNN can achieve predictive performance exceeding that of full-graph training across a range of model-dataset pairs.

Experiments in the intermediate-density regime, represented by the Coauthor Physics dataset, provide an initial indication of the advantages of Cluster-TNN. In this setting, Cluster-TNN produces results comparable in performance across multiple TNN models to that of the GCN baseline. However, none of the TNN models was able to outperform the GCN baseline.

Experiments in the large-scale regime further demonstrate the practical advantages of Cluster-TNN. The workflow enables, to our knowledge, the first memory-efficient training of multiple higher-order TNN models on datasets at this scale. On Reddit, we successfully train higher-order TNNs across different domains while achieving accuracies comparable to the GCN baseline. Notably, hypergraph and simplicial models outperform the GCN baseline on OGBN Products, while the remaining higher-order TNN models achieve performance comparable to the baseline. These results demonstrate that Cluster-TNN enables the application of higher-order TNNs to large-scale datasets while also achieving competitive performance in this regime.

4.4 Ablation studies

To address Q3, in Figure 2 and Appendix C.13.1 we quantify structural recovery under cluster recombination. The ablations below examine how partitioning, cross-cluster edges, and inference strategy affect predictive performance. Table 2 compares METIS with random partitioning, as well as our induced-subgraph construction with a variant that removes edges between co-sampled clusters. Also, we compare a single-pass mini-batch inference, an ensemble mini-batch inference that averages node predictions over ten independent cluster recombinations, and full-graph inference. All ablation experiments were performed in the controlled full-domain regime.

Partitioning method. We see from Table 2 that METIS yields the higher mean test metric in 15 of the 21 model-dataset comparisons, while random partitioning yields the higher mean in the remaining six. The cases where random partitioning outperforms METIS partitioning are not unexpected, since Wang et al. (2026) recently showed that training using random partitioning scheme can even outperform full-graph training.

Table 2: Partitioning and inference ablations (shaded columns). *Partitioner* compares METIS with random partitioning, and *Edges* compares preserving with removing cross-cluster edges. *Ensemble* and *Full-graph* compare the respective inference protocols with single-pass mini-batch inference. Entries report mean paired differences in test performance (percentage points) over five seeds.

	Cora Full				Amazon Ratings				Questions			
Model	**Partitioner**	**Edges**	**Ensemble**	**Full-graph**	**Partitioner**	**Edges**	**Ensemble**	**Full-graph**	**Partitioner**	**Edges**	**Ensemble**	**Full-graph**
GCN	$+0.98$	$+0.08$	$+2.51$	$+3.44$	$+2.24$	$+0.25$	$+0.05$	$+0.08$	$+0.52$	$+0.99$	$+0.65$	$+2.71$
EDHNN	$+6.28$	$+1.72$	$+1.55$	$+2.59$	$+2.88$	-0.44	$+0.45$	$+0.33$	$+1.25$	$+1.04$	$+0.42$	$+0.22$
UniGNN	$+6.45$	$+0.63$	$+0.04$	$+0.30$	$+2.47$	-0.43	$+0.33$	$+0.34$	-1.01	$+1.10$	$+1.35$	$+0.99$
CWN	$+1.91$	$+0.66$	$+2.57$	$+5.37$	$+0.18$	$+0.23$	$+0.10$	$+0.08$	-0.09	$+0.56$	$+0.14$	$+0.15$
TopoTune	-3.86	$+10.01$	$+4.65$	$+4.75$	-2.40	$+0.41$	$+1.56$	$+0.48$	$+1.15$	$+1.49$	$+0.30$	-0.58
SCN	$+1.78$	$+0.42$	$+1.27$	$+1.78$	$+2.51$	$+0.27$	$+0.71$	$+0.76$	-0.61	$+1.27$	$+0.34$	$+1.54$
SCCNN	$+5.32$	$+0.84$	$+1.40$	$+2.22$	$+2.99$	$+0.20$	$+0.31$	$+0.51$	-0.18	$+1.07$	$+1.37$	$+2.16$

Cross-cluster edges. Preserving cross-cluster edges yields the higher mean test metric in 19 of the 21 comparisons, supporting the use of the full induced subgraph of each sampled cluster group in the evaluated settings.

Inference strategy. The evaluation columns show that ensemble mini-batch inference yields higher mean than a single-pass in all 21 comparisons, while full-graph inference does so in 20 of 21. These descriptive results indicate that repeated cluster recombinations can recover useful predictive context without materializing the full lifted graph, while full-graph inference remains an alternative when memory permits.

5 Conclusion

We propose Cluster-TNN, a mini-batch lifting workflow for graph-supported TDL that partitions the graph once and reconstructs and lifts only the induced subgraph of each sampled cluster group while preserving cross-cluster edges. Under the whole-partition sampling and lifting-compatibility assumptions in Appendix C, structure recovery depends on the structure's cluster span relative to the mini-batch size q, with expected coverage approaching one under repeated independent recombinations when the span does not exceed q. By reusing the TopoBench data path across compatible higher-order domains and TNN architectures, Cluster-TNN avoids model-specific batching code and reduces peak GPU memory by 83.2% on average across the evaluated full-graph comparisons while retaining competitive predictive performance. Cluster-TNN also enables, to the best of our knowledge, the first training of multiple higher-order TNNs on large-scale graph benchmarks, including Reddit and OGBN Products, extending TDL beyond the limits of global domain construction.

Cluster-TNN inherits the computational cost of the underlying lifting procedures, which can remain substantial even when applied to smaller induced subgraphs. More efficient lifting algorithms could therefore further improve scalability. Moreover, our current implementation reconstructs lifted mini-batches without caching structures encountered across cluster recombinations. Evaluating static caches, on-demand caches, and hybrid recombination policies provides a natural direction for reducing repeated construction costs. Finally, Cluster-TNN targets graph-supported TDL, where higher-order domains are constructed from an underlying graph. Extending the framework to datasets provided natively as higher-order domains would require partitioning and recombination mechanisms operating directly on these domains. Such extensions remain comparatively underexplored, in part because native higher-order benchmarks are scarce and scalable partitioning methods for higher-order domains still emerging. Developing these mechanisms alongside suitable large-scale benchmarks represents an important direction for future work.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631. Association for Computing Machinery, 2019. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

Guillermo Bernárdez, Marco Montagna, Louis Van Langendonck, Martin Carrasco, Amirreza Akbari, Louisa Cornelis, Mathilde Papillon, Pere Barlet-Ros, Nina Miolane, and Lev Telyatnikov. HOPSE: Scalable Higher-Order Positional and Structural Encoder for Combinatorial Representations, 2025. URL https://arxiv.org/abs/2505.15405.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael Bronstein. Weisfeiler and Lehman Go Cellular: CW Networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 2625–2640, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/157792e4abb490f99dbd738483e0d2d4-Abstract.html.

Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velikovi. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, 2021. URL https://arxiv.org/abs/2104.13478.

Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rytstxWAW.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 257–266, 2019. doi: 10.1145/3292500.3330925. URL https://doi.org/10.1145/3292500.3330925.

Gabriele Corso, Hannes Stark, Stefanie Jegelka, Tommi Jaakkola, and Regina Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4(17), 2024. doi: 10.1038/s43586-024-00294-7. URL https://doi.org/10.1038/s43586-024-00294-7.

Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial Neural Networks. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. URL https://openreview.net/forum?id=nPCt39DVIfk.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Sravanthi Gurugubelli and Sundeep Prabhakar Chepuri. SaNN: Simple Yet Powerful Simplicial-aware Neural Networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=eUgS9Ig8JG.

Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological Deep Learning: Going Beyond Graph Data, 2023. URL https://arxiv.org/abs/2206.00606.

Mustafa Hajij, Mathilde Papillon, Florian Frantzen, Jens Agerberg, Ibrahem AlJabea, Rubén Ballester, Claudio Battiloro, Guillermo Bernárdez, Tolga Birdal, Aiden Brent, Peter Chin, Sergio Escalera, Simone Fiorellino, Odin Hoff Gardaa, Gurusankar Gopalakrishnan, Devendra Govil, Josef Hoppe, Maneel Reddy Karri, Jude Khouja, Manuel Lecha, Neal Livesay, Jan MeiÃner, Soham Mukherjee, Alexander Nikitin, Theodore Papamarkou, Jaro Prílepok, Karthikeyan Natesan Ramamurthy, Paul Rosen, Aldo Guzmán-Sáenz, Alessandro Salatiello, Shreyas N. Samaga, Simone Scardapane, Michael T. Schaub, Luca Scofano, Indro Spinelli, Lev Telyatnikov, Quang Truong, Robin Walters, Maosheng Yang, Olga Zaghen, Ghada Zamzmi, Ali Zia, and Nina Miolane. TopoX: A Suite of Python Packages for Machine Learning on Topological Domains. *Journal of Machine Learning Research*, 25(374):1–8, 2024. URL http://jmlr.org/papers/v25/24-0110.html.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.*, 40:52–74, 2017a. URL https://api.semanticscholar.org/CorpusID:3215337.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, 2017b. doi: 10.48550/arXiv.1706.02216. URL https://arxiv.org/abs/1706.02216.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

Jing Huang and Jie Yang. UniGNN: a Unified Framework for Graph and Hypergraph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2563–2569, 2021. doi: 10.24963/ijcai.2021/353. URL https://www.ijcai.org/proceedings/2021/353.

George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. doi: 10.1137/S1064827595287997. URL https://doi.org/10.1137/S1064827595287997.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Fengcheng Lu and Michael Kwok-Po Ng. FastHGNN: A New Sampling Technique for Learning with Hypergraph Neural Networks. *ACM Trans. Knowl. Discov. Data*, 18(8), July 2024. ISSN 1556-4681. doi: 10.1145/3663670. URL https://doi.org/10.1145/3663670.

Marco Montagna, Simone Scardapane, and Lev Telyatnikov. Topological deep learning with state-space models: A mamba approach for simplicial complexes. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2025.

Theodore Papamarkou, Tolga Birdal, Michael Bronstein, Gunnar Carlsson, Justin Curry, Yue Gao, Mustafa Hajij, Roland Kwitt, Pietro Liò, Paolo Di Lorenzo, Vasileios Maroulas, Nina Miolane, Farzana Nasrin, Karthikeyan Natesan Ramamurthy, Bastian Rieck, Simone Scardapane, Michael T. Schaub, Petar Veličković, Bei Wang, Yusu Wang, Guo-Wei Wei, and Ghada Zamzmi. Position: Topological Deep Learning is the New Frontier for Relational Learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39529–39555. PMLR, 2024. URL https://proceedings.mlr.press/v235/papamarkou24a.html.

Mathilde Papillon, Sophia Sanborn, Mustafa Hajij, and Nina Miolane. Architectures of Topological Deep Learning: A Survey of Message-Passing Topological Neural Networks, 2024. URL https://arxiv.org/abs/2304.10031.

Mathilde Papillon, Guillermo Bernárdez, Claudio Battiloro, and Nina Miolane. TopoTune: A Framework for Generalized Combinatorial Complex Neural Networks. In *Proceedings of the*

42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, pp. 47924–47952. PMLR, 2025. URL https://proceedings.mlr.press/v267/papillon25a.html.

Keith Paton. An Algorithm for Finding a Fundamental Set of Cycles of a Graph. *Communications of the ACM*, 12(9):514–518, sep 1969. doi: 10.1145/363219.363232. URL https://doi.org/10.1145/363219.363232.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A Critical Look at the Evaluation of GNNs Under Heterophily: Are We Really Making Progress? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

Karthikeyan Natesan Ramamurthy, Aldo Guzmán-Sáenz, and Mustafa Hajij. Topo-MLP : A Simplicial Network Without Message Passing, 2023. URL https://arxiv.org/abs/2312.11862.

Bastian Rieck. Have Graph – Will Lift? The Case for Higher-Order Benchmarks, 2026. URL https://arxiv.org/abs/2605.07397.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation, 2019. URL https://arxiv.org/abs/1811.05868.

Lev Telyatnikov, Guillermo Bernardez, Marco Montagna, Mustafa Hajij, Martin Carrasco, Pavlo Vasylenko, Mathilde Papillon, Ghada Zamzmi, Michael T Schaub, Jonas Verhellen, Pavel Snopov, Bertran Miquel-Oliver, Manel Gil-Sorribes, Alexis Molina, Victor Guallar, Theodore Long, Julian Suk, Patryk Rygiel, Alexander V Nikitin, Giordan Escalona, Michael Banf, Dominik Filipiak, Liliya Imasheva, Max Schattauer, Alvaro L. Martinez, Halley Fritze, Marissa Masden, Valentina Sánchez, Manuel Lecha, Andrea Cavallo, Claudio Battiloro, Matthew Piekenbrock, Mauricio Tec, George Dasoulas, Nina Miolane, Simone Scardapane, and Theodore Papamarkou. TopoBench: A Framework for Benchmarking Topological Deep Learning. *Journal of Data-centric Machine Learning Research*, 2025a. URL https://openreview.net/forum?id=07sTzyEVtY.

Lev Telyatnikov, Maria Sofia Bucarelli, Guillermo Bernardez, Olga Zaghen, Simone Scardapane, and Pietro Lio. Hypergraph Neural Networks through the Lens of Message Passing: A Common Perspective to Homophily and Architecture Design. *Transactions on Machine Learning Research*, 2025b. ISSN 2835-8856. URL https://openreview.net/forum?id=8rxtL0kZnX.

Clement Wang, Antoine Vialle, Robin Vaysse, and Thomas Bonald. Implicit Regularization of Mini-Batch Training in Graph Neural Networks, 2026. URL https://arxiv.org/abs/2605.22480.

Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant Hypergraph Diffusion Neural Operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=RiTjKoscnNd.

Shuhei Watanabe. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. *arXiv preprint arXiv:2304.11127*, 2023. doi: 10.48550/arXiv.2304.11127. URL https://arxiv.org/abs/2304.11127.

Maosheng Yang and Elvin Isufi. Convolutional Learning on Simplicial Complexes, 2023. URL https://arxiv.org/abs/2301.11163.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

APPENDIX CONTENTS

Reproducibility statement . . . 14
AI use statement . . . 14
Ethics statement . . . 14
Notation guide . . . 14
A Featured topological domains and lifting maps . . . 14
A.1 Featured topological domains . . . 14
A.2 Lifting maps . . . 15
A.2.1 Lifting maps used in the experiments . . . 16
B Practical scalability of graph-supported TDL: bottlenecks and solution paths . . . 17
B.1 Pipeline view of graph-supported TDL scalability . . . 17
B.2 Representation growth: from structure to storage . . . 17
B.2.1 Lifting-specific mechanisms of structural expansion . . . 17
B.2.2 Benchmark scale regimes and coverage . . . 18
B.2.3 Static feature-matrix storage . . . 19
B.3 Full-domain CPU preprocessing: construction time and memory . . . 20
B.4 Phase-specific memory during preprocessing and TNN execution . . . 21
B.5 End-to-end cost . . . 22
B.5.1 End-to-end runtime . . . 23
B.5.2 Run-wide peak memory . . . 24
B.6 Structural exposure and statistical efficiency . . . 26
B.7 Solution paths and practical guidance . . . 27
C Structural reconstruction . . . 28
C.1 Setup and notation . . . 28
C.2 Per-epoch recovery probability . . . 29
C.3 Repeated structural exposure . . . 30
C.4 Cumulative recovery probability . . . 31
C.5 Structural coverage convergence . . . 31
C.6 Repeated-exposure structural coverage . . . 32
C.7 Mean structural exposure rate . . . 33
C.8 Exposure epoch distribution . . . 33
C.9 Direct-incidence message availability . . . 35
C.10 Repeated direct-incidence message exposure . . . 36
C.11 Layerwise incidence aggregation . . . 37
C.12 Cumulative recovery entropy . . . 38
C.13 Empirical analysis of structural recovery . . . 39
C.13.1 Structural recovery across datasets . . . 39
C.13.2 Recovery across epochs on Cora Full . . . 41
D Additional results . . . 42
E Experimental protocol and hyperparameter optimization . . . 43

REPRODUCIBILITY STATEMENT

Sections 3.1 and 3.2 describe mini-batch construction, training, and inference. Appendix A.2.1 specifies the evaluated liftings, and Appendix E reports the experimental protocol, hyperparameter search spaces, and selected configurations. Appendix C.13.1 describes the structural-recovery experiments. Experiments were conducted on a Linux machine equipped with dual AMD EPYC 7543 processors, 64 physical CPU cores, 1 TB of system memory, and 8 NVIDIA A100 GPUs, each with 80 GiB of GPU memory. To facilitate reproducibility, we make our code implementation available at `https://anonymous.4open.science/r/cluster-tnn-233C/`.

AI USE STATEMENT

In this work, AI assistance was used as a supplementary tool during the research and writing process. Specifically, AI tools were used to polish and improve the clarity, grammar, and organization of the manuscript, identify and locate potentially relevant related work, assist in drafting and restructuring selected sections, and improve the presentation and exposition of some mathematical proofs. All AI-assisted content was reviewed, revised, and verified by the authors, who remain responsible for the accuracy, originality, interpretation, and final presentation of the work. AI tools were not used as a substitute for the authors' scientific judgment, experimental analysis, or verification of the reported results.

ETHICS STATEMENT

The work presented in this paper aims to advance the field of Topological Deep Learning. While it may have broader societal implications, we do not identify any that warrant specific discussion here.

NOTATION GUIDE

The table collects the symbols used across multiple appendix sections. Symbols introduced only inside individual proofs remain local to those proofs.

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, F_{\mathcal{V}}, F_{\mathcal{E}})$	Featured input graph used by the evaluated liftings.
$\mathcal{T} = (\mathcal{X}, F)$	Featured topological domain.
$\mathcal{X}_r$, F_r, $\mathbf{X}_r$, d_r	Rank-r elements, feature map, feature matrix, and feature width, respectively.
$\mathcal{L}$, $\mathcal{L}_S$, $\mathcal{L}_F$	Complete lifting and its structural and feature components, respectively.
$\mathbf{B}_{r,r+1}$, $\mathbf{H}$	Incidence matrix between ranks r and $r+1$, and node-hyperedge incidence matrix, respectively.
K	Total number of clusters in the fixed graph partition.
$\Pi = \{P_k\}_{k=1}^{K}$	Fixed partition of the graph nodes.
q, Z, B, $\mathcal{G}_B$	q is the number of clusters per mini-batch. Z is a set of cluster indices, $B = \bigcup_{k \in Z} P_k$ is the resulting node set, and $\mathcal{G}_B$ is the graph induced by B.

A FEATURED TOPOLOGICAL DOMAINS AND LIFTING MAPS

We adopt the featured-domain and lifting formalism of TopoBench (Telyatnikov et al., 2025a), restricting the presentation to the objects used in this study. The input to every lifting is a featured graph, and the output is a featured graph, hypergraph, simplicial complex, or cell complex.

A.1 FEATURED TOPOLOGICAL DOMAINS

Definition 1 (Featured graph)**.** *A featured graph is a tuple*

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, F_{\mathcal{V}}, F_{\mathcal{E}}), \qquad F_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}^{d_{\mathcal{V}}}, \qquad F_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}^{d_{\mathcal{E}}}.$$

Here, $(\mathcal{V}, \mathcal{E})$ specifies the underlying graph, with $\mathcal{V}$ being the set of nodes and $\mathcal{E}$ the set of edges. While $F_{\mathcal{V}}$ and $F_{\mathcal{E}}$ assign the node and edge features, respectively. We omit $F_{\mathcal{E}}$ when the edge features are not available.

