PLOS Complex Systems

RESEARCH ARTICLE

Topology-aware GNNs under structural perturbations: Empirical robustness across domains

Jelena Losic[1*], **Charles Fanning**[2]

1 Faculty of Mathematics, University of Belgrade, Belgrade, Serbia, **2** School of Data Science and Analytics, Kennesaw State University, Kennesaw, Georgia, United States of America

* jelenal_007@yahoo.com

OPEN ACCESS

Citation: Losic J, Fanning C (2026) Topology-aware GNNs under structural perturbations: Empirical robustness across domains. PLOS Complex Syst 3(9): e0000125. https://doi.org/10.1371/journal.pcsy.0000125

Editor: Hocine Cherifi, Université de Bourgogne: Universite de Bourgogne, FRANCE

Received: February 27, 2026

Accepted: August 3, 2026

Published: September 2, 2026

Data availability statement: No new datasets were generated in this study. All analyses used the following existing public benchmark datasets from the TUDataset repository (https://chrsmrrs.github.io/datasets/docs/datasets/): MUTAG, PROTEINS, ENZYMES, NCI1, COLLAB, and REDDIT-BINARY. Each dataset can be accessed directly from that repository by name.

Abstract

Graph neural networks learn from relational structure but can be sensitive to edge-level noise. We study a hybrid graph classifier that combines a message-passing branch (Graph Isomorphism Network, GIN) with a topological branch based on extended persistence diagrams and PersLay embeddings. Training optionally uses a paper-specific hinge penalty motivated by stable persistence-diagram representations and Lipschitz regularity, which we abbreviate as "HK-inspired." On six TUDataset benchmarks, we compare five ablations (Full, GIN+PersLay, GIN + HK, GIN only, PersLay only) and measure robustness as accuracy drop under 10% random edge removal at test time, with persistence diagrams fixed from the original graphs; we also report targeted perturbations on MUTAG and PROTEINS and runtime on all six datasets. The regularized configurations reduce relative accuracy degradation where GIN-only is sensitive, particularly on smaller molecular and protein graphs, but do not consistently maximize clean or perturbed accuracy. On large social-network benchmarks, robustness differences are small and clean accuracy is the main differentiator. Adding PersLay to GIN can improve accuracy on several datasets, while PersLay alone performs worst. Training the full model increases per-epoch cost relative to GIN-only, with most overhead during training rather than inference. Overall, the results support a conditional empirical conclusion: combining structure, topology, and stability-oriented training can improve robustness under specified structural perturbations, with dataset-dependent accuracy–stability trade-offs rather than universal guarantees.

Author summary

We study how graph neural networks behave when some edges in a graph are missing or altered. These models are widely used to classify molecules, proteins, and social networks, but their predictions can change if the observed

The analysis code used to produce the reported results is publicly available at https://github.com/jecarica/TopoGNN.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

connections are incomplete or noisy. We combine two complementary views of a graph: a standard message-passing network that looks at local neighborhoods, and a topological summary that captures global connectivity patterns through persistence diagrams. We also test a training penalty that discourages large prediction changes when inputs are perturbed. Across six public benchmarks, this combination can make predictions more stable under edge removal, especially on smaller molecular and protein graphs, but it does not always improve raw accuracy. On large social networks, ordinary message-passing models are already largely insensitive to removing 10% of edges. Our results therefore do not support a one-size-fits-all claim. Instead, they help practitioners decide when topology-aware features and stability-oriented training are worth the added cost, and when a simpler graph neural network is already sufficient.

Introduction

Graph neural networks (GNNs) learn from relational structure by passing information between neighboring nodes, but their predictions can change when edges are missing, noisy, or altered. This sensitivity matters in settings where observed graphs are incomplete or unreliable, and motivates methods that remain useful under structural perturbations [1].

Persistence diagrams (PDs) summarize multiscale topological structure and can complement local message passing. PersLay [2] maps PDs to fixed-size vectors for use in neural pipelines. For graphs, extended persistence with a vertex filtration such as the Heat Kernel Signature (HKS) yields finite diagram features [2]. Combining message passing with such topological descriptors is a natural hybrid strategy [3]. We ask whether this combination can improve robustness when adjacency changes at test time.

Theoretical persistence stability is informative but conditional: guarantees depend on how much the underlying filtration or metric changes, not simply on how many edges are edited [4–6]. At the graph level, removing a redundant edge may leave connectivity largely unchanged, whereas removing a bridge (a cut-edge whose deletion increases the number of connected components) can disconnect the graph [7]. The resulting change in persistence-based descriptors therefore depends on the perturbation type: diagram drift can be small for mild edits but large when the induced metric or filtration change is substantial [4,5]. Our regularizer is motivated specifically by the stable persistence-diagram representations developed by Kusano, Fukumizu, and Hiraoka [8,9]. For brevity, we call this motivation *HK-inspired*; the term is our shorthand for that line of work and does not denote a named loss introduced in those papers. The particular training penalty used here discourages large prediction changes under perturbed inputs. We treat it as a practical regularizer and evaluate task-level robustness empirically, rather than claiming an end-to-end theoretical guarantee.

We study a hybrid model with two branches: a Graph Isomorphism Network (GIN) for local structure and PersLay for global topology, optionally trained with the

HK-inspired stability term. Using six TUDataset benchmarks and five ablation configurations, we measure clean test accuracy and robustness as accuracy drop (Drop = Clean – Noisy) under 10% random undirected edge removal at test time, with persistence diagrams fixed from the original graphs. We also evaluate targeted perturbations (bridge, hub, and spurious or hard non-edges) on MUTAG and PROTEINS and report runtime costs across all six datasets.

Our main empirical pattern is dataset-dependent. HK-inspired training substantially reduces Drop where GIN-only is sensitive, notably on molecular and protein graphs, while GIN-only or GIN+PersLay without HK often achieves the highest clean accuracy. On large social-network graphs, Drop is near zero for all GIN-based models, so accuracy rather than robustness separates configurations. PersLay-only performs worst across datasets, indicating that topology alone is insufficient.

