TopoSHAP: Shapley for Explainable and Efficient Topological Deep Learning

Anonymous authors
Paper under double-blind review

Abstract

Topological deep learning (TDL) extends learning on graphs, such as molecules or social networks, to higher-order structures, such as rings in a molecule or groups in a social network. Explaining which of these structures drive a model's predictions, and which architectural components matter most for its performance, remains challenging. We introduce TopoSHAP, an efficient and general explanation framework that addresses both questions across topological domains and model architectures. TopoSHAP-cell assigns Shapley values to topological data cells and extracts compact explanations. These explanations match or outperform graph-based counterparts on GraphXAI benchmark data while running up to 112× faster. They also reveal advantageous topological feature design, leading to measured improvements in explanation recovery and predictive accuracy. On the architecture side, TopoSHAP-neighbor explains how neighborhoods contribute to model performance, and turns these evaluations into a backward-elimination search that identifies smaller, competitive architectures early in training. This is up to 11.3× cheaper than traditional TDL training sweeps.

1 Introduction

Graph neural networks (GNNs) learn from relational data, such as molecules and social networks (Kipf and Welling, 2017). They achieve this by explicitly modeling pairwise interactions between the nodes (atoms, people) of the data. However, real-world systems often exhibit interactions that are inherently multiway. For instance, atoms can form rings which have critical functional roles; people can form groups whose nature goes beyond pairwise relationships. These higher-order interactions are not explicitly represented in standard GNNs. Topological neural networks (TNNs) address this limitation by explicitly modeling higher-order structure (Hajij et al., 2023; Papillon et al., 2023). In particular, topological deep learning (TDL) uses *combinatorial complexes* to organize interactions into *cells* of different *ranks*: atoms are rank-0 cells, bonds are rank-1 cells, and rings are rank-2 cells. At each layer, TNNs update each cell feature based on the features of its neighboring cells. On standardized topological benchmarks, TNNs often match or exceed GNN accuracy (Telyatnikov et al., 2025).

As Topological Deep Learning (TDL) moves towards domain-specific applications like chemistry and materials science (Papamarkou et al., 2024), practitioners face two black box challenges. First, they need to understand which structures drive these predictions. For a molecular model, for example, one may be compelled to reveal the chemical features and molecular structures the model deems relevant to the target property. Second, TDL practitioners do not know *a priori* what topological relations, or *neighborhoods*, matter for their task. Current TDL pipelines rely on an expensive hyperparameter search to identify the best architecture (Papillon et al., 2025; Bernárdez et al., 2025). This is not only expensive but also fails to provide practical insight as to why a specific architecture worked better.

Machine learning *explainers* are a promising avenue to opening these black boxes. They investigate model behavior by attributing predictions to parts of the input (Doshi-Velez and Kim, 2017; Ribeiro et al., 2016; Lundberg and Lee, 2017). However, existing post-hoc explainers are limited to graph (Sinha et al., 2025; Ying et al., 2019; Luo et al., 2020; Yuan et al., 2021; Muschalik et al., 2025) and hypergraph (Maleki et al., 2023; Su et al., 2024) neural networks. We propose the first explainer for general topological domains, encompassing both prediction and performance explanation.

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.

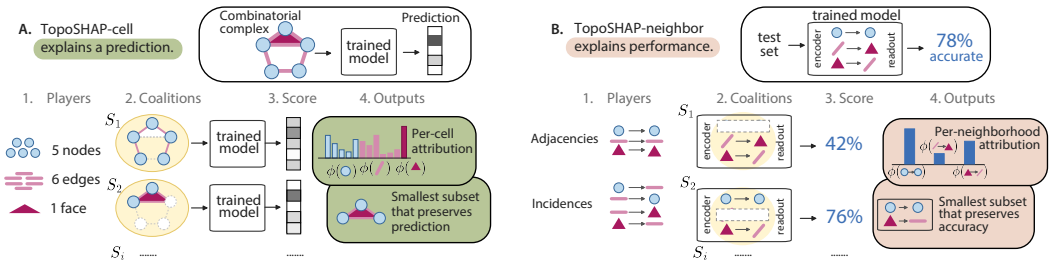

Figure 1: **TopoSHAP explains a Topological Neural Network's predictions and performance.** **A.** TopoSHAP-cell considers the cells of the input combinatorial complex as players (1). Given certain coalitions S_i of those players (2), it evaluates the model's predictions (3). Averaging marginal contributions over coalitions outputs one Shapley value per cell and a sub-complex best preserving the prediction (4). **B.** TopoSHAP-neighbor manipulates the neighborhoods that contribute to feature updates in the TNN (1). It briefly trains a model with all neighborhoods, then masks coalitions of them (2) to measure neighborhood impact on test performance (3). Averaging marginal contributions over coalitions outputs one attribution value per neighborhood and a subset of that preserves model performance (4).

Contributions. We introduce TopoSHAP, a unified game-theoretic framework for explaining TDL models, which attributes both predictions and performance to the cells and neighborhoods of a combinatorial complex. TopoSHAP is available at anonymous.4open.science/r/toposhap-3C75.

- **We explain predictions across ranks.** We formulate a cooperative game in which cells of every rank are players, producing per-cell Shapley attributions and compact explanations. Across both outputs, we match or outperform graph explanation methods on GraphXAI benchmarks. We also propose the first explanations of topological benchmark datasets going beyond hypergraphs. We call this component TopoSHAP-cell.
- **We use cell attributions to improve topological features.** Rank-specific attributions show that topological features, typically built by aggregating node features, retain information about lower-rank cells. Using this finding, we design new topological features and match or improve explanation recovery on GraphXAI benchmarks.
- **We explain performance and use it to select architectures.** We formulate a second game in which a model's neighborhood-specific modules are players and test performance is the payoff. We propose an inexpensive backward-elimination procedure that selects neighborhood-specific modules for full training. This reduces training compute by up to an order of magnitude relative to standard TDL training sweeps. We call this component TopoSHAP-neighbor.

2 Related Work

TopoSHAP builds on three lines of work: (i) topological deep learning, the subfield proposing the TNNs we explain; (ii) Shapley-value attribution in machine learning, the game-theoretic background we rely on; and (iii) GNN explainers, the methods we generalize. We review each below.

Topological deep learning (TDL). TDL proposes neural networks that learn from higher-order topological domains generalizing graphs, such as: simplicial complexes (Ebli et al., 2020), cellular complexes (Bodnar et al., 2021), hypergraphs (Feng et al., 2019) and combinatorial complexes (Hajij et al., 2023). TopoTune (Papillon et al., 2025) defines the architectural design space we study. Specifically, we consider Generalized Combinatorial Complex Neural Networks (GCCN) as they are general TNNs that are conveniently specified by the set of neighborhoods they each admit. We also consider the non-message passing version of GCCNs, introduced in HOPSE (Bernárdez et al., 2025), which leverage graph embeddings. Our framework treats HOPSE and message-passing TNNs identically because it only requires forward evaluations. Regarding implementation, TopoBench (Telyatnikov et al., 2025) standardizes liftings, models, and training across topological domains. We host our contribution and experiments there.

Shapley-value attribution in machine learning. The Shapley value credits each player in a cooperative game with its average marginal contribution across coalitions, and is the unique such credit assignment satisfying a standard set of fairness axioms (Shapley, 1953). SHAP brought this machinery to machine learning with input features as players (Lundberg and Lee, 2017), and interaction indices extend credit to coalitions (Grabisch and Roubens, 1999). In TopoSHAP-cell, our players are the cells (node, edge, hyperedge, face, and so on) that make up the topological domain. Because exact values cost exponential time in the number of players, larger complexes use unbiased permutation sampling (Castro et al., 2009; Štrumbelj and Kononenko, 2010). Shapley values are unsound for selecting players when two players can do the same job (Fryer et al., 2021). Our framework therefore uses the game only to order candidates, and compares the shortlisted candidates by validation of fully trained models (Section 4.2).

Explaining GNNs and Hypergraph Neural Networks. Existing GNN explainers broadly fall into two families. The first family of explainers learns continuous masks over the graph's nodes and edges. GNNExplainer learns a new mask for each instance, requiring a new optimization run per prediction (Ying et al., 2019). PGExplainer trains a single mask-generating model once over the dataset, which then generalizes to new instances without further optimization (Luo et al., 2020). Second, search-based explainers return discrete subgraphs scored by prediction preservation: SubgraphX explores connected subgraphs by Monte-Carlo tree search, scoring each with a sampled Shapley reward (Yuan et al., 2021), and Zorro greedily selects sparse masks with stability guarantees (Funke et al., 2023). GraphSVX (Duval and Malliaros, 2021) and GraphSHAP-IQ (Muschalik et al., 2025) bring Shapley values and exact any-order Shapley interactions to GNNs, with nodes and features as players, and Covert et al. (2021) unify such methods by their choice of removal operator, model behavior, and summary.TopoSHAP instantiates that framework with cells as players and detachment on the complex as the removal operator. Shapley values over internal model components also precede us in neuron attribution (Ghorbani and Zou, 2020) and structured pruning (Ancona et al., 2020). TopoSHAP-neighbor differs by scoring architectural neighborhoods with masked validation accuracy and by never selecting on the Shapley ranking itself (Section 4.2). All of the above operate on nodes, edges, features, or network units.

Beyond graphs, HyperEX explains hypergraph predictions through node–hyperedge associations (Maleki et al., 2023), and SHypX returns explanatory subhypergraphs (Su et al., 2024). Because hypergraphs are simply rank-≤ 1 combinatorial complexes, these approaches also do not handle cells across ranks. Counterfactual hypergraph explanations (Veglianti et al., 2026) pursue prediction-changing edits, complementary to our prediction-preserving objective. FORGE (Sinha et al., 2025) lifts graphs to cell complexes before explaining, yet still outputs node-level attributions. TopoSHAP instead defines games over cells across ranks of a combinatorial complex (of which graphs and hypergraphs are special cases) and over architectural neighborhoods. It is also the first to propose prediction attribution alongside validation-based architecture selection.

3 Preliminaries

TopoSHAP relies on three components: (i) *the data*, combinatorial complexes (Section 3.1), whose cells are the players of the cell game, (ii) *the models*, topological neural networks (TNNs) specified by a set of neighborhoods (Section 3.2), whose neighborhoods are the players of the neighbor game, and (iii) *the game-theoretic machinery*, Shapley values (Section 3.3).

3.1 Combinatorial Complexes and Their Cells

Combinatorial complexes are discrete topological domains that generalize graphs, as follows.

Definition 1 (Combinatorial complex and cells (Hajij et al., 2023))**.** A *combinatorial complex* is a triple $\mathcal{C} = (V, \mathcal{X}, \mathrm{k})$ where V is a finite set of nodes, $\mathcal{X} \subseteq 2^V \setminus \{\emptyset\}$ is a set of *cells*, and $\mathrm{k} : \mathcal{X} \to \mathbb{Z}_{\geq 0}$ is a rank function monotone with respect to inclusion: $x \subseteq y \implies \mathrm{k}(x) \leq \mathrm{k}(y)$.