Topological domains. A topological domain is a discrete representation of a system in terms of its constituent elements and the relations among them. Beyond pairwise graph edges, these relations may connect several nodes or organize elements through containment, incidence, or boundary relations. The term therefore provides a common language for graphs and their higher-order generalizations (Hajij et al., 2023; Papillon et al., 2024; Telyatnikov et al., 2025a).

Higher-order domains. We consider three families of higher-order domains. Hypergraphs represent set-type relations by allowing a hyperedge to connect any number of nodes, without imposing a hierarchy among hyperedges. Simplicial complexes add a strict part-whole hierarchy in which every simplex contains all of its lower-dimensional faces. Cell complexes retain a boundary-based hierarchy but admit non-simplicial cells, including polygonal faces that can represent graph cycles directly (Hajij et al., 2023; Papillon et al., 2024; Telyatnikov et al., 2025a).

Definition 2 (Hypergraph)**.** *Let $\mathcal{P}(\mathcal{V})$ denote the power set of a non-empty node set $\mathcal{V}$. A hypergraph on $\mathcal{V}$ is a pair $H = (\mathcal{V}, \mathcal{E}_H)$, where $\mathcal{E}_H \subseteq \mathcal{P}(\mathcal{V}) \setminus \{\emptyset\}$.*

Definition 3 (Simplicial complex)**.** *A simplicial complex on a non-empty node set $\mathcal{V}$ is a pair $S = (\mathcal{V}, \mathcal{X})$, where $\mathcal{X} \subseteq \mathcal{P}(\mathcal{V}) \setminus \{\emptyset\}$ and*

$$\sigma \in \mathcal{X}, \quad \emptyset \neq \tau \subseteq \sigma \quad \Longrightarrow \quad \tau \in \mathcal{X}.$$

The elements of $\mathcal{X}$ are called simplices.

Definition 4 (Cell complex)**.** *A regular cell complex is a topological space S partitioned into cells $\{x_\alpha\}_{\alpha \in P_S}$, where P_S is an index set, such that:*

1. *$S = \bigcup_{\alpha \in P_S} \operatorname{int}(x_\alpha)$.*
2. *for every $\alpha \in P_S$, the interior of x_α is homeomorphic to $\mathbb{R}^{n_\alpha}$ for some $n_\alpha \in \mathbb{Z}_{\geq 0}$, and n_α is the dimension of x_α.*
3. *the boundary of each cell is a union of finitely many cells of strictly lower dimension.*

The experimental liftings instantiate these domains using node neighbourhoods as hyperedges, cliques as simplicial cells, and graph cycles as cellular 2-cells, as detailed in Appendix A.2.1.

Definition 5 (Featured topological domain)**.** *Let $\mathcal{T} = (\mathcal{X}, F)$, where $\mathcal{X}$ is a topological domain, and*

$$F = \{F_r\}_{r \geq 0}, \qquad F_r : \mathcal{X}_r \to \mathbb{R}^{d_r}$$

assigns feature vectors to its rank-r elements. For hypergraphs the feature maps instead distinguish nodes and hyperedges.

A.2 Lifting maps

Definition 6 (Lifting between featured topological domains)**.** *Let $\mathcal{T}_{\text{src}} = (\mathcal{X}_{\text{src}}, F_{\text{src}})$ be the featured source domain and $\mathcal{T}_{\text{tgt}} = (\mathcal{X}_{\text{tgt}}, F_{\text{tgt}})$ the featured target domain. A lifting $\mathcal{L} : \mathcal{T}_{\text{src}} \to \mathcal{T}_{\text{tgt}}$ consists of a structural map $\mathcal{L}_S$ and a feature map $\mathcal{L}_F$:*

$$\mathcal{X}_{\text{tgt}} = \mathcal{L}_S(\mathcal{X}_{\text{src}}, F_{\text{src}}), \qquad F_{\text{tgt}} = \mathcal{L}_F(\mathcal{X}_{\text{src}}, F_{\text{src}}), \qquad \mathcal{L}(\mathcal{T}_{\text{src}}) = \mathcal{T}_{\text{tgt}}. \tag{1}$$

The structural map constructs the elements and incidence relations of $\mathcal{X}_{\text{tgt}}$, while the feature map transfers or constructs their features F_{tgt}.

This definition follows the lifting formalism of TopoBench (Telyatnikov et al., 2025a). In our experiments, $\mathcal{T}_{\text{src}}$ is a featured graph and $\mathcal{T}_{\text{tgt}}$ is the representation required by the model. For example, the clique-complex lifting constructs simplicial cells from graph cliques and assigns features to the resulting elements. All structural liftings used in this study derive the target domain from graph connectivity.

For $B \subseteq \mathcal{V}$, the featured graph provided to a mini-batch lifting is

$$\mathcal{G}_B = \left(B, \mathcal{E}_B, F_{\mathcal{V}}|_B, F_{\mathcal{E}}|_{\mathcal{E}_B}\right), \qquad \mathcal{E}_B = \{(u,v) \in \mathcal{E} : u, v \in B\},$$

with the edge-feature term omitted when unavailable. During mini-batch training and inference, the workflow constructs $\mathcal{L}(\mathcal{G}_B)$ directly without first materializing $\mathcal{L}(\mathcal{G})$.

A.2.1 Lifting maps used in the experiments

The experiments instantiate each lifting on a featured graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, F_{\mathcal{V}}, F_{\mathcal{E}})$. We first specify its structural component $\mathcal{L}_S$. The identity lifting preserves the input features, whereas the higher-order liftings use a common feature map that assigns features to the constructed elements. Each construction applies to either the full graph or an induced mini-batch graph, with $\mathcal{G}_B$ substituted for $\mathcal{G}$. The lifting-specific scale parameters and their experimental values are stated below.

Identity graph lifting. The identity lifting preserves the graph and its features,

$$\mathcal{L}_{\mathrm{id}}(\mathcal{G}) = \mathcal{G}.$$

We use it with GCN.

k-hop neighbourhood hypergraph lifting. The k-hop lifting assigns a hyperedge to the closed k-hop neighbourhood of each node (Telyatnikov et al., 2025a). For an integer $k \geq 1$ and a node $v \in \mathcal{V}$, define

$$N_{\mathcal{G}}^{(k)}[v] = \{u \in \mathcal{V} : \mathrm{dist}_{\mathcal{G}}(u,v) \leq k\},$$

where $\mathrm{dist}_{\mathcal{G}}$ denotes shortest-path distance in $\mathcal{G}$. The resulting node-indexed hyperedge family and structural lifting are

$$\mathcal{E}_H^{(k)} = \left(e_v\right)_{v \in \mathcal{V}}, \qquad e_v = N_{\mathcal{G}}^{(k)}[v], \qquad \mathcal{L}_{S,k}^{\mathrm{hypergraph}}(\mathcal{G}) = \left(\mathcal{V}, \mathcal{E}_H^{(k)}\right).$$

The center-node index is retained, so the implementation creates exactly one incidence-matrix column per node even when two indexed hyperedges have identical memberships. The parameter k determines the graph radius represented by each hyperedge. We set $k = 1$ and use this lifting with EDHNN and UniGNN.

Cycle-basis cell lifting. Cycle-based cellular liftings attach 2-cells along selected graph cycles (Bodnar et al., 2021; Telyatnikov et al., 2025a). Let $\mathcal{B}(\mathcal{G})$ denote a fundamental cycle basis obtained using Paton's algorithm (Paton, 1969). For a maximum cycle length $\ell_{\max}$, we retain

$$\mathcal{B}_{\leq \ell_{\max}}(\mathcal{G}) = \{c \in \mathcal{B}(\mathcal{G}) : |c| \leq \ell_{\max}\}.$$

Attaching one 2-cell along each retained cycle gives

$$\mathcal{L}_{S,\ell_{\max}}^{\mathrm{cell}}(\mathcal{G}) = (\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2), \qquad \mathcal{C}_0 = \mathcal{V}, \qquad \mathcal{C}_1 = \mathcal{E}, \qquad \mathcal{C}_2 = \mathcal{B}_{\leq \ell_{\max}}(\mathcal{G}).$$

We set $\ell_{\max} = 9$ and use this lifting with CWN and TopoTune.

Clique-complex lifting. The clique-complex lifting maps every clique of size $r+1$ to an r-simplex (Bodnar et al., 2021; Telyatnikov et al., 2025a). For a maximum simplex dimension $d_{\max}$, define the truncated clique complex

$$\mathrm{Cl}_{\leq d_{\max}}(\mathcal{G}) = \{\emptyset \neq \sigma \subseteq \mathcal{V} : |\sigma| \leq d_{\max}+1, (u,v) \in \mathcal{E} \text{ for all distinct } u, v \in \sigma\}.$$

The corresponding structural lifting is

$$\mathcal{L}_{S,d_{\max}}^{\mathrm{simplicial}}(\mathcal{G}) = (\mathcal{V}, \mathrm{Cl}_{\leq d_{\max}}(\mathcal{G})).$$

We set $d_{\max} = 2$, yielding nodes, edges, and filled triangles. We use this lifting with SCN and SCCNN.

Feature lifting. The preceding maps specify the structural component $\mathcal{L}_S$ of each lifting. In the experiments, the common feature component $\mathcal{L}_F$ retains existing features and constructs missing higher-rank features by summing over incident lower-rank elements (Telyatnikov et al., 2025a). For a ranked domain, let $\mathbf{B}_{r,r+1}$ be the incidence matrix between rank-r and rank-$(r+1)$ elements. The feature assignment from $\mathbf{X}_r$ to $\mathbf{X}_{r+1}$ is

$$\mathbf{X}_{r+1} = |\mathbf{B}_{r,r+1}|^{\mathsf{T}} \mathbf{X}_r.$$

The absolute value removes orientation signs, and the operation is applied successively across ranks. For a hypergraph with node-hyperedge incidence matrix $\mathbf{H}$, the corresponding assignment is $\mathbf{X}_H = |\mathbf{H}|^{\mathsf{T}}\mathbf{X}_0$, where $\mathbf{X}_0$ is the input node-feature matrix. Each new element therefore receives the sum of the features on its incident source elements. The feature assignment does not alter the structure produced by $\mathcal{L}_S$.

B Practical scalability of graph-supported TDL: bottlenecks and solution paths

B.1 Pipeline view of graph-supported TDL scalability

The scalability of graph-supported TDL depends on the entire computational pipeline, not only on the TNN architecture. The input graph must first be lifted by identifying higher-order structures and constructing their feature matrices and connectivity operators. In the evaluated implementation, the constructed representation is held in CPU memory and transferred to the GPU as needed for training or inference. A bottleneck at any one of these stages can make an otherwise efficient architecture impractical.

Where these costs occur depends on the execution regime. Full-domain execution constructs the topology and lifts features once during preprocessing, then reuses the complete lifted representation throughout training and evaluation. Cluster-TNN instead partitions the graph during preprocessing and reconstructs and lifts individual induced graph mini-batches during training and inference. This shift trades full-domain storage for repeated local construction and therefore changes preprocessing, execution time, and memory jointly. Both regimes use the same parameterized lifting procedure and structural restrictions. Applying that procedure to induced subgraphs, however, need not produce the restriction of the full-domain lift.

We use the notation introduced in Appendix A. For a lifting $\mathcal{L} = (\mathcal{L}_S, \mathcal{L}_F)$, the stored representation depends on the number of generated elements $|\mathcal{X}_r|$, the number of entries in its incidence or neighbourhood operators, and the feature width d_r at each rank. The structural counts in Table 3 determine the number of feature rows. The feature widths convert those row counts into the storage estimates in Table 4. Table 5 distinguishes models that request different operators from the same lifted domain.

B.2 Representation growth: from structure to storage

Structural expansion and feature storage are related but distinct. We first explain how liftings create higher-order structures and represent their connectivity (Sec. B.2.1). We then compare the structural characteristics of the benchmarks and their lifted domains (Sec. B.2.2). Finally, we combine these structural counts with feature dimensions to estimate feature-matrix storage (Sec. B.2.3).

B.2.1 Lifting-specific mechanisms of structural expansion

Different liftings can produce very different numbers of higher-order structures from the same graph. The connectivity information also varies, including node memberships in hyperedges and the edges forming cell boundaries. We illustrate these sources of growth using the neighbourhood-hypergraph, cycle-based cellular, and clique-complex liftings used in our experiments, which are TopoBench's default graph mappings to the corresponding domains.

Neighbourhood-hypergraph lifting. For the evaluated $k = 1$ neighbourhood-hypergraph lifting, each node defines one indexed hyperedge containing its closed one-hop neighbourhood. The number

of hyperedges is therefore exactly $|\mathcal{V}|$ for every dataset and provides no additional distinction beyond the node count. Connectivity nevertheless determines the incidence size. Across all neighbourhood hyperedges, the lifting stores $|\mathcal{V}| + 2|\mathcal{E}|$ node-hyperedge memberships, corresponding to an average hyperedge size of $1 + 2|\mathcal{E}|/|\mathcal{V}|$.

Cycle-based cellular lifting. The cycle-based cellular lifting first computes a fundamental cycle basis and retains cycles of length at most $\ell_{\max} = 9$, attaching one cellular 2-cell along every retained cycle. Each retained cycle contributes one 2-cell and one nonzero entry in the edge-cell incidence matrix for each edge on its boundary. Its storage requirements therefore depend on both the number and lengths of the retained cycles. The retained cycle basis can also change when the lifting is recomputed on an induced subgraph, as discussed in Appendix C.13.[1]

Clique-complex lifting. The clique-complex lifting fills every graph triangle with a simplicial 2-cell. Triangle-rich graphs can therefore produce far more 2-cells than input nodes or edges. In the stored representation, each generated 2-cell contributes one feature-vector row and three nonzero entries to the edge-cell incidence matrix, one for each boundary edge.

B.2.2 Benchmark scale regimes and coverage

Table 3 reports the numbers of input nodes and edges, together with the numbers of 2-cells generated by the cellular and simplicial liftings.[2] Mean node degree, calculated as $2|\mathcal{E}|/|\mathcal{V}|$, describes how many neighbours an input node has on average. It distinguishes graphs with similar edge counts but substantially different node counts.

For each lifting, the *Per node* column divides the total number of generated 2-cells by the number of input nodes. This shows how large the generated 2-cell set is relative to the original node set. The absolute counts describe the size of the lifted domain, while these ratios allow us to compare structural expansion across graphs of different sizes. They are ratios of total counts, not counts of the cells incident to an average node.

Table 3: Structural characteristics of the input graphs and their lifted domains. The edge column counts each undirected relation once, and mean node degree is derived as $2|\mathcal{E}|/|\mathcal{V}|$. Per-node values are the corresponding 2-cell counts divided by the number of input nodes. Cellular 2-cells are retained basis cycles of length at most nine, whereas simplicial 2-cells are filled graph triangles.

Dataset	Input graph			Cellular 2-cells		Simplicial 2-cells	
	Nodes	Edges	Mean node degree	Count	Per node	Count	Per node
Questions	48 921	153 540	6.28	24 405	0.50	110 209	2.25
Amazon Ratings	24 492	93 050	7.60	63 667	2.60	110 765	4.52
Cora Full	19 793	63 421	6.41	24 306	1.23	48 386	2.44
Coauthor Physics	34 493	247 962	14.38	102 152	2.96	468 550	13.58
Reddit	232 965	57 307 946	491.99	40 582 059	174.20	8 360 338 411	35 886.67
OGBN Products	2 449 029	61 859 140	50.52	34 226 557	13.98	621 082 120	253.60

Input and lifted scale regimes. Questions, Amazon Ratings, and Cora Full have mean node degrees between 6.28 and 7.60. Coauthor Physics reaches 14.38 and produces 13.58 simplicial 2-cells per node, placing it between the smaller controlled benchmarks and the two scalability-oriented cases.

Reddit and OGBN Products expose complementary scalability regimes. OGBN Products contains approximately $10.5\times$ more nodes, yet the two graphs have comparable edge counts. Reddit conse-

[1] The evaluated implementation computes the complete cycle basis before discarding cycles longer than $\ell_{\max}$. The length restriction therefore bounds the retained cell complex, but not necessarily the time or temporary memory required to construct the basis.

[2] For the evaluated $k = 1$ neighbourhood-hypergraph lifting, the hyperedge count is $|\mathcal{V}|$ and the node-hyperedge membership count is $|\mathcal{V}| + 2|\mathcal{E}|$. Both follow from the input-graph columns and are omitted from the table.

quently has a mean node degree of 491.99, compared with 50.52 for OGBN Products. The higher-order constructions also differ in their expansion. Reddit contains 174.20 cellular and 35886.67 simplicial 2-cells per node, whereas OGBN Products contains 13.98 and 253.60, respectively. These datasets therefore contrast greater higher-order expansion per node with a substantially larger input node count at comparable edge counts.

Why the benchmarks are complementary. The same progression gives the datasets complementary experimental roles. Questions, Amazon Ratings, and Cora Full support controlled comparisons between full-domain and Cluster-TNN partitioned execution because every evaluated lifting can be constructed on the complete graph. Coauthor Physics represents an intermediate case in which a moderate node count already produces substantially greater higher-order expansion. Reddit and OGBN Products extend the suite to graphs with tens of millions of edges and contrasting node counts. Together, these datasets support controlled comparisons with full-domain execution and allow us to examine how graph size, connectivity, and lifting-induced structural expansion affect computational cost.

The benchmarks also cover distinct application domains and relation semantics. Questions and Reddit represent online interactions, although Questions connects users through question-answer activity and Reddit connects posts through shared commenters (Platonov et al., 2023; Hamilton et al., 2017b). Cora Full and Coauthor Physics represent scientific-information networks through citation and coauthorship relations (Bojchevski & Günnemann, 2018; Shchur et al., 2019). Amazon Ratings and OGBN Products represent product co-purchasing networks at different scales (Platonov et al., 2023; Hu et al., 2020). Together, these datasets expose the pipeline to graph structures generated by different relational processes while keeping the downstream task fixed as node classification.

B.2.3 Static feature-matrix storage

Here, static feature-matrix storage denotes the persistent rank-wise input feature matrices of the full lifted domain. It excludes incidence tensors, derived operators, temporary construction objects, activations, gradients, and optimizer state.

Feature width provides a further source of variation across the benchmarks. Cora Full and Coauthor Physics have input widths of $8\,710$ and $8\,415$, respectively, compared with 602 for Reddit and 100 for OGBN Products. Their wider features increase storage per generated element, so a smaller graph need not have a proportionally smaller feature-storage requirement.

The memory required for the rank-wise feature matrices follows directly from the structural counts. At rank r, the feature matrix contains $|\mathcal{X}_r|$ rows and d_r values per row. We assume single-precision floating-point storage (float32), which requires four bytes per value. Storing the feature matrices through rank two therefore requires

$$M_{\text{features}} = 4\,(|\mathcal{X}_0|d_0 + |\mathcal{X}_1|d_1 + |\mathcal{X}_2|d_2)\ \text{bytes}. \tag{2}$$

For these estimates, each generated element receives a feature vector with the dataset's input feature width, consistent with the feature assignment described in Appendix A.2.1. For cellular and simplicial domains, this gives $d_0 = d_1 = d_2$ and $M_{\text{features}} = 4d_0 \sum_r |\mathcal{X}_r|$. For hypergraphs, we count the node and hyperedge feature matrices. Table 4 uses the structural counts in Table 3. Other numerical precisions may be appropriate depending on the features and numerical requirements. For unchanged matrix dimensions, storage scales directly with the number of bytes per value. These estimates account only for input feature matrices, not total preprocessing or training memory.