Our contributions are: (i) a hybrid GIN + PersLay architecture with HK-inspired stability regularization; (ii) a reproducible ablation protocol separating topology, stability, and their combination; (iii) a robustness evaluation based on 10% random edge removal with fixed original-graph diagrams across all six benchmarks, supplemented by targeted perturbations on MUTAG and PROTEINS; (iv) runtime measurements across all six benchmarks; (v) an analysis of dataset-dependent accuracy–stability–runtime trade-offs across molecular, protein, and social-network graphs.

Together, these results characterize when combining topology and HK-inspired training improves empirical robustness under structural perturbations, and when standard GIN or GIN+PersLay is already sufficient.

Related work

Graph neural networks and graph classification

Graph neural networks (GNNs) are neural models defined on graphs. In the standard *message-passing* setup, each node repeatedly aggregates information from its neighbors and updates its representation layer by layer [10]. For graph-level prediction, node representations are combined by a readout (for example, sum or mean pooling) and passed to a classifier.

Several architectures are widely used backbones. The Graph Convolutional Network (GCN) [11] performs neighborhood aggregation with normalized convolution-like updates. The Graph Attention Network (GAT) [12] learns attention weights over neighbors. The Graph Isomorphism Network (GIN) [10] uses sum aggregation and was designed to match the discriminative power of the one-dimensional Weisfeiler–Lehman test (1-WL), a classical heuristic for graph isomorphism. We use GIN as the structural branch in our hybrid model.

GNNs can be sensitive to structural perturbations. Nettack [13] studies adversarial attacks on attributed graphs for node classification and introduces efficient algorithms for both test-time (evasion) and training-time (poisoning) settings; its perturbations can modify graph structure and node features, and the original work focuses on *targeted* attacks that aim to misclassify a chosen node. Metattack [14] addresses a complementary poisoning setting on node classification: it uses meta-gradients to solve the bilevel optimization problem behind training-time attacks, treats the graph structure as a hyperparameter to optimize, and crafts small edge perturbations that reduce *global* node classification performance even without access to the target classifier. Both methods highlight that message passing can amplify the effect of localized structural edits, but they optimize adversarial perturbations under a node-classification threat model rather than evaluating robustness on graph-level benchmarks. Robustness has also been surveyed and studied more broadly through adversarial training, certifiable defenses, and structural regularization [1]. Our evaluation complements this line by using random and targeted edge perturbations as interpretable stress tests for graph-level classification, rather than optimizing adversarial edge edits.

Topological deep learning and persistence diagrams

Persistent homology tracks how topological features such as connected components and cycles appear and disappear across a filtration, and summarizes the result in a persistence diagram (PD) [6]. Classical stability results show that small changes

in the underlying filtration or metric imply bounded changes in PDs under bottleneck and Wasserstein distances [4,5]. For graphs, extended persistence with vertex filtrations [15] yields finite diagram coordinates and supports practical vectorization.

Topological information has been integrated into graph learning pipelines to improve expressivity and accuracy. Examples include TOGL [16] and the Persistence Enhanced GNN [3]. Our focus is different: we study whether topological features, together with an HK-inspired stability penalty, improve *empirical* robustness when adjacency is perturbed at test time.

PersLay and diagram representations

Early approaches summarize PDs as fixed-size features, for example persistence images [17] and (upon discretization) persistence landscapes [18], the latter being functional summaries that become finite vectors once sampled on a grid. PersLay [2] generalizes this idea with a learnable, permutation-invariant layer; we describe our implementation in the section PersLay: learnable PD vectorization. The same work introduces graph topological signatures based on extended persistence and the Heat Kernel Signature (HKS). Learnable diagram representations are also studied in kernel-based settings [19]. We use PersLay as the topology branch of our hybrid model, fused with GIN and optionally trained with an HK-inspired regularizer.

Stability and robustness in GNNs

Prior work on GNN robustness has emphasized adversarial structural perturbations on node classification, including targeted evasion and poisoning attacks [13] and global poisoning attacks found via meta-gradients [14], as well as broader surveys and defense strategies [1]. A separate line embeds persistence diagrams into stable feature spaces, notably via the Persistence Weighted Gaussian Kernel (PWGK) [8,9], which bounds feature drift under diagram perturbations. Lipschitz or spectrally normalized classifiers can further limit output drift [20,21]. These results motivate training-time regularization, but they do not by themselves guarantee end-to-end robustness of a full neural pipeline under arbitrary edge edits.

Our HK-inspired penalty follows this motivation as a *practical training heuristic*: we penalize prediction changes under perturbed inputs without claiming a full theoretical stability chain. Relative to prior adversarial-attack studies, which mainly target node classification and optimized poisoning or evasion, we provide a concrete GIN + PersLay + HK ablation study for graph classification, an empirical robustness protocol based on accuracy drop under random and targeted edge perturbations, runtime measurements, and an analysis of dataset-dependent accuracy–stability–runtime trade-offs.

Negative sampling and spurious edges

Structural robustness can also be discussed in the context of how models learn from non-edges during representation learning. Recent work on diverse negative sampling for GCNs [22] and layer-diverse negative sampling for GNNs [23] shows that hard or diverse negative edges carry useful training signal and can improve representation quality. These methods target link-prediction-style learning and training-time sampling of non-neighbors.

Our setting is complementary. We study graph *classification* under perturbations of an observed graph at test time (edge removal and edge addition), and we measure task-level robustness via accuracy drop. Negative-sampling methods address how models learn to separate true edges from misleading non-edges during training; our experiments address whether predictions remain stable when the edge set changes after training. We therefore view the two directions as related but not interchangeable, and we include spurious and hard non-edge addition in our targeted perturbation protocol as a light empirical counterpart to the negative-edge literature.

Background and preliminaries

This section introduces the components used in our hybrid model: GIN message passing, extended persistence diagrams with the Heat Kernel Signature (HKS) filtration, PersLay vectorization, and an HK-inspired stability penalty. We keep the presentation concise and define notation before equations.

