Enhancing Machine Learning Algorithms on Directed Graphs Using Persistent Path Homology

Sokolov Anton
Supervisor: Cand. Sc. Viktor A. Zamaraev

June 8, 2025

Abstract

This thesis introduces the Local Persistent Path Homology (LPPH) vector, a novel topological feature vector for enhancing directed link prediction machine learning algorithms on directed graphs. LPPH captures multi-scale topological features around graph edges by leveraging Persistent Path Homology (PPH), which tracks the evolution of directed path structures across a filtration. We propose a model incorporating LPPH for link prediction in directed graphs, addressing the challenge of directional asymmetry often overlooked by traditional graph neural networks. We provide a detailed construction of LPPH and demonstrate its effectiveness through extensive experiments on real-world datasets, showing improved performance of baseline directed GNN models, and even outperforms them as a standalone classifier in some scenarios, achieving competitive results, compared to state-of-the-art directed link prediction methods.

Nomenclature

PH	Persistent Homology
PPH	Persistent Path Homology
LPPH	Local Persistent Path Homology
GNN	Graph Neural Network
dirGNN	Directed Graph Neural Network
AUC	Area Under the ROC Curve

1 Introduction

Directed graphs are integral in modeling complex systems, from social networks and biological interactions to transportation flows and citation networks. Unlike undirected graphs, the

asymmetric nature of edges in directed graphs encodes critical information about influence, hierarchy, and causality. Link prediction in such graphs—the task of inferring missing or future connections—poses unique challenges due to this directional asymmetry. Traditional graph neural networks (GNNs), while powerful for undirected graphs, often struggle to capture the nuanced topological patterns inherent in directed structures. Moreover, recent work has highlighted that the theoretical understanding of GNNs—particularly their expressivity, generalization, and optimization dynamics—remains limited, emphasizing the need for more principled approaches in graph machine learning (Morris et al. 2024).

Persistent homology (PH) has emerged as a potent tool for topological data analysis, providing multi-scale summaries of features like connected components, loops, and higher-dimensional voids. However, standard PH operates on simplicial complexes derived from undirected proximity relations, making it ill-suited for directed graphs. Recent advances in *path homology* have addressed this gap by extending homological concepts to directed paths, enabling the detection of directional cycles and connectivity patterns. Building on this foundation, *Persistent Path Homology (PPH)* tracks the evolution of such features across filtrations, providing a theoretically grounded framework for directed graph analysis.

This work introduces the *Local Persistent Path Homology (LPPH)* vector, a novel topological feature vector designed to enhance machine learning models for directed link prediction. The LPPH vector quantifies multi-scale directional connectivity around graph edges by computing persistent path homology over edge-centric filtrations. By converting the resulting persistence diagrams into stable vector representations via persistence images, LPPH captures local topological signatures while focusing directional asymmetry. When integrated with directed GNNs, LPPH provides interpretable, topology-aware features that may complement learned node embeddings, addressing the limitations of purely message-passing approaches.

Our contributions are threefold:

- We propose LPPH, the first method to combine persistent path homology with persistence images for directed link prediction, enabling the encoding of multi-scale directional features in fixed-dimensional vectors.
- We develop an efficient algorithm for constructing edge-centric filtrations and computing LPPH, ensuring scalability to real-world networks.
- Through extensive experiments on different directed graph datasets, we demonstrate that LPPH improves the performance of baseline directed GNNs models and even outperforms them as a standalone classifier in certain scenarios, showing competitive results, compared to state-of-the-art directed link prediction methods.

2 Persistent Homology

Persistent homology (PH) is a mathematical framework for analyzing the topological structure of data across scales. It provides a multi-scale summary of features such as connected components, loops, cavities, and higher-dimensional voids. Unlike traditional homology, which computes topological invariants at a fixed scale, PH tracks the evolution of these features over a parameterized filtration, offering a robust tool for understanding the "shape" of data.

2.1 Key Definitions

Definition 2.1 (Simplicial Complex)**.** A *simplicial complex* $\mathscr{K}$ is a set of simplices (vertices, edges, triangles, tetrahedra, etc.) such that:

- Every face of a simplex in $\mathscr{K}$ is also in $\mathscr{K}$.
- The intersection of any two simplices $\sigma, \tau \in \mathscr{K}$ is a face of both σ and τ.

