EMPOOL: COLLAPSE VIA DISCRETE MORSE THEORY FOR REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite their success on non-Euclidean data, Message Passing Neural Networks (MPNNs) face fundamental depth and expressivity limitations rooted in topological bottlenecks, notably over-smoothing and over-squashing. To address these structural constraints, we propose *Empool*, an end-to-end differentiable graph pooling framework grounded in discrete Morse theory. By modeling graphs as Lefschetz complexes, Empool learns a discrete Morse function through a parameterized multi-layer perceptron that governs a Forman reduction, progressively collapsing the graph topology into a compact set of critical cells while rigorously preserving its homology. The resulting sparse representation mitigates the topological bottlenecks responsible for information degradation in deep networks. Extensive experiments on established node- and graph-prediction benchmarks demonstrate competitive and robust performance relative to state-of-the-art architectures.

1 INTRODUCTION

Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017) have established themselves as a foundational paradigm for representation learning on graph-structured data, driving breakthroughs across domains ranging from molecular property prediction in chemistry to relational modeling in social networks. Despite their empirical success, the ability of MPNNs to scale to deep architectures remains fundamentally constrained. As network depth increases, standard message passing suffers from profound structural and information-theoretic bottlenecks—most notably *over-smoothing*, where node representations converge indiscriminately, and *over-squashing*, where exponentially growing neighborhood information is compressed into fixed-size vector representations, severely hindering long-range signal propagation (Giraldo et al., 2023).

A wide array of strategies has been proposed to mitigate these limitations, including neighborhood sampling schemes (Hamilton et al., 2017; Zou et al., 2019; Balin & Çatalyürek, 2023), principled edge rewiring techniques (Topping et al., 2021), expressive attention mechanisms (Velickovic et al., 2017), and various graph pooling paradigms (Gao & Ji, 2019; Lee et al., 2019; Bianchi et al., 2019; Ying et al., 2018). While these methods offer partial remedies, they often rely on heuristics that disrupt the underlying structural geometry, introduce sampling variance, or fail to provide formal guarantees regarding the preservation of global topological properties. Consequently, designing an architecture that deepens message passing while safeguarding the underlying manifold and structural integrity remains a central open challenge.

In this work, we propose a radically different approach rooted in discrete topology and combinatorial Morse theory. Rather than treating graph coarsening as a heuristic downsampling step, we formalize graphs as Lefschetz complexes and extend Forman's discrete reduction framework to graph structures. This yields a homology-invariant reduction that serves as the mathematical foundation for *Empool*, an end-to-end differentiable pooling framework. Empool learns a vertex scalar field to dynamically induce a gradient structure, guiding a Forman reduction that progressively collapses redundant topology into a compact set of critical cells while rigorously preserving the homology of the graph. By operating on this structurally condensed proxy, Empool mitigates the topological bottlenecks responsible for information degradation.

It is worth noting that existing topology-aware pooling methods (Chen & Gel, 2023) frequently incorporate persistent homology, landmark selections, or witness complexes primarily into node scoring or heuristic graph coarsening. In contrast, Empool establishes a direct, differentiable link

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.

between discrete Morse theory and network coarsening, deriving its reduction straight from a parameterized gradient vector field.

Our main contributions are summarized as follows:

- We formulate a rigorous mathematical mapping from graphs to Lefschetz complexes, broadening the analytical utility of discrete Morse theory in deep learning. We define the *Forman–Lefschetz reduced* graph, formally prove its homology equivalence to the original structure under precise conditions, and introduce an efficient algorithm for its computation.
- We introduce the Empool architecture, seamlessly integrating the Forman–Lefschetz reduction into a graph pooling pipeline. We detail its design, addressing core challenges such as resolving dimensional discrepancies and circumventing non-differentiability through novel relaxation techniques.
- We conduct an extensive empirical benchmarking suite across diverse node- and graph-prediction datasets, demonstrating that Empool achieves highly competitive and robust predictive accuracy relative to state-of-the-art architectures without sacrificing structural fidelity.

All mathematical proofs and technical derivations are provided in Appendix B.

2 The Collapsibility of Graphs via Discrete Morse Theory

The motivation for modeling graphs as Lefschetz complexes stems from their natural correspondence with geometric and topological structures. Once this correspondence is established, Forman's discrete reduction allows us to retain the graph's "critical cells" while preserving its underlying homology. We therefore define the *Forman–Lefschetz reduction* of a graph, which we use as the structural basis for our pooling mechanism during GNN training.

In this study, we exclude multigraphs *i.e.* our study is restricted to graphs such that there exists at most one edge between each pair of vertices.

2.1 Graphs as Lefschetz Complexes

We formalize this correspondence by establishing that every graph induces a Lefschetz complex.

Theorem 1 (Lefschetz Complex of a Graph). *Let $\mathcal{G} = (V, E)$ be a graph. Let $\mathcal{K}_0 = \{\{v\} \mid v \in V\}$ and $\mathcal{K}_1 = E$. Let $X = \mathcal{K}_0 \cup \mathcal{K}_1$ and define $\kappa : X \times X \to \mathbb{Z}_2$ by*

$$\forall x, y \in X, \quad \kappa(x, y) = \begin{cases} 1, & x \in \mathcal{K}_1,\ y \in \mathcal{K}_0 \text{ and } y \subseteq x, \\ 0, & \text{otherwise}. \end{cases}$$

Then $\left(X, (\mathcal{K}_k)_{k=0,1}, \kappa\right)$ is a regular Lefschetz complex over $\mathbb{Z}_2$.

Definition 1 (Lefschetz Complex of a Graph). *For a fixed graph $\mathcal{G}$, the Lefschetz complex $\mathscr{L}_\mathcal{G}$ given by Theorem 1 is called the Lefschetz complex induced by $\mathcal{G}$.*

This incidence map induces the natural cell ordering of the graph: in particular, $v \prec e$ precisely when v is a vertex incident to the edge e. Moreover, vertices and edges naturally correspond to 0- and 1-dimensional cells, respectively, through the partition $(\mathcal{K}_k)_{k=0,1}$. Consequently, we obtain the notion of a critical cell of a graph.

Definition 2 (Critical Cell). *Let $\mathcal{G} = (V, E)$ be a graph with induced Lefschetz complex $\mathscr{L}_\mathcal{G}$, and let f be a discrete Morse function on $\mathscr{L}_\mathcal{G}$.*

A vertex $v \in V$ (resp. an edge $e \in E$) is called critical with respect to f if the corresponding cell $\{v\}$ (resp. e) is critical in $\mathscr{L}_\mathcal{G}$ with respect to f. When the distinction between critical vertices and critical edges is unnecessary, we refer to either as a critical cell.

Applying Forman's discrete reduction theorem (Forman, 2002), we obtain the following result.

Theorem 2 (Forman–Lefschetz Reduction of a Graph). *Let $\mathcal{G} = (V, E)$ be a graph and $\mathscr{L}_\mathcal{G}$ its induced Lefschetz complex. Let V_f be a discrete gradient vector field on $\mathscr{L}_\mathcal{G}$. The quotient graph*

$\mathcal{G}_{\text{Forman}}$ formed by identifying vertices along the acyclic gradient paths preserves the dominant homological invariants of $\mathscr{L}_{\mathcal{G}}$ under valid elementary collapses:

$$\mathcal{H}_*\left(\mathscr{L}_{\mathcal{G}};\mathbb{Z}_2\right) \cong \mathcal{H}_*\left(\mathscr{L}_{\mathcal{G}_{\text{Forman}}};\mathbb{Z}_2\right)$$

It is worth noting that the original theorem for cell complexes does not define the Forman collapse as a single "monolithic" process for obtaining a homology-equivalent complex; rather, "the" Forman collapse is not a unique procedure, but a family of valid, homology-invariant operations built from *elementary* collapses. Furthermore, these elementary collapses consist of removing non-critical cells one at a time. By extension, Theorem 2 ensures this principle applies to graphs in the sense defined above. We discuss in detail how we control the extent to which the original graph is collapsed in Section 3.