GIN and message passing

A *graph* is a pair $G=(V,E)$, where V is the set of nodes and E is the set of undirected edges. Each node $v \in V$ has a feature vector $\mathbf{x}_v \in \mathbb{R}^d$. The *neighborhood* of a node v is the set of nodes adjacent to it,

$$\mathcal{N}(v) = \{u \in V : (v, u) \in E\}.$$

Graph Isomorphism Networks (GINs) [10] are message-passing graph neural networks (GNNs). A GIN has K layers; at layer k, each node updates its representation by combining its current feature with the sum of its neighbors' features, then applying a small feedforward network. Formally, letting $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ and $\mathbf{h}_v^{(k)}$ denote the representation of node v after layer k, we write

$$\mathbf{h}_v^{(k)} = \mathrm{MLP}^{(k)}\left((1+\epsilon^{(k)})\cdot \mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(k-1)}\right), \tag{1}$$

where $\mathrm{MLP}^{(k)}$ is a multilayer perceptron (a small fully connected network applied independently to each node), and $\epsilon^{(k)}$ is a learnable scalar (often fixed to 0). For graph-level classification, node representations from the final layer are aggregated by a readout (we use sum pooling) and passed to a classifier.

GIN was designed to be as expressive as the one-dimensional Weisfeiler–Lehman test (1-WL), a classical heuristic for deciding whether two graphs are structurally identical. Because GIN propagates information along edges, its predictions depend on the adjacency structure: edge edits change neighborhoods and can alter graph-level outputs [1]. We use GIN as the structural branch of our model and combine it with a topological branch described below.

Extended persistence diagrams and the HKS filtration

Intuition. Extended persistence summarizes how connected components and cycles appear and disappear as we filter a graph by a vertex function. Informally, it records when topological features are born and when they die across scales, producing finite birth–death coordinates that can be vectorized for learning.

Ordinary vs. extended persistence. Persistent homology tracks changes in homology (connected components, cycles, and higher-dimensional features) as a simplicial complex is filtered. Ordinary persistence for a vertex function $f\colon V \to \mathbb{R}$ uses a sublevel-set filtration; on graphs, some cycle features may persist until infinity and complicate vectorization. Extended persistence [15] interleaves sublevel-set and superlevel-set filtrations, yielding finite birth–death pairs in all dimensions.

For a graph G with vertex function f, extended persistence produces four diagram families [15]:

- Ord_0: connected components in the sublevel-set filtration;
- Rel_1: cycles in the relative (superlevel-set) filtration;
- Ext_0^+: global connectivity features pairing sublevel births with superlevel deaths;
- Ext_1^-: global cycle features pairing superlevel births with sublevel deaths.

All coordinates are finite, so points at infinity need not be handled separately.

Heat Kernel Signature (HKS) filtration. The Heat Kernel Signature (HKS) [24] was introduced for shape analysis and can be adapted to graphs using the normalized graph Laplacian $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$. Let $\{\lambda_i, \phi_i\}_{i=1}^{|V|}$ be the eigenvalues and eigenvectors of $\mathcal{L}$. For diffusion time $t>0$, the HKS at vertex v is

$$\mathrm{HKS}(v,t) = \sum_{i=1}^{|V|} e^{-\lambda_i t}\,\phi_i(v)^2. \tag{2}$$

Small t emphasizes local structure; large t captures global diffusion. Using $\mathrm{HKS}(\cdot, t)$ as the filtration function yields extended persistence diagrams (Ord_0, Rel_1, Ext_0^+, Ext_1^-) that encode connectivity and cycle structure across scales [2].

PersLay: learnable PD vectorization

A persistence diagram is a multiset of birth–death pairs $\mathcal{D} = \{(b_i, d_i)\}_{i=1}^n$. PersLay [2] maps diagrams to fixed-size vectors with a learnable, permutation-invariant architecture:

$$\mathbf{z} = \rho\left(\bigoplus_{i=1}^{n} w(b_i, d_i)\cdot\phi(b_i, d_i)\right), \tag{3}$$

where $\phi\colon \mathbb{R}^2 \to \mathbb{R}^p$ is a pointwise feature map, $w(b_i, d_i)$ is a scalar weight, $\oplus$ is a permutation-invariant aggregation (sum, mean, max, or top-k), and ρ is optional postprocessing. Because diagram points are unordered, PersLay is invariant to point reordering.

Under suitable choices of ϕ and aggregation, PersLay is continuous with respect to diagram perturbations measured in Wasserstein distance [2,25]. In practice, weights that downweight near-diagonal (low-persistence) points improve numerical behavior. We use one PersLay module per diagram type and concatenate the resulting vectors to form the topology-branch input, as detailed in the Method section.

HK-inspired stability regularizer (conceptual motivation)

The abbreviation "HK-inspired" is manuscript shorthand for the stability motivation drawn from the persistence-diagram work of Kusano, Fukumizu, and Hiraoka [8,9]. It does not refer to a standard loss defined under that name in the cited papers.

It is useful to distinguish three related notions used in this paper:

- **Persistence stability (theory):** classical results bound diagram change under controlled filtration or metric perturbations [4,5];
- **HK-inspired regularization (training):** a penalty that discourages large prediction changes when inputs are perturbed, motivated by stable diagram embeddings such as the Persistence Weighted Gaussian Kernel (PWGK) [8,9];
- **Empirical robustness (evaluation):** accuracy drop under edge perturbations at test time, which we report as $\mathrm{Drop} = \mathrm{Clean} - \mathrm{Noisy}$.

These are related but not equivalent: small diagram change does not always imply unchanged class labels, and edge edits can induce large filtration changes (for example, when a bridge is removed) [4,5,7].

Motivation from stable diagram embeddings. PWGK embeds persistence diagrams into a reproducing kernel Hilbert space (RKHS) with bounded feature drift under weighted diagram perturbations [8,9]. Other stable diagram kernels provide similar guarantees [26,27], and PersLay provides a complementary learnable embedding [2]. Together with Lipschitz or spectrally normalized classifiers [20,21], this literature motivates penalizing excess prediction drift relative to representation drift. We use this idea as a training heuristic: the implemented hinge loss discourages prediction drift beyond a chosen input-drift proxy, but does not provide an end-to-end guarantee. The section HK-inspired stability loss gives the exact objective and the section Robustness evaluation protocols defines the empirical evaluation.