Cells are the building blocks of the combinatorial complex (depicted in Fig. 1, top left). Cells of rank 0, 1 and 2 are called nodes (depicted in blue), edges (depicted in pink) and faces (depicted in burgundy), respectively. Cells of higher ranks represent higher multi-way interactions between nodes. Graphs and hypergraphs are special cases of combinatorial complexes, where all cells have rank ≤ 1.

Other higher order domains used in TDL are also special cases, such as simplicial complexes, whose rank-2 cells are triangles and whose cells are closed under taking subsets, and cellular complexes, whose faces may be arbitrary polygons. Combinatorial complexes generalize them all (Hajij et al., 2023), which is why we define TopoSHAP on them.

Raw data is often provided in graph format. A topological *lifting* converts a graph into a higher order domain by promoting substructures of nodes to higher-order cells. For example a cycle of nodes in a graph can be defined as a face in a cellular complex (Bodnar et al., 2021). Of the datasets we use, only MANTRA (Ballester et al., 2025) is natively defined on a higher order domain, namely the simplicial complex. We lift the other (graph) datasets we use to the cellular complex domain using liftings available in TopoBench (Telyatnikov et al., 2025). We will see that the choice of lifting impacts both accuracy and explanation quality (Section 5).

3.2 Topological Neural Networks and Their Neighborhoods

TNNs generalize GNNs to consider higher order domains. Here, we specifically consider the combinatorial complex for its generality, encompassing graph, hypergraphs, simplicial/cell complexes, and more. A TNN architecture is specified by a set of *neighborhoods*, the relationships through which cells interact on the combinatorial complex. We first define neighborhoods formally, then provide preliminaries for the two model families we use in experiments: GCCNs, which pass messages along the chosen neighborhoods, and HOPSE, which encodes them without message passing.

Definition 2 (Neighborhood (Hajij et al., 2023; Bernárdez et al., 2025))**.** A neighborhood on a combinatorial complex $\mathcal{C} = (V, \mathcal{X}, \mathrm{k})$ is a function $N : \mathcal{X} \to 2^{\mathcal{X}}$ assigning to each cell x a set $N(x)$ of cells, its neighbors. The standard neighborhoods are indexed by ranks. For a cell x of rank t and a rank $s \neq t$, the *incidence*

$$I_{s\to t}(x) \;=\; \{\, y \in \mathcal{X} \mid \mathrm{k}(y) = s,\; x \subsetneq y \text{ or } y \subsetneq x \,\}$$

collects the rank-s cells that contain x (when $s > t$) or are contained in x (when $s < t$), and the *adjacency*

$$A_{t,s}(x) \;=\; \{\, y \in \mathcal{X} \setminus \{x\} \mid \mathrm{k}(y) = t,\; I_{s\to t}(x) \cap I_{s\to t}(y) \neq \emptyset \,\}$$

collects the rank-t cells that share a rank-s incident cell with x.

Intuitively, neighborhoods are the routes along which cells exchange information. Incidences connect cells of different ranks, e.g., $I_{1\to 0}$ links each node to the edges containing it; adjacencies connect cells of the same rank through a shared cell, e.g., $A_{0,1}$ links nodes joined by an edge. In practice each neighborhood is stored as a sparse matrix over cells.

Given a set $\mathcal{N} = \{N_1, \ldots, N_m\}$ of neighborhoods, each $N \in \mathcal{N}$ decomposes $\mathcal{C}$ into an ordinary graph on cells, where cells participating in the neighborhood are nodes and the relationships of the neighborhood are edges (Papillon et al., 2025). With this formulation, the combinatorial complex $\mathcal{C}$ can be represented as a collection of smaller graphs, one per neighborhood. Both TNN families we use are built directly on this collection. Because both models represent each neighborhood as an architectural component, removing a neighborhood from $\mathcal{N}$is a well defined operation on the model.

Definition 3 (Generalized combinatorial complex network (GCCN) (Papillon et al., 2025))**.** A GCCN stacks L message-passing layers over the strictly augmented Hasse graphs $\{\mathcal{G}_N\}_{N\in\mathcal{N}}$. At layer ℓ, every cell x exchanges a message with its neighbors along each graph $\mathcal{G}_N$, using a GNN layer ψ_N dedicated to that neighborhood; the messages arriving from a single neighborhood are pooled together, and the results are then pooled once more across neighborhoods to update x's feature:

$$\mathbf{h}_x^{(\ell+1)} \;=\; \phi\Big(\mathbf{h}_x^{(\ell)},\; \bigotimes_{N\in\mathcal{N}} \bigoplus_{y\in N(x)} \psi_N\big(\mathbf{h}_x^{(\ell)}, \mathbf{h}_y^{(\ell)}\big)\Big). \tag{1}$$

Definition 4 (Higher-Order Positional and Structural Encoder (HOPSE) (Bernárdez et al., 2025))**.** HOPSE replaces every GNN layer ψ_N with a fixed positional or structural encoder, such as Laplacian eigenvectors, computed once on the strictly augmented Hasse graph $\mathcal{G}_N$ rather than learned during training. For every cell x, HOPSE collects the encodings x receives this way from each of its Hasse graphs, concatenates them with x's own input feature, and passes the result through a small learnable network, shared across cells of the same rank, to produce x's final feature.

3.3 Measuring a Contribution to a Game: Shapley Values and Interactions

A *cooperative game* (P, v) consists of players P, $|P| = n$, and a value function $v : 2^P \to \mathbb{R}$ with $v(\emptyset) = 0$, where $v(S)$ is the gain that the *coalition* $S \subseteq P$ secures on its own. The question is how to divide the total gain $v(P)$ among the n players.

The *Shapley value* (Shapley, 1953) of player i averages its marginal contribution over all coalitions:

$$\phi_i(v) \;=\; \sum_{S \subseteq P \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \big[v(S \cup \{i\}) - v(S)\big]. \tag{2}$$

It is the unique attribution satisfying four desirable axioms: (i) *efficiency*, i.e., attributions credit the total gain exactly, $\sum_i \phi_i = v(P)$, (ii) *symmetry* where $\phi_i = \phi_j$ whenever $v(S \cup \{i\}) = v(S \cup \{j\})$ for all S, (iii) *null player* where $\phi_i = 0$ whenever $v(S \cup \{i\}) = v(S)$ for all S, and (iv) *linearity* which makes credit additive across games, $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$. These axioms explain the adoption of Shapley values in explainable machine learning. For this reason, this is the attribution that we will use in TopoSHAP.

4 Method: TopoSHAP

TopoSHAP asks two questions about a trained TNN. Which cells of an input drove its prediction? Which neighborhoods of the architecture drove its performance? Each question can be answered with a cooperative game (Section 3.3), which we call TopoSHAP-cell and TopoSHAP-neighbor, respectively. Figure 1 compares the two games.

4.1 TopoSHAP-cell: Explaining Predictions.

The cell game. The players P are the cells of the input complex, at every rank; $n = |P|$. A coalition $S \subseteq P$ specifies a perturbed input $\mathcal{C}[S]$ in which every cell outside S is *detached*. Specifically, its feature row is zeroed, neighborhoods are rebuilt without it, and it is excluded from readout pooling. Retained cells keep the input features they were supplied with; cells are independent players, so a face remains when one of its vertices is detached, losing only its incidence to that node. For HOPSE, whose inputs include precomputed encodings of the complex, the encodings are recomputed on $\mathcal{C}[S]$.

The value of a coalition is $v(S) = f(\mathcal{C}[S]) - f(\mathcal{C}[\emptyset])$, where $f(\mathcal{C}[S])$ is the model's prediction given the coalition, and $f(\mathcal{C}[\emptyset])$ is the model's prediction given an empty complex, leaving the encoder and readout biases. At $S = P$ the rebuilt input reproduces the model's stored forward pass. Section A.1 gives the full algorithm.

Output 1: cell attributions. The Shapley value $\phi_i(v)$ of Equation (2) credits each player with its average marginal contribution to the coalition value $v(S)$. The Shapley axioms ensure efficiency, meaning the Shapley values sum to $v(P)$, as well as null-player behavior, meaning a player *j* the model never uses gets $\phi_j(v) = 0$. To compute exact Shapley values, Equation (2) requires every possible coalition. While this is feasible for small numbers of players (such as in TopoSHAP-neighbor where $n = 11$ so $2^n = 2048$ coalitions), it is prohibitively expensive for many players (such as in TopoSHAP-cell for large complexes). In that case, we follow Castro et al. (2009) and draw T random orderings of the players, add them one at a time to a coalition, and average the marginal changes. This unbiased estimate costs $O(Tn)$ model evaluations.

Output 2: explanation via greedy pruning. An explanation is a small set of players that together preserve the model's prediction; we approximate it with one greedy candidate per size. Starting from all players P, we repeatedly eliminate the player whose removal lowers v the least. Because the steps are nested, one pass returns a candidate coalition $\hat{S}_b$ for every size b, at most $(n^2 + n)/2$ evaluations in total. Each $\hat{S}_b$ is a compact, high-scoring coalition on the greedy path. Greedy elimination carries no optimality guarantee, so $\hat{S}_b$ need not be the best size-b coalition.

Axioms as unit tests. For an exact game the Shapley axioms hold with equality; our implementation matches brute force to 2.254×10^{-14}, and separate consistency checks test the masking itself (Section A.1).

4.2 TopoSHAP-neighbor: Explaining Performance

TopoSHAP-neighbor evaluates how neighborhoods contribute to a trained model's validation accuracy. These evaluations support both neighborhood attribution and architecture selection. The players are the candidate neighborhoods $\mathcal{N}$, with $n = |\mathcal{N}|$. Our main rank-2 architecture-selection experiments consider eleven neighborhoods. Here, a coalition $S \subseteq \mathcal{N}$ specifies a model where the neighborhood-specific modules (message-passing modules in GCCN or encoding MLP blocks in HOPSE) are constrained to neighborhoods in S. The encoder and readout backbones are left untouched.

Evaluating coalitions. In practice, training a separate model for every coalition requires 2^n training runs. With the goal of TopoSHAP-neighbor being practical in applied TDL settings, we propose a cheaper alternative. Specifically, we briefly train a full version of the model containing all neighborhoods, and then evaluate coalitions on a frozen version of it. For each S, we mask any neighborhood-specific modules outside S and measure the resulting validation accuracy, $a(S)$. We define the value function $v(S) = a(S) - a(\emptyset)$ where $a(\emptyset)$ is the accuracy of the same model with all neighborhood components masked, leaving its encoder and readout active. Each coalition therefore requires one validation pass. This brief initial training follows prior work on identifying ideal architectures from early epochs (You et al., 2020; Frankle et al., 2020) and low-fidelity candidate ranking in architecture and hyperparameter search (Li et al., 2018; Mellor et al., 2021).

Output 1: neighborhood attributions. The Shapley value of a neighborhood measures its average marginal contribution to $v(\mathcal{N})$. We compute these values exactly by evaluating all 2^n coalitions $S \subseteq \mathcal{N}$. We note that these attributions describe neighborhood contributions in one frozen model, meaning they do not establish which neighborhoods a separately trained model would need. For example, a model may learn to rely on one of two neighborhoods carrying equally useful information.