2.2 The Lower Star Reduction Algorithm

The framework introduced by Forman (Forman, 1998) offers an algorithmic way to collapse complexes by iteratively removing non-critical cells. However, requiring a discrete gradient vector field (or discrete Morse function) as input imposes two primary constraints:

- **Semantic constraint:** Beyond theoretical correctness, the gradient field must align with the GNN's downstream task to effectively guide the graph collapse.
- **Algorithmic constraint:** An optimal gradient field minimizes the number of critical cells to the lower bound established by the Betti numbers (Forman, 2002). While this yields a "perfect gradient field," computing it is NP-hard (Joswig & Pfetsch, 2006).

To bypass these obstacles, we adopt a data-driven Lower Star approach (Robins et al., 2011). First, we restrict Morse values to the 0-cells (vertices) and compute them using a parameterized multilayer perceptron, allowing the model to learn the optimal gradient end-to-end. Second, critical nodes behave, by design, as local minima in terms of Morse value and act as "attractors" of their neighbors. Thus, we construct the gradient field greedily: each node collapses toward the adjacent node offering the steepest descent in Morse value.

Formally, let $f : V \to \mathbb{R}$ be the learned vertex function. We define the downward gradient magnitude for adjacent nodes u, v as $\Delta f(v,u) = \text{ReLU}\left(f(v) - f(u)\right)$. The negative gradient of a node v is thus:

$$-\nabla f(v) = \underset{u\in\mathcal{N}(v)}{\arg\max}\, \Delta f(v,u).$$

Traversing vertices sorted by $f(v)$, if $\Delta f\left(v, -\nabla f(v)\right) = 0$, v is a local minimum and flagged as a critical node. Otherwise, we draw a gradient arrow from v to $-\nabla f(v)$. To track cluster assignments, we define a parent function π:

$$\forall v \in V, \quad \pi(v) = \begin{cases} v, & \text{if } v \in \mathcal{C}_0, \\ -\nabla f(v), & \text{otherwise.} \end{cases}$$

Applying this parent-pointer jumping for a fixed number T of times routes nodes deeper into their basins without distorting critical nodes, drastically reducing the graph size. To enable an optimized pooling and unpooling mechanisms, we define a Sink matrix $\boldsymbol{S}$:

Definition 3 (Sink Matrix)**.** *Let $\mathcal{G} = (V, E)$ have n nodes, and $\mathcal{R}_T = \{v_{i_1}, \cdots, v_{i_k}\}$ be the set of remaining nodes after $T \geq 1$ pointer-jumping steps. The **Sink Matrix** $\boldsymbol{S} \in \{0,1\}^{n\times k}$ is defined as $\boldsymbol{S}_{p,q} = 1$ if v_p collapses into v_{i_q}, and 0 otherwise. Equivalently,*

$$\boldsymbol{S}_{p,q} = 1 \iff \underbrace{\pi \circ \cdots \circ \pi}_{T \text{ times}}(v_p) = v_{i_q}.$$

We emphasize that, for a fixed cell complex and discrete gradient vector field, Forman reduction does not define a unique intermediate reduced complex. Rather, different admissible sequences of non-critical cell removals yield a finite family of homology-equivalent reductions, whose maximally reduced representative retains only the critical cells.

Empool explicitly controls this reduction spectrum through T, the number of pointer-jumping iterations. After T iterations,

$$\mathcal{R}_T = \mathrm{Im}\left(\pi^{2^T}\right).$$

Thus $\mathcal{R}_T$ contains the critical vertices but need not coincide with them for finite T. Since the parent map is acyclic on a finite graph, there exists T_0 such that

$$\forall T \geq T_0, \qquad \mathcal{R}_T = \mathcal{C}_0.$$

Thus, T controls the depth to which vertices are propagated along the discrete gradient field, yielding a nested family of representative sets $\mathcal{C}_0 = \mathcal{R}_{T_0} \subset \cdots \subset \mathcal{R}_2 \subset \mathcal{R}_1$ and controlling the strength of graph coarsening.

Algorithm 1: Discrete Gradient Field, Parent-Pointer Tracking & Collapse Matrix

Input: Graph $\mathcal{G} = (V, E)$, Morse function $f : V \longrightarrow \mathbb{R}$, max iterations T
Output: Sink matrix $\boldsymbol{S}$
Sort V ascending by $f(v)$ into $[v_1, \ldots, v_n]$;
Initialize $\mathcal{C}_0 \leftarrow \varnothing$, $V_f \leftarrow \varnothing$, $\pi(v) \leftarrow v$;
for $i = 1, \ldots, n$ **do**
  $u^* \leftarrow -\nabla f(v_i)$;
  if $\Delta f(v_i, u^*) > 0$ **then**
    $V_f \leftarrow V_f \cup \{(v_i, \{v_i, u^*\})\}$;
    $\pi(v_i) \leftarrow u^*$
  else
    $\mathcal{C}_0 \leftarrow \mathcal{C}_0 \cup \{v_i\}$
for $t = 1, \ldots, T$ **do**
  for each $v \in V$ **do**
    $\pi(v) \leftarrow \pi(\pi(v))$;
Enumerate $\mathcal{R}_T = \{r_1, r_2, \ldots, r_k\}$;
Initialize $\boldsymbol{S} \leftarrow \mathbf{0}_{|V| \times k}$;
for $p = 1, \ldots, |V|$ **do**
  Let $r_q \in \mathcal{R}_T$ be the node such that $\pi(v_p) = r_q$;
  $\boldsymbol{S}_{p,q} \leftarrow 1$;
return *Sink matrix* $\boldsymbol{S}$

The construction is topologically valid by the following result.

Theorem 3. *The discrete vector field V_f produced by Algorithm 1 is strictly acyclic.*

Induced quotient adjacency. Each gradient pair $(v, \{u, v\})$ is resolved immediately by identifying v with u, inducing a vertex map $\pi : V \to \mathcal{R}_T$. Let $\mathcal{R}_T = \{r_1, \ldots, r_k\}$ denote the resulting representative set. The Sink Matrix $\boldsymbol{S} \in \{0, 1\}^{n \times k}$ is the matrix representation of this identification, with

$$S_{p,q} = 1 \quad \Longleftrightarrow \quad \pi(v_p) = r_q.$$

The graph obtained by transporting the original edge set through these identifications has adjacency support

$$\mathsf{Pool}_{\mathrm{adj}}(\boldsymbol{A}) = \mathbf{1}[\boldsymbol{S}^\top \boldsymbol{A} \boldsymbol{S} > 0].$$

Indeed, $(\mathsf{Pool}_{\mathrm{adj}}(\boldsymbol{A}))_{cd} = 1$ if and only if there exists an original edge $(u, v) \in E$ such that $\pi(u) = r_c$ and $\pi(v) = r_d$. Thus, the pooled graph is the quotient graph induced by the same sequence of elementary vertex identifications used by the collapse procedure. In particular, an original edge becomes a self-loop whenever its two endpoints are identified with the same representative (more details are given in Appendix B.1).