Method

We describe the hybrid GIN + PersLay architecture, stability loss, training procedure, and ablations.

Model architecture: GIN + PersLay, fusion, and classifier

The model has two parallel branches whose outputs are fused before a final classifier.

Structure branch (GIN). The graph G is processed by a GIN with K message-passing layers [10] (see GIN and message passing), producing node representations $\mathbf{h}_v^{(K)}$. For graph-level prediction we apply a sum readout:

$$\mathbf{g}_{\text{GIN}} = \sum_{v \in V} \mathbf{h}_v^{(K)} \in \mathbb{R}^{d_{\text{GIN}}}. \tag{4}$$

Topology branch (PersLay). For each graph we compute extended persistence diagrams from the HKS filtration (see Extended persistence diagrams and the HKS filtration), yielding four diagram types [15]: Ord_0, Rel_1, Ext_0^+, and Ext_1^-. Each type is vectorized by a dedicated PersLay layer [2] with its own learned ϕ, weight function, and aggregation operator. The topology-branch output is the concatenation of the four vectors:

$$\mathbf{g}_{\text{topo}} = \left[\mathbf{z}_{\text{Ord}_0} \mid \mathbf{z}_{\text{Rel}_1} \mid \mathbf{z}_{\text{Ext}_0^+} \mid \mathbf{z}_{\text{Ext}_1^-} \right] \in \mathbb{R}^{d_{\text{topo}}}. \tag{5}$$

Fusion and classifier. The two branch outputs are concatenated and passed through a classification MLP:

$$\mathbf{g} = \left[\mathbf{g}_{\text{GIN}} \mid \mathbf{g}_{\text{topo}} \right], \qquad \hat{y} = \text{MLP}_{\text{cls}}(\mathbf{g}). \tag{6}$$

We use concatenation to keep the ablations simple.

HK-inspired stability loss

The reported experiments use a Euclidean proxy on padded and masked diagram tensors.

Diagram-tensor stability loss (used in the reported experiments). Let $\mathbf{f}_\theta(G, \mathcal{D})$ and $\mathbf{f}_\theta(G, \mathcal{D}')$ be the vectors of class logits obtained with the original and perturbed diagram inputs, respectively. After padding or truncating each diagram to a common size and masking invalid entries, let $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{D}}'$ denote the resulting tensors. We use their Euclidean distance as a computational proxy for diagram drift and penalize prediction drift in excess of this proxy:

$$\mathcal{L}_{\text{stab}}^{\text{diag}} = \frac{1}{|\mathcal{B}'|} \sum_{(G, \mathcal{D}, \mathcal{D}') \in \mathcal{B}'} \max\left(0,\ \|\mathbf{f}_\theta(G, \mathcal{D}) - \mathbf{f}_\theta(G, \mathcal{D}')\|_2 - L_\pi \left\| \widetilde{\mathcal{D}} - \widetilde{\mathcal{D}}' \right\|_2 \right). \tag{7}$$

Here $L_\pi > 0$ sets the allowed logit drift per unit proxy drift. This padded-tensor distance is an implementation surrogate, not a bottleneck or Wasserstein distance; the cited stability results motivate the constraint but do not establish a theorem for it.

Combined objective. The total training loss is

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{stab}} \mathcal{L}_{\text{stab}}, \tag{8}$$

where $\mathcal{L}_{\text{CE}}$ is cross-entropy. We use $\lambda_{\text{stab}} = 0.3$ and $L_\pi = 1.0$.

Training: graph vs. diagram-input perturbation

The stability pairs can be formed by changing the graph adjacency seen by the structural branch or by changing the diagram input seen by the topological branch; we use these two designs in different configurations.

Graph perturbation. A perturbed graph G' is formed by randomly removing edges with probability p_{edge} [1]. This mirrors a simple structural-noise model and trains the model to limit prediction change under missing edges.

Diagram-input perturbation. For the Full configuration, we first remove edges from a copy of each graph and recompute its HKS-based persistence diagrams offline. During training, both forward passes use the original adjacency G, but one receives the original diagrams $\mathcal{D}$ and the other receives the precomputed diagrams $\mathcal{D}'$ from the edge-removed counterpart. Thus this configuration regularizes sensitivity to the topology-branch input while holding the structure-branch input fixed.

Practical details. Gradients do not backpropagate through the persistent-homology computation; they flow through the PersLay layers and the rest of the network [2]. The Full model uses one precomputed diagram counterpart generated by 5% random edge removal, whereas GIN + HK samples a 10% edge-dropped adjacency during training for its second forward pass. These rates serve different roles and should not be interpreted as matched perturbation strengths: the former defines offline diagram-input pairs, while the latter matches the primary test-time edge-removal rate. Accordingly, the comparison is between the implemented regularization configurations, not a controlled comparison at identical perturbation severity. The remaining optimizer, epoch, and architecture settings are specified in the experimental setup (see Experimental setup).

Ablation configurations

To separate the topology branch from the stability regularizer, we compare five configurations (Table 1). The classifier head is shared across configurations and always receives a fixed-size input; when a branch is disabled, its output is replaced by a zero vector of the same size, so architecture and parameter count remain comparable.

Full activates both branches and the diagram-input stability loss. **GIN+PersLay** uses the same architecture with $\lambda_{\text{stab}} = 0$. **GIN + HK** disables PersLay and applies the graph-perturbation penalty $\max(0, \|\mathbf{f}(G) - \mathbf{f}(G')\|_2 - L_\pi p_{\text{edge}})$. **GIN only** and **PersLay only** retain just the indicated branch and use no stability loss.

Experiments

Datasets

We evaluate on six standard graph classification benchmarks from the TUDataset collection [28] (Table 2), spanning molecular, protein, and social-network graphs.