Output 2: architecture selection. We use the coalition evaluations to select architectures through backward elimination, following backward stepwise selection and recursive feature elimination (Guyon et al., 2002; Hastie et al., 2009). Starting from $S = \mathcal{N}$, we evaluate every possible single-neighborhood removal and retain the coalition with the highest validation accuracy. We repeat this process down to one neighborhood, obtaining one candidate per size. Each step reassesses removal effects given the neighborhoods that remain (Algorithm 2).

From these n candidates, we choose B to train further, where $1 \leq B \leq n$. We first select the *stop candidate*: the architecture immediately before the first removal whose accuracy loss exceeds

$$\tau = \sqrt{\frac{p(1-p)}{m}}, \qquad p = a(\mathcal{N}),$$

where m is the number of validation examples and p is the briefly trained full model's validation accuracy. This estimated binomial standard error provides a data-dependent pruning threshold. If no removal exceeds it, the stop candidate is the last remaining neighborhood. The threshold marks a candidate without halting elimination, and does not bound the cumulative accuracy loss relative to the full model.

We select the remaining $B - 1$ architectures by selecting the remaining candidates with the highest validation accuracy. Each of the B selected architectures is initialized using the full frozen model and then trained to completion. Of the B trained models, we select the one with the highest final validation accuracy. This pipeline provides an inexpensive alternative to proposing architectures based on early neighborhood contributions. Algorithm 2 gives the complete procedure.

5 Results I: Explaining Predictions with TopoSHAP-cell

This section asks which cells a trained TNN relies on, considering benchmark explanation graph datasets, lifted topological versions of those datasets, as well as unlabeled standard topological datasets.

Setup. We use four GraphXAI molecular benchmark data with ground truth explanations from Agarwal et al. (2023), four lifted topological benchmark data from TopoBench (Telyatnikov et al., 2025), and the MANTRA dataset (Ballester et al., 2025), a native simplicial complex dataset. All

Table 1: **Explanation accuracy on graphs.** GEA on parameter-matched GNN models; per method the better of GCN/GIN is shown (both in Section C.1). Last column: measured seconds per instance; †plus one-time training (142–1501 s per dataset). Bold is best and gray is within $\leq 1\sigma$.

	Benzene	FluorideCarbonyl	Mutagenicity	AlkaneCarbonyl	s / inst.
TopoSHAP (attribution)	0.859 ± 0.009	0.507 ± 0.047	$\mathbf{0.798 \pm 0.124}$	$\mathbf{0.500 \pm 0.081}$	3.3–4.0
TopoSHAP (explanation)	$\mathbf{0.969 \pm 0.009}$	0.340 ± 0.010	0.480 ± 0.135	0.354 ± 0.125	0.33–0.67
SubgraphX	0.796 ± 0.029	0.439 ± 0.107	0.344 ± 0.036	0.296 ± 0.010	21–52
PGExplainer	0.642 ± 0.044	$\mathbf{0.555 \pm 0.067}$	0.290 ± 0.102	0.205 ± 0.160	$<0.01^{\dagger}$
GNNExplainer	0.443 ± 0.054	0.210 ± 0.014	0.158 ± 0.023	0.162 ± 0.049	0.80–0.82
Random	0.250 ± 0.000	0.219 ± 0.000	0.132 ± 0.000	0.173 ± 0.000	<0.01

comparisons use identical splits and three seeds (mean $\pm$ sd). Explanation accuracy is Graph Explanation Accuracy (GEA), the Jaccard overlap between a size-b explanation and a same-size ground-truth motif (Agarwal et al., 2023). In cases where no ground truth is available, we measure explanation faithfulness using deletion-Area-Under-the-Curve (deletion-AUC,). To compute it, cells are detached cumulatively from most to least important and the predicted-class probability is integrated over the removal process. A faithful ranking collapses the prediction quickly, so lower is better (formula in App. A.1). Lastly, the *alignment gap* $\Delta = v(\hat{S}_b) - v(S_{\mathrm{gt}})$ compares the greedy candidate $\hat{S}_b$ to the annotated ground truth motif S_{gt}. A positive Δ occurs when a same-size coalition scored above the annotated motif, indicating that the annotation is in fact not the game's optimum on that instance. Datasets and training details are in Appendix B; evaluation settings in Appendix C.

5.1 TopoSHAP-cell on Graphs Matches GNN Explainers for Cheap

TopoSHAP-cell matches within 1σ or outperforms every baseline. In Table 1, all methods explain the same parameter-matched models of two types: GCN (Kipf and Welling, 2017) and GIN (Xu et al., 2019). The table shows each method's better model-type. Per-model-type results are in Table 4. We evaluate both outputs by their overlap with the annotated motif (the ground truth): attribution selects the b highest-scoring nodes, while greedy pruning retains a size-b set through successive removals. Greedy pruning achieves higher motif overlap on Benzene with GIN; attribution-based selection performs better on Mutagenicity and Alkane-Carbonyl for both model types.

TopoSHAP's explanation output is 64–112$\times$ cheaper per instance than search-based explanation. The explanation output takes 0.33–0.67 s per instance and the attribution output 3.3–4.0 s, against 21–52 s for its graph counterpart SubgraphX (Table 1). PGExplainer has a costly training setup: for computing explanations it exceeds TopoSHAP's cost on every dataset (Table 5).

5.2 TopoSHAP-cell on Higher-Order Data Recovers Explanations

We consider the rank-2 GCCN containing all eleven possible neighborhoods. We lift the GraphXAI datasets conventionally by converting chordless cycles of up to 10 nodes to a face (Appendix B.1).

Standard lifting preserves aggregated node information. The standard lifting assigns face features using the sum of their incident nodes' features. In TopoSHAP-cell, because retained cells keep their original features during masking, this summary information remains available regardless of node or edge masking. We find that selected coalitions sometimes retain faces while discarding edges (Benzene, App. C.2), suggesting some models use these summaries. The selected coalitions also outscore the annotated motifs on most evaluated instances ($\Delta > 0$; Table 2), showing that motif recovery and the model's preferences under masking can diverge.

Constant higher-rank features improve motif recovery on three of four datasets. Building on our finding of summary information caching structural information, we devise a new lifting where every higher-rank feature is instead the same constant all-ones vector. Under this "structure-only" lifting, greedy explanations achieve higher mean GEA on three datasets and attribution results are mixed (Table 2). Mean predictive accuracy also increases on three datasets. We consider models trained on Benzene as a specific example. Moving from standard to structure-only lifting increases face retention from 0.84 ± 0.14 to 0.95 ± 0.05 and boundary-edge recovery from 0.40 ± 0.14 to

Table 2: **Explanation accuracy on lifted complexes.** Node-projected GEA for both TopoSHAP outputs of the GCCN with all eleven neighborhoods under the two liftings; the lower block gives each model's test accuracy and its share of candidates with $\Delta > 0$. Bold is best and gray is within $\leq 1\sigma$.

	Benzene	FluorideCarbonyl	Mutagenicity	AlkaneCarbonyl
attribution, structural lifting	0.471 ± 0.018	0.491 ± 0.035	0.326 ± 0.065	0.333 ± 0.097
attribution, standard lifting	0.459 ± 0.192	$\mathbf{0.516 \pm 0.089}$	0.257 ± 0.077	0.348 ± 0.043
explanation, structural lifting	$\mathbf{0.725 \pm 0.105}$	0.426 ± 0.027	$\mathbf{0.436 \pm 0.081}$	0.362 ± 0.130
explanation, standard lifting	0.509 ± 0.188	0.372 ± 0.004	0.289 ± 0.117	$\mathbf{0.364 \pm 0.053}$
model accuracy, structural lifting	0.942 ± 0.005	0.947 ± 0.003	0.976 ± 0.002	0.967 ± 0.006
model accuracy, standard lifting	0.928 ± 0.003	0.936 ± 0.003	0.969 ± 0.009	0.971 ± 0.005
share with $\Delta > 0$, structural lifting	0.74	0.93	0.97	0.99
share with $\Delta > 0$, standard lifting	0.97	0.94	1.00	0.99

0.86 ± 0.06 (App. C.2). Thus, explanations more often retain both the face and its boundary. More broadly, these findings support the importance of feature lifting for explanation recovery.

5.3 TopoSHAP-cell Is the Most Faithful Explainer Without Annotations

TopoSHAP-cell beats occlusion, gradient-times-input, and random on every unannotated dataset. Here we apply TopoSHAP-cell to benchmark topological datasets hosted in TopoBench (Telyatnikov et al., 2025). Because there is no ground truth annotation, we score explanations against the model's own prediction by deletion-AUC (Table 7; protocol in Section C.3). We hypothesize that TopoSHAP outperforms other baselines by attributing cells across coalitions rather than in isolation.

Rank-specific attributions help explain models' prediction strategies. Here, we consider MANTRA (Ballester et al., 2025), the natively topological dataset, and its orientability classification task. HOPSE performs best overall (AUROC 0.93), followed by GCCN (0.79) and the published TopoBench baseline (0.52). However, when we restrict evaluation to surfaces with matched cell counts, GCCN falls to near chance (0.51), while HOPSE retains 0.79. Thus, HOPSE can distinguish orientability using information beyond the numbers of nodes, edges, and triangles (protocol in Section C.4). Cell attributions help interpret this difference. Consider an orientable torus and a non-orientable Klein bottle with identical cell counts (Figure 3). Attributions show GCCN leverages triangles, but fails to distinguish triangles of orientable and the non-orientable surfaces. HOPSE instead concentrates its credit on nodes and edges, but successfully assigns attributions of opposite sign. We hypothesize that HOPSE succeeds because its spectral encodings provide global information about the complex's topology. Giving GCCN access to these encodings supports the hypothesis: with Hodge Laplacian eigenvectors concatenated to its input features, GCCN reaches a balanced accuracy of 0.88, exceeding HOPSE's 0.79 (Table 8). This is an example of how TopoSHAP-cell can shed light on models' inner workings and help design better TNNs for specific tasks.

6 Results II: Explaining Performance with TopoSHAP-neighbor

6.1 Architecture Selection in Practice

TopoSHAP-neighbor reduces total training compute by a factor of 4.5–11.3 relative to the evaluated architecture sweeps. The resulting accuracy–compute trade-off varies by dataset, as shown in Figure 2 and Table 9. We evaluate GCCN and HOPSE across 11 datasets against their respective reported architecture search grids (Papillon et al., 2025; Bernárdez et al., 2025). Our pipeline uses a five-epoch full model to select $B = 2$ smaller architectures, completes the training of these smaller architectures, and selects one by validation accuracy. Reported costs include search, candidate training, and preprocessing (Section D.1). Controls using the same initialization and continuation settings compare elimination with a random two-candidate shortlist and the full-model default (Table 10). We find that elimination generally improves mean accuracy.