For OGBN Products, substituting $d_0 = d_1 = d_2 = 100$ and the structural counts from Table 3 into Eq. (2) gives 39.414 GB for the cellular feature matrices and 274.156 GB for the simplicial feature matrices.

For Reddit, the cellular calculation is explicit:

$$\begin{aligned} M^{\text{cell}}_{\text{features}} &= 4 \times 602 \times (232\,965 + 57\,307\,946 + 40\,582\,059) \\ &= 236\,280\,111\,760\ \text{bytes} \approx 236.280\ \text{GB}. \end{aligned} \tag{3}$$

Replacing the cycle-cell count with Reddit's 8.36B simplicial 2-cells in the same calculation gives the 20.270 TB simplicial estimate reported in Table 4. Thus, before adding sparse incidence tensors or any neural-network state, the feature matrices alone can require hundreds of gigabytes for the

Table 4: Estimated memory required to store the float32 input feature matrices of each full-domain lifting. Values use the input feature dimension of each dataset. Observed preprocessing memory is reported in Table 5.

Dataset	**Input width** d_0	$k = 1$ **hypergraph**	**Cycle cell complex**	**Clique simplicial complex**
Questions	301	0.118 GB	0.273 GB	0.376 GB
Amazon Ratings	300	0.059 GB	0.217 GB	0.274 GB
Cora Full	8 710	1.379 GB	3.746 GB	4.585 GB
Coauthor Physics	8 415	2.322 GB	12.946 GB	25.279 GB
Reddit	602	1.122 GB	236.280 GB	20.270 TB
OGBN Products	100	1.959 GB	39.414 GB	274.156 GB

cellular lifting and tens of terabytes for the simplicial lifting. This estimate results from storing a 602-dimensional float32 feature vector for every node, edge, and generated higher-order element.

B.3 Full-domain CPU preprocessing: construction time and memory

The analytical estimate above accounts only for persistent rank-wise feature matrices. The evaluated implementation stores incidence relations sparsely and constructs only the additional operators requested by a model.[3] Cellular models may derive adjacency or Laplacian operators from cell incidences, while simplicial models may derive Laplacians from simplicial incidences. Full-domain preprocessing can therefore hold incidence matrices, derived operators, temporary construction objects, and implementation overhead in memory at the same time. We measure peak CPU memory as the largest resident-set size (RSS) sampled for the main Python process during full-domain preprocessing. It should be interpreted as the largest sampled resident-memory footprint of the main process, rather than as the size of any single tensor. Sampling may miss short-lived peaks and does not include memory held exclusively by child processes. Table 5 reports the available measurements for the evaluated dataset-lifting-model configurations.

Table 5: Available full-graph preprocessing time and peak CPU memory measurements for the evaluated dataset-lifting-model configurations. Models are distinguished because they may request different operators from the same lifted domain. Values are reported as the mean ± standard deviation across five runs. Memory usage is reported in decimal GB. Red placeholders mark pending measurements.

Dataset	**Lifting**	**Model**	**Additional operators**	**Preprocessing time (s)**	**Peak CPU memory (GB)**
Amazon Ratings	$k = 1$ neighbourhood hypergraph	EDHNN	None beyond the node-hyperedge incidence	0.42 ± 0.07	1.00 ± 0.01
Amazon Ratings	$k = 1$ neighbourhood hypergraph	UniGNN	None beyond the node-hyperedge incidence	0.39 ± 0.02	1.01 ± 0.01
Amazon Ratings	Cycle-basis cell	CWN	Rank-1 upper adjacency	5.7 ± 0.2	1.38 ± 0.00
Amazon Ratings	Cycle-basis cell	TopoTune	Rank-1 and two-hop rank-0 upper adjacencies	8.5 ± 0.2	1.39 ± 0.02
Amazon Ratings	Dimension-2 clique complex	SCN	None beyond the rank-1 and rank-2 incidences	4.3 ± 0.3	1.44 ± 0.00
Amazon Ratings	Dimension-2 clique complex	SCCNN	None beyond the rank-1 and rank-2 incidences	4.3 ± 0.3	1.44 ± 0.00
Questions	$k = 1$ neighbourhood hypergraph	EDHNN	None beyond the node-hyperedge incidence	0.63 ± 0.09	1.09 ± 0.02
Questions	$k = 1$ neighbourhood hypergraph	UniGNN	None beyond the node-hyperedge incidence	0.61 ± 0.05	1.10 ± 0.03
Questions	Cycle-basis cell	CWN	Rank-1 upper adjacency	6.1 ± 0.1	1.52 ± 0.08

Continued on next page

[3] See the reproducibility statement for implementation and reproduction information.

Table 5 continued

Dataset	Lifting	Model	Additional operators	Preprocessing time (s)	Peak CPU memory (GB)
Questions	Cycle-basis cell	TopoTune	Rank-1 and two-hop rank-0 upper adjacencies	7.3 ± 0.4	1.51 ± 0.07
Questions	Dimension-2 clique complex	SCN	None beyond the rank-1 and rank-2 incidences	6.2 ± 0.3	1.67 ± 0.03
Questions	Dimension-2 clique complex	SCCNN	None beyond the rank-1 and rank-2 incidences	6.2 ± 0.4	1.65 ± 0.06
Cora Full	$k = 1$ neighbourhood hypergraph	EDHNN	None beyond the node-hyperedge incidence	3.3 ± 0.1	2.98 ± 0.00
Cora Full	$k = 1$ neighbourhood hypergraph	UniGNN	None beyond the node-hyperedge incidence	3.4 ± 0.1	2.98 ± 0.00
Cora Full	Cycle-basis cell	CWN	Rank-1 upper adjacency	12.7 ± 1.5	6.14 ± 0.01
Cora Full	Cycle-basis cell	TopoTune	Rank-1 and two-hop rank-0 upper adjacencies	13.6 ± 0.7	6.14 ± 0.01
Cora Full	Dimension-2 clique complex	SCN	None beyond the rank-1 and rank-2 incidences	13.7 ± 0.7	7.28 ± 0.01
Cora Full	Dimension-2 clique complex	SCCNN	None beyond the rank-1 and rank-2 incidences	14.8 ± 3.3	7.28 ± 0.01
Coauthor Physics	$k = 1$ neighbourhood hypergraph	EDHNN	None beyond the node-hyperedge incidence	6.2 ± 0.7	4.48 ± 0.01
Coauthor Physics	$k = 1$ neighbourhood hypergraph	UniGNN	None beyond the node-hyperedge incidence	6.5 ± 0.8	4.48 ± 0.01
Coauthor Physics	Cycle-basis cell	CWN	Rank-1 upper adjacency	51.0 ± 9.3	19.79 ± 0.01
Coauthor Physics	Cycle-basis cell	TopoTune	Rank-1 and two-hop rank-0 upper adjacencies	55.3 ± 7.0	19.79 ± 0.01
Coauthor Physics	Dimension-2 clique complex	SCN	None beyond the rank-1 and rank-2 incidences	75.9 ± 5.6	42.42 ± 0.02
Coauthor Physics	Dimension-2 clique complex	SCCNN	None beyond the rank-1 and rank-2 incidences	77.3 ± 5.3	42.42 ± 0.01
Reddit	$k = 1$ neighbourhood hypergraph	EDHNN	None beyond the node-hyperedge incidence	137.7 ± 7.6	20.77 ± 0.08
Reddit	$k = 1$ neighbourhood hypergraph	UniGNN	None beyond the node-hyperedge incidence	136.9 ± 6.9	20.79 ± 0.12
Reddit	Cycle-basis cell	CWN	Rank-1 upper adjacency	4684.8 ± 304.8	479.96 ± 0.12
Reddit	Cycle-basis cell	TopoTune	Rank-1 and two-hop rank-0 upper adjacencies	6965.2 ± 682.4	480.06 ± 0.01
Reddit	Dimension-2 clique complex	SCN	None beyond the rank-1 and rank-2 incidences	OOM	OOM
Reddit	Dimension-2 clique complex	SCCNN	None beyond the rank-1 and rank-2 incidences	OOM	OOM

For OGBN Products, the higher-order structures can be constructed, but retaining their complete feature matrices exceeds the available CPU RAM in the affected full-graph configurations. Even if this preprocessing bottleneck were overcome, full-graph training is expected to exceed the available GPU memory. We therefore report partitioned results with Cluster-TNN, while omitting incomplete full-graph preprocessing measurements from Table 5.

Reddit full-domain bottlenecks. For Reddit, Eq. (3) predicts 236.280 GB for the cellular feature matrices alone, before accounting for incidence matrices, model-requested operators, and temporary allocations. Across five runs, CWN preprocessing takes 4684.8 ± 304.8 s and reaches a peak CPU memory usage of 479.96 ± 0.12 GB.

Storing the full Reddit simplicial input feature matrices at the original feature width in float32 would require an estimated 20.270 TB (Table 4).

B.4 Phase-specific memory during preprocessing and TNN execution

The following paired analyses focus on Questions, Amazon Ratings, and Cora Full, the three controlled benchmarks with complete full-graph and Cluster-TNN measurements across all seven evaluated models. Together, they provide 21 matched model-dataset comparisons. Each execution setting uses its independently selected hyperparameters, as reported in Appendix E. These comparisons

therefore characterize the resource requirements and predictive performance of the tuned configurations.

Successfully constructing a lifted full domain does not imply that the complete representation can be processed by a TNN in one optimization step. A TNN layer may update several ranks and retain intermediate messages for backpropagation. For example, an edge representation can receive separate messages from incident nodes and 2-cells. These intermediate tensors add to the GPU memory required by the input features and connectivity operators.

Figure 5 places preprocessing CPU-memory peaks alongside training and evaluation GPU-memory peaks for all seven evaluated models, complementing the joint memory-runtime comparison in Figure 3.

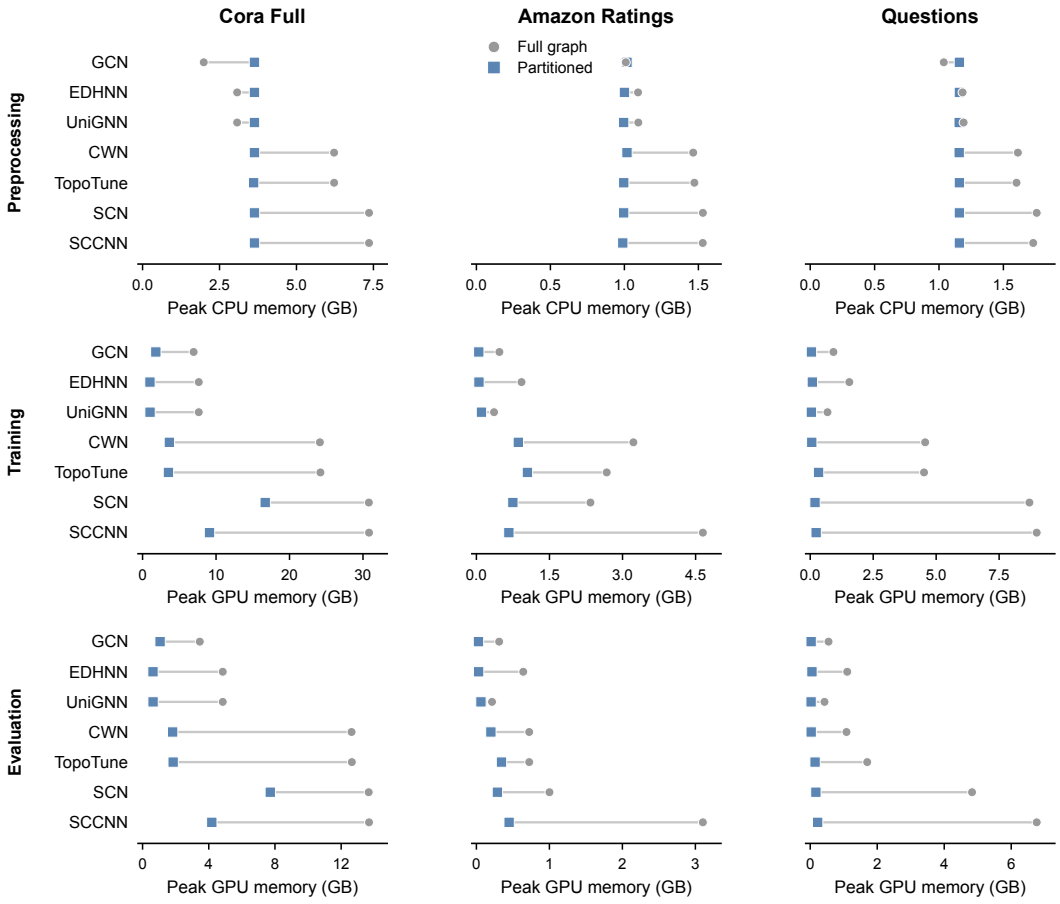


Figure 5: Cluster-TNN consistently reduces peak GPU memory during training and evaluation, whereas preprocessing memory varies by configuration and can increase when partition construction exceeds the cost of full-graph lifting. Values are means across five seeds.

Across the displayed comparisons, Cluster-TNN execution lowers peak GPU memory during both training and evaluation, whereas preprocessing CPU memory varies by configuration. These phase-specific measurements locate where memory pressure occurs. The next subsection reports elapsed time and peak CPU and GPU memory over a complete run.

B.5 End-to-end cost

End-to-end scalability has two distinct empirical dimensions: elapsed time across workflow phases and the peak CPU and GPU memory reached anywhere during a run. We report them separately because Cluster-TNN execution can reduce peak memory while increasing repeated lifting, validation, or inference time.

B.5.1 END-TO-END RUNTIME

An end-to-end comparison must account for when each method incurs its costs. Full-domain execution constructs the lifted domain once before training, whereas Cluster-TNN constructs the graph partition once and lifts mini-batches repeatedly. Table 6 reports preprocessing, training including validation passes, final validation and test inference, and total runtime for the 21 matched model-dataset comparisons. Table 5 reports the separate full-domain preprocessing measurements.

Table 6: End-to-end runtime in seconds under full-graph and Cluster-TNN settings. Preprocessing, training, and final evaluation are reported separately. Training is the complete fit call, including periodic validation. Final evaluation spans the end of fitting through checkpoint validation and test inference. Total is the sum of these non-overlapping components. For Cluster-TNN inference, we use the ensemble protocol and average predictions over ten cluster recombinations. Values are mean $\pm$ sample standard deviation in seconds across five seeds.

Dataset	Model	Setting	K	q	Preprocessing	Train	Final evaluation	Total
	GCN	Full-graph	-	-	4.0 ± 2.5	222.3 ± 57.4	3.4 ± 0.2	229.7 ± 57.5
		Cluster-TNN	32	8	11.2 ± 1.1	83.7 ± 8.8	47.2 ± 0.5	142.1 ± 8.8
	EDHNN	Full-graph	-	-	8.0 ± 0.8	438.7 ± 114.3	7.2 ± 0.3	453.9 ± 113.8
		Cluster-TNN	32	4	11.2 ± 1.1	180.5 ± 40.3	48.7 ± 0.9	240.3 ± 40.4
	UniGNN	Full-graph	-	-	8.4 ± 0.6	343.8 ± 71.9	7.1 ± 0.1	359.3 ± 71.8
		Cluster-TNN	32	4	11.2 ± 1.1	150.7 ± 21.9	50.4 ± 1.0	212.2 ± 22.3
Cora Full	CWN	Full-graph	-	-	18.7 ± 1.8	909.6 ± 117.6	14.6 ± 0.5	942.8 ± 118.9
		Cluster-TNN	32	4	11.2 ± 1.1	776.2 ± 134.2	272.4 ± 8.9	1059.8 ± 135.7
	TopoTune	Full-graph	-	-	19.4 ± 1.4	401.1 ± 53.1	14.4 ± 0.4	434.9 ± 51.9
		Cluster-TNN	64	8	11.4 ± 2.0	1211.6 ± 316.8	312.7 ± 10.6	1535.8 ± 309.1
	SCN	Full-graph	-	-	22.0 ± 4.2	473.5 ± 97.5	17.0 ± 0.9	512.5 ± 100.7
		Cluster-TNN	32	16	11.2 ± 1.1	562.9 ± 37.0	405.2 ± 25.6	979.3 ± 38.4
	SCCNN	Full-graph	-	-	21.6 ± 0.9	458.9 ± 22.7	17.3 ± 0.8	497.9 ± 22.6
		Cluster-TNN	32	8	11.2 ± 1.1	627.1 ± 118.0	455.6 ± 64.4	1093.9 ± 173.4
	GCN	Full-graph	-	-	2.6 ± 0.8	81.3 ± 16.6	1.3 ± 0.2	85.2 ± 15.8
		Cluster-TNN	64	4	2.6 ± 0.9	29.0 ± 8.7	6.2 ± 0.4	37.8 ± 8.9
	EDHNN	Full-graph	-	-	3.4 ± 0.3	143.9 ± 40.5	1.5 ± 0.1	148.8 ± 40.3
		Cluster-TNN	128	4	1.8 ± 0.1	100.5 ± 23.0	12.9 ± 2.7	115.3 ± 24.6
	UniGNN	Full-graph	-	-	3.5 ± 0.2	119.7 ± 54.2	1.6 ± 0.0	124.8 ± 54.1
		Cluster-TNN	32	8	1.8 ± 0.2	26.8 ± 6.8	6.0 ± 0.2	34.7 ± 6.9
Amazon Ratings	CWN	Full-graph	-	-	10.6 ± 1.6	150.7 ± 19.5	2.3 ± 0.1	163.5 ± 19.2
		Cluster-TNN	64	16	2.6 ± 0.9	849.4 ± 112.4	84.0 ± 8.9	936.1 ± 105.9
	TopoTune	Full-graph	-	-	13.4 ± 1.3	68.3 ± 11.9	2.4 ± 0.2	84.1 ± 12.4
		Cluster-TNN	32	8	1.8 ± 0.2	561.2 ± 108.0	120.7 ± 10.3	683.7 ± 112.1
	SCN	Full-graph	-	-	7.6 ± 0.3	240.0 ± 41.4	2.4 ± 0.1	250.0 ± 41.5
		Cluster-TNN	32	8	1.8 ± 0.2	572.4 ± 180.9	61.2 ± 3.2	635.4 ± 180.0
	SCCNN	Full-graph	-	-	7.7 ± 0.3	209.5 ± 47.8	2.3 ± 0.1	219.5 ± 47.8
		Cluster-TNN	32	4	1.8 ± 0.2	525.1 ± 89.3	76.4 ± 5.2	603.3 ± 93.8
	GCN	Full-graph	-	-	4.6 ± 2.1	164.5 ± 29.2	1.3 ± 0.1	170.4 ± 28.9
		Cluster-TNN	500	20	3.8 ± 0.4	42.0 ± 8.9	7.0 ± 0.1	52.9 ± 9.1
	EDHNN	Full-graph	-	-	4.0 ± 0.5	203.1 ± 42.2	1.7 ± 0.1	208.8 ± 42.4
		Cluster-TNN	500	50	3.8 ± 0.4	19.5 ± 4.7	7.1 ± 0.2	30.4 ± 5.0
	UniGNN	Full-graph	-	-	3.8 ± 0.3	260.0 ± 10.0	1.7 ± 0.1	265.5 ± 10.2
		Cluster-TNN	1000	40	4.2 ± 0.1	23.2 ± 11.6	7.3 ± 0.2	34.8 ± 11.6
Questions	CWN	Full-graph	-	-	9.8 ± 0.4	68.4 ± 23.6	2.5 ± 0.1	80.7 ± 23.8
		Cluster-TNN	1000	40	4.2 ± 0.1	116.4 ± 31.1	22.8 ± 0.4	143.4 ± 30.9
	TopoTune	Full-graph	-	-	12.1 ± 1.9	170.3 ± 21.6	2.8 ± 0.2	185.2 ± 20.8
		Cluster-TNN	500	50	3.8 ± 0.4	226.6 ± 61.0	32.9 ± 1.0	263.4 ± 61.0
	SCN	Full-graph	-	-	10.2 ± 0.3	378.6 ± 119.1	2.8 ± 0.1	391.6 ± 119.1
		Cluster-TNN	500	50	3.8 ± 0.4	107.6 ± 9.8	30.6 ± 1.5	142.0 ± 11.2
	SCCNN	Full-graph	-	-	10.2 ± 0.5	131.8 ± 14.5	2.9 ± 0.2	144.9 ± 14.7
		Cluster-TNN	500	30	3.8 ± 0.4	91.5 ± 9.9	36.1 ± 1.5	131.5 ± 10.5
	GCN	Cluster-TNN	500	20	10.4 ± 1.1	126.2 ± 52.2	91.4 ± 1.4	228.1 ± 51.4
	EDHNN	Cluster-TNN	2 000	30	11.2 ± 0.1	284.2 ± 56.8	55.3 ± 9.1	350.6 ± 61.6
	UniGNN	Cluster-TNN	2 000	20	11.2 ± 0.1	258.4 ± 52.6	42.9 ± 2.2	312.5 ± 52.7
Coauthor Physics	CWN	Cluster-TNN	2 000	20	11.2 ± 0.1	316.7 ± 114.0	80.3 ± 6.6	408.2 ± 110.9
	TopoTune	Cluster-TNN	2 000	20	11.2 ± 0.1	263.4 ± 38.4	86.3 ± 6.3	360.9 ± 32.9
	SCN	Cluster-TNN	2 000	20	11.2 ± 0.1	339.0 ± 113.2	96.8 ± 7.0	447.0 ± 111.2
	SCCNN	Cluster-TNN	2 000	20	11.2 ± 0.1	599.8 ± 215.2	161.3 ± 17.6	772.2 ± 211.1