- **MUTAG**: 188 graphs, 2 classes (mutagenicity); small molecular graphs.
- **PROTEINS**: 1,113 graphs, 2 classes; nodes are secondary structure elements, edges represent proximity.
- **ENZYMES**: 600 graphs, 6 classes; protein tertiary structures.
- **NCI1**: 4,110 graphs, 2 classes; balanced screen for anticancer activity.

Table 1. Ablation configurations. HK stability denotes the HK-inspired loss; Diagram pairs an original graph with precomputed edge-removed diagrams; Graph pairs the original graph with an edge-dropped graph while keeping diagrams fixed.

Configuration	GIN	PersLay	HK stability	Perturbation
Full (GIN+PersLay + HK)	yes	yes	yes	Diagram
GIN+PersLay (no stability)	yes	yes	no	—
GIN + HK	yes	no	yes	Graph
GIN only	yes	no	no	—
PersLay only	no	yes	no	—

https://doi.org/10.1371/journal.pcsy.0000125.t001

Table 2. Dataset statistics.

Dataset	Graphs	Classes	Filtrations	Avg nodes (approx.)
MUTAG	188	2	4	small
PROTEINS	1,113	2	4	medium
ENZYMES	600	6	4	medium
NCI1	4,110	2	8	medium
COLLAB	5,000	3	8	large
REDDIT-B	2,000	2	8	large

https://doi.org/10.1371/journal.pcsy.0000125.t002

- **COLLAB**: 5,000 graphs, 3 classes; ego-networks from scientific collaboration.
- **REDDIT-BINARY**: 2,000 graphs, 2 classes; thread discussion networks.

For each dataset we use extended persistence diagrams with HKS filtrations: MUTAG, PROTEINS, and ENZYMES use 4 filtrations (single diffusion time), while NCI1, COLLAB, and REDDIT-BINARY use 8 filtrations (two times). Diagrams are padded or truncated to 50 points per diagram type, with binary masks so that only valid points are passed to PersLay. Node features are one-hot degree (maximum degree 50). We use a stratified 80%/20% train/test split; the split and all randomness are fixed per run by seed. Unless noted otherwise, we report results over 5 runs (seeds 17–21) as mean $\pm$ standard deviation; 95% confidence intervals and paired *t*-tests are computed as well and reported where relevant.

Experimental setup

We train all configurations for 100 epochs with AdamW (learning rate 10^{-3}, weight decay 10^{-4}), batch size 32, and cosine annealing of the learning rate. The GIN branch has hidden dimension 64; the PersLay branch uses learned permutation-equivariant layers (output 25 per filtration) followed by a small MLP to 64 dimensions. The fusion classifier has two layers with spectral normalization and dropout 0.5. When stability is used, we set $\lambda_{\text{stab}} = 0.3$ and $L_\pi = 1.0$. The two stability configurations use different training perturbations: for *Full (GIN+PersLay+HK)* the second forward pass uses perturbed diagrams precomputed from a graph copy with 5% random edge removal, whereas for *GIN+HK* the second forward pass uses the same diagrams and a graph with 10% of its edges dropped at training time. These rates are not intended to represent matched perturbation severity: the 5% rate constructs offline diagram-input pairs, whereas the 10% graph-drop rate matches the primary test-time protocol. The corresponding comparison is therefore an ablation of the implemented configurations, not a controlled comparison of equal-strength perturbations. All configurations share the same base architecture and optimization settings; the active branches, stability term, and associated perturbation path differ as specified above.

Computational cost

Table 3 reports measured wall-clock runtime on CPU (3 epochs for the train-time estimate, and a full test-set inference pass). As expected, adding PersLay and HK increases training cost, because Full performs two forward passes and includes the topology branch. The overhead is dataset-dependent but consistent: Full is approximately 1.8$\times$ slower than GIN-only on MUTAG (0.068s vs 0.038s per epoch), 2.6$\times$ on PROTEINS (0.449s vs 0.173s), 3.7$\times$ on NCI1 (2.153s vs 0.589s), and 2.1$\times$ on COLLAB (7.709s vs 3.662s). Inference overhead is smaller (e.g., NCI1: 0.129s vs 0.063s; COLLAB: 0.464s vs 0.469s), indicating that most of the additional cost is incurred during training. These estimates use three measured epochs and do not separately report offline persistence-diagram construction.

Table 3. Runtime comparison on CPU. Mean train seconds/epoch over 3 measured epochs; inference seconds for a full test-set pass.

Dataset	GIN train s/epoch	Full train s/epoch	GIN inference s	Full inference s
MUTAG	0.038	0.068	0.0043	0.0051
PROTEINS	0.173	0.449	0.0195	0.0279
ENZYMES	0.089	0.328	0.0095	0.0215
NCI1	0.589	2.153	0.0629	0.1287
COLLAB	3.662	7.709	0.4692	0.4641
REDDIT-BINARY	1.417	3.457	0.1524	0.1980

https://doi.org/10.1371/journal.pcsy.0000125.t003

Robustness evaluation protocols

We evaluate robustness by measuring how test accuracy changes when the edge set is perturbed at test time. For each perturbation we report Clean accuracy (original graph), Noisy accuracy (perturbed graph), and their difference,

$$\text{Drop} = \text{Clean} - \text{Noisy}.$$

Crucially, persistence diagrams are held *fixed* to the original graphs; only the GIN branch sees the perturbed adjacency. This isolates the structural branch's sensitivity and lets us measure how much a model's prediction depends on the exact edge set when the topological input is unchanged. It is an intervention-style ablation of the two model branches, not an assumption that persistence diagrams would remain unchanged when a perturbed graph is observed in deployment; recomputing the diagrams would answer a different end-to-end robustness question.

Random edge removal (primary). Our primary protocol removes a fraction p of edges uniformly at random (undirected, per graph). We use $p = 0.10$ (i.e., 10% of each graph's edges) as the main setting and report mean Drop $\pm$ std over the 5 runs.