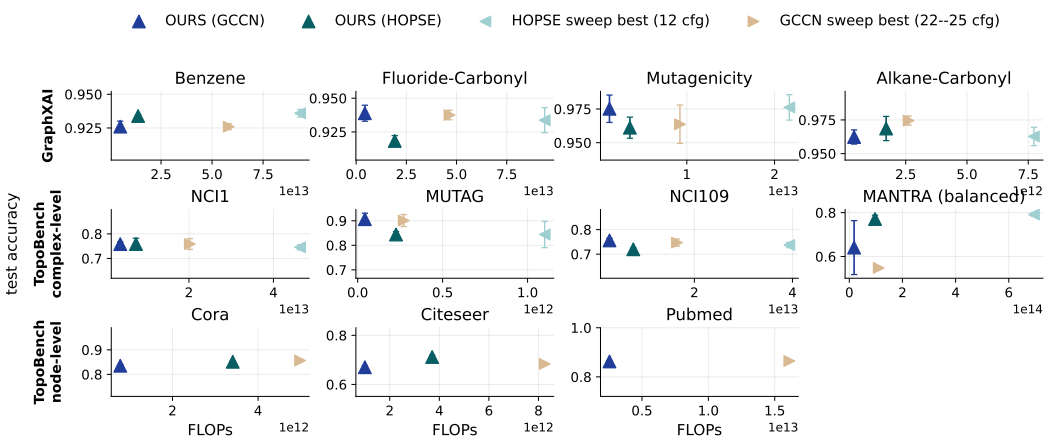

Figure 2: **Architecture selection: accuracy versus total compute.** Dark markers denote TopoSHAP-neighbor; pale markers denote each family's validation-selected sweep winner, charged for the full sweep. Error bars show standard deviation over five seeds for GCCN $B = 2$ and three otherwise. Due to its practical costs, HOPSE sweeps are limited to complex-level datasets and we avoid training any HOPSE models on Pubmed. MANTRA uses balanced test accuracy.

6.2 Controlled Evaluation with Exhaustive Training

We separately train all $2^6 = 64$ subsets of six neighborhoods across twelve GCCN dataset/hyperparameter settings and three seeds (Section D.3). This exhaustive retraining allows us to compute the neighborhoods' Shapley values from separately trained coalitions and compare them with attributions from the masked five-epoch model; the selection pipeline of Section 6.1 itself never consumes attributions, scoring candidate removals directly by masked coalition accuracy.

Masked attributions describe reliance in the trained model. Masked-model and retrained-model Shapley values show weak pooled agreement ($\rho = 0.23$), rising to 0.32 when masking is scored by accuracy rather than logits (Table 12). Masking uses validation data, whereas retraining is scored by test accuracy or negative MAE, so this comparison also reflects differences in data and scoring (Section D.3). The results caution against interpreting fixed-model attributions as necessity after retraining.

Backward elimination generates competitive candidates in an oracle analysis. We score removals using the separately trained coalitions' seed-averaged test performance. On five held-out settings across four datasets, the cheap greedy architecture selection (outlined in Section 4.2) contains the exhaustive benchmark's best architecture in four settings (Table 13). This is encouraging evidence that our pipeline can construct competitive candidate sequences (Section D.4).

7 Limitations and Conclusion

TopoSHAP brings prediction and architecture explanations into a unified explainability framework for Topological Deep Learning, providing insight into model predictions and performance. We conclude by describing two limitations which are also promising avenues for future work. First, scaling to larger complexes requires more efficient coalition evaluation. Although greedy pruning is substantially cheaper than SubgraphX on our benchmarks (Table 1), it requires $O(n^2)$ model evaluations for n cells, while attribution requires $O(Tn)$ evaluations for T sampling passes. Truncated or sampled search offers a path toward complexes with thousands of cells. Second, our evaluation against ground truth explanations remains graph-based in this work. Developing explanation-annotated benchmarks on general combinatorial complexes, building on graph benchmarks and emerging synthetic hypergraph benchmarks (Su et al., 2024), will be paramount for developing better explanations for, and thus more trustworthy, Topological Deep Leaning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing and debugging the proposed methods, running experiments, and drafting initial versions of portions of the manuscript. We also used generative AI tools to assist with summarizing existing literature; all cited literature and associated claims were independently reviewed by the authors. AI tools were not used to conceptualize the project, develop its core research ideas or methodology or interpret the experimental results. All datasets used in this work were publicly available prior to this study and were not generated using AI. AI-assisted code was reviewed, modified, and tested by the authors, and all experimental results and analyses were independently verified by the authors. The authors wrote a substantial portion of the main text and heavily revised and edited all AI-assisted text. Appendix text a more equal combination of author and AI written text. We take full responsibility for the final content of the paper, including all claims, analyses, code, and artifacts.

REPRODUCIBILITY STATEMENT

Datasets, splits, and all model, training, and explainer settings are specified in Section B; evaluation protocols and full results in Sections C and D; the software environment in Section E. The implementation is made available at anonymous.4open.science/r/toposhap-3C75.

REFERENCES

Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. Evaluating explainability for graph neural networks. *Scientific Data*, 10(144), 2023.

Marco Ancona, Cengiz Öztireli, and Markus Gross. Shapley value as principled metric for structured network pruning. *arXiv preprint arXiv:2006.01795*, 2020.

Rubén Ballester, Ernst Röell, Daniel Bin Schmid, Mathieu Alain, Sergio Escalera, Carles Casacuberta, and Bastian Rieck. MANTRA: The manifold triangulations assemblage. In *International Conference on Learning Representations (ICLR)*, 2025.

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. *arXiv preprint arXiv:2505.15405*, 2025.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2625–2640. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/157792e4abb490f99dbd738483e0d2d4-Paper.pdf.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2008.04.004. URL https://www.sciencedirect.com/science/article/pii/S0305054808000804. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Alexandre Duval and Fragkiskos D Malliaros. Graphsvx: Shapley value explanations for graph neural networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 302–318. Springer, 2021.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. In *Topological Data Analysis and Beyond workshop at NeurIPS*, 2020. URL https://arxiv.org/abs/2010.03633.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 3558–3565. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013558. URL https://ojs.aaai.org/index.php/AAAI/article/view/4235.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations (ICLR)*, 2020.

Daniel Fryer, Inga Strumke, and Hien Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. *IEEE Access*, 9:144352–144360, 01 2021. doi: 10.1109/ACCESS.2021.3119110.

Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. Zorro: Valid, sparse, and stable explanations in graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8687–8698, 2023.

Amirata Ghorbani and James Zou. Neuron shapley: Discovering the responsible neurons. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5922–5932. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/41c542dfe6e4fc3deb251d64cf6ed2e4-Paper.pdf.

Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of Game Theory*, 28(4):547–565, 1999.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

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. *arXiv preprint arXiv:2206.00606*, 2023.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

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.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Sepideh Maleki, Ehsan Hajiramezanali, Gabriele Scalia, Tommaso Biancalani, and Kangway V. Chuang. Learning to explain hypergraph neural networks. In *2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, 2023. URL https://openreview.net/forum?id=B6YeDatcFw.

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020. arXiv:2007.08663.

Maximilian Muschalik, Fabian Fumagalli, Paolo Frazzetto, Janine Strotherm, Luca Hermes, Alessandro Sperduti, Eyke Hüllermeier, and Barbara Hammer. Exact computation of any-order Shapley interactions for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2025.

Theodore Papamarkou, Tolga Birdal, Michael M. 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 (ICML)*, 2024.

Mathilde Papillon, Sophia Sanborn, Mustafa Hajij, and Nina Miolane. Architectures of topological deep learning: A survey of message-passing topological neural networks. *arXiv preprint arXiv:2304.10031*, 2023.

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 (ICML)*, 2025.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Benjamin Sanchez-Lengeling, Jennifer Wei, Brian Lee, Emily Reif, Peter Wang, Wesley Wei Qian, Kevin McCloskey, Lucy Colwell, and Alexander Wiltschko. Evaluating attribution for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 5898–5910, 2020.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, Princeton, NJ, 1953.

Akshit Sinha, Sreeram Vennam, Charu Sharma, and Ponnurangam Kumaraguru. Higher order structures for graph explanations. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i19.34260. URL https://doi.org/10.1609/aaai.v39i19.34260.

Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 2010.

Shiye Su, Iulia Duta, Lucie Charlotte Magister, and Pietro Liò. Explaining hypergraph neural networks: From local explanations to global concepts. *arXiv preprint arXiv:2410.07764*, 2024.

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*, 2025. URL https://openreview.net/forum?id=07sTzyEVtY.

Fabiano Veglianti, Lorenzo Antonelli, and Gabriele Tolomei. Counterfactual explanations for hypergraph neural networks. *arXiv preprint arXiv:2602.04360*, 2026.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 40–48, 2016.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Yingyan Lin, Zhangyang Wang, and Richard G. Baraniuk. Drawing early-bird tickets: Towards more efficient training of deep networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 12241–12252, 2021.

A Additional Method Details

A.1 Cell Masking and Explanation Algorithms

Masked input. Given a complex $\mathcal{C}$ with cells P and a coalition $S \subseteq P$, the masked input $\mathcal{C}[S]$ is built by detaching every cell outside S: its feature row is zeroed, its entries are removed from every neighborhood matrix in which it appears (as source, target, or intermediate cell, with composed neighborhoods rebuilt from the incidences of the remaining cells, using the same constructors that built the original input), and it is excluded from readout pooling. Retained cells keep the input features they were supplied with; because cells are independent players, a face can remain after one of its nodes is detached, so $\mathcal{C}[S]$ is a masked complex and not in general a node-induced subcomplex. For HOPSE, whose inputs include precomputed positional and structural encodings, the encodings are recomputed on each $\mathcal{C}[S]$. On plain graphs we take the nodes as players and let incident edges follow their endpoints, in which case $\mathcal{C}[S]$ is the node-induced subgraph.

Target score. f reads one fixed class logit, held fixed across all coalitions of an instance: the annotated class on labeled benchmarks, the predicted class on datasets without annotations (no labels consulted), and the orientable-class logit margin on MANTRA. Coalition values are $v(S) = f(\mathcal{C}[S]) - f(\mathcal{C}[\emptyset])$, where $\mathcal{C}[\emptyset]$ detaches every cell, leaving the encoder and readout biases.

Attribution (Output 1). Exact Shapley values enumerate all 2^n coalitions when n is small. Otherwise we sample player orderings, add players in the sampled order, and average the marginal changes (Castro et al., 2009). The budget T counts every evaluated ordering: orderings are drawn uniformly at random and each drawn ordering is followed by its reverse, so T orderings comprise $T/2$ random draws and their reversals. A reversed ordering visits its own coalitions; all coalition values are cached by bitmask, so the total cost is at most $Tn + 1$ model evaluations and fewer in practice. Explanation experiments use $T = 128$ with estimator seed 0, distinct from the training seeds; the per-cell estimates of Figure 3 use $T = 512$.

Explanation (Output 2). Greedy backward pruning (Algorithm 1) starts from P and repeatedly removes the cell whose removal lowers v the least, producing one nested candidate $\hat{S}_b$ per size b in at most $(n^2 + n)/2$ evaluations. The candidates are compact, high-scoring coalitions on the greedy path and carry no optimality guarantee.