Continued on next page

Table 6 continued

Dataset	Model	Setting	K	q	Preprocessing	Train	Final evaluation	Total
Reddit	GCN	Cluster-TNN	7 000	25	500.4 ± 3.7	449.7 ± 61.9	106.3 ± 3.1	1056 ± 62
	EDHNN	Cluster-TNN	7 000	20	500.4 ± 3.7	1165 ± 255	170.8 ± 4.6	1836 ± 253
	UniGNN	Cluster-TNN	7 000	25	500.4 ± 3.7	967.1 ± 208.2	116.8 ± 3.4	1584 ± 202
	CWN	Cluster-TNN	7 000	20	500.4 ± 3.7	27148 ± 1703	1659 ± 26	29307 ± 1703
	TopoTune	Cluster-TNN	7 000	20	500.4 ± 3.7	18696 ± 2834	2617 ± 35	21813 ± 2858
	SCN	Cluster-TNN	7 000	25	500.4 ± 3.7	26897 ± 981	4342 ± 95	31739 ± 950
	SCCNN	Cluster-TNN	7 000	20	500.4 ± 3.7	28571 ± 8714	5145 ± 128	34217 ± 8659
OGBN Products	GCN	Cluster-TNN	15 000	25	245.2 ± 5.6	3665 ± 153	2729 ± 24	6639 ± 161
	EDHNN	Cluster-TNN	20 000	25	263.7 ± 3.5	4169 ± 1019	3229 ± 18	7662 ± 1010
	UniGNN	Cluster-TNN	15 000	20	245.2 ± 5.6	5170 ± 195	2691 ± 10	8106 ± 202
	CWN	Cluster-TNN	15 000	20	245.2 ± 5.6	234389 ± 1997	17057 ± 140	251691 ± 2042
	TopoTune	Cluster-TNN	15 000	25	245.2 ± 5.6	207711 ± 63892	25215 ± 398	233172 ± 63530
	SCN	Cluster-TNN	15 000	20	245.2 ± 5.6	275955 ± 3781	23405 ± 1106	299605 ± 3287
	SCCNN	Cluster-TNN	15 000	20	245.2 ± 5.6	345176 ± 3392	19851 ± 491	365272 ± 3407

The following summaries are unweighted across the 21 model-dataset comparisons and are computed from the configuration-level means in Table 6. The preprocessing component is shorter under Cluster-TNN partitioned execution in 16 comparisons, longer in four, and unchanged in one. Among the 16 decreases, the mean and median reductions are 53.9% and 53.1%. Among the four increases, the mean and median increases are 66.0% and 36.7%. These partitioned preprocessing measurements include the one-off graph-partitioning cost. Training time is lower in 12 comparisons by 60.2% on average (median 63.3%) and higher in nine by 203.9% on average (median 138.5%).

CWN on Cora Full illustrates why training time and total runtime must be distinguished. Under partitioned execution, mean training time, including periodic validation, decreases from 909.6 to 776.2 seconds, a reduction of 14.7%. However, final evaluation increases from 14.6 to 272.4 seconds, resulting in a 12.4% increase in total runtime. In this configuration, the additional evaluation cost outweighs the training-time reduction.

Final evaluation is longer for every partitioned configuration, with a median ratio of $11.75\times$, under the evaluated ten-recombination protocol compared with one full-graph pass. This measured difference includes both repeated evaluation and batch-construction costs. It is not an intrinsic lower bound for partitioned inference.

Validation caching convention. In these experiments, we do not cache lifted validation batches. We reconstruct and lift each validation batch online. We include validation-batch construction costs in the training component for validation during fitting and in the final-evaluation component for checkpoint validation. Final evaluation includes checkpoint validation and test inference.

B.5.2 Run-wide peak memory

Unlike Table 5, which isolates full-domain preprocessing, Table 7 and 8 report peak CPU and GPU memory over the complete execution of each full-graph and Cluster-TNN run. CPU memory is the maximum sampled RSS of the main Python process, while GPU memory is the maximum CUDA memory allocated by the process.

Across the 21 matched full-graph and partitioned comparisons in Table 7,, run-wide peak CPU memory is lower under partitioned execution in 15 comparisons by 14.8% on average (median 2.4%) and higher in six by 6.8% on average (median 3.7%). CPU memory therefore varies by configuration rather than exhibiting the uniform decrease observed for GPU memory. Peak GPU memory is lower in all 21 comparisons by 83.2% on average (median 86.7%).

Table 8 extends the measurements to Coauthor Physics, Reddit, and OGBN Products, reporting absolute memory requirements where matched full-graph results are unavailable. The larger-dataset results show that large input graphs can be processed with modest GPU memory requirements under Cluster-TNN execution. On Reddit, mean peak GPU memory ranges from 0.04 GB for UniGNN to 7.45 GB for SCCNN. On OGBN Products, TopoTune and SCN have mean peaks of 1.46 and 5.42 GB, respectively. Memory requirements therefore depend strongly on the sampled graph context and model configuration, rather than input graph size alone.

Table 7: Run-wide peak CPU and GPU memory under full-graph and Cluster-TNN execution. Values are mean ± sample standard deviation in decimal GB across five seeds. CPU values are main-process RSS peaks. GPU values are CUDA allocated-memory peaks.

				Full-graph		Cluster-TNN	
Dataset	**Model**	K	q	**Peak CPU memory (GB)**	**Peak GPU memory (GB)**	**Peak CPU memory (GB)**	**Peak GPU memory (GB)**
Cora Full	GCN	32	8	2.96 ± 0.01	6.95 ± 0.00	3.64 ± 0.03	1.80 ± 0.01
	EDHNN	32	4	3.68 ± 0.02	7.65 ± 0.00	3.64 ± 0.03	1.01 ± 0.00
	UniGNN	32	4	3.71 ± 0.01	7.65 ± 0.00	3.64 ± 0.03	1.02 ± 0.00
	CWN	32	4	7.63 ± 0.01	24.14 ± 0.00	3.89 ± 0.07	3.65 ± 0.09
	TopoTune	64	8	7.61 ± 0.03	24.21 ± 0.00	3.98 ± 0.12	3.53 ± 0.09
	SCN	32	16	8.47 ± 0.02	30.61 ± 0.00	8.33 ± 0.06	16.42 ± 0.23
	SCCNN	32	8	8.47 ± 0.03	30.62 ± 0.00	5.74 ± 0.05	8.92 ± 0.26
Amazon Ratings	GCN	64	4	1.60 ± 0.01	0.47 ± 0.00	1.66 ± 0.01	0.05 ± 0.00
	EDHNN	128	4	1.66 ± 0.01	0.93 ± 0.00	1.68 ± 0.01	0.05 ± 0.00
	UniGNN	32	8	1.66 ± 0.01	0.36 ± 0.00	1.72 ± 0.02	0.10 ± 0.00
	CWN	64	16	2.01 ± 0.01	3.22 ± 0.00	2.00 ± 0.02	0.86 ± 0.01
	TopoTune	32	8	2.03 ± 0.02	2.68 ± 0.00	2.19 ± 0.01	1.05 ± 0.01
	SCN	32	8	2.11 ± 0.00	2.10 ± 0.00	2.06 ± 0.04	0.65 ± 0.00
	SCCNN	32	4	2.13 ± 0.01	4.38 ± 0.00	2.16 ± 0.02	0.62 ± 0.00
Questions	GCN	500	20	1.70 ± 0.01	0.92 ± 0.00	1.69 ± 0.02	0.05 ± 0.00
	EDHNN	500	50	1.76 ± 0.01	1.55 ± 0.00	1.73 ± 0.01	0.08 ± 0.00
	UniGNN	1000	40	1.76 ± 0.01	0.69 ± 0.00	1.72 ± 0.01	0.05 ± 0.00
	CWN	1000	40	2.25 ± 0.03	4.56 ± 0.00	1.73 ± 0.01	0.06 ± 0.00
	TopoTune	500	50	2.27 ± 0.01	4.52 ± 0.00	1.87 ± 0.02	0.33 ± 0.01
	SCN	500	50	2.31 ± 0.02	3.40 ± 0.00	1.81 ± 0.02	0.14 ± 0.00
	SCCNN	500	30	2.30 ± 0.01	6.10 ± 0.00	1.88 ± 0.01	0.18 ± 0.00

Table 8: Cluster-TNN run-wide peak memory on Coauthor Physics, Reddit, and OGBN Products, using the measurement definitions of Table 7. CPU values are main-process RSS peaks; GPU values are CUDA allocated-memory peaks, in decimal GB.

Dataset	**Model**	K	q	**Peak CPU memory (GB)**	**Peak GPU memory (GB)**
Coauthor Physics	GCN	500	20	6.68 ± 0.03	0.51 ± 0.00
	EDHNN	2 000	30	6.07 ± 0.05	0.24 ± 0.00
	UniGNN	2 000	20	6.07 ± 0.05	0.16 ± 0.00
	CWN	2 000	20	6.07 ± 0.05	0.38 ± 0.01
	TopoTune	2 000	20	6.07 ± 0.05	0.38 ± 0.01
	SCN	2 000	20	6.07 ± 0.05	0.67 ± 0.05
	SCCNN	2 000	20	6.07 ± 0.05	0.72 ± 0.04
Reddit	GCN	7 000	25	11.81 ± 0.02	0.07 ± 0.00
	EDHNN	7 000	20	11.81 ± 0.02	0.13 ± 0.01
	UniGNN	7 000	25	11.81 ± 0.02	0.04 ± 0.00
	CWN	7 000	20	11.81 ± 0.02	0.51 ± 0.04
	TopoTune	7 000	20	11.81 ± 0.02	0.57 ± 0.04
	SCN	7 000	25	11.81 ± 0.02	6.84 ± 1.91
	SCCNN	7 000	20	11.81 ± 0.02	7.45 ± 2.47
OGBN Products	GCN	15 000	25	12.88 ± 0.02	0.19 ± 0.00
	EDHNN	20 000	25	13.19 ± 0.02	0.38 ± 0.01
	UniGNN	15 000	20	12.87 ± 0.02	0.05 ± 0.00
	CWN	15 000	20	12.87 ± 0.02	1.45 ± 0.02
	TopoTune	15 000	25	12.87 ± 0.02	1.46 ± 0.02
	SCN	15 000	20	12.87 ± 0.02	5.42 ± 0.52
	SCCNN	15 000	20	12.87 ± 0.02	11.94 ± 1.10

B.6 STRUCTURAL EXPOSURE AND STATISTICAL EFFICIENCY

Memory savings are only one consequence of batch-before-lift execution. Because each update is formed from an induced graph batch, the method also changes which full-graph structures can occur together and how often the model encounters them. We refer to this availability across cluster recombinations as structural exposure.

An epoch under Cluster-TNN visits all active training clusters once and groups them into mini-batches of at most q clusters. Its number of optimization steps is therefore the number of active clusters divided into groups of size at most q. The theory in Appendix C considers the simplified case in which all K clusters are active and $q \mid K$, giving exactly $M = K/q$ steps. Increasing q uses more memory per mini-batch but can expose more cross-cluster higher-order structures. Consequently, equal epoch counts need not represent equal update counts, wall-clock time, or structural information.

The recovery theory describes when structures and messages are available. It does not establish convergence to the model trained on the full lifted domain. We therefore assess statistical efficiency through validation trajectories, optimization steps, and test performance. The primary validation metric selects the checkpoint for each run. Test predictions from that checkpoint use full-graph inference for full-graph setting and ten-recombination ensemble inference for Cluster-TNN setting.

Table 9 and 10 summarize each validation trajectory by the checkpoint attaining the best primary validation metric. For each run, we associate this checkpoint with its completed optimizer-step count and its final test result.

Table 9: Statistical-efficiency for full-graph and Cluster-TNN training. E^* is the epoch selected by the validation metric, and U^* is the number of optimizer updates completed at that checkpoint. Validation and test results are reported. The metric is accuracy for Cora Full and Amazon Ratings and AUROC for Questions. Full-graph uses one full-domain test pass, whereas Cluster-TNN uses the ten-pass ensemble protocol. Each entry is the mean ± sample standard deviation across five seeds.

				Full-graph				Cluster-TNN			
Dataset	**Model**	K	q	E^*	U^*	**Val.**	**Test**	E^*	U^*	**Val.**	**Test**
Cora Full	GCN	32	8	94 ± 30	94 ± 30	70.02 ± 1.23	70.18 ± 0.42	44 ± 10	176 ± 39	68.42 ± 1.63	69.19 ± 0.66
	EDHNN	32	4	82 ± 29	82 ± 29	70.07 ± 0.97	69.74 ± 0.89	59 ± 19	472 ± 153	66.89 ± 1.15	67.06 ± 1.19
	UniGNN	32	4	58 ± 18	58 ± 18	68.92 ± 1.32	68.44 ± 0.79	44 ± 10	352 ± 82	67.43 ± 1.23	67.61 ± 0.56
	CWN	32	4	83 ± 12	83 ± 12	61.29 ± 0.82	60.50 ± 0.60	57 ± 3	456 ± 22	62.37 ± 1.04	62.66 ± 1.11
	TopoTune	64	8	22 ± 6	22 ± 6	46.87 ± 0.62	46.40 ± 0.47	67 ± 21	536 ± 171	55.92 ± 0.92	56.45 ± 1.42
	SCN	32	16	23 ± 10	23 ± 10	70.81 ± 0.53	70.45 ± 0.35	14 ± 2	28 ± 5	70.01 ± 0.82	70.32 ± 0.46
	SCCNN	32	8	21 ± 2	21 ± 2	71.19 ± 0.71	70.81 ± 0.58	19 ± 11	76 ± 43	69.52 ± 0.95	70.20 ± 0.29
Amazon Ratings	GCN	64	4	120 ± 29	120 ± 29	49.36 ± 1.39	48.29 ± 1.31	75 ± 31	1200 ± 493	49.70 ± 1.03	49.62 ± 0.96
	EDHNN	128	4	181 ± 58	181 ± 58	48.86 ± 1.08	48.41 ± 1.18	134 ± 29	4288 ± 936	51.71 ± 1.21	51.41 ± 0.72
	UniGNN	32	8	151 ± 82	151 ± 82	47.35 ± 1.32	47.05 ± 1.76	69 ± 20	276 ± 82	47.45 ± 0.95	47.24 ± 0.70
	CWN	64	16	106 ± 20	106 ± 20	45.13 ± 1.02	44.07 ± 0.64	78 ± 14	312 ± 58	46.63 ± 0.97	46.25 ± 0.67
	TopoTune	32	8	32 ± 10	32 ± 10	43.41 ± 1.00	43.18 ± 1.03	21 ± 9	84 ± 36	43.60 ± 0.85	43.27 ± 0.68
	SCN	32	8	174 ± 37	174 ± 37	50.36 ± 0.74	50.08 ± 0.36	78 ± 34	312 ± 137	50.99 ± 1.01	50.65 ± 0.64
	SCCNN	32	4	141 ± 37	141 ± 37	51.33 ± 0.83	51.20 ± 1.10	67 ± 10	536 ± 83	51.45 ± 0.71	51.24 ± 0.79
Questions	GCN	500	20	254 ± 50	254 ± 50	76.32 ± 1.25	76.69 ± 1.18	98 ± 30	2450 ± 748	72.85 ± 1.66	73.83 ± 2.01
	EDHNN	500	50	211 ± 52	211 ± 52	76.54 ± 1.27	75.73 ± 0.92	42 ± 20	420 ± 196	75.78 ± 1.01	76.23 ± 1.05
	UniGNN	1000	40	300 ± 0	300 ± 0	70.41 ± 1.41	69.44 ± 1.68	27 ± 28	675 ± 694	71.25 ± 1.58	71.09 ± 0.98
	CWN	1000	40	26 ± 18	26 ± 18	68.56 ± 2.56	68.21 ± 3.48	27 ± 14	675 ± 360	71.62 ± 0.99	70.20 ± 0.88
	TopoTune	500	50	95 ± 17	95 ± 17	69.98 ± 1.88	71.53 ± 1.73	54 ± 22	540 ± 222	71.97 ± 1.90	73.61 ± 1.88
	SCN	500	50	236 ± 90	236 ± 90	73.96 ± 1.29	74.77 ± 1.34	11 ± 2	110 ± 22	71.96 ± 1.26	73.04 ± 1.84
	SCCNN	500	30	56 ± 8	56 ± 8	76.26 ± 1.15	77.14 ± 0.99	6 ± 2	102 ± 38	73.13 ± 1.50	74.10 ± 1.85

Based on the reported means, partitioned training reaches the selected checkpoint in fewer epochs in 19 of 21 comparisons, but uses more optimizer updates in 20. Because a partitioned epoch contains multiple updates, epoch counts alone do not establish greater optimization efficiency. They should be interpreted alongside optimizer updates, elapsed time, and predictive performance.