Multiple removal rates. To show how sensitivity scales with perturbation magnitude, we additionally sweep $p \in \{0.05, 0.10, 0.15, 0.20\}$ for the main configurations; these curves are provided in S1 File.

Targeted and spurious perturbations. Because uniform random removal is a comparatively mild stress test, we also evaluate structurally targeted perturbations on MUTAG and PROTEINS at $p = 0.10$: bridge-targeted removal (edges whose deletion disconnects components), hub-edge removal (edges incident to high-degree nodes), and spurious/hard non-edge *addition* (inserting edges the model did not observe during training). These are interpretable stress tests motivated by established graph concepts [7], adversarial structural-perturbation studies [1,13,14], and work on informative hard or diverse non-edges [22,23]; they are not implementations of Nettack, Metattack, or a negative-sampling training algorithm. These protocols probe whether robustness persists beyond random deletion, addressing cases where a few structurally critical edits can cause large changes.

Results

Test accuracy across configurations

Table 4 reports test accuracy (mean $\pm$ std over 5 runs). GIN-only or GIN+PersLay (no stability) often attain the highest or near-highest accuracy: GIN+PersLay is best on MUTAG (80.0%) and REDDIT-BINARY (88.9%), and GIN-only is best on ENZYMES (40.5%) and NCI1 (78.4%). The Full model is competitive but does not consistently outperform, and is slightly below GIN-only or GIN+PersLay on NCI1 and ENZYMES. PersLay-only is always worst, so topology alone is insufficient for these tasks. GIN + HK is close to Full on several datasets (e.g., COLLAB, PROTEINS) but much worse on ENZYMES (30.3% vs 38–40%), indicating that a strong graph-level stability term can cost accuracy on difficult tasks.

Table 4. Test accuracy (mean ± std, 5 runs). Best per dataset in bold.

Config.	MUTAG	PROTEINS	ENZYMES	NCI1	COLLAB	REDDIT-B
Full	75.8±6.6	77.8±1.3	38.2±4.4	77.1±1.5	82.8±1.1	87.5±2.5
GIN+PersLay	**80.0**±8.7	76.7±1.1	39.7±1.9	78.2±1.4	81.7±1.5	**88.9**±3.2
GIN + HK	76.3±5.6	76.6±3.9	30.3±3.2	74.9±1.8	82.8±1.4	86.6±1.7
GIN only	77.9±5.5	75.2±2.5	**40.5**±4.5	**78.4**±1.7	81.7±1.5	87.2±4.6
PersLay only	65.8±0.0	70.8±1.5	23.0±6.3	62.7±4.1	70.4±1.7	84.1±1.7

Full denotes GIN+PersLay + HK. GIN+PersLay is trained without the stability penalty.

https://doi.org/10.1371/journal.pcsy.0000125.t004

Stability under edge removal

Table 5 reports Clean accuracy, Noisy accuracy (10% random edge removal at test time), and mean Drop ± std. PersLay-only has Drop = 0 by construction (no edges used at test time). On MUTAG and PROTEINS, GIN-only has the largest Drop (17.9% and 6.5% mean), while Full and GIN + HK are substantially more stable (e.g., MUTAG: Full 5.3%, GIN + HK 2.1%). On ENZYMES the pattern holds for Drop: GIN + HK has the smallest mean Drop (0.5%), GIN-only the largest (5.0%), although GIN + HK also has the lowest Clean and Noisy accuracy. On COLLAB and REDDIT-BINARY, all GIN-based configurations have Drop near zero (sometimes slightly negative), so 10% edge removal does not materially harm accuracy. On NCI1, Drop is moderate (2–3.6%), with Full and GIN + HK slightly more stable than GIN-only.

Why does adding PersLay help even without the stability loss?. On most datasets (MUTAG, PROTEINS, NCI1), GIN+PersLay already shows a smaller Drop than GIN-only, before any HK penalty is applied. This follows from the evaluation design: persistence diagrams are computed once from the original graph and held fixed at test time, so the topology branch supplies a perturbation-invariant global signal even when edges are dropped. Because the fused classifier can partly rely on this stable branch, its prediction is anchored and drifts less than a purely structural GIN when the adjacency changes. The HK penalty then adds an explicit constraint on prediction drift, further reducing Drop where GIN-only is most sensitive. The effect is not universal: on ENZYMES, where accuracy is low and variance high, GIN+PersLay does not reduce Drop relative to GIN-only.

A paired *t*-test of Full vs. GIN-only accuracy is not significant on MUTAG, NCI1, or REDDIT-BINARY ($p \approx 0.68$, 0.21, and 0.90) and is marginal on PROTEINS ($p \approx 0.09$). Accuracy and Drop should therefore be interpreted together rather than as evidence of a universal winner.

Additional perturbation protocols (targeted and spurious)

Because uniform random removal is a mild stress test, we additionally evaluate structurally targeted perturbations on MUTAG and PROTEINS at $p = 0.10$ (Table 6): bridge-targeted removal, hub-edge removal, and spurious/hard non-edge addition. These runs use a single held-out split (seed 17) and are therefore not directly comparable to the 5-run means in Table 5; they are intended to compare perturbation *types* within the same model. On MUTAG, GIN-only is highly sensitive to hub-edge and random removal (Drop 36.8% and 31.6%), while Full stays far lower (7.9% in both cases) and is essentially unaffected by bridge removal (2.6% vs 13.2%). On PROTEINS, Full is more stable than GIN-only under random (6.3% vs 10.8%), bridge (6.7% vs 12.1%), and hub (4.9% vs 9.9%) removal. Edge *addition* (spurious or hard non-edges) has a small effect on both datasets and is occasionally slightly beneficial (e.g., PROTEINS Full –2.7%), consistent with mild noise acting as regularization rather than pure degradation. Overall, the advantage of the topology-aware Full model over GIN-only persists beyond random deletion, though its magnitude depends on perturbation type and dataset.

Table 5. Stability under edge removal. Drop = Clean - Noisy, $p = 0.1$. Mean ± std over 5 runs.