For the feature space, we use a mean-pooling mechanism via the formula

$$\mathsf{Pool}_{\mathrm{feat}}(\boldsymbol{H}) = \boldsymbol{\Delta}_S^{-1} \boldsymbol{S}^\top \boldsymbol{H}$$

where $\mathbf{\Delta}_S = \mathrm{diag}(\boldsymbol{S}^\top \mathbf{1}_n)$ and $\mathbf{1}_n \in \mathbb{R}^n$ is the all-ones vector. Analogously, to restore the features to their original spatial dimension, we use the unpooling formula:

$$\mathsf{Unpool}\left(\boldsymbol{H}_{\mathrm{pool}}\right) = \boldsymbol{S}\boldsymbol{H}_{\mathrm{pool}}.$$

At each valid Forman–Lefschetz graph reduction, the resulting Sink Matrix $\boldsymbol{S} \in \{0,1\}^{n\times k}$ contains exactly one nonzero entry per row. Hence, it need not be materialized explicitly. Instead, we represent $\boldsymbol{S}$ by a $1 \times |V|$ index vector $\boldsymbol{s}$, where $s_p = q$ denotes the unique nonzero entry $S_{p,q} = 1$. This representation reduces the storage requirement from $\mathcal{O}(nk)$ for a dense matrix to $\mathcal{O}(n)$ and avoids the overhead of explicitly constructing a sparse matrix. In particular, feature unpooling can be implemented directly as

$$(\mathsf{Unpool}(\boldsymbol{H}))_{p:} = \boldsymbol{H}_{\mathrm{coarse},\, s_p},$$

while pooling can be implemented using indexed aggregation over the assignment vector $\boldsymbol{s}$. These operations are highly parallelizable and admit efficient GPU implementations.

Time complexity. Let $n = |V|$, $m = |E|$, d denote the node feature dimension, and T the number of pointer-jumping rounds. The construction of the Sink Matrix requires $\mathcal{O}(n \log n + m + Tn)$ time: vertex ordering costs $\mathcal{O}(n \log n)$, downhill-parent construction costs $\mathcal{O}(n + m)$, and the T pointer-jumping rounds cost $\mathcal{O}(Tn)$. Once the sink assignment has been constructed, feature pooling is implemented by indexed aggregation and costs $\mathcal{O}(nd)$, while constructing the quotient adjacency requires $\mathcal{O}(m)$ sparse edge processing. Thus, the complete collapse-and-pooling operation costs

$$\mathcal{O}(n \log n + m + Tn + nd).$$

For sparse graphs ($m = \mathcal{O}(n)$) and fixed T and feature dimension d, this reduces to $\mathcal{O}(n \log n)$.

3 Implementation Details

3.1 The Architecture: Empool Pipeline

Inspired by the Graph U-Net framework (Gao & Ji, 2019), Empool employs a hierarchical encoder–decoder architecture. Within the encoder, each layer executes a sequential cycle. First, message passing updates the node representations,

$$\boldsymbol{H}^{(\ell)}_{\mathrm{skip}} = \mathsf{MPNN}^{(\ell)}_{\mathrm{enc}}\left(\boldsymbol{H}^{(\ell)}, \boldsymbol{A}^{(\ell)}; \Theta\right).$$

These updated representations are then processed by the Emhead, a multi-layer perceptron (MLP) that outputs a learned vertex scalar field $\boldsymbol{f}^{(\ell)} \in \mathbb{R}^{n_\ell}$. The scalar field induces the discrete gradient and the corresponding sink assignments, from which the Sink Matrix $\boldsymbol{S}^{(\ell)}$ is constructed. Rather than materializing this matrix explicitly, its one-hot assignments are stored in the Sate Stack, along with the coarsened graph representation and the skip features to enable unpooling in the decoder phase:

$$\boldsymbol{A}^{(\ell+1)} = \mathsf{Pool}_{\mathrm{adj}}\left(\boldsymbol{A}^{(\ell)}\right), \qquad \boldsymbol{H}^{(\ell+1)} = \mathsf{Pool}_{\mathrm{feat}}\left(\boldsymbol{H}^{(\ell)}_{\mathrm{skip}}\right).$$

Repeating this cycle for L encoder stages yields a progressively coarsened graph and a compressed representation at the bottleneck. For graph-level tasks, the bottleneck node representations are aggregated using a permutation-invariant global readout and passed to a task-specific classifier.

For node- or edge-level tasks, the decoder progressively restores the original graph resolution. At decoder stage ℓ, the coarse representation is first unpooled using the sink assignments cached during the corresponding encoder stage:

$$\widehat{\boldsymbol{H}}^{(\ell)} = \boldsymbol{S}^{(\ell)}\widetilde{\boldsymbol{H}}^{(\ell+1)}.$$

The unpooled representation is then concatenated with the corresponding encoder skip features ($\boldsymbol{H}^{(\ell)}_{\mathrm{cat}} = \left[\widehat{\boldsymbol{H}}^{(\ell)} \,\|\, \boldsymbol{H}^{(\ell)}_{\mathrm{skip}}\right]$) and processed by a decoder message-passing layer on the restored fine graph:

$$\widetilde{\boldsymbol{H}}^{(\ell)} = \mathsf{MPNN}^{(\ell)}_{\mathrm{dec}}\left(\left[\widehat{\boldsymbol{H}}^{(\ell)} \,\|\, \boldsymbol{H}^{(\ell)}_{\mathrm{skip}}\right], \boldsymbol{A}^{(\ell)}; \Theta\right).$$

Here, $\|$ denotes feature-wise concatenation. In the implementation, the unpooling operation is realized directly through the cached sink-assignment vector, without explicitly constructing $\boldsymbol{S}^{(\ell)}$. The

Sink Stack provides a natural last-in-first-out (LIFO) mechanism for retrieving these assignments, ensuring that each decoder stage uses the Sink Matrix and fine-grained graph structure associated with its corresponding encoder stage. After the final decoder stage, the resulting representations are passed to the task-specific node or edge classifier (Figure 1).