Table 10: Cluster-TNN statistical efficiency on Coauthor Physics, Reddit, and OGBN Products, using the checkpoint and inference definitions of Table 9. E^* is the selected epoch, U^* the completed optimizer updates, and validation and test scores are accuracies (%).

Dataset	Model	K	q	E^*	U^*	Val.	Test
Coauthor Physics	GCN	500	20	39 ± 27	975 ± 681	96.32 ± 0.12	96.68 ± 0.08
	EDHNN	2 000	30	58 ± 20	3886 ± 1332	96.13 ± 0.11	96.31 ± 0.32
	UniGNN	2 000	20	68 ± 17	6800 ± 1681	94.66 ± 0.19	94.97 ± 0.37
	CWN	2 000	20	34 ± 24	3400 ± 2434	95.69 ± 0.10	95.80 ± 0.24
	TopoTune	2 000	20	18 ± 7	1800 ± 671	94.70 ± 0.27	95.22 ± 0.20
	SCN	2 000	20	32 ± 20	3200 ± 1956	95.84 ± 0.20	96.22 ± 0.22
	SCCNN	2 000	20	60 ± 35	6000 ± 3500	95.84 ± 0.31	96.00 ± 0.25
Reddit	GCN	7 000	25	62 ± 8	17360 ± 2123	92.42 ± 0.15	92.62 ± 0.10
	EDHNN	7 000	20	134 ± 40	46900 ± 14076	91.73 ± 0.75	92.17 ± 0.73
	UniGNN	7 000	25	108 ± 37	30240 ± 10458	90.73 ± 0.11	91.28 ± 0.16
	CWN	7 000	20	113 ± 8	39550 ± 2654	91.02 ± 0.19	91.29 ± 0.18
	TopoTune	7 000	20	33 ± 8	11550 ± 2928	89.12 ± 0.14	89.55 ± 0.09
	SCN	7 000	25	56 ± 2	15680 ± 626	91.44 ± 0.14	91.80 ± 0.13
	SCCNN	7 000	20	65 ± 29	22750 ± 9977	92.16 ± 0.19	92.45 ± 0.25
OGBN Products	GCN	15 000	25	128 ± 3	76800 ± 1643	85.50 ± 0.08	85.81 ± 0.04
	EDHNN	20 000	25	95 ± 28	76000 ± 22627	86.27 ± 0.28	86.51 ± 0.25
	UniGNN	15 000	20	125 ± 11	93750 ± 8385	85.61 ± 0.08	85.87 ± 0.07
	CWN	15 000	20	128 ± 3	96000 ± 2054	83.43 ± 0.09	83.93 ± 0.06
	TopoTune	15 000	25	61 ± 22	36600 ± 13315	83.09 ± 0.48	83.79 ± 0.55
	SCN	15 000	20	127 ± 3	95250 ± 2054	86.57 ± 0.06	87.03 ± 0.05
	SCCNN	15 000	20	167 ± 3	125250 ± 2054	88.14 ± 0.04	88.63 ± 0.05

B.7 Solution paths and practical guidance

The measurements above show a phase-specific trade-off rather than a uniform reduction in computational cost. Partitioned execution lowers peak GPU memory in all 21 direct comparisons, while preprocessing time, training time, and run-wide CPU memory change in both directions across configurations. Final evaluation is longer under the evaluated ten-recombination protocol, and statistical outcomes vary by dataset and model. A suitable remedy must therefore be matched to the stage that constitutes the binding bottleneck.

Alternative solution paths. Different remedies address different stages of the workflow. Restricting neighbourhood radius, simplex dimension, or retained cycle length can reduce the number of retained higher-order elements or incidences, but may discard useful structure. For the evaluated neighbourhood lifting, reducing the radius changes membership volume rather than the fixed count of one indexed hyperedge per node. For the evaluated cycle lifting, the length cap filters retained cells after a complete cycle basis is computed, so it need not reduce basis-construction time or temporary memory. Reducing feature width or numerical precision lowers feature storage. Computing higher-order features when needed can avoid retaining their complete matrices, but introduces repeated computation and does not by itself remove the need to provide the required connectivity.

When full-domain construction exceeds CPU memory, batch-before-lift, distributed construction, or out-of-core processing address the construction stage in different ways. Distributed construction spreads work and allocations across workers, while out-of-core processing moves part of the representation to secondary storage. Neither reduces the representation itself. When the domain can be constructed but model execution exceeds GPU memory, sampling within the higher-order domain becomes a viable alternative. When repeated local lifting dominates runtime, caching a fixed set of lifted batches can reduce reconstruction work at the cost of storage and a fixed collection of cluster combinations.

Published lift-then-batch methods provide concrete examples in the hypergraph setting. Telyatnikov et al. (Telyatnikov et al., 2025b) sample hyperedges and then nodes within the sampled hyperedges, while FastHGNN (Lu & Ng, 2024) performs layer-wise node and hyperedge sampling. These methods reduce model-execution cost after a hypergraph is available, but do not address the cost of

constructing a lifted domain from a source graph. They also do not directly specify how to sample simplicial or cellular domains while preserving the face or boundary relations used by those models.

Batch-before-lift: scope and trade-offs. Batch-before-lift instead limits how much of the source graph is presented to the lifting and the TNN at one time. It is operationally applicable to graph-to-higher-order liftings that can be evaluated on induced subgraphs. This condition does not guarantee that locally lifted structures coincide with restrictions of the full-domain lift. For non-compatible liftings, local recomputation can alter the selected structures in addition to omitting structures whose supporting nodes do not appear together. Its principal costs are therefore repeated local lifting and reduced or altered structural exposure. The number of clusters grouped into a mini-batch controls part of this trade-off: larger batches can expose more cross-cluster structure, but require more CPU and GPU memory. Lift-then-batch methods address a complementary setting, particularly when data are observed natively as higher-order domains, but require sampling rules that preserve the relations used by the target hypergraph, simplicial, or cellular model. Standardized, intrinsically higher-order benchmark datasets remain comparatively scarce (Papamarkou et al., 2024; Rieck, 2026). Our implementation realizes batch-before-lift through TopoBench's existing dataset, lifting, and model interfaces.

Reporting guidance. At minimum, scalable graph-supported TDL studies should report lifting-induced structural growth, phase-wise time and peak memory with the device and measurement scope identified, end-to-end runtime, and predictive results. Where applicable, they should also report partition and mini-batch parameters, caching and inference protocols, the number of cluster recombinations, and optimizer updates. Together, these quantities distinguish whether a method reduces the representation itself, constructs it incrementally or on demand, limits only the model input, or trades repeated computation against storage. The realized memory savings and structural exposure depend on the dataset and lifting, as well as method-specific choices such as the partition, total number of clusters, mini-batch grouping, and number of recombinations. Appendix C next states the compatibility assumptions and formalizes the structural-exposure side of this trade-off.

C STRUCTURAL RECONSTRUCTION

This appendix analyzes which full-graph structures Cluster-TNN workflow recovers, how often it exposes them, and when their direct-incidence messages are available. The graph partition is fixed, and the only randomness comes from independently reshuffling its cluster indices at each epoch. Throughout the paper, K denotes the total number of clusters in this partition. The implementation retains only clusters containing at least one target node from the current split (Sec. 3.1). For example, a cluster containing validation or test nodes but no training nodes is inactive for training. The number of active clusters can therefore be smaller than K. The analysis below assumes that all K partition clusters participate in recombination. For notational simplicity, we assume $q \mid K$, so every mini-batch contains exactly q clusters. This removes a possible smaller remainder mini-batch without changing the qualitative conclusions. The following results state the compatibility assumptions required for structural recovery, repeated exposure, and direct-message availability.

The recovery analysis assumes compatibility with induced subgraphs, as formalized in Definition 7. This condition holds for the identity lifting, the evaluated k-hop neighbourhood liftings, and the clique-complex lifting. It can fail for the cycle-basis lifting because restricting the graph can change the selected basis. The cycle-basis recovery probabilities therefore require stable basis membership. Appendix C.13 examines this condition empirically on Cora Full dataset.

C.1 SETUP AND NOTATION

Because features do not affect structural recovery, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the underlying undirected graph. Let

$$\Pi = \{P_1, \ldots, P_K\}$$

be the fixed partition of $\mathcal{V}$ into K clusters considered in this analysis. We write $[K] = \{1, \ldots, K\}$ and use $q \leq K$ clusters per mini-batch. Under the divisibility assumption above, each epoch contains $M = K/q$ mini-batches.

At each epoch $t \in \{1, \dots, T\}$, we independently draw a uniformly random permutation of $[K]$. We divide the permutation into the mini-batch index sets

$$\mathcal{Z}^{(t)} = \{Z_1^{(t)}, \dots, Z_M^{(t)}\}, \qquad Z_i^{(t)} \subseteq [K], \qquad |Z_i^{(t)}| = q.$$

For each mini-batch index set $Z_i^{(t)}$, we define the induced mini-batch graph

$$\mathcal{G}_i^{(t)} = \big(\mathcal{V}_i^{(t)}, \mathcal{E}_i^{(t)}\big), \qquad \mathcal{V}_i^{(t)} = \bigcup_{k \in Z_i^{(t)}} P_k, \qquad \mathcal{E}_i^{(t)} = \{(u,v) \in \mathcal{E} : u, v \in \mathcal{V}_i^{(t)}\}.$$

Thus, $\mathcal{G}_i^{(t)}$ is the graph induced by the mini-batch node set $B = \mathcal{V}_i^{(t)}$.

Let $\mathcal{L}_S$ be a deterministic structural lifting from graphs to a chosen topological domain. Let $\mathcal{S}$ denote the set of possible topological structures. For any graph $\mathcal{G}$, we write $\mathrm{Str}(\mathcal{L}_S(\mathcal{G})) \subseteq \mathcal{S}$ for the structures in the resulting lifted domain. The set observed in epoch t is

$$S^{(t)} = \bigcup_{i=1}^{M} \mathrm{Str}\left(\mathcal{L}_S\left(\mathcal{G}_i^{(t)}\right)\right) . \tag{4}$$

The cumulative observed set of structures after T epochs is $\widehat{S}_T = \bigcup_{t=1}^{T} S^{(t)}$.

The global structure set is $S^* = \mathrm{Str}\left(\mathcal{L}_S\left(\mathcal{G}\right)\right)$, obtained by lifting the full graph. We assume that $\mathcal{G}$, and hence S^*, is finite. Mini-batch lifting may also produce structures outside S^*. We therefore define the observed global structures as

$$\widehat{S}_T^* = \widehat{S}_T \cap S^*.$$

Thus, the recovery analysis only tracks global structures and ignores mini-batch-local artifacts.

For $s \in S^*$, given its vertex set $\mathcal{V}_s \subseteq \mathcal{V}$, we define its partition-index support as

$$I_s = \{i \in [K] : P_i \cap \mathcal{V}_s \neq \emptyset\}$$

and its cluster span $c_s = |I_s|$ as the number of clusters that s intersects. Since every structure contains at least one vertex, we have $c_s \geq 1$. A structure with $c_s = 1$ is fully contained in one cluster.

Definition 7 (Lifting compatibility)**.** *The lifting map is compatible with induced subgraphs if the following equivalence holds for every $s \in S^*$ and every $Z_i^{(t)}$. The global structure s appears in the lifted mini-batch exactly when that mini-batch contains all its supporting cluster indices:*

$$s \in \mathrm{Str}\left(\mathcal{L}_S\left(\mathcal{G}_i^{(t)}\right)\right) \iff I_s \subseteq Z_i^{(t)} .$$

C.2 Per-epoch recovery probability

We first compute the probability that a fixed global structure $s \in S^*$ is recovered during one epoch. By Definition 7, all indices in I_s must occur in the same mini-batch. Thus, recovery depends on the cluster span $c_s = |I_s|$. It is impossible when $c_s > q$ and becomes a combinatorial co-occurrence event when $c_s \leq q$.

Let

$$p_s^{(t)} = \mathbb{P}\left(s \in S^{(t)}\right)$$

denote this per-epoch recovery probability. Since the reshuffling distribution is identical across epochs, $p_s^{(t)}$ does not depend on t, and we write p_s.

Lemma 1 (Per-epoch recovery probability)**.** *For any global structure $s \in S^*$,*

$$p_s = \begin{cases} \dfrac{\binom{q-1}{c_s-1}}{\binom{K-1}{c_s-1}}, & c_s \leq q, \\ 0, & c_s > q. \end{cases}$$

Proof. If $c_s > q$, no mini-batch can contain all c_s supporting cluster indices, so $p_s = 0$. Now suppose $c_s \le q$. Fix $k \in I_s$, and let $Z^{(t)}(k)$ be its unique mini-batch in epoch t. Under a uniform shuffle, the other $q-1$ members of $Z^{(t)}(k)$ form a uniformly random subset of $[K] \setminus \{k\}$. Recovery requires this subset to contain the fixed set $I_s \setminus \{k\}$. Therefore,

$$p_s = \frac{\binom{K-c_s}{q-c_s}}{\binom{K-1}{q-1}} = \frac{\binom{q-1}{c_s-1}}{\binom{K-1}{c_s-1}} .$$

□

Structures contained in one cluster are recovered every epoch. When $q < K$, the recovery probability decreases with cluster span, and structures with $c_s > q$ remain unobservable.

C.3 Repeated structural exposure

Per-epoch recovery determines whether a structure is available during one epoch. To track repeated availability, for each $s \in S^*$ we define the exposure indicator

$$X_s^{(t)} = \mathbf{1}\{s \in S^{(t)}\} , \tag{5}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function, and the number of epochs in which s has been observed by epoch T,

$$N_s(T) = \sum_{t=1}^{T} X_s^{(t)} . \tag{6}$$

The mini-batches partition the K clusters in each epoch. A compatible global structure can therefore occur in at most one mini-batch per epoch. Thus, $N_s(T)$ counts both exposure epochs and the corresponding optimization steps containing s.

Proposition 1 (Repeated structural exposure)**.** *Under independent epoch reshuffling and lifting compatibility, for every fixed $s \in S^*$,*

$$X_s^{(t)} \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p_s), \qquad N_s(T) \sim \text{Binomial}(T, p_s). \tag{7}$$

Consequently,

$$\mathbb{E}[N_s(T)] = Tp_s, \qquad \text{Var}[N_s(T)] = Tp_s(1-p_s), \tag{8}$$

and

$$\frac{N_s(T)}{T} \longrightarrow p_s \qquad \textit{almost surely as } T \to \infty. \tag{9}$$

Proof. By Lemma 1, each $X_s^{(t)}$ is Bernoulli with success probability p_s. It is a function of the permutation drawn in epoch t. Independent and identically distributed epoch permutations therefore produce independent and identically distributed exposure indicators. Their sum is binomial, giving Eq. (7) and the moments in Eq. (8). Eq. (9) follows from the strong law of large numbers. □

Proposition 1 gives p_s two interpretations. It is the one-epoch recovery probability and the asymptotic fraction of epochs containing s. If $c_s = 1$, then $N_s(T) = T$ almost surely. If $c_s > q$, then $N_s(T) = 0$ almost surely. When $q < K$, structures with $1 < c_s \le q$ remain observable but occur less frequently than structures contained in one cluster.

Since every epoch contains $M = K/q$ optimization steps, Eq. (9) also gives the asymptotic fraction of all training steps in which s is present:

$$\frac{N_s(T)}{MT} \longrightarrow \frac{p_s}{M} \qquad \text{almost surely as } T \to \infty. \tag{10}$$

The distinction is important: p_s is a per-epoch exposure probability, whereas p_s/M is the corresponding frequency among optimization steps.

Remark 1 (Dependence between structures)**.** *Proposition 1 concerns one fixed structure across epochs. Exposure indicators for different structures within an epoch are generally dependent because they share the same cluster grouping. None of the expectations below require independence between structures.*

C.4 CUMULATIVE RECOVERY PROBABILITY

Cumulative recovery records whether a global structure has been exposed at least once by epoch T. For every $s \in S^*$, this event is equivalent to

$$\{s \in \widehat{S}_T^*\} = \{N_s(T) \geq 1\}.$$

By Proposition 1, the probability that s remains unobserved after T epochs is

$$\mathbb{P}\left(s \notin \widehat{S}_T^*\right) = \mathbb{P}\left(N_s(T) = 0\right) = (1 - p_s)^T.$$

Hence,

$$\mathbb{P}\left(s \in \widehat{S}_T^*\right) = 1 - (1 - p_s)^T \,. \tag{11}$$

Theorem 1 (Recovery in probability of the q-observable structure set)**.** *For every $s \in S^*$,*

$$\lim_{T\to\infty} \mathbb{P}\left(s \in \widehat{S}_T^*\right) = \begin{cases} 1, & c_s \leq q, \\ 0, & c_s > q. \end{cases}$$

Consequently, with

$$S_q^* = \{s \in S^* : c_s \leq q\} \,,$$

we have $\mathbb{P}(\widehat{S}_T^* = S_q^*) \to 1$. In particular, if $q \geq \max_{s\in S^*} c_s$, then $\mathbb{P}(\widehat{S}_T^* = S^*) \to 1$.

Proof. If $c_s > q$, then Lemma 1 gives $p_s = 0$, and Eq. (11) gives $\mathbb{P}(s \in \widehat{S}_T^*) = 0$ for all T. If $c_s \leq q$, then $p_s > 0$, so $(1 - p_s)^T \to 0$ and Eq. (11) gives $\mathbb{P}(s \in \widehat{S}_T^*) \to 1$.

Structures in $S^* \setminus S_q^*$ are never observed with q clusters per mini-batch. Thus, $\widehat{S}_T^* \subseteq S_q^*$ for every T. Since S_q^* is finite, the union bound gives

$$\mathbb{P}\left(\widehat{S}_T^* \neq S_q^*\right) = \mathbb{P}\Big(\bigcup_{s\in S_q^*} \{s \notin \widehat{S}_T^*\}\Big) \leq \sum_{s\in S_q^*} \mathbb{P}\left(s \notin \widehat{S}_T^*\right) = \sum_{s\in S_q^*} (1 - p_s)^T \,.$$

The right-hand side tends to zero as $T \to \infty$, so $\mathbb{P}(\widehat{S}_T^* = S_q^*) \to 1$. If $q \geq \max_{s\in S^*} c_s$, then $S_q^* = S^*$, which gives the full-recovery claim. □

C.5 STRUCTURAL COVERAGE CONVERGENCE

We now aggregate the per-structure recovery probabilities into a global coverage quantity. The realized structural coverage after T epochs is the random variable

$$R_{q,T} = \frac{\left|\widehat{S}_T^*\right|}{|S^*|} = \frac{1}{|S^*|} \sum_{s\in S_q^*} \mathbf{1}\{s \in \widehat{S}_T^*\} \,. \tag{12}$$

Thus, $R_{q,T}$ is the fraction of global structures recovered by epoch T. Its expectation defines the corresponding deterministic coverage curve. By linearity of expectation and Eq. (11),

$$\mathbb{E}[R_{q,T}] = \frac{1}{|S^*|} \sum_{s\in S_q^*} \mathbb{P}\left(s \in \widehat{S}_T^*\right) = \frac{1}{|S^*|} \sum_{s\in S_q^*} \left(1 - (1 - p_s)^T\right) \,. \tag{13}$$

Proposition 2 (Expected structural coverage convergence)**.** *We have*

$$\lim_{T\to\infty} \mathbb{E}[R_{q,T}] = \frac{\left|S_q^*\right|}{|S^*|} \,.$$

In particular, if $q \geq \max_{s\in S^*} c_s$, then $\lim_{T\to\infty} \mathbb{E}[R_{q,T}] = 1$.