Algorithm 1 TopoSHAP-cell on one instance

Require: complex $\mathcal{C}$ with cells P, $n = |P|$; model logit f; sampling budget T
1: define $v(S) = f(\mathcal{C}[S]) - f(\mathcal{C}[\emptyset])$, cached by coalition
2: **Output 1:** $\phi \leftarrow$ exact Shapley values if 2^n is affordable, else the T-ordering estimate (each drawn ordering followed by its reverse)
3: **Output 2:** $S \leftarrow P$; $\hat{S}_n \leftarrow P$
4: **for** $b = n - 1, \ldots, 1$ **do**
5: $\quad i^* \leftarrow \arg\max_{i \in S} v(S \setminus \{i\})$, ties toward the lower cell index
6: $\quad S \leftarrow S \setminus \{i^*\}$; $\hat{S}_b \leftarrow S$
7: **end for**
8: **return** ϕ and $\{\hat{S}_b\}_{b=1}^{n}$

Implementation checks. At $S = P$ the rebuilt input must reproduce the model's stored forward pass exactly; this check runs at construction, and the HOPSE variant additionally checks that recomputed full-coalition encodings reproduce the stored prediction within tolerance. The Shapley computation is verified against brute-force enumeration to 2.254×10^{-14}on small games. These checks verify the implementation of the specified game; separate consistency checks (attribution invariance under batch permutation, equality on provably symmetric cells) test the masking itself.

A.2 Neighborhood Masking and Architecture Selection

The eleven neighborhoods. Every subset mask indexes neighborhoods in the order listed below, with the code label followed by the notation of Definition 2 ($I_{s \to t}$: messages from rank-s cells to their incident rank-t cells): up_adjacency-0 $= A_{0,1}$, up_incidence-0 $= I_{0 \to 1}$, down_incidence-1 $= I_{1 \to 0}$,

Algorithm 2 TopoSHAP-neighbor: architecture selection

Require: Full model with neighborhoods $\mathcal{N}$; training and validation sets $D_{\text{train}}, D_{\text{val}}$; candidate budget $1 \le B \le |\mathcal{N}|$
Ensure: Selected trained model
1: Train the full model for five epochs, obtaining weights θ_0
2: Define $a(S)$ as its validation accuracy with neighborhoods outside S masked and weights fixed at θ_0
3: Cache all evaluations of $a(S)$
4: $m \leftarrow |D_{\text{val}}|$, $p \leftarrow a(\mathcal{N})$, $\tau \leftarrow \sqrt{p(1-p)/m}$
5: $S \leftarrow \mathcal{N}$, $\mathcal{K} \leftarrow \{\mathcal{N}\}$, $S_{\text{stop}} \leftarrow \varnothing$
6: **for** $k = |\mathcal{N}|, \dots, 2$ **do**
7: $\quad i^* \leftarrow \arg\max_{i \in S} a(S \setminus \{i\})$
8: $\quad S' \leftarrow S \setminus \{i^*\}$
9: $\quad$ **if** $S_{\text{stop}} = \varnothing$ **and** $a(S) - a(S') > \tau$ **then**
10: $\quad\quad S_{\text{stop}} \leftarrow S$ ▷ Mark the candidate before this removal
11: $\quad$ **end if**
12: $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{S'\}; \quad S \leftarrow S'$ ▷ Continue even after marking a candidate
13: **end for**
14: **if** $S_{\text{stop}} = \varnothing$ **then**
15: $\quad S_{\text{stop}} \leftarrow S$ ▷ Use the singleton if no loss exceeded τ
16: **end if**
17: $\mathcal{T} \leftarrow \{S_{\text{stop}}\} \cup \{B-1$ candidates in $\mathcal{K} \setminus \{S_{\text{stop}}\}$ with highest $a(S)\}$
18: **for** each $T' \in \mathcal{T}$ **do**
19: $\quad$ Initialize $f_{T'}$ from a separate copy of θ_0, removing components outside T'
20: $\quad$ Continue training $f_{T'}$ to completion under standard settings
21: $\quad q(T') \leftarrow \text{Accuracy}(f_{T'}, D_{\text{val}})$
22: **end for**
23: **return** f_{T^*}, $T^* = \arg\max_{T' \in \mathcal{T}} q(T')$ (ties toward S_{stop})

up_adjacency-1 $= A_{1,2}$, down_incidence-2 $= I_{2\to1}$, 2-up_adjacency-0 $= A_{0,2}$, down_adjacency-1 $= A_{1,0}$, up_incidence-1 $= I_{1\to2}$, down_adjacency-2 $= A_{2,1}$, 2-up_incidence-0 $= I_{0\to2}$, 2-down_incidence-2 $= I_{2\to0}$. The exhaustive benchmark of Section D.3 uses the first six.

Masking a neighborhood. In a GCCN, each neighborhood owns one message-passing module per layer; in HOPSE, one encoding block (the fixed encoder on that neighborhood's graph plus its learned MLP). Masking a neighborhood silences its modules; removing it deletes them, and for HOPSE also removes that block's preprocessing, which is charged accordingly (Section D.1).

Selection. Algorithm 2 restates the procedure of Section 4.2. The five-epoch initial model defines $a(S)$, the masked validation accuracy; backward elimination removes, at each step, the neighborhood whose removal keeps a highest (ties toward the first neighborhood in the order listed above), continuing down to one neighborhood and yielding one candidate per size. The stop candidate is the architecture immediately before the first removal whose measured loss exceeds $\tau = \sqrt{p(1-p)/m}$; τ is a heuristic based on the estimated standard error of a single accuracy, and it does not bound the cumulative loss relative to the full model. The remaining $B-1$ shortlist places go to the candidates with the highest masked accuracies at sizes not yet taken. Each shortlisted candidate starts from a separate copy of the initial model with excluded components removed and continues training under the dataset's standard settings, with epochs and early stopping counted from the continuation start; the five initial epochs are charged additionally in every reported cost. The full architecture is a candidate whenever the elimination path marks it; the empty architecture is never a candidate and serves as the baseline $a(\emptyset)$. The final choice is the trained candidate with the highest validation accuracy, ties toward the stop candidate. Only this final comparison uses fully trained models; all intermediate removals use the masked game.

Table 3: **Dataset statistics** after preprocessing. Cells per rank are means over complexes (rank 0 nodes, rank 1 edges, rank 2 2-cells; MANTRA is natively simplicial). Node-level datasets are one complex whose splits partition the nodes (transductive, random 50/25/25 split, split seed 0); their split columns count nodes.

Dataset	Train	Val	Test	Cells r0	r1	r2	Classes
GraphXAI benchmarks (lifted to cellular complexes)							
Benzene	8400	1800	1800	21	22	2	2
Fluoride-Carbonyl	6069	1300	1302	21	23	2	2
Alkane-Carbonyl	787	168	170	21	23	2	2
Mutagenicity	1237	265	266	29	30	2	2
Graph benchmarks (lifted to cellular complexes)							
MUTAG	94	47	47	18	20	3	2
NCI1	2055	1027	1028	30	32	4	2
NCI109	2063	1031	1033	30	32	4	2
PROTEINS	556	278	279	39	73	33	2
Cora	1354	677	677	2,708	5,278	1,648	7
Citeseer	1663	831	833	3,327	4,552	1,259	6
Pubmed	9858	4929	4930	19,717	44,324	9,325	3
Natively higher-order (simplicial)							
MANTRA (orientability)	21568	10785	10785	10	34	23	2

B Datasets and Training Settings

B.1 Datasets, Splits, and Liftings

Our experiments use three groups of datasets (Table 3). Benzene, Fluoride-Carbonyl, Alkane-Carbonyl, and Mutagenicity are binary graph-classification tasks from the GraphXAI suite (Agarwal et al., 2023), built on the attribution benchmarks of Sanchez-Lengeling et al. (2020); each molecule's label is determined by a known functional group, so every graph carries a ground-truth explanation. MUTAG, NCI1, NCI109, and PROTEINS are graph-classification benchmarks from the TUDataset collection (Morris et al., 2020); Cora, Citeseer, and Pubmed (Yang et al., 2016) are citation networks with node-level classification (one complex each; transductive, random 50/25/25 node split, split seed 0). MANTRA (Ballester et al., 2025) is a collection of triangulated 2-manifolds, natively simplicial; we use the orientability task. The ZINC subset (Dwivedi et al., 2023) (regression, scored by mean absolute error) appears only in the exhaustive six-neighborhood benchmark of Section D.3, under two training configurations described in Section B.2; both use ZINC's predefined subset splits of 10,000/1,000/1,000 graphs (train/validation/test, about 23 nodes and 50 edges per graph). It is not part of the selection or explanation experiments.

Liftings. Graph datasets are lifted to cellular complexes with TopoBench's cycle lifting (Telyatnikov et al., 2025): chordless cycles of up to ten nodes become rank-2 cells. Two feature liftings appear. The *standard* lifting (ProjectionSum) gives each higher-rank cell the sum of its constituent nodes' features. The *structure-only* lifting gives every cell above rank 0 a constant all-ones feature vector, identical across cells, at the width of the rank below (MANTRA's native convention); rank-1 and rank-2 cells are treated the same way, node features are kept, and the complex itself is unchanged. MANTRA needs no lifting.

Participation. The prediction-explanation experiments of Section 5 use the four GraphXAI datasets (GNN models on graphs; GCCN models on their lifted complexes), the four unannotated TU datasets for faithfulness, and MANTRA for the model diagnostics. The architecture-selection experiments of Section 6 use eleven datasets (all of Table 3 except PROTEINS, which appears in the faithfulness study only). On Pubmed, HOPSE preprocessing does not complete at practical cost (its eigensolvers exceed eight GPU-hours on the lifted complex), so Pubmed carries no HOPSE runs; the HOPSE sweep covers the complex-level datasets.

B.2 Models, Training, and Explainer Settings

TNN models (architecture selection). All GCCN selection experiments use the `topotune_nonlinear_exact` configuration: a TopoTune GCCN backbone with two layers, hidden width 32, one two-layer GCN sub-network per neighborhood (BatchNorm, ReLU), and a linear sum-pool readout. HOPSE models replace each neighborhood's message-passing module with fixed encodings of that neighborhood's graph plus a learned MLP block (Definition 4): four encodings per neighborhood (Laplacian-eigenvector positional, random-walk structural, electrostatic positional, and heat-kernel-diagonal structural), each targeting 20 dimensions per cell and concatenated with the cell's input feature; the encoder width is 64, with two backbone layers and a sum-pool readout. Both families train with Adam (learning rate 10^{-3}, no weight decay), the learning rate halved every 50 epochs, up to 300 epochs with early stopping (patience 25) on the dataset's monitored validation metric, and batch size 64 for graph datasets, 256 for MANTRA, and full-batch for the citation complexes. Checkpoint selection is validation-best. Continuations in the selection pipeline restart these settings from the five-epoch initial model.

ZINC benchmark configurations. The exhaustive benchmark's two ZINC settings share ZINC's predefined subset splits and are scored by negative test MAE. The fixed-recipe setting uses the shared GCCN configuration above (26,112 backbone parameters at the full six-neighborhood model); the strong setting replaces the GCN sub-networks with GIN sub-networks (104,448 backbone parameters).

GNN models (graph explanation). Parameter-matched GCN (Kipf and Welling, 2017) and GIN (Xu et al., 2019) models: three layers, input width 14, hidden width 32, two classes; Adam with learning rate 10^{-3} and weight decay 5×10^{-4}; class-weighted cross-entropy; batch size 64; up to 300 epochs with early stopping on validation balanced accuracy (patience 30); checkpoint = best validation balanced accuracy; stratified shuffle splits with seed 42; three training seeds per model.