Configuration	Clean (mean±std)	Noisy (mean±std)	Drop (mean±std)
MUTAG			
Full (GIN+PersLay+HK)	75.8±6.6	70.5±12.7	5.3±16.2
GIN+PersLay, no stab.	80.0±8.7	67.4±3.0	12.6±8.6
GIN+HK	76.3±5.6	74.2±4.3	2.1±2.2
GIN only	77.9±5.5	60.0±9.4	17.9±14.3
PersLay only	65.8±0.0	65.8±0.0	0.0±0.0
PROTEINS			
Full	77.8±1.3	73.7±1.9	4.0±2.0
GIN+PersLay	76.7±1.1	72.4±4.1	4.3±4.1
GIN+HK	76.6±3.9	73.4±4.7	3.2±1.7
GIN only	75.2±2.5	68.7±6.0	6.5±3.5
PersLay only	70.8±1.5	70.8±1.5	0.0
ENZYMES			
Full	38.2±4.4	34.3±5.0	3.8±3.7
GIN+PersLay	39.7±1.9	33.8±2.5	5.8±2.0
GIN+HK	30.3±3.2	29.8±2.1	0.5±2.5
GIN only	40.5±4.5	35.5±3.6	5.0±1.6
PersLay only	23.0±6.3	23.0±6.3	0.0
NCI1			
Full	77.1±1.5	73.6±0.7	3.6±1.1
GIN+PersLay	78.2±1.4	75.8±0.8	2.4±1.4
GIN+HK	74.9±1.8	72.9±1.2	2.0±1.0
GIN only	78.4±1.7	74.9±2.3	3.5±1.5
PersLay only	62.7±4.1	62.7±4.1	0.0
COLLAB			
Full	82.8±1.1	82.3±1.0	0.4±0.4
GIN+PersLay	81.7±1.5	81.4±2.0	0.4±0.9
GIN+HK	82.8±1.4	82.4±1.4	0.4±0.6
GIN only	81.7±1.5	81.4±1.1	0.3±0.4
PersLay only	70.4±1.7	70.4±1.7	0.0
REDDIT-BINARY			
Full	87.5±2.5	87.5±3.3	0.1±1.0
GIN+PersLay	88.9±3.2	89.5±1.9	−0.6±1.8
GIN+HK	86.6±1.7	86.6±1.5	−0.1±1.2
GIN only	87.2±4.6	87.8±3.9	−0.7±1.3
PersLay only	84.1±1.7	84.1±1.7	0.0

https://doi.org/10.1371/journal.pcsy.0000125.t005

Discussion

Dataset-dependent trade-offs

The trade-off is clearest on ENZYMES and PROTEINS, where unregularized models maximize clean accuracy while regularized models reduce Drop. Possible explanations include graph size and density, task difficulty, and run-to-run variance, but the present experiments do not isolate these factors. On COLLAB and REDDIT-BINARY there is little baseline sensitivity for the regularizer to reduce; NCI1 shows a weaker intermediate pattern. PersLay alone is insufficient, yet its

Table 6. Targeted perturbation study at $p = 0.10$ on MUTAG and PROTEINS (single split, seed 17). Drop = Clean - Noisy in percentage points; lower is more robust, and negative means accuracy improved under perturbation.

Dataset	Perturbation mode	Full Drop (%)	GIN-only Drop (%)
MUTAG	random removal	7.89	31.58
MUTAG	bridge-targeted removal	2.63	13.16
MUTAG	hub-edge removal	7.89	36.84
MUTAG	spurious edge addition	13.16	13.16
MUTAG	hard non-edge addition	10.53	10.53
PROTEINS	random removal	6.28	10.76
PROTEINS	bridge-targeted removal	6.73	12.11
PROTEINS	hub-edge removal	4.93	9.87
PROTEINS	spurious edge addition	−2.69	−0.90
PROTEINS	hard non-edge addition	0.90	−1.79

https://doi.org/10.1371/journal.pcsy.0000125.t006

fixed diagram input can anchor a fused prediction when GIN sees a perturbed adjacency. Lower Drop does not necessarily mean higher perturbed accuracy: GIN + HK is most stable on ENZYMES but also less accurate. The preferred configuration therefore depends on whether clean accuracy, perturbed accuracy, or relative degradation is the primary objective.

Scope of the robustness claim

The claim is conditional and empirical, not an end-to-end guarantee. Persistence stability requires controlled filtration change, whereas even one critical edge edit can alter topology substantially. Topological stability also differs from semantic robustness: a small diagram change need not preserve class evidence. Because the primary protocol fixes the original diagrams and perturbs only the adjacency seen by GIN, it measures structural-branch sensitivity and the buffering effect of an unchanged topology input, not deployment behavior after recomputing diagrams.

Targeted tests support this limited interpretation: Full is less sensitive than GIN-only under random, bridge, and hub removal on MUTAG and PROTEINS, while edge-addition effects are mixed. The latter connect qualitatively to negative sampling [22,23]: our test-time graph-classification setting complements, but does not implement, methods that use hard non-edges as training signal.

Limitations

Limitations include fixed hyperparameters across datasets, unmatched 5%/10% training perturbations, and single-split targeted experiments on only two datasets. Runtime estimates use three measured epochs and omit offline diagram construction; Full is about $1.8\times$ – $3.7\times$ slower per training epoch than GIN-only. The targeted results are therefore stress tests rather than population-level estimates.

Conclusion

Across six benchmarks, regularized GIN/PersLay configurations reduce relative degradation where GIN-only is sensitive, especially on MUTAG and PROTEINS, but can sacrifice accuracy on ENZYMES; the social-network benchmarks are already largely insensitive to 10% edge removal. PersLay helps when fused with GIN but not alone, and most added cost occurs during training. Thus topology-aware fusion and stability-oriented training are practical, dataset-dependent options rather than guaranteed improvements. Future work should broaden multi-seed targeted tests, recompute diagrams after test-time edits, match training perturbation strengths, and include offline topology computation in runtime measurements.