Proof. Since S^* is finite, we can pass the limit through the finite sum in Eq. (13):

$$\lim_{T\to\infty} \mathbb{E}[R_{q,T}] = \frac{1}{|S^*|} \sum_{s\in S_q^*} \lim_{T\to\infty} \left(1-(1-p_s)^T\right) .$$

For $s \in S_q^*$, Lemma 1 gives $p_s > 0$, so the corresponding limit is 1. Therefore,

$$\lim_{T\to\infty} \mathbb{E}[R_{q,T}] = \frac{1}{|S^*|} \sum_{s\in S_q^*} 1 = \frac{|S_q^*|}{|S^*|} .$$

If $q \geq \max_{s\in S^*} c_s$, then $S_q^* = S^*$, which gives the full-coverage claim. □

C.6 Repeated-exposure structural coverage

One-exposure coverage does not distinguish structures observed once from those observed repeatedly. For a threshold $m \in \{1,\dots,T\}$, define the realized m-exposure structural coverage as

$$R_{q,T}^{(m)} = \frac{1}{|S^*|} \sum_{s\in S^*} \mathbf{1}\{N_s(T) \geq m\} = \frac{1}{|S^*|} \sum_{s\in S_q^*} \mathbf{1}\{N_s(T) \geq m\} . \tag{14}$$

The second equality follows because $N_s(T) = 0$ almost surely for $s \notin S_q^*$. Moreover,

$$R_{q,T}^{(1)} = R_{q,T}, \tag{15}$$

Thus, cumulative structural coverage is the case $m = 1$.

Proposition 3 (Expected repeated-exposure coverage)**.** *For every $m \in \{1,\dots,T\}$,*

$$\mathbb{E}\left[R_{q,T}^{(m)}\right] = \frac{1}{|S^*|} \sum_{s\in S_q^*} \left[1 - \sum_{j=0}^{m-1} \binom{T}{j} p_s^j (1-p_s)^{T-j}\right] . \tag{16}$$

Proof. By linearity of expectation,

$$\mathbb{E}\left[R_{q,T}^{(m)}\right] = \frac{1}{|S^*|} \sum_{s\in S_q^*} \mathbb{P}\left(N_s(T) \geq m\right).$$

Proposition 1 gives $N_s(T) \sim \text{Binomial}(T, p_s)$. Complementing the events $N_s(T) = j$ for $j \in \{0,\dots,m-1\}$ gives Eq. (16). Independence between different structures is not required. □

For a fixed partition, define the cluster-span counts and span-specific recovery probabilities as

$$n_c = \left|\{s \in S^* : c_s = c\}\right|, \qquad p_c(q) = \begin{cases} \frac{\binom{q-1}{c-1}}{\binom{K-1}{c-1}}, & c \leq q, \\ 0, & c > q. \end{cases} \tag{17}$$

Since $p_s = p_{c_s}(q)$, grouping Eq. (16) by cluster span gives

$$\mathbb{E}\left[R_{q,T}^{(m)}\right] = \frac{1}{|S^*|} \sum_{c=1}^{q} n_c \left[1 - \sum_{j=0}^{m-1} \binom{T}{j} p_c(q)^j \left(1-p_c(q)\right)^{T-j}\right] . \tag{18}$$

For $m = 1$, Eq. (18) reduces to the expected cumulative coverage in Eq. (13).

Corollary 1 (Fixed-threshold coverage convergence)**.** *For every fixed $m \geq 1$,*

$$R_{q,T}^{(m)} \longrightarrow \frac{|S_q^*|}{|S^*|} \qquad \textit{almost surely as } T \to \infty. \tag{19}$$

Consequently,

$$\lim_{T\to\infty} \mathbb{E}\left[R_{q,T}^{(m)}\right] = \frac{|S_q^*|}{|S^*|} .$$

Proof. Every $s \in S_q^*$ has $p_s > 0$. Eq. (9) therefore implies $N_s(T) \to \infty$ almost surely. Hence, $\mathbf{1}\{N_s(T) \geq m\} \to 1$ almost surely for each fixed m. Structures outside S_q^* have zero exposure, and S_q^* is finite. Eq. (14) then gives the almost-sure limit. Bounded convergence gives the corresponding expectation because $0 \leq R_{q,T}^{(m)} \leq 1$. □

At finite T, exposure differs across cluster spans because the expected count is $Tp_c(q)$.

C.7 Mean structural exposure rate

Thresholded coverage discards differences beyond its prescribed number of exposures. We therefore define the realized mean structural exposure rate as

$$\Phi_{q,T} = \frac{1}{|S^*|} \sum_{s \in S^*} \frac{N_s(T)}{T} . \tag{20}$$

Equivalently, $\Phi_{q,T}$ is the fraction of structure-epoch pairs in which the corresponding global structure is available.

Proposition 4 (Mean structural exposure rate)**.** *For every $T \geq 1$,*

$$\mathbb{E}[\Phi_{q,T}] = \Phi_q := \frac{1}{|S^*|} \sum_{s \in S^*} p_s = \frac{1}{|S^*|} \sum_{c=1}^{q} n_c p_c(q) . \tag{21}$$

Moreover,

$$\Phi_{q,T} \longrightarrow \Phi_q \qquad \textit{almost surely as } T \to \infty. \tag{22}$$

Proof. Eq. (8) gives $\mathbb{E}[N_s(T)/T] = p_s$. Linearity of expectation and grouping by cluster span give Eq. (21). For each $s \in S^*$, Eq. (9) gives $N_s(T)/T \to p_s$ almost surely. Averaging over the finite set S^* gives Eq. (22). □

Unlike fixed-threshold coverage, Φ_q retains span-dependent exposure imbalance as thresholded coverage approaches its limit. Since each epoch contains M optimization steps, the corresponding average frequency among training steps is

$$\frac{1}{|S^*|} \sum_{s \in S^*} \frac{N_s(T)}{MT} = \frac{\Phi_{q,T}}{M} \longrightarrow \frac{\Phi_q}{M} \qquad \text{almost surely.} \tag{23}$$

These quantities alone describe structural availability. They do not determine optimization behavior or predictive performance.

C.8 Exposure epoch distribution

The exposure count also identifies when a structure accumulates a prescribed number of exposures. For $m \geq 1$, define the epoch of the m-th exposure as

$$T_s^{(m)} = \inf\{t \in \mathbb{N} : N_s(t) \geq m\}, \tag{24}$$

with the convention that the infimum of the empty set is ∞.

Proposition 5 (Epoch of the m-th structural exposure)**.** *If $0 < p_s < 1$, then $T_s^{(m)}$ follows the negative-binomial distribution*

$$\mathbb{P}\left(T_s^{(m)} = \ell\right) = \binom{\ell - 1}{m - 1} p_s^m (1 - p_s)^{\ell - m}, \qquad \ell = m, m+1, \ldots . \tag{25}$$

If $p_s = 1$, then $T_s^{(m)} = m$ almost surely. For every $p_s > 0$, its expectation and variance are

$$\mathbb{E}\left[T_s^{(m)}\right] = \frac{m}{p_s}, \qquad \mathrm{Var}\left[T_s^{(m)}\right] = \frac{m(1 - p_s)}{p_s^2} . \tag{26}$$

If $p_s = 0$, then $T_s^{(m)} = \infty$ almost surely.

Proof. Let $X_s^{(t)} = \mathbf{1}\{s \in S^{(t)}\}$ denote the exposure indicator of structure s at epoch t, which forms an i.i.d. sequence of $\mathrm{Bernoulli}(p_s)$ random variables. Let $N_s(t) = \sum_{i=1}^{t} X_s^{(i)}$ be the cumulative exposure count up to epoch t.

Case 1: $0 < p_s < 1$**.** For any integer $\ell \geq m$, the event $T_s^{(m)} = \ell$ indicates that the m-th exposure of structure s occurs precisely at epoch ℓ. This occurs if and only if structure s is exposed exactly $m-1$ times in the first $\ell - 1$ epochs and is exposed again at epoch ℓ:

$$\{T_s^{(m)} = \ell\} = \{N_s(\ell - 1) = m - 1\} \cap \{X_s^{(\ell)} = 1\}.$$

Because epoch reshufflings are independent, $N_s(\ell-1)$ and $X_s^{(\ell)}$ are independent random variables. Using $N_s(\ell - 1) \sim \text{Binomial}(\ell - 1, p_s)$, we evaluate:

$$\begin{aligned}\mathbb{P}\left(T_s^{(m)} = \ell\right) &= \mathbb{P}\left(N_s(\ell-1) = m-1\right) \cdot \mathbb{P}\left(X_s^{(\ell)} = 1\right) \\ &= \left[\binom{\ell-1}{m-1} p_s^{m-1}(1-p_s)^{(\ell-1)-(m-1)}\right] \cdot p_s \\ &= \binom{\ell-1}{m-1} p_s^m (1-p_s)^{\ell-m}, \qquad \ell = m, m+1, \ldots,\end{aligned}$$

which establishes Eq. (25).

To derive the moments in Eq. (26), we represent $T_s^{(m)}$ as the sum of m inter-arrival waiting times:

$$T_s^{(m)} = \sum_{k=1}^{m} W_k,$$

where W_k represents the number of epochs elapsed between the $(k-1)$-th and k-th exposure of structure s. By the memoryless property of independent Bernoulli trials, $W_1, W_2, \ldots, W_m$ are independent and identically distributed Geometric random variables on $\{1, 2, \ldots\}$ with success probability p_s:

$$\mathbb{P}(W_k = w) = (1-p_s)^{w-1} p_s, \qquad w \in \{1, 2, \ldots\}.$$

First, applying linearity of expectation gives $\mathbb{E}\left[T_s^{(m)}\right] = \sum_{k=1}^{m} \mathbb{E}[W_k]$. Setting $\overline{p}_s = 1 - p_s$, we compute the expectation of an individual waiting time $\mathbb{E}[W_k]$:

$$\begin{aligned}\mathbb{E}[W_k] &= \sum_{w=1}^{\infty} w(1-p_s)^{w-1} p_s = p_s \sum_{w=1}^{\infty} w \overline{p}_s^{w-1} \\ &= p_s \frac{\mathrm{d}}{\mathrm{d}\overline{p}_s}\left(\sum_{w=0}^{\infty} \overline{p}_s^w\right) = p_s \frac{\mathrm{d}}{\mathrm{d}\overline{p}_s}\left(\frac{1}{1-\overline{p}_s}\right) = p_s \cdot \frac{1}{(1-\overline{p}_s)^2} = \frac{p_s}{p_s^2} = \frac{1}{p_s}.\end{aligned}$$

Summing over the m independent inter-arrival periods yields:

$$\mathbb{E}\left[T_s^{(m)}\right] = \sum_{k=1}^{m} \mathbb{E}[W_k] = \sum_{k=1}^{m} \frac{1}{p_s} = \frac{m}{p_s}.$$

Second, for the variance of the finite sum, we expand:

$$\text{Var}\left[T_s^{(m)}\right] = \sum_{k=1}^{m} \text{Var}(W_k) + 2 \sum_{1 \leq i < j \leq m} \text{Cov}(W_i, W_j).$$

Because the epoch trials determining each W_k are stochastically independent across disjoint epoch intervals, all pairwise covariances vanish ($\text{Cov}(W_i, W_j) = 0$ for $i \neq j$). Thus we have, $\text{Var}\left[T_s^{(m)}\right] = \sum_{k=1}^{m} \text{Var}(W_k)$.

To determine $\text{Var}(W_k) = \mathbb{E}[W_k^2] - (\mathbb{E}[W_k])^2$, we first evaluate the second factorial moment:

$$\begin{aligned}\mathbb{E}[W_k(W_k-1)] &= \sum_{w=2}^{\infty} w(w-1)(1-p_s)^{w-1} p_s = p_s \overline{p}_s \sum_{w=2}^{\infty} w(w-1)\overline{p}_s^{w-2} \\ &= p_s \overline{p}_s \frac{\mathrm{d}^2}{\mathrm{d}\overline{p}_s^2}\left(\frac{1}{1-\overline{p}_s}\right) = p_s \overline{p}_s \cdot \frac{2}{(1-\overline{p}_s)^3} = \frac{2p_s(1-p_s)}{p_s^3} = \frac{2(1-p_s)}{p_s^2}.\end{aligned}$$

The second raw moment is therefore:

$$\mathbb{E}[W_k^2] = \mathbb{E}[W_k(W_k-1)] + \mathbb{E}[W_k] = \frac{2(1-p_s)}{p_s^2} + \frac{1}{p_s} = \frac{2-p_s}{p_s^2},$$

which yields the single-variable variance:

$$\mathrm{Var}(W_k) = \frac{2-p_s}{p_s^2} - \frac{1}{p_s^2} = \frac{1-p_s}{p_s^2}.$$

Summing across m independent variables completes the variance expression in Eq. (26):

$$\mathrm{Var}\left[T_s^{(m)}\right] = \sum_{k=1}^{m} \mathrm{Var}(W_k) = \sum_{k=1}^{m} \frac{1-p_s}{p_s^2} = \frac{m(1-p_s)}{p_s^2}.$$

Case 2: $p_s = 1$**.** If $p_s = 1$, then $X_s^{(t)} = 1$ almost surely for all $t \in \mathbb{N}$, so $N_s(t) = t$ almost surely. The stopping condition $N_s(t) \geq m$ simplifies to $t \geq m$, yielding $T_s^{(m)} = m$ almost surely. Substituting $p_s = 1$ into Eq. (26) confirms $\mathbb{E}\left[T_s^{(m)}\right] = m$ and $\mathrm{Var}\left[T_s^{(m)}\right] = 0$.

Case 3: $p_s = 0$**.** If $p_s = 0$, Proposition 1 gives $N_s(T) = 0$ almost surely for all $T \in \mathbb{N}$. Since $m \geq 1$, the exposure threshold is never satisfied, meaning $\{t \in \mathbb{N} : N_s(t) \geq m\} = \emptyset$ almost surely. By convention, $\inf \emptyset = \infty$, so $T_s^{(m)} = \infty$ almost surely. □

The expected exposure epoch characterizes an implicit curriculum induced by random recombination. A one-cluster structure reaches its m-th exposure at epoch m. When $q < K$, observable structures with larger spans reach the same threshold later on average. Structures with $c_s > q$ never reach it.

C.9 Direct-incidence message availability

Structural exposure affects message passing through the incidences available in each lifted mini-batch. Write $\tau \prec \sigma$ for a direct incidence satisfying $\mathcal{V}_\tau \subseteq \mathcal{V}_\sigma$. We call τ the lower structure and σ the higher structure. A model may pass messages in either or both directions. We denote these directed instances by $\mu_{\tau\to\sigma}$ and $\mu_{\sigma\to\tau}$.

Definition 8 (Direct-incidence message compatibility)**.** *A direct-incidence message rule is compatible with mini-batch lifting if the following condition holds for every global direct incidence $\tau \prec \sigma$. Each direction used by the model is available exactly when the lifted mini-batch contains the higher structure σ. Equivalently, for $\mu \in \{\mu_{\tau\to\sigma}, \mu_{\sigma\to\tau}\}$,*

$$\{\mu \text{ is available in mini-batch } i \text{ of epoch } t\} \iff I_\sigma \subseteq Z_i^{(t)}\ .$$

This condition isolates structural availability. It does not require the numerical value of a message to remain unchanged when it is computed on a mini-batch.

For a directed message μ associated with $\tau \prec \sigma$, define its per-epoch availability indicator as

$$J_\mu^{(t)} = \mathbf{1}\{\mu \text{ is available in epoch } t\}\ . \tag{27}$$

The inclusion $\mathcal{V}_\tau \subseteq \mathcal{V}_\sigma$ implies $I_\tau \subseteq I_\sigma$. Definitions 7 and 8 therefore give

$$J_\mu^{(t)} = X_\sigma^{(t)} \qquad \text{for either message direction.} \tag{28}$$

Thus, the unconditional availability of either directed message is governed by the cluster span of its higher endpoint.

For target updates, the relevant quantity is conditional message availability. Let $b(\mu) \in \{\tau, \sigma\}$ denote the target of μ. For any target with $p_{b(\mu)} > 0$, define

$$\alpha_{\mu|b(\mu)} = \mathbb{P}\left(J_\mu^{(t)} = 1 \mid X_{b(\mu)}^{(t)} = 1\right). \tag{29}$$

Proposition 6 (Conditional availability of a direct-incidence message)**.** *Let $\tau \prec \sigma$. If $c_\sigma \le q$, then a lower-to-higher message satisfies*

$$\alpha_{\mu_{\tau\to\sigma}|\sigma} = 1\ . \tag{30}$$

If $c_\tau \le q$, then a higher-to-lower message satisfies

$$\alpha_{\mu_{\sigma\to\tau}|\tau} = \frac{p_\sigma}{p_\tau} = \begin{cases} \frac{\binom{q-c_\tau}{c_\sigma-c_\tau}}{\binom{K-c_\tau}{c_\sigma-c_\tau}}, & c_\sigma \le q, \\ 0, & c_\sigma > q. \end{cases} \tag{31}$$

Proof. Eq. (28) shows that either directed message is available exactly when σ is recovered. For the lower-to-higher direction, target recovery and message availability are the same event. This gives Eq. (30).

For the higher-to-lower direction, $I_\tau \subseteq I_\sigma$ implies $\{X_\sigma^{(t)} = 1\} \subseteq \{X_\tau^{(t)} = 1\}$. Hence,

$$\alpha_{\mu_{\sigma\to\tau}|\tau} = \mathbb{P}\left(X_\sigma^{(t)} = 1 \mid X_\tau^{(t)} = 1\right) = \frac{p_\sigma}{p_\tau}\ .$$

Conditional on recovery of τ, the other $q - c_\tau$ batch members form a uniform subset of the $K - c_\tau$ clusters outside I_τ. Message availability requires this subset to contain the fixed set $I_\sigma \setminus I_\tau$. When $c_\sigma \le q$, its conditional probability is

$$\frac{\binom{K-c_\sigma}{q-c_\sigma}}{\binom{K-c_\tau}{q-c_\tau}} = \frac{\binom{q-c_\tau}{c_\sigma-c_\tau}}{\binom{K-c_\tau}{c_\sigma-c_\tau}}\ .$$

If $c_\sigma > q$, the higher structure and its incident message are unavailable. □

The two directions have the same unconditional availability but different conditional availability. Every directly incident lower-to-higher message is present when σ is updated. A higher-to-lower message is present only when the additional clusters supporting σ accompany I_τ. The asymmetry therefore comes from the target-update event, not from a direction-dependent sampling rule.