Baseline explainers (settings from the saved suite): SubgraphX with 10 Monte-Carlo tree-search rollouts, minimum subgraph size 3, $c_{\mathrm{puct}} = 10$, 14 expansion atoms, local radius 4, and 100 reward samples per evaluation; GNNExplainer with learning rate 0.05 for 1000 epochs (selected on training data; the GraphXAI defaults of 0.01/200 were also run); PGExplainer trained once per dataset-model pair on up to 1000 training graphs (30 epochs, learning rate 0.003 selected on training data; defaults 10 epochs/0.1 also run). Every method receives the annotated motif's size and is evaluated on the same instances. Occlusion scores each cell by the leave-one-out drop $v(P) - v(P \setminus \{i\})$ under the same masked game; gradient-times-input scores each cell by the gradient of the target margin with respect to its raw input features, dotted with those features and summed over the feature dimension (signed); the random baseline draws one seeded uniform score per cell. All methods are ranked by signed score, descending.

Seeds, hardware, software. Comparisons share seeds 42–44; the GCCN B=2 selection additionally carries seeds 45–46 (Table 9). All local runs use one NVIDIA H200 GPU per job; PyTorch 2.3.0 (CUDA 12.1) and a pinned TopoBench commit, listed in Section E.

C Additional Results for TopoSHAP-cell

C.1 Graph Explanation Accuracy and Runtime

From scores to node sets. On graphs, every method is scored as a node set of the annotated motif's size b: the attribution output takes the b highest-scoring nodes, the explanation output takes the greedy candidate $\hat{S}_b$, and baselines are converted with the same top-b rule (GNNExplainer and PGExplainer score a node by the sum of its incident edges' mask weights; SubgraphX scores a node by its best explored-coalition reward, plus one for membership in the returned subgraph; ties break toward the lower node index). On lifted complexes, sizes are matched at the cell level instead: the size-b cell set is projected to the union of its constituent nodes without re-truncation, so the projected node set can exceed b nodes and the Jaccard score penalizes the excess. GEA is the best Jaccard over the instance's admissible annotated candidates. The evaluated population is every ground-truth-positive complex in the test split, shared by every method: 917 (Benzene), 231 (Fluoride-Carbonyl), 96 (Mutagenicity), and 58 (Alkane-Carbonyl) instances.

Table 4: **Explanation accuracy by model type.** Node-level GEA (mean ± sd, 3 seeds) for each parameter-matched model type, one panel per type. The zero-fill row runs our explanation search with feature zero-filling in place of detachment. Bold: column best; gray: within one sd.

GCN models	Benzene	Fluoride	Mutagenicity	Alkane
TopoSHAP (attribution)	0.467 ± 0.078	0.398 ± 0.040	**0.723 ± 0.074**	**0.411 ± 0.058**
TopoSHAP (explanation)	0.377 ± 0.012	0.289 ± 0.094	0.317 ± 0.062	0.318 ± 0.074
TopoSHAP (explanation, zero-fill)	0.648 ± 0.055	0.244 ± 0.034	0.561 ± 0.062	0.378 ± 0.075
SubgraphX	**0.664 ± 0.027**	0.226 ± 0.024	0.344 ± 0.036	0.276 ± 0.013
PGExplainer	0.642 ± 0.044	**0.429 ± 0.092**	0.290 ± 0.102	0.203 ± 0.137
GNNExplainer	0.443 ± 0.054	0.210 ± 0.014	0.152 ± 0.033	0.066 ± 0.050
Random	0.250 ± 0.000	0.219 ± 0.000	0.132 ± 0.000	0.173 ± 0.000

GIN models	Benzene	Fluoride	Mutagenicity	Alkane
TopoSHAP (attribution)	0.859 ± 0.009	0.507 ± 0.047	**0.798 ± 0.124**	**0.500 ± 0.081**
TopoSHAP (explanation)	**0.969 ± 0.009**	0.340 ± 0.010	0.480 ± 0.135	0.354 ± 0.125
TopoSHAP (explanation, zero-fill)	0.569 ± 0.064	0.313 ± 0.038	0.273 ± 0.164	0.391 ± 0.028
SubgraphX	0.796 ± 0.029	0.439 ± 0.107	0.274 ± 0.055	0.296 ± 0.010
PGExplainer	0.453 ± 0.239	**0.555 ± 0.067**	0.175 ± 0.200	0.205 ± 0.160
GNNExplainer	0.256 ± 0.081	0.166 ± 0.035	0.158 ± 0.023	0.162 ± 0.049
Random	0.250 ± 0.000	0.219 ± 0.000	0.132 ± 0.000	0.173 ± 0.000

Table 5: **Per-instance explanation cost.** Measured wall-clock seconds per explained instance (mean ± sd over both model types and 3 seeds), on one NVIDIA H200 per run. †PGExplainer's per-instance entry is inference only; the last row is its one-time training.

Method	Benzene	Fluoride	Mutagenicity	Alkane
TopoSHAP (attribution)	3.3 ± 0.9	3.3 ± 0.8	4.0 ± 3.1	3.3 ± 0.8
TopoSHAP (explanation)	0.33 ± 0.11	0.35 ± 0.11	0.67 ± 1.05	0.36 ± 0.11
SubgraphX	21.1 ± 16.1	34.3 ± 28.9	51.6 ± 79.4	40.4 ± 28.8
PGExplainer†	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
GNNExplainer	0.81 ± 0.08	0.80 ± 0.08	0.80 ± 0.07	0.82 ± 0.08
Random	0.001 ± 0.001	0.003 ± 0.010	0.001 ± 0.001	0.002 ± 0.003
PGExplainer training (one-time)†	1500.8 ± 159.5	1081.3 ± 115.9	221.3 ± 21.7	142.5 ± 15.5

Runtime protocol. Table 5 reports measured wall-clock seconds per explained instance, averaged over both model types and three seeds, on one NVIDIA H200 GPU per run, with each method timed end to end on the same instances; PGExplainer's one-time per-dataset training is reported as its own row.

C.2 Feature Lifting and Motif Recovery

Table 6 reports the lifting diagnostics behind Section 5.2, on the explanation output $\hat{S}_b$ of the GCCN with all eleven neighborhoods. Face retention is the fraction of instances whose selected coalition contains at least one rank-2 cell; boundary recall is the mean recall of the annotated motif's rank-1 cells. The two statistics are computed separately, and both use the full ground-truth-positive test populations and three training seeds. The final column re-scores the standard lifting with its summed higher-rank features recomputed on every masked complex, so a retained face carries only the features of its retained nodes; everything else (weights, players, target) is unchanged. The difference in motif recovery between the two liftings persists when the summed features are recomputed after masking.

C.3 Faithfulness Without Annotations

Deletion-AUC detaches cells cumulatively, most important first. With m_k the target margin after detaching the k highest-ranked of n cells and $s = \mathrm{sign}(m_0)$ orienting the margin toward the predicted

Table 6: **Lifting diagnostics for the explanation output.** Face retention: fraction of instances whose selected coalition contains at least one rank-2 cell. Boundary recall: mean recall of the annotated motif's rank-1 cells. Both are computed on the full ground-truth-positive test populations (mean ± sd over three training seeds). The last column rescores the standard lifting with its summed features recomputed on every masked complex instead of held fixed (node-projected GEA); its structural-lifting counterpart is in Table 2.

	Face retention		Boundary recall		GEA, recomputed feat.
Dataset	standard	structure-only	standard	structure-only	(standard lifting)
Benzene	0.84 ± 0.14	0.95 ± 0.05	0.40 ± 0.14	0.86 ± 0.06	0.56 ± 0.09
Fluoride-Carbonyl	0.00 ± 0.00	0.00 ± 0.00	0.16 ± 0.06	0.53 ± 0.04	0.28 ± 0.02
Mutagenicity	0.05 ± 0.04	0.13 ± 0.15	0.08 ± 0.04	0.30 ± 0.04	0.18 ± 0.02
Alkane-Carbonyl	0.02 ± 0.03	0.03 ± 0.05	0.07 ± 0.13	0.50 ± 0.24	0.25 ± 0.17

Table 7: **Faithfulness without ground truth.** Deletion-AUC (lower is better; a fixed draw of 50 test complexes per dataset, except MUTAG, which uses its whole 47-complex test split). Subjects are GCCNs with all eleven neighborhoods under the structure-only lifting; explanations target the model's own prediction.

	NCI1	NCI109	MUTAG	PROTEINS
TopoSHAP	$\mathbf{0.218 \pm 0.005}$	$\mathbf{0.203 \pm 0.044}$	$\mathbf{0.175 \pm 0.043}$	$\mathbf{0.338 \pm 0.067}$
Occlusion	0.254 ± 0.021	0.240 ± 0.038	0.231 ± 0.006	0.431 ± 0.065
Gradient×Input	0.552 ± 0.016	0.507 ± 0.049	0.499 ± 0.073	0.638 ± 0.047
Random	0.545 ± 0.004	0.491 ± 0.035	0.481 ± 0.014	0.666 ± 0.058

class,

$$\mathrm{AUC} = \sum_{k=1}^{n} \frac{\sigma(s\, m_{k-1}) + \sigma(s\, m_k)}{2} \cdot \frac{1}{n},$$

the trapezoid rule applied to the predicted-class probability (σ: logistic function) over the removed fraction, from the full complex to full removal. Lower is better. The explained models are GCCNs with all eleven neighborhoods under the structure-only lifting; explanations target the model's own prediction. The evaluated population is a fixed seed-0 draw of 50 test complexes per dataset, except MUTAG, which uses its entire 47-complex test split.

C.4 MANTRA: Attribution Patterns and Feature Ablations

Metrics. The overall AUROC in Section 5.3 is computed on MANTRA's full test split from each model's orientable-class score. The count-conditional AUROC restricts comparisons to pairs of test complexes with identical (V, E, F) cell counts: over all count groups containing both classes (four groups), it is the pooled probability that an orientable complex outscores a non-orientable one within the same group; between-group pairs are excluded, so a model that predicts from counts alone cannot exceed 0.5. These AUROCs are model diagnostics and are distinct from the balanced test accuracy used for selection results.

Feature ablations. Table 8 collects the comparison behind Section 5.3: models restricted to cell counts or purely structural features, the GCCN and HOPSE selections of Section 6.1, and a GCCN trained with Hodge Laplacian eigenvectors concatenated to its input features. The rows differ in model-selection procedure and seed count (stated per row in the table), so the table compares feature sets across configurations rather than isolating one controlled feature change.