Definition 2.2 (Filtration)**.** A *filtration* of a simplicial complex $\mathscr{K}$ is a nested sequence of subcomplexes:

$$\emptyset = \mathscr{K}_0 \subseteq \mathscr{K}_1 \subseteq \cdots \subseteq \mathscr{K}_n = \mathscr{K},$$

parameterized by a scale $\varepsilon \in \mathbb{R}$. At each scale ε_i, simplices are added according to a certain rule (e.g. proximity in a point cloud in Čech complex).

Definition 2.3 (Homology Groups)**.** Let K be a simplicial complex. The ***k*-th homology group** $H_k(K)$ describes the k-dimensional topological features of K. It is constructed using the chain complex of K, which consists of chain groups, boundary maps, and their corresponding quotient structure.

1. **Chain Groups:** Define the group of k-chains as the free abelian group $C_k(K)$ generated by the k-simplices of K:

$$C_k(K) = \mathbb{Z}\langle \sigma \mid \sigma \text{ is a } k\text{-simplex of } K \rangle.$$

2. **Boundary Operators:** The boundary operator $\partial_k : C_k(K) \to C_{k-1}(K)$ maps a k-simplex to the formal sum of its $(k-1)$-dimensional faces with alternating signs:

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i [v_0, \ldots, \hat{v}_i, \ldots, v_k],$$

where $[v_0, \ldots, \hat{v}_i, \ldots, v_k]$ is the $(k-1)$-dimensional face obtained by removing the vertex v_i. These maps satisfy $\partial_{k-1} \circ \partial_k = 0$, ensuring that the image of ∂_k is always contained in the kernel of ∂_{k-1}.

3. **Cycles and Boundaries:**

   - The group of k**-cycles** is the kernel of ∂_k, consisting of chains whose boundary is zero:
   $$Z_k(K) = \ker \partial_k = \{c \in C_k(K) \mid \partial_k c = 0\}.$$
   - The group of k**-boundaries** is the image of ∂_{k+1}, consisting of boundaries of $(k+1)$-chains:
   $$B_k(K) = \operatorname{im} \partial_{k+1} = \{\partial_{k+1} c \mid c \in C_{k+1}(K)\}.$$

4. **Homology Groups:** The k-th homology group is defined as the quotient:
   $$H_k(K) = Z_k(K)/B_k(K),$$
   which identifies cycles differing by boundaries as equivalent. This captures the number of "holes" in K that are not filled by higher-dimensional simplices.

5. **Interpretation of Low-Dimensional Homology:**

   - $H_0(K)$ (0-th homology group) corresponds to the number of **connected components** of K.
   - $H_1(K)$ (1st homology group) detects **loops** (1-cycles that are not boundaries of 2-simplices).
   - $H_2(K)$ (2nd homology group) identifies **voids or cavities** (2-cycles not filled by 3-simplices).

6. **Betti Numbers:** The **Betti number** β_k is the rank of $H_k(K)$:
   $$\beta_k = \operatorname{rank} H_k(K),$$
   which represents the number of independent k-dimensional topological features.

Definition 2.4 (Čech Complex)**.** Let X be a finite set of points in a metric space (M,d) and let $\varepsilon > 0$. The Čech complex $\mathscr{C}_\varepsilon(X)$ is the abstract simplicial complex with simplices corresponding to finite subsets $\{x_0,\ldots,x_k\} \subseteq X$ such that
$$\bigcap_{i=0}^{k} B_\varepsilon(x_i) \neq \emptyset,$$
where $B_\varepsilon(x_i)$ denotes the open ball of radius ε centered at x_i.

Definition 2.5 (Vietoris–Rips Complex)**.** Let X be a finite set of points in a metric space (M,d) and let $\varepsilon > 0$. The Vietoris–Rips complex $\mathrm{VR}_\varepsilon(X)$ is the abstract simplicial complex with

simplices corresponding to finite subsets $\{x_0, \ldots, x_k\} \subseteq X$ such that