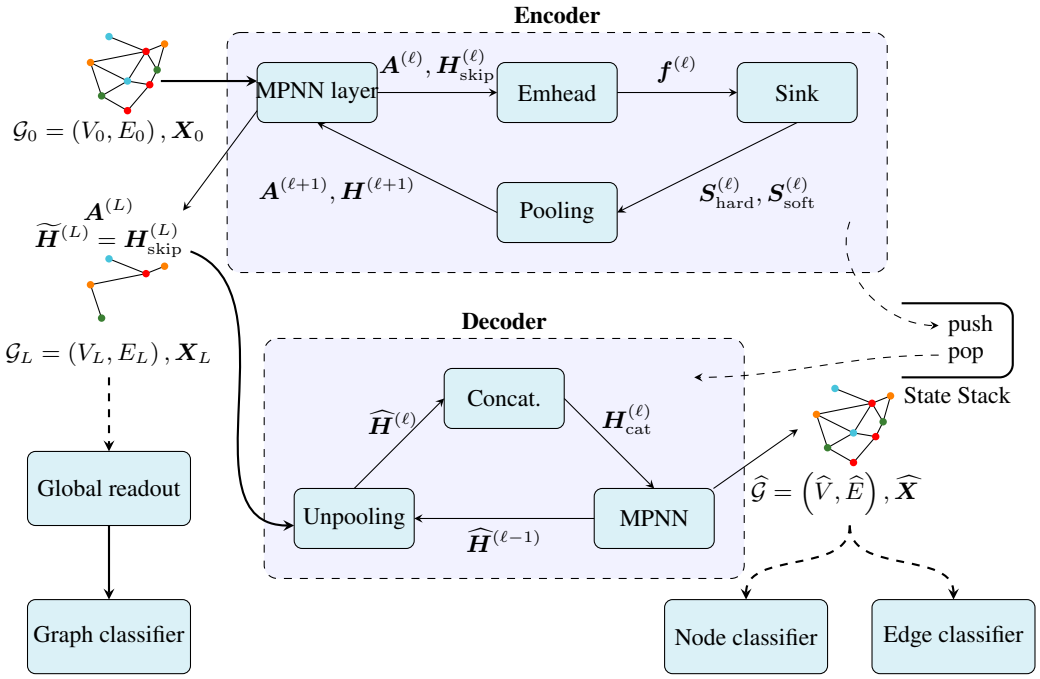

Figure 1: Diagram of the Empool architecture pipeline.

3.2 Generation of the Morse vector, Regularization and the Differentiability of Empool:

As mentioned previously, a scalar Morse function on vertices is essential to the topological collapse and determines the semantic relevance of nodes with respect to their local neighborhoods. Therefore, at each layer ℓ, we compute a Morse vector $\boldsymbol{f}^{(\ell)} = (f(v_1), \ldots, f(v_n))^\top$ over the nodes of the current graph using a multi-layer perceptron that takes the current feature matrix $\boldsymbol{H}^{(\ell)}$ as input:

$$\boldsymbol{f}^{(\ell)} = \mathsf{MLP}_\ell \left(\boldsymbol{H}^{(\ell)}; \Theta \right) \tag{1}$$

These parameters are learned jointly with the GNN parameters; thus, the model can learn the optimal Morse vector that maximizes accuracy without requiring an explicitly defined formula or a manual compression ratio.

The hard Sink Matrix $\boldsymbol{S}_{\text{hard}}$ is generated through non-differentiable argmax operations, preventing gradients from reaching the learned Morse field. We therefore introduce a localized Straight-Through Estimator (STE).

For each vertex v_i, let

$$\mathcal{Q}_i = \left\{ \pi^{2^T}(u) \mid u \in \mathcal{N}(v_i) \cup \{v_i\} \right\}$$

be its candidate representative set *i.e.* the set of vertices that v_i and its neighbors are *susceptible* to collapse into. For $c \in \mathcal{Q}_i$, define a score

$$S_i(c) = (f(v_i) - f(c))^2,$$

and

$$p_i(c) = \frac{\exp(-S_i(c)/\tau)}{\sum_{c' \in \mathcal{Q}_i} \exp(-S_i(c')/\tau)}.$$

This yields a sparse assignment with memory complexity $\mathcal{O}(n \max_i |\mathcal{Q}_i|)$ rather than $\mathcal{O}(nk)$. This is especially efficient in terms of memory given that $|\mathcal{Q}_i| \leq \Delta$, where Δ is the maximum degree of the graph. During the forward pass we retain the exact hard topological assignment, while the differentiable soft assignment supplies the surrogate gradient. We demonstrate that this surrogate matrix is indeed a "faithful" approximation to the original hard assignment via the following Theorem 4:

Theorem 4. *Let $\boldsymbol{S}_{\text{hard}}$ be the assignment matrix generated by the discrete Forman–Lefschetz collapse, and let $\boldsymbol{S}_{\text{soft}}^{(\tau)}$ be the continuous surrogate parameterized by temperature τ. For each vertex v_i, let c_i^{hard} denote its hard sink and let $\mathcal{Q}_i$ be its candidate sink set. Assume that the hard sink is the unique minimizer of the surrogate score, i.e.,*

$$\left(f(v_i) - f\left(c_i^{\text{hard}}\right)\right)^2 < \left(f(v_i) - f(c)\right)^2 \qquad \forall c \in \mathcal{Q}_i \setminus \{c_i^{\text{hard}}\}.$$

Then

$$\lim_{\tau \to 0^+} \boldsymbol{S}_{\text{soft}}^{(\tau)} = \boldsymbol{S}_{\text{hard}}.$$

To encourage a meaningful Morse field, we use the following regularization:

$$\mathcal{L}_{\text{reg}} = \xi\, \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f} + \frac{\lambda}{\text{Var}(\boldsymbol{f})}$$

where

$$\text{Var}(\boldsymbol{f}) = \frac{1}{n} \sum_{i=1}^{n} \left(f_i - \widehat{\boldsymbol{f}}\right)^2, \qquad \widehat{\boldsymbol{f}} = \frac{1}{n} \sum_{i=1}^{n} f_i.$$

The regularizer's influence is hyperparameterized by $\xi, \lambda \geq 0$. The first term penalizes local variation across edges, while the second prevents degenerate constant fields (see Appendix B.2 for details).

4 Benchmarking Setup

We evaluate Empool across node- and graph-level prediction tasks and compare its predictive performance against established message-passing and graph-pooling baselines.

4.1 Datasets, Baselines, and Evaluation Metrics

We consider six benchmark datasets. Cora, PubMed, and ogbn-arxiv are used for node prediction, while MUTAG, COLLAB, and ogbg-molhiv are used for graph prediction.

For node prediction, Empool is compared against Graph Convolutional Networks (GCN) (Kipf & Welling, 2016), Graph Attention Networks (GAT) (Velickovic et al., 2017), and GraphSAGE (Hamilton et al., 2017). For graph prediction, we compare against Graph Isomorphism Networks (GIN) (Xu et al., 2018), MinCutPool (Bianchi et al., 2019), and StructPool (Yuan & Ji, 2020).

For Cora, PubMed, ogbn-arxiv, MUTAG, and COLLAB, predictive performance is reported as classification accuracy. For ogbg-molhiv, we report ROC-AUC. In all cases, the test set is evaluated only after selecting a model checkpoint using validation data.

For MUTAG and COLLAB, we use a fixed 80/10/10 train/validation/test split that is shared across all models and random seeds. Thus, the reported standard deviation on these datasets measures variation due to model initialization, minibatch ordering, dropout, and optimization, rather than variation across dataset splits.

4.2 Benchmarking Hardware and Software Setup

All benchmark experiments are executed on NVIDIA A100 80GB GPUs. Complete details of the software environment, including Python, PyTorch, and PyTorch Geometric (PyG) versions, are provided in Table 2.

4.3 Benchmarking Methodology and Hyperparameter Selection

Each architecture is tuned independently on each dataset using validation performance only. Consequently, hyperparameters such as learning rate, dropout probability, hidden dimensionality, and pooling configuration may differ across architectures and datasets. The exact configuration used for every model–dataset pair is reported in Appendix C.2.

To ensure reproducibility, we predefine a deterministic random-seed schedule generated from a master seed using SHA-256 and use the same schedule for all compared methods. We report mean $\pm$ standard deviation over 100 independent training runs for Cora, PubMed, MUTAG, and COLLAB, and over 10 runs for ogbn-arxiv and ogbg-molhiv.

All models use early stopping with a patience of 100 epochs. Model selection is performed exclusively using the validation set; the test set is evaluated only after the selected checkpoint has been fixed.