D Additional Results for TopoSHAP-neighbor

D.1 Architecture Selection, Baselines, and Compute

Sweeps. The GCCN sweep is a TopoTune-style neighborhood grid of 22–25 configurations per dataset: the saved record's per-dataset draws on the eight datasets it covered, and one fixed 23-

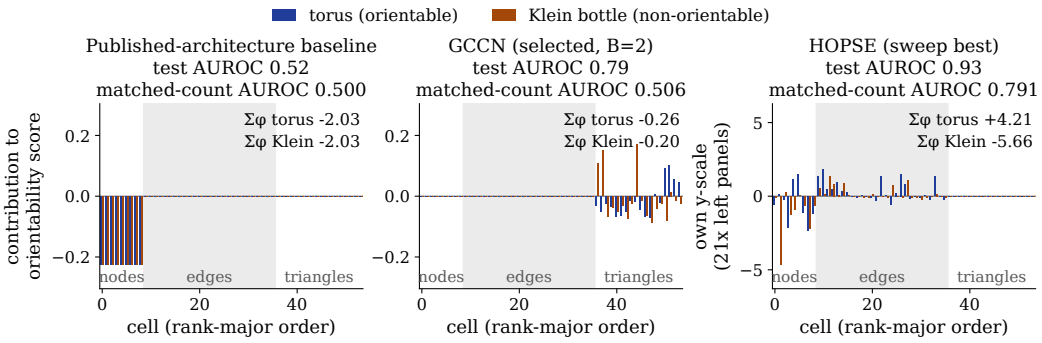

Figure 3: **MANTRA attributions on a count-matched pair.** One orientable torus and one non-orientable Klein-bottle-family surface with identical cell counts (9 nodes, 27 edges, 18 triangles), drawn from the smallest count group containing both classes: per-cell contribution to the orientable-class logit margin for the three models of Section 5.3 (published-architecture baseline; the GCCN selected by TopoSHAP-neighbor at $B{=}2$; the sweep-best HOPSE). Panel headers give each model's overall and count-conditional test AUROC. HOPSE is drawn on its own y-scale, stated on the axis. Per-cell values are single sampled-game estimates at $T = 512$ orderings (Section A.1); no aggregation, hence no error bars.

Table 8: **MANTRA orientability: feature and model comparison.** Balanced test accuracy. The count-only row is a single logistic regression fit; the structural-features and eigenvector rows are means over three training seeds whose per-seed values were not exported, so no sd is shown. The feature study trains its own GCCN configuration, distinct from the selected models, so rows compare feature sets across configurations. Selection rows report mean ± sd over seeds.

Model / input features	Balanced test acc.	Runs
Logistic regression on cell counts	0.505	1 fit
Structural features only	0.541	3 seeds (means only)
GCCN, selected ($B{=}2$, plain-accuracy selection)	0.641 ± 0.123	5 seeds
GCCN + Hodge Laplacian eigenvectors	0.876	3 seeds (means only)
HOPSE, selected ($B{=}2$)	0.771 ± 0.017	3 seeds
HOPSE, sweep best	0.792 ± 0.009	3 seeds

configuration draw reused verbatim on the three datasets swept for this paper (per-dataset counts in Figure 2's legend; the configuration lists ship with the released pipeline). The HOPSE sweep is its published 12-configuration grid. Sweep winners are selected per seed by validation accuracy and charged the full grid.

Shortlist controls. Each control starts from the same five-epoch initial model and continues training under the standard settings of the selection pipeline. The random $B{=}2$ control draws two architectures uniformly at random from the 2047 non-empty neighborhood subsets (draw seeded by dataset and seed, redrawing duplicates), continues both, and selects between them by validation accuracy: two continuations, like the selection pipeline, but without backward elimination's 66 validation passes. The full-model default continues one model, the all-neighborhood architecture, with no search and no selection step.

Compute accounting. All FLOPs are counted under one convention: PyTorch's `FlopCounterMode` with dense pricing, explicit formulas for operations the counter does not price (eigendecomposition, SVD, sparse products), and every stage charged, including HOPSE's encoding preprocessing. Runs that predate the profiler are priced by rebuilding the recorded architecture, profiling one training epoch and the evaluation passes under the same convention, and combining these per-epoch counts with the recorded epoch and evaluation tallies; we describe such costs as profiler-based reconstructions. Exact neighborhood attribution requires all $2^{11} = 2048$ masked validation passes; backward elimination requires $(n^2 + n)/2 = 66$; sampled cell attribution requires

Table 9: **Architecture selection: full results by model family.** Test accuracy (mean ± sd; GCCN B=2 uses five seeds 42–46, all other columns the shared seeds 42–44; [b]balanced test accuracy on MANTRA) and, in parentheses, total training FLOPs as a multiple of that dataset's B=1 run (Section D.1); sweeps are charged their full grid. Gap = selection B=2 minus that family's sweep; sweeps use per-seed validation-best configurations. —: not run. Pubmed carries no HOPSE runs (Section B.1).

	GCCN family			
Dataset	B=1	B=2	sweep	gap
Benzene	0.925 ± 0.007 (×1.0)	0.926 ± 0.004 (×2.0)	0.926 ± 0.002 (×22.7)	+0.000
Fluoride-Carb.	0.901 ± 0.057 (×1.0)	0.939 ± 0.006 (×1.9)	0.938 ± 0.004 (×20.6)	+0.001
Mutagenicity	0.967 ± 0.011 (×1.0)	0.975 ± 0.010 (×2.0)	0.964 ± 0.014 (×15.4)	+0.012
Alkane-Carb.	0.965 ± 0.006 (×1.0)	0.962 ± 0.005 (×2.4)	0.975 ± 0.003 (×16.2)	-0.012
NCI1	0.764 ± 0.011 (×1.0)	0.758 ± 0.010 (×2.8)	0.759 ± 0.022 (×15.0)	-0.000
MUTAG	0.879 ± 0.025 (×1.0)	0.906 ± 0.024 (×2.1)	0.901 ± 0.025 (×13.1)	+0.006
NCI109	0.758 ± 0.011 (×1.0)	0.756 ± 0.009 (×1.8)	0.747 ± 0.015 (×11.3)	+0.009
Cora	0.838 ± 0.008 (×1.0)	0.835 ± 0.005 (×2.3)	0.856 ± 0.006 (×14.9)	-0.021
Citeseer	0.673 ± 0.014 (×1.0)	0.670 ± 0.007 (×1.8)	0.683 ± 0.002 (×15.1)	-0.013
Pubmed	0.860 ± 0.004 (×1.0)	0.863 ± 0.004 (×2.3)	0.865 ± 0.001 (×14.4)	-0.002
MANTRA[b]	0.532 ± 0.028 (×1.0)	0.641 ± 0.123 (×1.8)	0.548 ± 0.000 (×11.0)	+0.093

	HOPSE family		
Dataset	B=2	sweep	gap
Benzene	0.934 ± 0.001 (×5.4)	0.936 ± 0.003 (×36.4)	-0.002
Fluoride-Carb.	0.918 ± 0.004 (×8.5)	0.934 ± 0.009 (×41.6)	-0.015
Mutagenicity	0.961 ± 0.008 (×5.9)	0.976 ± 0.009 (×36.3)	-0.015
Alkane-Carb.	0.969 ± 0.009 (×10.5)	0.963 ± 0.007 (×47.8)	+0.006
NCI1	0.759 ± 0.024 (×5.6)	0.745 ± 0.011 (×34.5)	+0.013
MUTAG	0.844 ± 0.012 (×10.9)	0.844 ± 0.054 (×53.3)	+0.000
NCI109	0.720 ± 0.005 (×5.2)	0.737 ± 0.011 (×27.1)	-0.017
Cora	0.851 ± 0.005 (×10.2)	—	—
Citeseer	0.712 ± 0.007 (×6.8)	—	—
MANTRA[b]	0.771 ± 0.017 (×9.6)	0.792 ± 0.009 (×68.7)	-0.021

Table 10: **Shortlist baselines with shared initialization and training settings.** Test accuracy (mean ± sd, shared seeds 42–44; [b]balanced on MANTRA). All three columns share the five-epoch initial model and standard continuation; random B=2 replaces backward elimination with two uniformly random architectures, and the full-model default trains all 11 neighborhoods with no search.

Dataset	TopoSHAP-neighbor B=2	Random B=2	Full default
Benzene	0.927 ± 0.000	0.925 ± 0.013	0.927 ± 0.003
Fluoride-Carbonyl	0.939 ± 0.007	0.929 ± 0.002	0.938 ± 0.003
Mutagenicity	0.976 ± 0.012	0.932 ± 0.061	0.972 ± 0.015
Alkane-Carbonyl	0.965 ± 0.006	0.973 ± 0.003	0.955 ± 0.007
NCI1	0.757 ± 0.013	0.755 ± 0.026	0.766 ± 0.005
MUTAG	0.915 ± 0.021	0.879 ± 0.025	0.879 ± 0.012
NCI109	0.755 ± 0.011	0.755 ± 0.011	0.749 ± 0.001
Cora	0.839 ± 0.002	0.831 ± 0.021	0.836 ± 0.003
Citeseer	0.669 ± 0.001	0.660 ± 0.013	0.666 ± 0.010
Pubmed	0.862 ± 0.004	0.853 ± 0.006	0.861 ± 0.002
MANTRA[b]	0.615 ± 0.103	0.687 ± 0.114	0.777 ± 0.035

at most $Tn + 1$ forward passes per instance, with T counting every evaluated ordering (Section A.1 and Table 5). Wall-clock comparisons run on one NVIDIA H200 GPU per run.

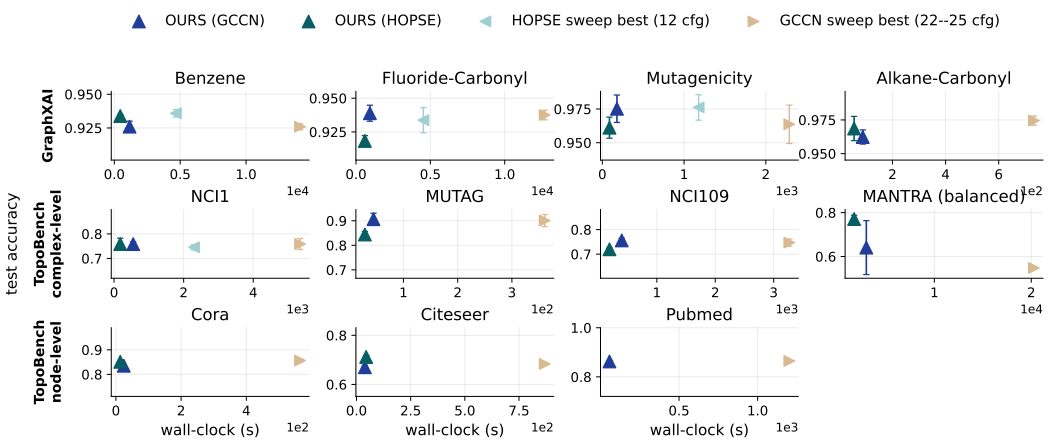

Figure 4: **Wall-clock variant of Figure 2.** The same four marker groups as Figure 2 (selected GCCN, selected HOPSE, and each family's sweep), with the horizontal axis showing measured wall-clock seconds (one NVIDIA H200 GPU per run) instead of FLOPs. Wall-clock time includes HOPSE's preprocessing.

Table 11: **Sensitivity to the brief-run epoch count.** Test accuracy of the $B{=}2$ selection (mean $\pm$ sd, 3 seeds) with the initial model trained for 3, 5 (default), or 10 epochs.

Brief-run epochs	Benzene	NCI1	MUTAG
3	0.926 ± 0.008	0.763 ± 0.006	0.879 ± 0.033
5 (protocol)	0.927 ± 0.000	0.757 ± 0.013	0.915 ± 0.021
10	0.925 ± 0.006	0.769 ± 0.008	0.887 ± 0.033

D.2 Sensitivity Analyses

Initial-training length. Table 11 repeats the $B{=}2$ selection with the initial model trained for 3, 5 (default), or 10 epochs. Mean selected test accuracy moves by at most 0.012 on Benzene and NCI1 and by 0.036 on MUTAG; the selected neighborhood sets themselves differ across settings.