$$d(x_i, x_j) \leq 2\varepsilon \quad \text{for all } 0 \leq i, j \leq k.$$

2.2 Persistent Homology Framework

Let $\{\mathscr{K}_\varepsilon\}_{\varepsilon \in \mathbb{R}}$ be a *filtration* of a topological space (or simplicial complex), meaning that for any $\varepsilon \leq \varepsilon'$, we have $\mathscr{K}_\varepsilon \subseteq \mathscr{K}_{\varepsilon'}$. For each ε, one computes the k-th homology group $H_k(\mathscr{K}_\varepsilon)$. The inclusion $\mathscr{K}_\varepsilon \hookrightarrow \mathscr{K}_{\varepsilon'}$ induces a homomorphism

$$\varphi_{\varepsilon,\varepsilon'} : H_k(\mathscr{K}_\varepsilon) \to H_k(\mathscr{K}_{\varepsilon'}),$$

which allows us to track how individual homology classes evolve with the scale parameter ε.

A homology class $\gamma \in H_k(\mathscr{K}_\varepsilon)$ is said to be *born* at the scale $\varepsilon = b$ if, for a sufficiently small $\delta > 0$, γ is not in the image of the map

$$\varphi_{b-\delta,b} : H_k(\mathscr{K}_{b-\delta}) \to H_k(\mathscr{K}_b).$$

In other words, γ represents a new topological feature that appears at scale b. Conversely, the class γ *dies* at the scale $\varepsilon = d$ if, for a small $\delta > 0$, its image under the inclusion

$$\varphi_{d-\delta,d} : H_k(\mathscr{K}_{d-\delta}) \to H_k(\mathscr{K}_d)$$

merges with an older homology class. This merging indicates that the feature corresponding to γ is no longer distinct beyond the scale d.

The persistence of these homology classes—their lifespans characterized by their birth and death scales—is visualized using *persistence diagrams*. Each point (b,d) in a persistence diagram represents a feature born at b and dying at d, with the difference $d-b$ indicating the significance of the feature.

Definition 2.6 (Hausdorff Distance)**.** Let (X,d) be a metric space and let $A, B \subset X$ be two nonempty subsets. The Hausdorff distance between A and B is defined as

$$d_H(A,B) = \max\left\{ \sup_{a\in A} \inf_{b\in B} d(a,b), \sup_{b\in B} \inf_{a\in A} d(a,b) \right\}.$$

Definition 2.7 (Wasserstein Distance)**.** Let D and D' be two persistence diagrams viewed as multisets of points in the extended plane $\overline{\mathbb{R}}^2$. For $p \in [1,\infty)$, the p-Wasserstein distance between D and D' is

$$W_p(D,D') = \inf_{\gamma} \left(\sum_{x\in D} \|x - \gamma(x)\|_\infty^p \right)^{1/p},$$

where the infimum is taken over all bijections $\gamma : D \to D'$ (allowing points to match to the diagonal), and $\|\cdot\|_\infty$ is the uniform norm.

Definition 2.8 (Bottleneck Distance)**.** The bottleneck distance between two persistence diagrams D and D' is the ∞-Wasserstein distance,

$$W_\infty(D, D') = \inf_\gamma \sup_{x \in D} \|x - \gamma(x)\|_\infty,$$

where the infimum is over all bijections γ between D and D' (again allowing matches to the diagonal).

Theorem 1 (Stability of Persistence Diagrams)**.** *Let X and Y be two point clouds in a metric space (M, d), with persistence diagrams D_X and D_Y obtained from the same filtration method. Then for any $p \in [1, \infty]$, the following holds:*

$$W_p(D_X, D_Y) \leq d_H(X, Y),$$

where $d_H(X, Y)$ is the Hausdorff distance between X and Y. In particular, for $p = \infty$ this gives the bottleneck distance bound

$$W_\infty(D_X, D_Y) \leq C * d_H(X, Y).$$

This inequality ensures that small perturbations in the input data produce only small changes in their persistence diagrams, confirming the robustness of persistent homology.

2.3 Example and Illustration

Consider a simplicial complex filtration on fig 1. As the scale ε increases, edges and triangles form:

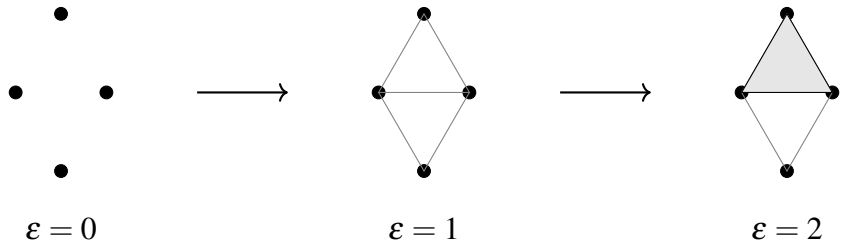

Figure 1: Filtration of a point cloud. At $\varepsilon = 0$, only vertices exist. At $\varepsilon = 1$, edges form two cycles (loops). At $\varepsilon = 2$, one of the loops is filled by a triangle, causing its "death".