5 Results

Table 1 summarizes predictive performance across the six benchmarks. Bold and underlined entries indicate the highest and second-highest empirical means, respectively. These markings are descriptive rankings of the observed means and should not be interpreted as evidence of statistical significance. Datasets marked with a dagger (†) were symmetrized by treating their edge sets as undirected prior to training.

Table 1: Predictive performance across node- and graph-level benchmarks, reported as mean $\pm$ standard deviation over independent training runs. Bold and underlined values indicate the highest and second-highest empirical means, respectively, and do not imply statistical significance.

Task	Dataset	GCN	GAT	SAGE	Empool (ours)
Node Prediction	Cora (%)	$\underline{82.29 \pm 0.57}$	$\mathbf{82.77 \pm 0.73}$	80.66 ± 0.62	82.26 ± 1.09
	PubMed (%)	$\mathbf{79.32 \pm 0.46}$	$\underline{78.35 \pm 0.49}$	77.43 ± 0.42	78.34 ± 0.96
	ogbn-arxiv† (%)	$\mathbf{71.67 \pm 0.27}$	70.82 ± 0.33	70.00 ± 0.73	$\underline{71.01 \pm 0.94}$
		GIN	**MinCutPool**	**StructPool**	**Empool (ours)**
Graph Prediction	MUTAG (%)	$\underline{89.12 \pm 2.14}$	79.17 ± 0.53	89.02 ± 0.43	$\mathbf{90.00 \pm 1.42}$
	COLLAB (%)	80.09 ± 1.45	$\underline{81.10 \pm 1.43}$	80.22 ± 0.82	$\mathbf{81.11 \pm 1.96}$
	ogbg-molhiv (ROC-AUC)	$\mathbf{0.76 \pm 0.02}$	0.74 ± 0.03	$\underline{0.75 \pm 0.02}$	$\mathbf{0.76 \pm 0.10}$

Across node-prediction benchmarks, Empool remains competitive with standard message-passing architectures. It obtains $82.26 \pm 1.09\%$ on Cora, $78.34 \pm 0.96\%$ on PubMed, and $71.01 \pm 0.94\%$ on ogbn-arxiv. In particular, its mean performance on ogbn-arxiv is second only to GCN among the evaluated models.

The graph-prediction experiments show similarly competitive behavior. Empool obtains $90.00 \pm 1.42\%$ on MUTAG and $81.11 \pm 1.96\%$ on COLLAB, corresponding to the highest empirical means among the evaluated methods on these datasets. On ogbg-molhiv, Empool reaches a mean ROC-AUC of 0.76, matching the highest rounded empirical mean, although with greater run-to-run variability than GIN.

Taken together, these results indicate that Empool can perform substantial learned graph reduction while retaining task-relevant information across both node- and graph-level prediction tasks.

6 Limitations and Future Work

First, although Empool is based on formal homology preservation under the Forman–Lefschetz reductions considered in Theorem 2, we do not establish conditions under which graph homological invariants are sufficient to guarantee improvements on downstream tasks tied to specific data semantics. In particular, there is *a priori* no analytical bound guaranteeing that topological homology alone dictates task-relevant feature learning, meaning the translation from topological reduction to

inductive utility relies on structural heuristics. Establishing a formal bridge between predictive performance and graph homology remains an important direction for future work.

Second, our current empirical evaluation is confined to node- and graph-level classification tasks, omitting link-level prediction. Additionally, our experiments are presently restricted to small- and medium-scale benchmarks. Extending the evaluation framework to incorporate large-scale graphs and link prediction tasks represents an immediate priority for future work.

Third, Empool trades wall-clock runtime for structural preservation and memory benefits. The primary overhead stems from the iterative execution of message passing, Forman–Lefschetz reduction, and dynamic graph restructuring across multi-scale encoder levels. Developing custom, optimized GPU kernels for sink assignment, gradient construction, and sparse pooling represents an important avenue for future work to mitigate this overhead.

Fourth, while Empool demonstrates strong competitive performance – particularly on graph-level classification tasks where global topological awareness is paramount – certain limitations remain. As observed in our empirical evaluation, Empool exhibits higher variance across random seeds compared to traditional message-passing or continuous pooling baselines (e.g., on ogbg-molhiv). We attribute this instability to the optimization landscape of discrete relaxations: mapping a continuous vertex scalar field to a discrete Forman–Lefschetz reduction introduces non-smooth transitions that can make training sensitive to weight initializations. Investigating smoother continuous relaxations of discrete Morse functions, variance-reduction regularizers, or deterministic cell-selection strategies represents an important direction for future work.

Finally, Empool introduces several specialized hyperparameters governing the discrete Morse matching process (*e.g.*, surrogate gradients, backtracking iterations, and topological regularization weights). While these parameters offer fine-grained control over the topological collapse, they also increase the hyperparameter search space compared to standard, unpooled GNN architectures.

7 Conclusion

In this work, we introduced *Empool*, a principled graph pooling framework grounded in discrete Morse theory and Lefschetz complex topology. By formalizing graph reduction through a Forman–Lefschetz collapse, Empool progressively condenses redundant structural topology into a compact set of critical cells while rigorously preserving homology. Our approach bridges discrete topology and deep learning via an end-to-end differentiable relaxation, addressing structural information bottlenecks without sacrificing global manifold properties. Extensive benchmarking across diverse node- and graph-level prediction tasks demonstrates that Empool achieves highly competitive and robust performance relative to established message-passing and pooling architectures. Future work will focus on scaling the framework to massive graphs, refining the optimization landscape to mitigate training variance, and exploring broader topological priors for representation learning.

AI Use Statement

In this work, we used generative AI tools to provide feedback on research methodology and experiments. We have not used generative AI tools for other tasks with required disclosure, such as generating synthetic datasets, developing theoretical or conceptual frameworks, formulating mathematical claims, or assisting in the writing of proofs. Additionally, we used generative AI tools to create or edit software code, source information, and edit the research paper to improve readability.

We have reviewed all AI-assisted work. Specifically, the feedback on research methodology and experiments was cross-referenced against existing literature and verified in consultation with department faculty. Furthermore, all AI-assisted software code was thoroughly tested and executed on a remote HPC cluster. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work does not involve human subjects, human-subject experimentation, or the collection of personally identifiable or otherwise private data. All experiments are conducted using publicly available

benchmark datasets for graph representation learning. We do not introduce new data collection procedures or applications that raise additional ethical, privacy, or security concerns. The author has reviewed the ICLR Code of Ethics and conducted the work in accordance with its principles.

Reproducibility statement

To ensure reproducibility, the authors have included in the appendix all proofs of the mathematical statements made in the main body of the paper. Furthermore, to ensure reproducibility of empirical results, the source code of the architecture is provided as supplementary material, and the fine-tuning details and processing of the datasets (if applicable) are presented in the appropriate appendix sections.

References

Muhammed Fatih Balin and Ümit Çatalyürek. Layer-neighbor sampling—defusing neighborhood explosion in gnns. *Advances in Neural Information Processing Systems*, 36:25819–25836, 2023.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut pooling in graph neural networks. 2019.

Yuzhou Chen and Yulia R Gel. Topological pooling on graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 7096–7103, 2023.

Robin Forman. *Morse Theory for Cell Complexes*. American Mathematical Society, 1998. ISBN 978-0821808603.