Selection metric on MANTRA. A separately run three-seed variant replaces ordinary validation accuracy with balanced validation accuracy in the masked game, the threshold's p, and the final comparison (τ's binomial form is then an approximation, since balanced accuracy is not a binomial proportion). It selects 11, 11, and 9 neighborhoods across seeds 42–44 and reaches 0.804 ± 0.011 balanced test accuracy, against 0.615 ± 0.103 for the original plain-accuracy run on the same seeds, consistent with ordinary validation accuracy rewarding majority-class predictions on this imbalanced dataset. Figure 2 and Table 9 report the original plain-accuracy selection.

D.3 Exhaustive Benchmark and Attribution Agreement

The controlled evaluation of Section 6.2 uses all $2^6 = 64$ subsets of the first six neighborhoods of Section A.2, each trained separately from scratch to the standard budget, under twelve dataset/hyperparameter settings and seeds 42–44; the empty subset is a trained model with node features only. The twelve settings are the four GraphXAI molecule datasets, MUTAG, NCI1, NCI109, the two ZINC configurations of Section B.2, and the three citation graphs. Seven settings were available while the selection procedure was developed; five (MUTAG, NCI109, Alkane-Carbonyl, and the two ZINC configurations) were held out and first evaluated after the procedure was fixed.

Table 12 compares exact Shapley attributions over the six neighborhoods computed two ways: from masked evaluations of the five-epoch model (the logit column scores validation logits, the primary comparison; the accuracy column re-scores the masked side by validation accuracy) and from the separately trained subsets' test values. Agreement is the Spearman correlation between seed-averaged

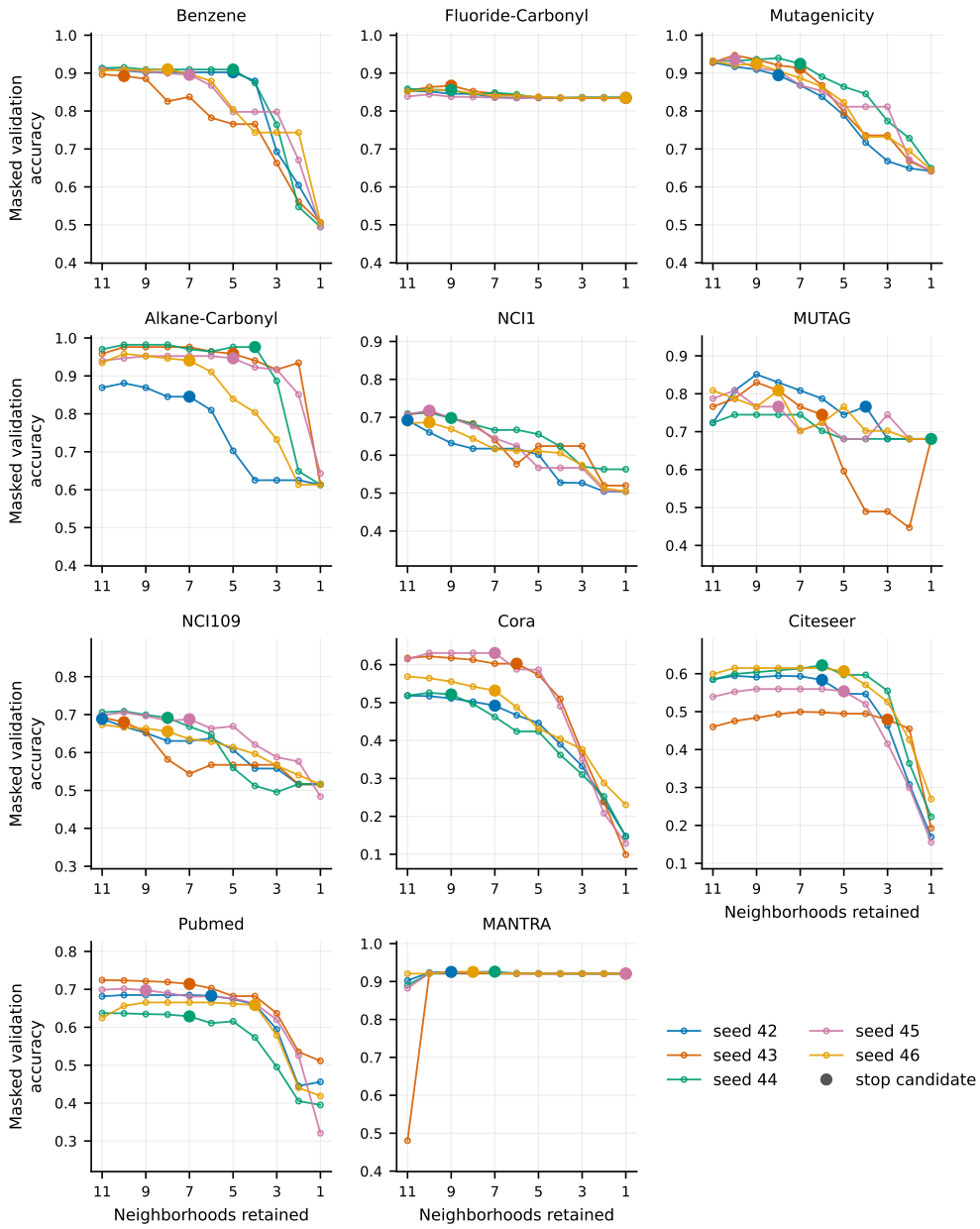

Figure 5: **Backward-elimination traces on all eleven datasets.** Masked validation accuracy along the GCCN elimination path, from eleven neighborhoods to one, for training seeds 42–46. Markers identify the stop candidate selected before the first removal whose accuracy loss exceeds τ; elimination continues after this candidate is marked. All panels share one y-axis span, positioned per panel. MANTRA uses ordinary validation accuracy in this experiment.

Table 12: **Attribution agreement with retraining.** Spearman correlation between masked-game and retraining-game Shapley profiles over the six neighborhoods (masked: logit-scored, validation split; retraining: test metric). Agreement: correlation between seed-averaged attribution profiles. Stability: per-seed correlation, mean ± sd (seeds 42–44). Pooled row: 95% CI in brackets.

Dataset	Agreement ρ (logit)	(accuracy)	Per-seed ρ (stability)
MUTAG	-0.37	+0.89	-0.16 ± 0.74
NCI1	-0.31	+0.14	-0.12 ± 0.95
NCI109	+0.09	+0.09	+0.20 ± 0.70
ZINC-subset (fixed recipe)	-0.09	-0.09	+0.14 ± 0.29
ZINC-subset (strong)	-0.31	-0.31	-0.45 ± 0.14
Alkane-Carbonyl	+0.49	+0.49	+0.50 ± 0.36
Benzene	+0.83	+0.31	+0.43 ± 0.37
Citeseer	+0.66	+0.66	+0.56 ± 0.12
Cora	+0.94	+0.94	+0.94 ± 0.00
PubMed	+0.54	+0.37	+0.39 ± 0.22
Fluoride-Carbonyl	-0.09	+0.77	+0.26 ± 0.45
Mutagenicity	+0.94	+0.77	+0.71 ± 0.20
Pooled (72 points)	+0.23 [-0.05, +0.52]	+0.32 [+0.06, +0.60]	

attribution profiles; stability is the mean ± sd of the per-seed correlations, which mixes seed-to-seed variation of the masked profile into the same statistic and is reported separately for that reason. The pooled row correlates $12 \times 6 = 72$ points (one per setting and neighborhood, each a three-seed mean); intervals are 95% bootstrap over the twelve setting blocks (10,000 resamples), and the primary statistic's permutation p is 0.048 (20,000 permutations). The two games differ in split and score (validation logits or accuracy versus test accuracy or negative MAE), a mismatch the saved record does not allow us to remove.

D.4 Oracle Evaluation of Candidate Generation

Table 13 reruns backward elimination with each removal scored by the benchmark's seed-averaged test values, per setting. Because it consults test scores, this analysis is an oracle diagnostic of the candidate generator; it is not a validation-only application of the practical method, and it does not validate the masked scores the practical pipeline uses. On the five held-out settings the greedy path contains the benchmark's best architecture in four, with worst regret $+0.0068$ (the GIN-backbone ZINC recipe); on the seven development settings it contains the best architecture in five and trails by at most $+0.0042$.

E Reproduction Instructions

Environment. Python 3.11; PyTorch 2.3.0 (CUDA 12.1); `torch-geometric` with matching scatter/sparse wheels; TopoBench pinned at the commit recorded in the repository. The repository's README gives the exact installation commands.

Released code. The release contains the explainer library with its test suite, including the exactness check of the Shapley implementation against brute-force enumeration, a tutorial, and the experiment runners and configuration files used in this paper.

Rerunning experiments. Each experiment has one runner under `experiments/` (for example, architecture selection, the sweeps, the shortlist controls, the balanced-selection MANTRA variant, and the faithfulness study), with seeds and output locations set by command-line arguments. The exhaustive six-neighborhood benchmark and the GraphXAI explainer suite were produced on a separate cluster; their configurations and saved numerical results are included with the release.

Table 13: **Oracle evaluation of candidate generation on the exhaustive benchmark.** Backward elimination is rerun with each removal scored by the separately trained coalitions' seed-averaged test values (three seeds); ties break toward the first neighborhood in the order of Section A.2. Because it consults test scores, this is a diagnostic of the candidate generator, not a deployable selection. Empty = trained model with node features only; Full = all six neighborhoods; Best = best of all 64 subsets, with its size; Path best = best subset on the greedy elimination path; Regret = Best − Path best. ZINC values are negative test MAE (higher is better).

Setting	Status	Empty	Full	Best (k)	Path best (k)	Regret
Benzene	development	0.739	0.921	0.925 (3)	0.925 (3)	+0.0000
Mutagenicity	development	0.821	0.956	0.957 (5)	0.957 (5)	+0.0000
Alkane-Carbonyl	held-out	0.967	0.967	0.984 (4)	0.984 (4)	+0.0000
Fluoride-Carbonyl	development	0.822	0.935	0.943 (4)	0.943 (4)	+0.0000
MUTAG	held-out	0.723	0.901	0.922 (5)	0.922 (5)	+0.0000
NCI1	development	0.667	0.759	0.768 (3)	0.768 (5)	+0.0000
NCI109	held-out	0.680	0.756	0.762 (3)	0.762 (3)	+0.0000
ZINC-subset (fixed recipe)^m	held-out	-0.699	-0.548	-0.534 (5)	-0.534 (5)	+0.0000
ZINC-subset (strong)^m	held-out	-0.723	-0.225	-0.177 (4)	-0.184 (5)	+0.0068
Cora	development	0.632	0.848	0.861 (3)	0.861 (3)	+0.0000
Citeseer	development	0.644	0.681	0.688 (3)	0.685 (3)	+0.0024
PubMed	development	0.854	0.861	0.867 (3)	0.863 (5)	+0.0042

^{m}negative test MAE.