2.4 Algorithmic Computation

The standard algorithm for PH (Persistent homology) involves the following steps:

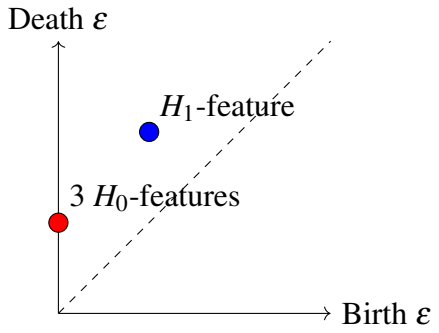

Figure 2: Persistence diagram for the filtration in Figure 1. The H_1-feature (loop) is born at $\varepsilon = 1$ and dies at $\varepsilon = 2$. H_0-features (components) merge as the filtration evolves.

1. **Filtration Construction:** Build a nested sequence of complexes (e.g., using Vietoris-Rips or Čech complexes or any other way).

2. **Boundary Matrix Reduction:** Compute the persistence pairs using matrix reduction.

2.4.1 Boundary Matrix Reduction

To compute persistence pairs one builds the *boundary matrix* and then performs a column-wise reduction (essentially a form of Gaussian elimination over a field) (Edelsbrunner and Harer 2010).

The Boundary Matrix. Order all simplices in the filtration as $\sigma_1, \sigma_2, \ldots, \sigma_N$ so that if σ_i appears before σ_j in the filtration then $i < j$. For each k-simplex σ_j, its boundary is

$$\partial_k(\sigma_j) \;=\; \sum_{i:\,\sigma_i \subset \sigma_j} [\sigma_i]\,,$$

and we form an $N \times N$ matrix D over $\mathbb{Z}_2$ (for simplicity) by

$$D_{i,j} = \begin{cases} 1, & \text{if } \sigma_i \subset \partial\sigma_j, \\ 0, & \text{otherwise.} \end{cases}$$

Columns are indexed by simplices and rows by their faces.

Reduction Algorithm. We reduce D by adding columns to the right (mod 2) so that each nonzero column has a unique lowest-one (the index of its lowest 1-entry), and no two columns share the same lowest-one. Formally:

Algorithm 1 Column-Reduction of Boundary Matrix

Require: Boundary matrix $D \in \mathbb{Z}_2^{N\times N}$

Initialize $R \leftarrow D$
for $j \leftarrow 1$ to N **do**
  while $\exists i < j$ with $\text{low}(R(:,i)) = \text{low}(R(:,j)) \neq \emptyset$ **do**
    $R(:,j) \leftarrow R(:,j) + R(:,i)$ ▷ column addition over $\mathbb{Z}_2$
  end while
end for

Ensure: Reduced matrix R

Here $\text{low}(C)$ returns the index of the lowest row in column C that is nonzero (or $\emptyset$ if $C \equiv 0$). After reduction:

2.4.2 Computing Persistent Diagrams

For each column i full of zeros, a column j with $\text{low}(R(:,j)) = i$ indicates that the k-simplex σ_j (column) is the *death* of the homology class born when σ_i (row) was added. Hence we pair σ_i and σ_j as a *persistent pair*. Assuming the time in which both simplices were added we can calculate birth-death pairs (b,d), where $b < d$ and construct persistent diagrams for each dimension.

Complexity. Naïve worst-case is $O(N^3)$ field operations, but in practice with sparse data and heuristics (clearing, compression) it is far more efficient (Bauer, Kerber, Reininghaus, and Wagner 2017; Chen and Kerber 2011; Bauer, Kerber, and Reininghaus 2014)).

With this reduction in hand, one directly reads off all (b,d) pairs to construct the persistence diagram as in Figure 2.