C.10 Repeated direct-incidence message exposure

For a directed message μ associated with $\tau \prec \sigma$, define its exposure count by

$$N_\mu(T) = \sum_{t=1}^{T} J_\mu^{(t)}\ . \tag{32}$$

Proposition 7 (Repeated direct-incidence message exposure)**.** *Assume independent epoch reshuffling and both compatibility conditions. Either direction of a direct-incidence message then satisfies*

$$N_\mu(T) = N_\sigma(T) \sim \mathrm{Binomial}(T, p_\sigma)\ . \tag{33}$$

Let $b = b(\mu)$ be an observable target, so that $p_b > 0$. For every n with $\mathbb{P}(N_b(T) = n) > 0$,

$$N_\mu(T) \mid N_b(T) = n \sim \mathrm{Binomial}\left(n, \alpha_{\mu|b}\right). \tag{34}$$

Consequently,

$$\mathbb{E}\left[N_\mu(T) \mid N_b(T) = n\right] = n\alpha_{\mu|b}, \qquad \mathrm{Var}\left[N_\mu(T) \mid N_b(T) = n\right] = n\alpha_{\mu|b}\left(1 - \alpha_{\mu|b}\right), \tag{35}$$

and

$$\frac{N_\mu(T)}{N_b(T)} \longrightarrow \alpha_{\mu|b} \qquad \textit{almost surely as } T \to \infty. \tag{36}$$

The ratio in Eq. (36) *is considered once $N_b(T) > 0$, which occurs eventually almost surely.*

Proof. By direct-incidence message compatibility,

$$J_\mu^{(t)} = X_\sigma^{(t)}$$

for every epoch t, where $X_\sigma^{(t)}$ indicates that the higher structure σ is recovered. Hence

$$N_\mu(T) = \sum_{t=1}^{T} J_\mu^{(t)} = \sum_{t=1}^{T} X_\sigma^{(t)} = N_\sigma(T) \, .$$

Under independent epoch reshuffling, the variables $X_\sigma^{(t)}$ are i.i.d. Bernoulli(p_σ). Therefore

$$N_\mu(T) = N_\sigma(T) \sim \mathrm{Binomial}(T, p_\sigma) \, .$$

Now let $b = b(\mu)$ be the target and assume $p_b > 0$. Since message availability implies target availability,

$$\mathbb{P}\Big(J_\mu^{(t)} = 1 \mid X_b^{(t)} = 1\Big) = \frac{p_\sigma}{p_b} =: \alpha_{\mu|b}.$$

Since message availability implies target availability, each epoch belongs to one of three categories. Their probabilities are p_σ, $p_b - p_\sigma$, and $1 - p_b$. These categories correspond to target with message, target without message, and absent target. They are independent and identically distributed across epochs. Conditional on $N_b(T) = n$, the n epochs in which the target is present therefore contain a binomial number of message exposures:

$$N_\mu(T) \mid N_b(T) = n \sim \mathrm{Binomial}(n, \alpha_{\mu|b}).$$

Thus

$$\mathbb{E}[N_\mu(T) \mid N_b(T) = n] = n\alpha_{\mu|b},$$

and

$$\mathrm{Var}(N_\mu(T) \mid N_b(T) = n) = n\alpha_{\mu|b}(1 - \alpha_{\mu|b}).$$

Finally, by the strong law of large numbers we have

$$\frac{N_\mu(T)}{T} \to p_\sigma, \quad \frac{N_b(T)}{T} \to p_b \qquad \text{almost surely as } T \to \infty.$$

Since $p_b > 0$,

$$\frac{N_\mu(T)}{N_b(T)} = \frac{N_\mu(T)/T}{N_b(T)/T} \to \frac{p_\sigma}{p_b} = \alpha_{\mu|b} \qquad \text{almost surely as } T \to \infty.$$

□

The conditional rate $\alpha_{\mu|b}$ is the asymptotic fraction of updates of b that can include μ. It equals one for lower-to-higher messages. For higher-to-lower messages, it depends on the additional cluster span of the higher structure. It may remain small even when the lower target is updated frequently.

C.11 Layerwise incidence aggregation

The availability probabilities yield a layerwise consequence. Let $\mathcal{M}_{\mathrm{inc}}(b)$ contain the full-domain direct-incidence messages entering an observable target b. Fix a layer ℓ, its inputs, its parameters, and every message coefficient. Let $\mathbf{m}_\mu^{(\ell)}$ be the resulting full-domain contribution of $\mu \in \mathcal{M}_{\mathrm{inc}}(b)$. The additive direct-incidence component of the full-domain pre-activation is

$$\mathbf{z}_{b,\mathrm{inc}}^{(\ell+1)} = \sum_{\mu \in \mathcal{M}_{\mathrm{inc}}(b)} \mathbf{m}_\mu^{(\ell)} \, . \tag{37}$$

When b is recovered in epoch t, define the availability-masked component as

$$\widehat{\mathbf{z}}_{b,\mathrm{inc}}^{(\ell+1,t)} = \sum_{\mu \in \mathcal{M}_{\mathrm{inc}}(b)} J_\mu^{(t)} \mathbf{m}_\mu^{(\ell)} \, . \tag{38}$$

Proposition 8 (Expected direct-incidence aggregation)**.** *Under the fixed-input availability mask above,*

$$\mathbb{E}\left[\widehat{\mathbf{z}}_{b,\mathrm{inc}}^{(\ell+1,t)} \mid X_b^{(t)} = 1\right] = \sum_{\mu \in \mathcal{M}_{\mathrm{inc}}(b)} \alpha_{\mu|b} \mathbf{m}_\mu^{(\ell)} \, . \tag{39}$$

Therefore, the conditional mean aggregation discrepancy satisfies

$$\mathbf{z}_{b,\mathrm{inc}}^{(\ell+1)} - \mathbb{E}\left[\widehat{\mathbf{z}}_{b,\mathrm{inc}}^{(\ell+1,t)} \mid X_b^{(t)} = 1\right] = \sum_{\mu \in \mathcal{M}_{\mathrm{inc}}(b)} \left(1 - \alpha_{\mu|b}\right) \mathbf{m}_\mu^{(\ell)}, \tag{40}$$

and, for any norm,

$$\left\| \mathbf{z}_{b,\mathrm{inc}}^{(\ell+1)} - \mathbb{E}\left[\widehat{\mathbf{z}}_{b,\mathrm{inc}}^{(\ell+1,t)} \mid X_b^{(t)} = 1\right] \right\| \leq \sum_{\mu \in \mathcal{M}_{\mathrm{inc}}(b)} \left(1 - \alpha_{\mu|b}\right) \left\| \mathbf{m}_\mu^{(\ell)} \right\| . \tag{41}$$

Proof. Conditional linearity of expectation and Eq. (29) give

$$\mathbb{E}\left[J_\mu^{(t)} \mathbf{m}_\mu^{(\ell)} \mid X_b^{(t)} = 1\right] = \alpha_{\mu|b} \mathbf{m}_\mu^{(\ell)}$$

for each incoming message. Summing proves Eq. (39). Subtraction from Eq. (37) gives Eq. (40). The triangle inequality gives Eq. (41). Independence between incident messages is not required. □

For lower-to-higher aggregation into σ, every direct boundary message has unit availability whenever σ is updated. For higher-to-lower aggregation into τ, Eq. (31) scales each coface contribution in conditional expectation. The coefficient reduction depends on the extra clusters required by that coface. Vector cancellation prevents these coefficients from implying a monotone change in aggregate norm. Eq. (39)-(41) instead characterize the conditional mean and bound its discrepancy.

This result is deliberately layerwise. Batch-dependent degree, incidence, Hodge, or attention normalization can change numerical message values. Representations produced by earlier sampled layers also depend on the mini-batch. Nonlinear updates prevent the expectation from passing directly through later layers. The proposition isolates structural availability at fixed inputs and parameters. It does not establish unbiasedness or convergence of the complete nonlinear network. Same-rank messages mediated by incidence paths or Hodge products require a separate witness-level analysis and are not covered here.

C.12 Cumulative recovery entropy

We associate an entropy with the event that an observable global structure has appeared by epoch T. Recall that

$$S_q^* = \{s \in S^* : c_s \leq q\}$$

is the set of global structures whose cluster span fits within a mini-batch of size q. For each $s \in S_q^*$, define the cumulative recovery indicator

$$Y_s^{(T)} = \mathbf{1}\{s \in \widehat{S}_T^*\}.$$

By Eq. (11),

$$Y_s^{(T)} \sim \mathrm{Bernoulli}(\rho_s(T)), \qquad \rho_s(T) = \mathbb{P}\left(s \in \widehat{S}_T^*\right) = 1 - (1 - p_s)^T.$$

Let

$$h(x) = -x \ln x - (1 - x) \ln(1 - x)$$

denote the Bernoulli entropy in nats, with $0 \ln 0 = 0$. Since $Y_s^{(T)}$ has success probability $\rho_s(T)$, its entropy is $h(\rho_s(T))$. We define the aggregate marginal cumulative recovery entropy as

$$H_{q,T} = \frac{1}{|S^*|} \sum_{s \in S_q^*} h\left(\rho_s(T)\right) . \tag{42}$$

Equivalently, the normalization assigns zero entropy to unobservable structures because their recovery events have probability zero.

Proposition 9 (Cumulative recovery entropy convergence)**.** *The aggregate marginal cumulative recovery entropy satisfies*

$$H_{q,T} \to 0 \qquad \textit{as} \qquad T \to \infty.$$

Proof. For every $s \in S_q^*$, Lemma 1 gives $p_s > 0$. Hence

$$\rho_s(T) = 1 - (1 - p_s)^T \to 1 \qquad \text{as} \qquad T \to \infty.$$

The function h is continuous on $[0, 1]$ and satisfies $h(1) = 0$. Hence, $h(\rho_s(T)) \to 0$ for every $s \in S_q^*$. Summing over the finite set S_q^* gives $H_{q,T} \to 0$. □

Thus, the marginal recovery uncertainty of every structure in S_q^* vanishes with increasing T. Theorem 1 also gives joint recovery of S_q^* with probability tending to one. Full recovery occurs when $S_q^* = S^*$, equivalently when $q \geq \max_{s \in S^*} c_s$.

C.13 Empirical analysis of structural recovery

C.13.1 Structural recovery across datasets

Experimental setup. A full-graph structure can fit within some group of q clusters if its cluster span satisfies $c_s \leq q$, giving the observable fraction $|S_q^*|/|S^*|$. Figures 2 and 6 instead measure how many structures' supporting nodes actually appear together within 200 epochs, with the latter extending the comparison to all four datasets. We fix one METIS partition per dataset, using $K = 32$ for Cora Full and Amazon Ratings, $K = 500$ for Questions, and $K = 2\,000$ for Coauthor Physics. At each epoch, we independently reshuffle all clusters and group them into mini-batches of q clusters. For each structure, we record whether all its supporting nodes occur together in at least one mini-batch by epoch 200. We divide the number of such structures by the total number of full-graph reference structures in the corresponding family, and report the mean and sample standard deviation over ten independent reshuffling runs for each q. No model is trained in this diagnostic.

The reference families comprise one centre-indexed neighbourhood hyperedge per node, cellular 2-cells from a fixed full-graph cycle basis with cycle length at most nine, and simplicial 2-cells corresponding to graph triangles. Membership-identical neighbourhood hyperedges retain their centre identities. The cellular curve measures joint sampling of a reference cycle's nodes, not whether recomputing a cycle basis on the induced mini-batch selects the identical cycle. A graph can admit multiple cycle bases, and changing the node or neighbour traversal order can change the basis selected by the lifting, even without changing the graph. Each partition and reference family remains fixed as q varies, so the results characterize the evaluated graph-lifting-partition configurations.

At $q = 1$, reshuffling cannot change which nodes occur together, so support recovery is determined entirely by structures contained within individual clusters. At $q = K$, all nodes occur in one batch and support recovery is necessarily complete. Both endpoint values are reached after the first epoch.

Cora Full and Amazon Ratings. Most cellular and simplicial 2-cells already lie within individual clusters under these partitions. At $q = 1$, the recovered fractions are 83.3% and 86.4% on Cora Full, and 97.7% and 99.0% on Amazon Ratings, respectively. Neighbourhood hyperedges have lower initial recovery, 59.7% and 90.0%. Increasing q closes much of the remaining gap. At $q = 4$, recovery exceeds 99% for both 2-cell families on Cora Full and for all three families on Amazon Ratings. Cora Full neighbourhood hyperedges reach 91.1% at the same value. Thus, these partitions preserve much of the reference structure within single clusters, while recombination exposes additional structures crossing cluster boundaries.

Questions and Coauthor Physics. These configurations exhibit a larger gap between observability in principle and recovery within a finite number of epochs. On Questions at $q = 20$, all cellular and simplicial 2-cells are observable, yet only 22.0% and 31.9% of their supports are encountered by epoch 200, respectively. On Coauthor Physics at $q = 100$, only 59.3% of cellular and 58.9% of simplicial supports are encountered by epoch 200, although each individual reference 2-cell could fit within a mini-batch of this size. Having enough clusters in a mini-batch to accommodate a structure does not ensure that its particular supporting clusters are sampled together within the available epochs. Larger groups increase these opportunities. On Questions at $q = 100$, simplicial recovery exceeds 99.9%, while cellular recovery is 64.0%. On Coauthor Physics at $q = 500$, the corresponding fractions exceed 99.9% and reach 97.0%.

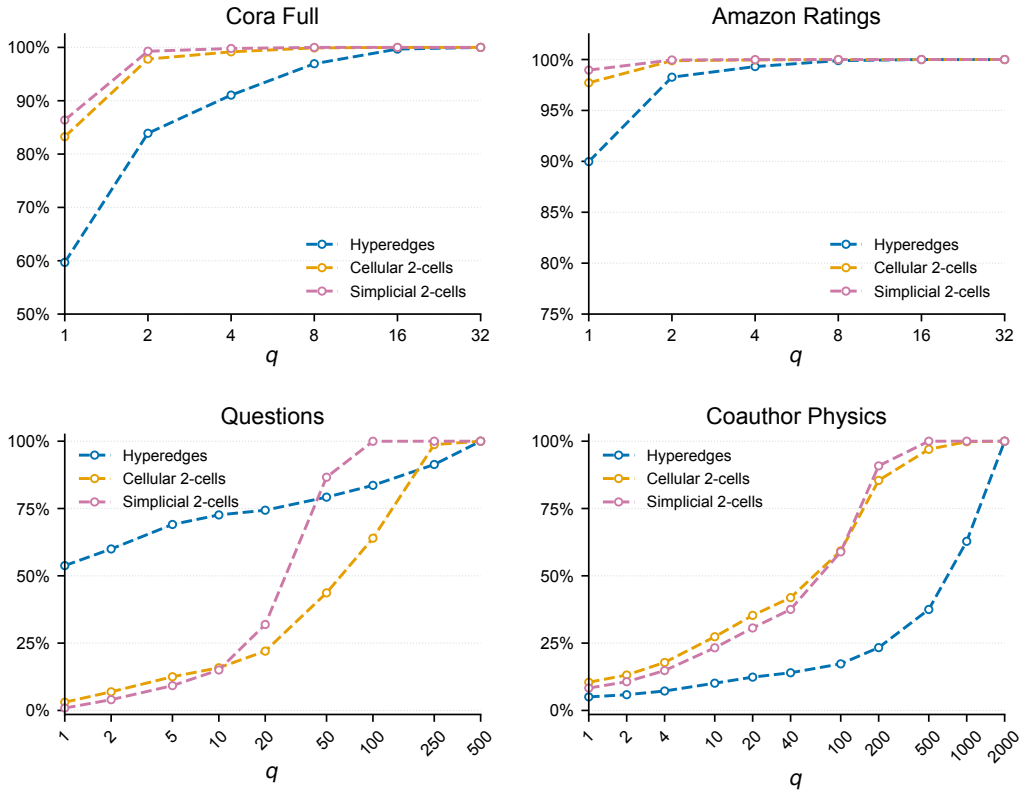


Figure 6: Fraction of full-graph reference structures whose supporting nodes appear together in at least one mini-batch within 200 epochs, as a function of the number of clusters per mini-batch q. Panels show Cora Full and Amazon Ratings ($K = 32$, top row), Questions ($K = 500$, bottom left), and Coauthor Physics ($K = 2\,000$, bottom right). Each dataset uses one fixed METIS partition. Curves show means over ten independent reshuffling runs. Shading indicates ± 1 standard deviation, generally narrower than the plotted lines. Cellular recovery refers to the supporting nodes of fixed full-graph cycle-basis cells, not their selection by a recomputed local basis. Vertical scales begin at 50% for Cora Full, 75% for Amazon Ratings, and 0% for the other datasets.

Differences between higher-order domains. The lifting definitions help explain why the three families respond differently to recombination. Each simplicial 2-cell contains three nodes, and each retained cellular 2-cell contains at most nine. A neighbourhood hyperedge contains its centre and all its neighbours, with no corresponding fixed size limit. Its complete support can therefore span many more clusters, making joint sampling less likely. On Coauthor Physics at $q = 1\,000$, both 2-cell families exceed 99.8% recovery, whereas neighbourhood hyperedges reach 62.8%. However, neighbourhood hyperedges are not uniformly harder to recover. On Questions at $q = 1$, 53.8% of neighbourhood hyperedges lie within individual clusters, compared with 3.0% of cellular and 0.9% of simplicial 2-cells. Both support size and its distribution across the partition matter. Low recovery of complete neighbourhoods also does not imply an absence of local hypergraph structure, because lifting a mini-batch constructs hyperedges from the neighbors present. Likewise, a batch-local cycle basis can contain cells different from the full-graph references.

Implications for repeated sampling. Addressing the structural component of Q3, the finite-epoch results distinguish structures that can become available from those encountered within a given sampling horizon. This distinction motivates considering both mini-batch size and repeated sampling when assessing structural exposure. Ensemble inference similarly evaluates nodes under several independently sampled cluster groupings. Table 2 provides complementary predictive evidence, with ensemble inference achieving a higher mean test metric than single-pass inference in all 21 reported

model-dataset comparisons. These results support evaluating multiple structural contexts, but do not establish that recovery of full-graph reference structures explains the predictive improvement.