Supporting information

S1 File. Multi-p edge-removal stability curves. Accuracy and Drop = Clean - Noisy under random undirected edge removal at $p \in \{0.05, 0.10, 0.15, 0.20\}$ for the main ablation configurations on MUTAG, PROTEINS, ENZYMES, NCI1, COLLAB, and REDDIT-BINARY. Persistence diagrams are held fixed to the original graphs; only the GIN branch observes the perturbed adjacency.
(CSV)

Acknowledgments

We thank the anonymous reviewers for helpful comments that improved the clarity and framing of this work.

Author contributions

Conceptualization: Charles Fanning.

Formal analysis: Charles Fanning.

Investigation: Charles Fanning.

Resources: Jelena Losic.

Software: Jelena Losic.

Validation: Jelena Losic.

Writing – original draft: Jelena Losic.

Writing – review & editing: Jelena Losic.

References

1. Günnemann S. Graph neural networks: adversarial robustness. In: Graph Neural Networks: Foundations, Frontiers, and Applications. Springer Nature Singapore; 2022. p. 149–76. https://doi.org/10.1007/978-981-16-6054-2_8
2. Carrière M, Chazal F, Ike Y, Lacombe T, Royer M, Umeda Y. PersLay: a neural network layer for persistence diagrams and new graph topological signatures. In: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS); 2020. pp. 2786–96.
3. Zhao Q, Ye Z, Chen C, Wang Y. Persistence enhanced graph neural network. In: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS); 2020. pp. 2896–906.
4. Cohen-Steiner D, Edelsbrunner H, Harer J. Stability of persistence diagrams. Discrete Comput Geom. 2006;37(1):103–20. https://doi.org/10.1007/s00454-006-1276-5
5. Chazal F, de Silva V, Oudot S. Persistence stability for geometric complexes. Geom Dedicata. 2013;173(1):193–214. https://doi.org/10.1007/s10711-013-9937-z
6. Edelsbrunner H, Harer J. Computational topology: an introduction. Providence, RI: American Mathematical Society; 2010.
7. Diestel R. Graph theory. 5th ed. Berlin: Springer; 2017.
8. Kusano G, Fukumizu K, Hiraoka Y. Persistence weighted Gaussian kernel for topological data analysis. In: Proceedings of the 33rd International Conference on Machine Learning (ICML); 2016. pp. 2004–13.
9. Kusano G, Hiraoka Y, Fukumizu K. Kernel method for persistence diagrams via kernel embedding and weight factor. J Mach Learn Res. 2018;18(189):1–41.
10. Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? In: Proceedings of the 7th International Conference on Learning Representations (ICLR); 2019. https://doi.org/10.48550/arXiv.1810.00826
11. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR); 2017. Available from: https://openreview.net/forum?id=SJU4ayYgl
12. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In: International Conference on Learning Representations (ICLR); 2018. Available from: https://openreview.net/forum?id=rJXMpikCZ
13. Zügner D, Akbarnejad A, Günnemann S. Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. pp. 2847–56. https://doi.org/10.1145/3219819.3220078

14. Zügner D, Günnemann S. Adversarial attacks on graph neural networks via meta learning. In: International Conference on Learning Representations (ICLR); 2019. Available from: https://openreview.net/forum?id=Bylnx209YX
15. Cohen-Steiner D, Edelsbrunner H, Harer J. Extending persistence using poincaré and lefschetz duality. Found Comput Math. 2008;9(1):79–103. https://doi.org/10.1007/s10208-008-9027-z
16. Horn M, De Brouwer E, Moor M, Moreau Y, Rieck B, Borgwardt K. Topological graph neural networks. In: International Conference on Learning Representations (ICLR); 2022. Available from: https://openreview.net/forum?id=oxxUMeFwEHd
17. Adams H, Emerson T, Kirby M, Neville R, Peterson C, Shipman P. Persistence images: a stable vector representation of persistent homology. J Mach Learn Res. 2017;18(8):1–35.
18. Bubenik P. Statistical topological data analysis using persistence landscapes. J Mach Learn Res. 2015;16(3):77–102.
19. Hofer C, Kwitt R, Niethammer M. Learning representations of persistence barcodes. J Mach Learn Res. 2019;20(126):1–45.
20. Virmaux A, Scaman K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: Advances in Neural Information Processing Systems (NeurIPS); 2018. Available from: https://proceedings.neurips.cc/paper/2018/hash/d54e99a6c03704e95e6965532dec148b-Abstract.html
21. Miyato T, Kataoka T, Koyama M, Yoshida Y. Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (ICLR); 2018. Available from: https://openreview.net/forum?id=B1QRgziT
22. Duan W, Xuan J, Qiao M, Lu J. Learning from the dark: boosting graph convolutional neural networks with diverse negative samples. AAAI. 2022;36(6):6550–8. https://doi.org/10.1609/aaai.v36i6.20608
23. Duan W, Lu J, Wang YG, Xuan J. Layer-diverse negative sampling for graph neural networks. Trans Mach Learn Res. 2024.
24. Sun J, Ovsjanikov M, Guibas L. A concise and provably informative multi-scale signature based on heat diffusion. In: Proceedings of the Symposium on Geometry Processing (SGP); 2009. pp. 1383–92.
25. Divol V, Lacombe T. Understanding the topology and the geometry of the space of persistence diagrams via optimal partial transport. J Appl and Comput Topology. 2020;5(1):1–53. https://doi.org/10.1007/s41468-020-00061-z
26. Reininghaus J, Huber S, Bauer U, Kwitt R. A stable multi-scale kernel for topological machine learning. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. pp. 4741–8. https://doi.org/10.1109/cvpr.2015.7299106
27. Carrière M, Cuturi M, Oudot S. Sliced Wasserstein kernel for persistence diagrams. In: Proceedings of the 34th International Conference on Machine Learning (ICML); 2017. pp. 664–73.
28. Morris C, Kriege NM, Bause F, Kersting K, Mutzel P, Neumann M. TUDataset: a collection of benchmark datasets for learning with graphs. In: ICML Workshop on Graph Representation Learning and Beyond (GRL+); 2020. Available from: https://arxiv.org/abs/2007.08663