Robin Forman. A user's guide to discrete morse theory. *Séminaire Lotharingien de Combinatoire [electronic only]*, 48:B48c–35, 2002.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Jhony H Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D Malliaros. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 566–576, 2023.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

Michael Joswig and Marc E Pfetsch. Computing optimal morse matchings. *SIAM Journal on Discrete Mathematics*, 20(1):11–25, 2006.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Dmitry N Kozlov. *Organized collapse: an introduction to discrete Morse theory*, volume 207. American mathematical society, 2021.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. pmlr, 2019.

Marian Mrozek and Thomas Wanner. *Connection Matrices in Combinatorial Topological Dynamics*. Springer Nature, 2025.

Vanessa Robins, Peter John Wood, and Adrian P Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Transactions on pattern analysis and machine intelligence*, 33(8):1646–1658, 2011.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *CoRR*, abs/1806.08804, 2018. URL http://arxiv.org/abs/1806.08804.

Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *International conference on learning representations*, 2020.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems*, 32, 2019.

A Definitions, Preliminaries and Notations

Throughout this paper, we adopt the following definitions, notations and preliminary results. Some of them may be classified as "standard". However, we make sure to appropriately cite the less common ones.

A.1 Lefschetz Complexes

Definition 4 (Lefschetz Complex (Mrozek & Wanner, 2025)). *Let $(R, +, \cdot)$ be a ring and X a set. A Lefschetz complex over R is a triplet $(X, \mathcal{K}, \kappa)$ such that:*

- *X is a finite set.*
- *$\mathcal{K} = (\mathcal{K}_k)_{0 \leq k \leq n}$ is a partition of X.*
- *κ is a map $\kappa : X \times X \longrightarrow R$ called the **incidence coefficient map** such that*

$$\forall x, y \in X, \quad \kappa(x, y) \neq 0 \Longrightarrow \exists k \in \{1, \cdots, n\} / \quad x \in \mathcal{K}_k \text{ and } y \in \mathcal{K}_{k-1}$$

and

$$\forall x, y \in X, \quad \sum_{z \in X} \kappa(x, z)\kappa(z, y) = 0.$$

*The elements of X are referred to as **cells**, $\kappa(x, y)$ is called the **incidence coefficient** of x and y.*

*For every cell x, there exists one, and only one index k such that $x \in \mathcal{K}_k$, called the **dimension** of the cell.*

*A Lefschetz complex is said to be **regular** if for any pair of cells (x, y), $\kappa(x, y)$ is either zero or invertible in R.*

If understood from context, the Lefschetz complex will simply be denoted X.

Given a Lefschetz complex $(X, \mathcal{K}, \kappa)$, we can construct a partial order. Namely let $\prec_\kappa$ be the binary relation defined as

$$\forall x, y \in X, \quad y \prec_\kappa x \Longleftrightarrow \kappa(x, y) \neq 0.$$

When $y \prec_\kappa x$, we say that y is a **facet** of x. The reflexive and transitive closure of $\prec_\kappa$ defines a partial order on X called the **face relation** that we will be denoting $\leq_\kappa$.

A.2 Morse Functions and Gradient Fields

The framework of Forman (1998) introduced the notion of Morse function in order to assign to each cell within a complex a real-value function that, given specific conditions, can construct a gradient field.

Definition 5 (Discrete Morse Function[1]). *Let $(X, \mathcal{K}, \kappa)$ be a Lefschetz complex. A function $f : X \longrightarrow \mathbb{R}$ is called a **discrete Morse function** if for every cell $\alpha \in X$, the two following conditions hold:*

- $|\{\gamma \in X \mid \gamma \prec_\kappa \alpha \text{ and } f(\gamma) \geq f(\alpha)\}| \leq 1$,
- $|\{\beta \in X \mid \alpha \prec_\kappa \beta \text{ and } f(\beta) \leq f(\alpha)\}| \leq 1$.

The two sets will respectively be denoted as $M_f^-(\alpha)$ and $M_f^+(\alpha)$.

The first condition dictates that the faces γ of α (*i.e.* cells γ such that $\gamma \prec_\kappa \alpha$) must all have Morse function values strictly less than α, with *possibly* one exception.

Analogously, the second dictates that the cells whose one of the faces is α (*i.e.* cells β such that $\alpha \prec_\kappa \beta$) must all have Morse function values strictly greater than α, with *possibly* one exception.

[1] While the notion of a discrete Morse function is broadly defined on simplicial complexes and chain complexes, we focus our definition here particularly on Lefschetz complexes.

Heuristically, the conditions above are hence equivalent to saying that f is an "order-preserving function that eventually allows one exception per cell".

A fundamental correspondence exists between discrete Morse functions and discrete vector fields within the complex. We recall that:

Definition 6 (Discrete Vector Field). *A **discrete vector field** V on X is a collection of pairs of cells (α, β) such that:*

- $\alpha \prec_\kappa \beta$ *(i.e., α is a facet of β).*
- *Every cell in X belongs to at most one pair in V.*

*If V is acyclic, we say that V is a **discrete gradient vector field**.*

If V is constructed from a discrete Morse function f by pairing cells where $f(\beta) \leq f(\alpha)$, we call it the vector field induced by f and denote it V_f.

The first fundamental theorem in this framework is the following:

Theorem 5. *Given a discrete Morse function f, V_f is acyclic.*

Conversely, given a discrete gradient field V, there exists a discrete Morse function f such that $V = V_f$ if and only if V is acyclic.

A.3 Criticality and the Collapse Theorem

The collapse mechanism on which our whole model is based on is due to Forman's theorem: "every simplicial complex of finite dimension can be collapsed into its critical cells in a homology-preserving mechanism" (Forman, 2002). First,

Definition 7 (Critical Cell). *A cell $\alpha \in X$ is called **critical** with respect to a discrete Morse function f if it does not belong to any pair in the induced gradient vector field V_f. Equivalently, α is critical if:*

- $\forall \gamma \prec_\kappa \alpha, \quad f(\gamma) < f(\alpha)$,
- $\forall \beta \succ_\kappa \alpha, \quad f(\beta) > f(\alpha)$

which in turn is equivalent to $\left|M_f^+(\alpha)\right| = \left|M_f^-(\alpha)\right| = 0$.

Following up the heuristic from Definition 5, a critical cell is one that allows "no exceptions" with respect to the Morse function.

The core theorem regarding collapsibility and homology preservation, due to Forman, can be stated in several ways. We adopt the following formulation by Kozlov (2021):

Theorem 6 (Forman's Collapse Theorem). *Suppose V is a discrete gradient vector field on a simplicial complex K. Then there exists a cell complex X such that*

- *for each $d \in \mathbb{N}$, the number of cells of dimension d in X is equal to the number of critical d-cells with respect to V in K.*
- *there is a homotopy equivalence $K \cong X$.*

B Proofs of Theorems

B.1 Proofs of Section 2

We establish the proof for Theorem 1.

Proof. (Lefschetz Complex of a Graph) The set X is finite since it is the union of two finite sets. By definition, the union of $\mathcal{K}_0$ and $\mathcal{K}_1$ is the set X. Furthermore, the two sets are disjoint since they respectively contain singletons and pairs (each edge is represented by the pair of its endpoints, whether the graph is directed or

not). $(\mathcal{K}_0, \mathcal{K}_1)$ is indeed a partition of X.