C.13.2 Recovery across epochs on Cora Full

For liftings compatible with induced subgraphs, the span counts n_c and probabilities $p_c(q)$ from Eq. (17) give the expected one-exposure coverage

$$\mathbb{E}[R_{q,T}] = \frac{1}{|S^*|} \sum_{c=1}^{q} n_c \left(1 - (1 - p_c(q))^T\right) . \tag{43}$$

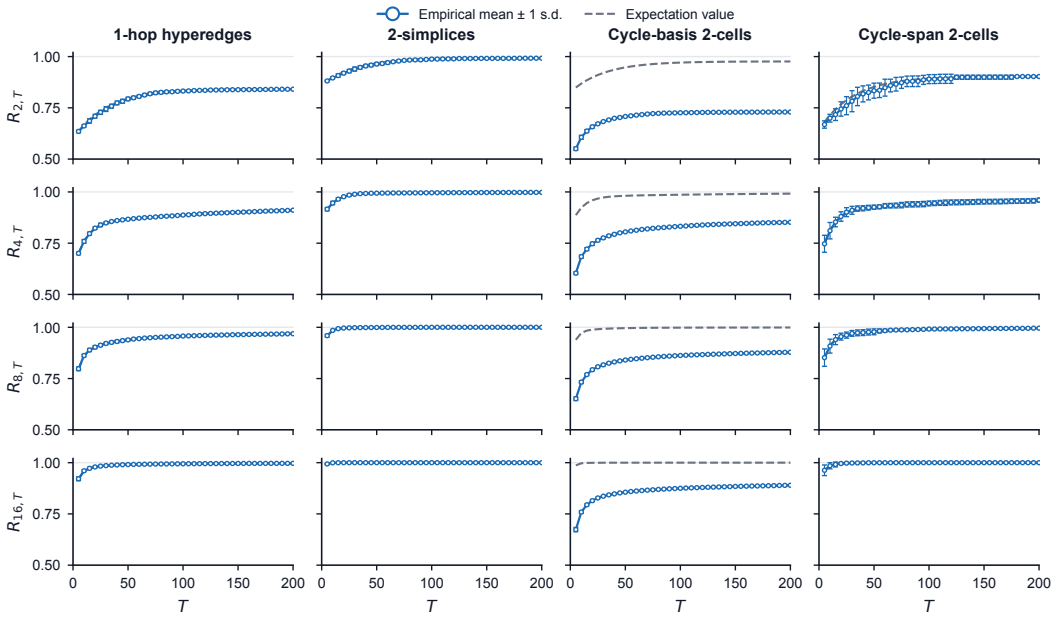


Figure 7: Comparison of empirical and analytical structural recovery on Cora Full with $K = 32$ and $q \in \{2, 4, 8, 16\}$. Empirical coverage is shown as the mean ± 1 sample standard deviation across ten independent runs and compared with the analytical coverage from Eq. (43) over 200 epochs.

We also evaluate the aggregate marginal entropy of cumulative support availability from the same span counts:

$$H_{q,T} = \frac{1}{|S^*|} \sum_{c=1}^{q} n_c \, h\left(1 - (1 - p_c(q))^T\right) , \tag{44}$$

where h is the Bernoulli entropy defined in Sec. C.12. Eq. (18) provides the corresponding expression for any exposure threshold $m \in \{1, \ldots, T\}$.

We evaluate Cora Full using one fixed METIS partition with $K = 32$ and mini-batches of $q \in \{2, 4, 8, 16\}$ clusters. For each q, we simulate ten independent cluster-reshuffling sequences, each lasting 200 epochs and using a different seed. The reference families comprise center-indexed 1-hop hyperedges, graph triangles, full-graph cycle-basis cells of length at most nine, and cycle-span cells. The cycle-basis lifting need not satisfy the compatibility condition in Definition 7: recomputing a basis can omit a reference cycle even when its complete support is present. We therefore include the cycle-span family as a compatible reference for testing the support-based recovery prediction. This family comprises all distinct undirected simple cycles of lengths three to eight. Each cycle appears in an induced mini-batch whenever all its supporting clusters occur together, independently of any basis selection. We count recovery exactly by grouping cycles with the same supporting cluster set. For cycle-basis cells, we instead record whether the recomputed mini-batch basis selects each full-graph reference cycle. The expanded family is used only in this structural diagnostic. CWN and TopoTune retain the cycle-basis lifting described in Appendix A.2.1.

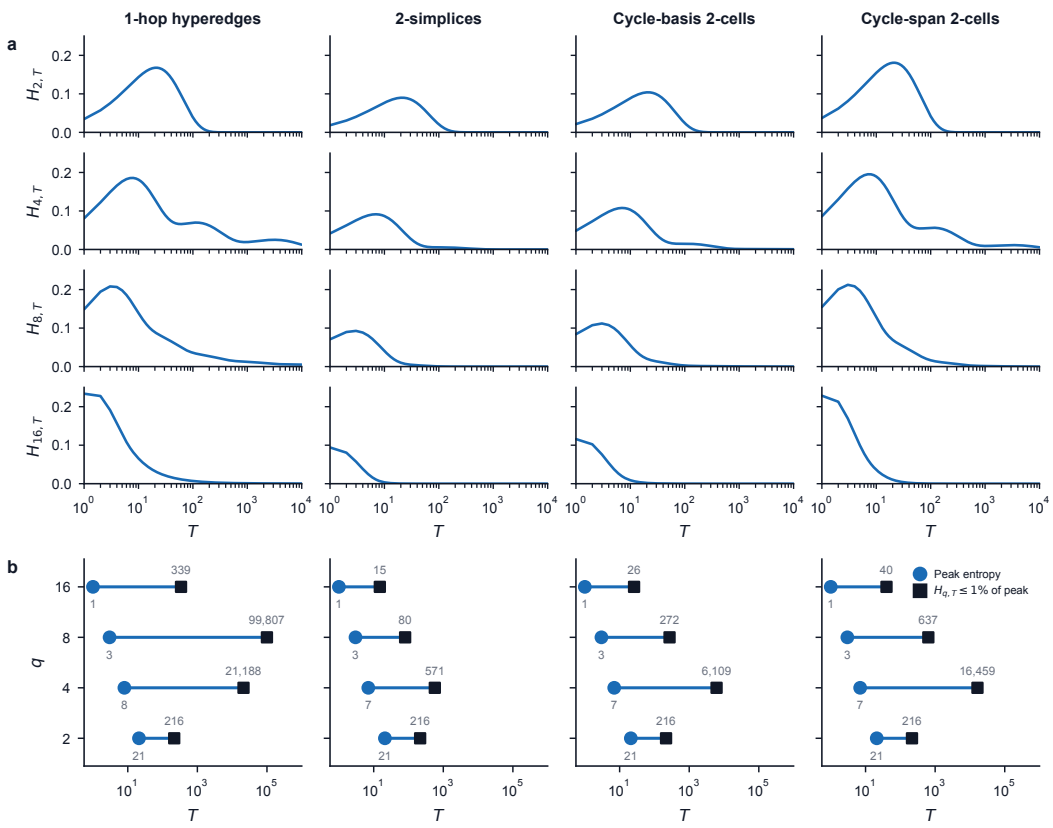


Figure 8: Analytical entropy of cumulative support availability on Cora Full ($K = 32$). **a** Eq. (44) for $T \leq 10^4$. **b** Blue circles mark peak epochs, while black squares mark the final crossing below 1% of peak entropy.

Figure 7 compares cumulative empirical recovery with Eq. (43) for these reference families. Empirical means agree closely with the prediction for 1-hop hyperedges, triangles, and expanded cycles. Cycle-basis recovery falls below the support-based prediction because restricting the graph can change which cycles the basis selects.

Figure 8 evaluates Eq. (44) analytically from each reference family's fixed cluster-span histogram, normalized by the total number of reference structures. It measures the average marginal uncertainty about whether each reference structure's complete support has appeared together by epoch T. For cycle-basis cells, support availability does not guarantee selection by a recomputed local basis. Their entropy curves therefore describe support availability rather than the actual basis recovery measured in Figure 7.

We define the decay epoch as the earliest epoch after the entropy peak from which entropy remains at or below 1% of its maximum. Both the peak and decay epochs are analytical quantities, including beyond the 200-epoch sampling horizon. They do not measure training convergence.

D Additional Results

GPU memory across partition configurations. Figure 9 examines how partition granularity and mini-batch size affect peak GPU memory on Cora Full, Amazon Ratings, and Questions. We evaluate $K \in \{32, 64, 128, 256\}$ with $q/K \in \{0.125, 0.25, 0.5, 0.75\}$. For each model-dataset pair, we fix the model hyperparameters to the configuration selected for Cluster-TNN training and use the same settings for the full-graph baseline. We report peak CUDA allocated memory across training and validation in decimal GB. The measurements characterize memory sensitivity to K and q at fixed model hyperparameters.

On Cora Full, peak GPU memory increases with the sampled cluster fraction across all seven models. At $q/K = 0.125$, Cluster-TNN reduces peak memory by 85.4-88.7% relative to the corresponding full-graph baseline across the four partition counts. At $q/K = 0.75$, these reductions narrow to 24.7-29.7%. For SCCNN with $K = 32$, increasing q from 4 to 24 raises peak memory from 4.08 to 22.05 GB, compared with 30.58 GB for full-graph execution. Amazon Ratings and Questions exhibit the same qualitative trend. These results illustrate how Cluster-TNN can be configured through K and q to balance GPU memory use and the amount of graph context processed per mini-batch.



Figure 9: Peak GPU memory under Cluster-TNN and full-graph execution on three datasets. Rows correspond to models, while columns vary the number of partitions K and clusters per mini-batch q, with percentages indicating q/K. The rightmost column shows the full-graph baseline. Each dataset has a separate color scale.

E Experimental protocol and hyperparameter optimization

Dataset splits and random seeds. Our predictive-performance experiments use transductive node classification with generated, non-stratified random node splits rather than predefined public splits. The training/validation/test proportions are $70\%/15\%/15\%$ for Cora Full, Coauthor Physics, and OGBN Products, and $50\%/25\%/25\%$ for Amazon Ratings, Questions, and Reddit. For each dataset, we generate ten random splits with a fixed seed of 42 and use split seeds $0, 1, 2, 3, 4$ for final evaluation. The same splits are used across models and between full-graph and Cluster-TNN execution wherever both are evaluated. For run s, the corresponding seed also controls model initialization, stochastic training, and cluster recombinations during partitioned training and inference. Thus, variation across the five runs reflects both data-split and model-training randomness.

Search objective and procedure. We optimize each model-dataset pair separately for full-graph and Cluster-TNN partitioned training using Optuna's TPESampler (Akiba et al., 2019; Watanabe, 2023). Each trial maximizes the best validation metric attained during training. Test data are not evaluated during hyperparameter optimization. Dataset-specific metrics are listed in Table 11. Graph and lifted-domain structural counts are reported separately in Table 3.

Each study comprises 100 trials for Questions, Amazon Ratings, Cora Full, and Coauthor Physics, 50 trials for Reddit, and 30 trials for OGBN Products. The first ten trials use random sampling, after

which the sampler uses previous trial outcomes to guide subsequent choices. Search trials run for at most 300 epochs, with validation every five epochs and early stopping after five validation checks without improvement.

Table 11: Dataset tasks, input feature dimensions, and class counts. Structural counts are reported separately in Table 3.

Dataset	Metric	Input feature dimension	Classes
Cora Full	Accuracy	8 710	70
Amazon Ratings	Accuracy	300	5
Questions	AUROC	301	2
Coauthor Physics	Accuracy	8 415	5
Reddit	Accuracy	602	41
OGBN Products	Accuracy	100	47

Model hyperparameters. The model search varies optimization step size, regularization, and representation width. Learning rates are selected from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, covering several orders of magnitude. Weight decay is selected from $\{0, 10^{-5}, 10^{-4}, 10^{-3}\}$, allowing comparison between no weight decay and different regularization strengths. We search encoder widths of $\{32, 64, 128\}$ channels to vary representation capacity and the memory occupied by hidden features during model execution.

Projection dropout is selected from $\{0, 0.1, 0.2, 0.3\}$. For GCN, EDHNN, and TopoTune, we additionally search backbone dropout in $\{0, 0.2, 0.5\}$. Both dropout spaces include a setting without dropout. These model search spaces are shared across datasets and between full-graph and Cluster-TNN training wherever the corresponding parameter applies.

Partition hyperparameters. For partitioned training, we jointly optimize the number of graph partitions K and the number of clusters combined into each training mini-batch q. Increasing K produces smaller clusters on average. For a fixed partition, increasing q expands the sampled graph context and allows higher-order structures spanning more clusters to be recovered, but can also increase lifting and model-execution costs. Searching both parameters allows validation-based selection among different partition granularities and amounts of graph context.

Table 12 lists the dataset-specific candidate sets. Within each dataset, all models use the same K, q search space.

Table 12: Dataset-specific partition search spaces. K denotes the total number of graph partitions, and q denotes the number of clusters combined into one training mini-batch. These parameters are optimized jointly with the model hyperparameters.

Dataset	Candidate K values	Candidate q values
Questions	$\{500, 1\,000, 1\,500, 2\,000\}$	$\{20, 30, 40, 50\}$
Amazon Ratings	$\{32, 64, 128, 256\}$	$\{4, 8, 16, 32\}$
Cora Full	$\{32, 64, 128, 256\}$	$\{4, 8, 16, 32\}$
Coauthor Physics	$\{500, 1\,000, 1\,500, 2\,000\}$	$\{20, 30, 40, 50\}$
Reddit	$\{7\,000, 10\,000\}$	$\{20, 25\}$
OGBN Products	$\{15\,000, 20\,000\}$	$\{20, 25\}$

With the same number of partitions K, a graph containing more nodes has more nodes per cluster on average. Combining q clusters can therefore produce very different mini-batch sizes across datasets. Connectivity further determines how many higher-order cells the lifting constructs from those nodes. These differences affect the time and memory required to construct and process each mini-batch.

We selected the dataset-specific candidate ranges through preliminary runs assessing the memory required for mini-batch lifting and model execution.

Selected configurations and final evaluation. Table 13 reports the selected hyperparameters. Final evaluations use five seeds. For OGBN Products, final training runs for at most 130 epochs, with validation every five epochs and early stopping after two validation checks without improvement. For final training and evaluation, validation and test mini-batches use the selected training q, as reported in Table 13. Test predictions for P are obtained using ensemble inference with ten independent cluster recombinations.

Table 13: Selected hyperparameters for full-graph and Cluster-TNN training. Shading denotes parameters that do not apply to the corresponding model or protocol.

Dataset	Protocol	Learning Rate	Weight Decay	Hidden Channels	Projection Dropout	K	q	Dropout
GCN								
Questions	Full-graph	10^{-2}	10^{-4}	128	0.3			0.5
	Cluster-TNN	10^{-4}	0	128	0.2	500	20	0.2
Amazon Ratings	Full-graph	10^{-2}	10^{-3}	128	0			0
	Cluster-TNN	10^{-2}	10^{-4}	128	0.3	64	4	0
Cora Full	Full-graph	10^{-3}	10^{-5}	128	0.3			0
	Cluster-TNN	10^{-3}	10^{-5}	128	0.3	32	8	0
Coauthor Physics	Cluster-TNN	10^{-2}	10^{-4}	64	0.3	500	20	0.2
Reddit	Cluster-TNN	10^{-4}	10^{-4}	128	0.2	7 000	25	0.2
OGBN Products	Cluster-TNN	10^{-4}	10^{-4}	128	0.1	15 000	25	0.2
EDHNN								
Questions	Full-graph	10^{-3}	10^{-3}	128	0			0.5
	Cluster-TNN	10^{-2}	10^{-3}	32	0.2	500	50	0
Amazon Ratings	Full-graph	10^{-3}	10^{-3}	128	0.1			0
	Cluster-TNN	10^{-3}	10^{-4}	128	0.3	128	4	0
Cora Full	Full-graph	10^{-3}	10^{-3}	128	0.3			0
	Cluster-TNN	10^{-2}	10^{-4}	64	0.2	32	4	0.2
Coauthor Physics	Cluster-TNN	10^{-2}	0	128	0.2	2 000	30	0.5
Reddit	Cluster-TNN	10^{-2}	10^{-4}	128	0.3	7 000	20	0.2
OGBN Products	Cluster-TNN	10^{-3}	10^{-4}	128	0.1	20 000	25	0.2
UniGNN								
Questions	Full-graph	10^{-2}	0	32	0			
	Cluster-TNN	10^{-3}	10^{-3}	128	0	1 000	40	
Amazon Ratings	Full-graph	10^{-2}	10^{-3}	128	0			
	Cluster-TNN	10^{-2}	10^{-4}	64	0	32	8	
Cora Full	Full-graph	10^{-3}	10^{-4}	128	0.3			
	Cluster-TNN	10^{-4}	10^{-4}	128	0.1	32	4	
Coauthor Physics	Cluster-TNN	10^{-2}	10^{-5}	32	0.2	2 000	20	
Reddit	Cluster-TNN	10^{-3}	10^{-3}	64	0.3	7 000	25	
OGBN Products	Cluster-TNN	10^{-4}	10^{-4}	128	0	15 000	20	
CWN								
Questions	Full-graph	10^{-2}	10^{-3}	128	0.3			
	Cluster-TNN	10^{-2}	10^{-3}	32	0.3	1 000	40	
Amazon Ratings	Full-graph	10^{-2}	10^{-3}	128	0.3			
	Cluster-TNN	10^{-2}	10^{-3}	128	0.2	64	16	
Cora Full	Full-graph	10^{-2}	10^{-3}	64	0.3			
	Cluster-TNN	10^{-2}	10^{-3}	32	0.3	32	4	
Coauthor Physics	Cluster-TNN	10^{-2}	0	32	0.2	2 000	20	
Reddit	Cluster-TNN	10^{-2}	10^{-4}	64	0.3	7 000	20	
OGBN Products	Cluster-TNN	10^{-4}	10^{-4}	128	0	15 000	20	

Continued on next page

Table 13 continued

Dataset	Protocol	Learning Rate	Weight Decay	Hidden Channels	Projection Dropout	K	q	Dropout
TopoTune								
Questions	Full-graph	10^{-2}	10^{-5}	128	0.1			0
	Cluster-TNN	10^{-2}	10^{-3}	128	0.3	500	50	0.2
Amazon Ratings	Full-graph	10^{-2}	10^{-5}	64	0.1			0.2
	Cluster-TNN	10^{-2}	10^{-5}	128	0	32	8	0.5
Cora Full	Full-graph	10^{-2}	10^{-3}	128	0.2			0.2
	Cluster-TNN	10^{-2}	10^{-3}	128	0.2	64	8	0.5
Coauthor Physics	Cluster-TNN	10^{-3}	10^{-3}	32	0.3	2 000	20	0.5
Reddit	Cluster-TNN	10^{-3}	10^{-4}	64	0.2	7 000	20	0.2
OGBN Products	Cluster-TNN	10^{-3}	10^{-5}	64	0.2	15 000	25	0.5
SCN								
Questions	Full-graph	10^{-2}	10^{-4}	128	0.3			
	Cluster-TNN	10^{-3}	10^{-3}	64	0.1	500	50	
Amazon Ratings	Full-graph	10^{-2}	0	64	0.3			
	Cluster-TNN	10^{-3}	0	128	0.2	32	8	
Cora Full	Full-graph	10^{-3}	10^{-3}	128	0.3			
	Cluster-TNN	10^{-3}	10^{-4}	128	0.3	32	16	
Coauthor Physics	Cluster-TNN	10^{-2}	0	32	0.3	2 000	20	
Reddit	Cluster-TNN	10^{-3}	10^{-3}	64	0.3	7 000	25	
OGBN Products	Cluster-TNN	10^{-4}	10^{-4}	128	0	15 000	20	
SCCNN								
Questions	Full-graph	10^{-3}	10^{-5}	128	0.1			
	Cluster-TNN	10^{-3}	10^{-5}	128	0.3	500	30	
Amazon Ratings	Full-graph	10^{-3}	0	128	0.3			
	Cluster-TNN	10^{-3}	10^{-3}	128	0.2	32	4	
Cora Full	Full-graph	10^{-3}	10^{-3}	128	0.3			
	Cluster-TNN	10^{-3}	10^{-3}	128	0.1	32	8	
Coauthor Physics	Cluster-TNN	10^{-2}	0	128	0.3	2 000	20	
Reddit	Cluster-TNN	10^{-3}	10^{-3}	64	0.3	7 000	20	
OGBN Products	Cluster-TNN	10^{-4}	10^{-4}	128	0	15 000	20	