Now, let $x, y \in X$. Assume $\kappa(x, y) \neq 0$. Hence, $\kappa(x, y) = 1$ which by definition implies $x \in \mathcal{K}_1$ and $y \in \mathcal{K}_0$.

Finally, let $x, y \in X$. Let us establish $\sum_{z \in X} \kappa(x, z)\kappa(z, y) = 0$. The result is obvious if $x \in \mathcal{K}_0$ or $y \in \mathcal{K}_1$. From now on, assume that $x \in \mathcal{K}_1$ and $y \in \mathcal{K}_0$. Let $z \in X$. This means either $z \in \mathcal{K}_0$ or $z \in \mathcal{K}_1$, yielding respectively $\kappa(z, y) = 0$ and $\kappa(x, z) = 0$. In both scenarios, the product $\kappa(x, z)\kappa(z, y)$ is zero, hence proving the equality. ■

Next is the corrected formulation and proof for Theorem 2:

Proof. (Forman-Lefschetz Reduction of a Graph) Fix a graph $\mathcal{G}$ and its induced Lefschetz complex $\mathscr{L}_\mathcal{G}$, and let V_f be a discrete gradient vector field on $\mathscr{L}_\mathcal{G}$. By Forman's Collapse Theorem (Theorem 6), a sequence of elementary collapses along an acyclic discrete gradient vector field reduces the complex to a homology-equivalent subcomplex consisting of the critical cells. In the context of graph pooling via Empool, these elementary collapses correspond to identifying non-critical nodes and edges with their downstream basins of attraction along the gradient flow paths. The resulting quotient graph $\mathcal{G}_{\text{Forman}}$ inherits the structural connectivity and preserves the core topological invariants through these homology-preserving cell identifications. ■

Now, we prove the strict acyclicity theorem (Theorem 3):

Proof. We first prove that V_f is a valid discrete vector field on $\mathscr{L}_\mathcal{G}$. By definition, we must show that every cell in $\mathscr{L}_\mathcal{G}$ belongs to at most one pair in V_f.

Case 1: 0-cells (Vertices). Because $\mathscr{L}_\mathcal{G}$ contains no cells of negative dimension, a vertex $v \in V$ can only appear as the first component of a pair (v, e). In Algorithm 1, the outer loop iterates over each vertex v_i exactly once. If v_i is not critical, it is assigned to a unique edge $\{v_i, -\nabla f(v_i)\}$. Thus, every 0-cell belongs to at most one pair in V_f.

Case 2: 1-cells (Edges). Since $\mathscr{L}_\mathcal{G}$ has maximum dimension 1, no 2-cells exist, meaning an edge $e = \{u, v\} \in E$ cannot appear as the first component of any pair. Thus, e can only appear as the second component in a pair of the form (w, e) with $w \in \{u, v\}$.

Suppose, for the sake of contradiction, that e belongs to two distinct pairs in V_f. This requires both $(u, e) \in V_f$ and $(v, e) \in V_f$. By construction of the lower-star gradient field, $(u, e) \in V_f \implies \Delta f(u, v) > 0$, which implies $f(u) > f(v)$. Symmetric reasoning for $(v, e) \in V_f$ yields $f(v) > f(u)$. Combining these inequalities gives $f(u) > f(u)$, a contradiction.

Hence, every 1-cell appears in at most one pair in V_f. Consequently, V_f satisfies the conditions of a discrete vector field on $\mathscr{L}_\mathcal{G}$.

Now, we prove that V_f is strictly acyclic. Assume otherwise, *i.e.* V_f contains a closed V-path. This means that there exists $k \geq 1$ and two sequences of vertices $(v_0, \cdots, v_k)$ and of edges $(e_0, \cdots, e_{k-1})$ such that for every $j \in \{0, \cdots, k-1\}$:

$$\begin{cases} (v_j, e_j) \in V_f, & \text{(2a)} \\ v_{j+1} \prec e_j, & \text{(2b)} \\ v_j \neq v_{j+1}, & \text{(2c)} \\ v_0 = v_k. & \text{(2d)} \end{cases}$$

Let $j \in \{0, \cdots, k-1\}$. By Algorithm 1, $(v_j, e_j) \in V_f$ implies $e_j = \{v_j, -\nabla f(v_j)\}$ and $\Delta f(v_j, -\nabla f(v_j)) > 0$. However, by Eq. (2b), v_{j+1} is a facet of e_j. Hence, either $v_{j+1} = -\nabla f(v_j)$ or $v_{j+1} = v_j$. However, the latter is excluded given Eq. (2c). This yields that $f(v_j) > f(v_{j+1})$. By chaining these inequalities together, we obtain:

$$f(v_k) < f(v_{k-1}) < \cdots < f(v_1) < f(v_0)$$

and, crucially, $f(v_k) < f(v_0)$. However, $v_0 = v_k$, and by consequence, $f(v_0) = f(v_k)$. This yields a contradiction. We have proven that V_f contains no closed V-paths, hence is acyclic. ■

Pooled adjacency induced by the Sink Matrix. Let $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ denote the adjacency matrix of the current graph and let $\boldsymbol{S} \in \{0, 1\}^{n \times k}$ denote the Sink Matrix. Each row of $\boldsymbol{S}$ contains exactly

one nonzero entry, so for every original vertex v_r there exists a unique reduced-node index $c_r \in \{1, \dots, k\}$ such that

$$S_{r,c_r} = 1.$$

Thus c_r is the representative to which v_r is assigned.

The matrix product $\boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S}$ aggregates edges according to their endpoint assignments. In particular, for $i, j \in \{1, \dots, k\}$,

$$\begin{aligned}(\boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S})_{ij} &= \sum_{r=1}^{n}\sum_{s=1}^{n} S_{r,i}\, A_{rs}\, S_{s,j} \\ &= \sum_{r=1}^{n}\sum_{s=1}^{n} \mathbf{1}[c_r = i]\, A_{rs}\, \mathbf{1}[c_s = j].\end{aligned}$$

For a simple unweighted graph, where $A_{rs} = \mathbf{1}[(v_r, v_s) \in E]$, this is precisely the number of original edges whose source is assigned to reduced node i and whose target is assigned to reduced node j:

$$(\boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S})_{ij} = \left|\{(v_r, v_s) \in E : c_r = i,\ c_s = j\}\right|.$$

Consequently, the simple unweighted quotient graph induced by the assignment is represented by

$$\boldsymbol{A}_{\text{pool}} = \mathbf{1}\left[\boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S} > 0\right],$$

where the indicator is applied entry-wise.

In the present implementation, this latter operation is realized implicitly in sparse form: endpoints of each original edge are mapped to their sink indices, and duplicate reduced edges are merged by `coalesce`. Hence the implementation retains only the support of $\boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S}$, rather than its edge multiplicities. With `remove_self_loops=True`, entries on the diagonal are removed as well. Self-loop handling is deliberately decoupled from the pooling operation; self-loops can instead be introduced by the subsequent message-passing layer (*e.g.*, GCN) according to the requirements of the chosen operator.

B.2 Proofs of Section 3

We begin by proving the convergence of the soft surrogate (Theorem 4)

Proof. Let $\tau > 0$, and fix $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, k\}$. If $c_j \notin \mathcal{Q}_i$, then by definition

$$\left(S^{(\tau)}_{\text{soft}}\right)_{i,j} = 0$$

for every τ, and since the hard assignment also assigns v_i to a single sink, the desired convergence holds for this entry.

Now suppose that $c_j \in \mathcal{Q}_i$. We have

$$\begin{aligned}\left(S^{(\tau)}_{\text{soft}}\right)_{i,j} &= p_i\left(c_j\right) \\ &= \frac{\exp\left(-S_i\left(c_j\right)/\tau\right)}{\sum_{c \in \mathcal{Q}_i} \exp\left(-S_i(c)/\tau\right)} \\ &= \frac{\exp\left(-\dfrac{S_i\left(c_j\right) - S_i\left(c_i^{\text{hard}}\right)}{\tau}\right)}{1 + \sum\limits_{\substack{c \in \mathcal{Q}_i \\ c \neq c_i^{\text{hard}}}} \exp\left(-\dfrac{S_i(c) - S_i\left(c_i^{\text{hard}}\right)}{\tau}\right)}.\end{aligned}$$

By assumption, the hard sink c_i^{hard} is the unique minimizer of S_i on $\mathcal{Q}_i$. Hence

$$S_i(c) - S_i\left(c_i^{\text{hard}}\right) > 0 \qquad \forall c \in \mathcal{Q}_i \setminus \{c_i^{\text{hard}}\}.$$

Since $\mathcal{Q}_i$ is finite, every term in the sum satisfies

$$\exp\left(-\frac{S_i(c) - S_i\left(c_i^{\text{hard}}\right)}{\tau}\right) \longrightarrow 0 \qquad (\tau \to 0^+),$$

and therefore

$$\sum_{\substack{c \in \mathcal{Q}_i \\ c \neq c_i^{\text{hard}}}} \exp\left(-\frac{S_i(c) - S_i\left(c_i^{\text{hard}}\right)}{\tau}\right) \longrightarrow 0.$$

Thus,

$$\left(S_{\text{soft}}^{(\tau)}\right)_{i,j} \longrightarrow \begin{cases} 1, & c_j = c_i^{\text{hard}}, \\ 0, & c_j \neq c_i^{\text{hard}}. \end{cases}$$

The right-hand side is exactly $(S_{\text{hard}})_{i,j}$. Hence every entry converges:

$$\lim_{\tau \to 0^+} \left(S_{\text{soft}}^{(\tau)}\right)_{i,j} = (S_{\text{hard}})_{i,j}\,.$$

Since the matrices are finite-dimensional, entrywise convergence implies convergence in any matrix norm, and therefore

$$\lim_{\tau \to 0^+} S_{\text{soft}}^{(\tau)} = S_{\text{hard}}.$$

■

Note on the regularization term. The quadratic form $\boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f}$ measures the variation of the Morse values across adjacent vertices. Recall that

$$\boldsymbol{f} = \begin{pmatrix} f(v_1) \\ \vdots \\ f(v_n) \end{pmatrix},$$

and let $\boldsymbol{A}$ denote the symmetric adjacency matrix of the graph and $\boldsymbol{D}$ its degree matrix. Then

$$\begin{aligned}
\sum_{\{u,v\} \in E} (f(u) - f(v))^2 &= \sum_{1 \le i < j \le n} A_{ij} \left(f(v_i) - f(v_j)\right)^2 \\
&= \frac{1}{2} \sum_{i,j=1}^{n} A_{ij} \left(f(v_i) - f(v_j)\right)^2 \\
&= \frac{1}{2} \sum_{i,j=1}^{n} A_{ij} f(v_i)^2 + \frac{1}{2} \sum_{i,j=1}^{n} A_{ij} f(v_j)^2 - \sum_{i,j=1}^{n} A_{ij} f(v_i) f(v_j) \\
&= \sum_{i=1}^{n} d_i f(v_i)^2 - \boldsymbol{f}^\top \boldsymbol{A} \boldsymbol{f} \\
&= \boldsymbol{f}^\top \boldsymbol{D} \boldsymbol{f} - \boldsymbol{f}^\top \boldsymbol{A} \boldsymbol{f} \\
&= \boldsymbol{f}^\top (\boldsymbol{D} - \boldsymbol{A}) \boldsymbol{f} \\
&= \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f}.
\end{aligned}$$

Thus, $\boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f}$ penalizes squared differences in Morse values between adjacent vertices, encouraging a locally smooth Morse field. This discourages abrupt variations of f across edges and can thereby promote more stable gradient and sink assignments.

The second regularization term,

$$\frac{1}{\operatorname{Var}(\boldsymbol{f})},$$

prevents the Morse field from collapsing to a constant-valued solution. Indeed, for a connected graph, the smoothness term $\boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f}$ is minimized when all Morse values are equal, in which case $\operatorname{Var}(\boldsymbol{f}) = 0$. The reciprocal-variance term therefore penalizes this degenerate solution.

The two terms consequently impose competing objectives: the Laplacian quadratic form favors smoothness of the Morse field, while the reciprocal-variance term prevents its variation from collapsing completely. Their combination is intended to maintain a non-trivial yet locally coherent Morse function.

C EXPERIMENTAL SETUP

C.1 HARDWARE AND SOFTWARE ENVIRONMENT

Next, we present detailed information about software and hardware settings of our benchmark (Table 2):

Table 2: Hardware environment and software runtime specifications.

Component	**Specification**
GPU Accelerator	NVIDIA A100-SXM4-80GB
Python Version	Python `3.10.21`
Deep Learning Framework	PyTorch `2.5.1` + CUDA `12.1`
Graph Processing Library	PyTorch Geometric (PyG) `2.8.0`
OGB Version	ogb `1.3.6`

C.2 HYPERPARAMETERS

In this section, we present the hyperparameters that were used in our experiments across each model-dataset pair:

First, we detail hyperparameters for baseline models on node-prediction tasks (Table 3):

Table 3: Hyperparameter configuration for node-prediction tasks

Model	**Hyperparameter**	Cora	PubMed	ogbn-arxiv
GCN	Number of Layers	2	2	3
	Hidden Channels	96	128	256
	Learning Rate	5e-3	1e-2	1e-2
	Weight Decay	5e-4	5e-4	0
	Dropout	0.9	0.2	0.5
GraphSAGE	Number of Layers	2	2	3
	Hidden Channels	64	128	256
	Learning Rate	1e-2	1e-2	1e-2
	Weight Decay	5e-4	1e-3	0
	Dropout	0.8	0.4	0.5
GAT	Number of Layers	2	2	4
	Hidden Channels	64	64	256
	Learning Rate	1e-2	1e-2	1e-2
	Weight Decay	1e-3	1e-4	0
	Dropout	0.6	0.5	0.5
	Attention Heads	8	8	16
	Output Attention Heads	1	8	8

Next, we detail hyperparameters for baseline models on graph-prediction tasks (Table 4) (the same configuration has been used for all models):

For Empool-specific hyperparameters, we use $\tau = 0.85$, pointer-jumping steps $T = 2$, $\lambda = 10^{-3}$ and $\xi = 5 \times 10^{-4}$ across all datasets. Finally, the Emhead (the MLP responsible for generating the vertex Morse vector) is configured with 3 layers, 48 hidden dimensions, and a dropout probability of 0.2.

Table 4: Hyperparameter configuration for graph-prediction tasks

Hyperparameter	MUTAG	COLLAB	ogbg-molhiv
Number of Layers	5	5	5
Hidden Channels	32	64	300
Learning Rate	5e-3	1e-2	1e-2
Weight Decay	5e-4	5e-4	0
Dropout	0.5	0.5	0.5
Graph Readout	Sum Pooling		