Persistent Homology (PH) has been successfully applied in various domains, including neuroscience (Giusti et al. 2015; Lee and Chung 2023), materials science (Hiraoka et al. 2016), machine learning (Hofer et al. 2020; Chen, Ni, et al. 2021; Zhao et al. 2022; Horn et al. 2022a), and network analysis (Chowdhury and Mémoli 2018; Liao et al. 2023).

3 Persistent Image

The persistence diagram, while rich in topological information, is an unstructured multiset of points unsuitable for direct use in machine learning models. To bridge this gap, the *Persistent Image* (PI) provides a vectorized representation of persistence diagrams by converting them into finite-dimensional vectors (Adams et al. 2017). This transformation enables compatibility with standard machine learning algorithms.

3.1 Definition and Construction

Given a persistence diagram $D = \{(b_i, d_i)\}_{i=1}^n$, the Persistent Image is generated through the following constructive process:

1. **Weighting Scheme:** Assign weights via a Lipschitz-continuous function $f : \mathbb{R}^2 \to \mathbb{R}^+$ that vanishes along the diagonal $y = x$. For stability, a standard choice is persistence-weighted Gaussians:

$$w_i = f(b_i, d_i) = c \cdot (d_i - b_i)^\alpha \quad \text{with } \alpha \geq 0, c > 0$$

   This continuous weighting avoids instability from diagonal perturbations.

2. **Grid Construction:** Define an $r \times r$ grid over the birth-persistence coordinates (b, p) where $p = d - b$. The grid often bounds $[b_{\min}, b_{\max}] \times [0, p_{\max}]$ are determined by:

$$b_{\min} = \min_i b_i, \quad b_{\max} = \max_i b_i, \quad p_{\max} = \max_i (d_i - b_i)$$

3. **Kernel Density Integration:** Construct a persistence surface through Gaussian convolution:

$$\rho_D(x, y) = \sum_{i=1}^n w_i \cdot \mathcal{N}\left((x, y); (b_i, p_i), \sigma^2 I\right)$$

   where $\mathcal{N}(\cdot; \mu, \Sigma)$ is the Gaussian PDF. The persistence image values are obtained by integrating over grid cells Ω_{kl}:

$$I_{kl} = \iint_{\Omega_{kl}} \rho_D(x, y)\, dx\, dy = \sum_{i=1}^n w_i \iint_{\Omega_{kl}} \mathcal{N}\left((x, y); (b_i, p_i), \sigma^2 I\right) dx\, dy$$

   This integration provides stability against diagram perturbations.

4. **Vector Representation:** Flatten the integrated grid values into a feature vector:

$$\mathbf{v} = [I_{11}, I_{12}, \ldots, I_{rr}]^\top \in \mathbb{R}^{r^2}$$

The final persistence image provides a fixed-dimensional representation while preserving the stability properties of persistence diagrams, enabling direct use with machine learning algorithms.

3.2 Stability Theorem

Theorem 2 (Persistence Image Stability)**.** *Let B, B' be persistence diagrams with 1-Wasserstein distance $W_1(B, B')$. For persistence images $I(\rho_B), I(\rho_{B'})$ generated using:*

- *A Lipschitz-continuous weighting function f with $\|\nabla f\|_\infty \leq L$*
- *Gaussian kernels with variance σ^2*
- *Normalized persistence coordinates $T(b,d) = (b, d-b)$*

The difference between persistence images is bounded in multiple norms:

$$\|I(\rho_B) - I(\rho_{B'})\|_1 \leq \left(\sqrt{5}L + \sqrt{\frac{10}{\pi}}\frac{\|f\|_\infty}{\sigma}\right) W_1(B,B') \tag{1}$$

$$\|I(\rho_B) - I(\rho_{B'})\|_2 \leq \left(\sqrt{5}L + \sqrt{\frac{10}{\pi}}\frac{\|f\|_\infty}{\sigma}\right) W_1(B,B') \tag{2}$$

$$\|I(\rho_B) - I(\rho_{B'})\|_\infty \leq \left(\sqrt{5}L + \sqrt{\frac{10}{\pi}}\frac{\|f\|_\infty}{\sigma}\right) W_1(B,B') \tag{3}$$

where:

$$\|v\|_1 := \sum_{i=1}^{n} |v_i|$$

$$\|v\|_2 := \sqrt{\sum_{i=1}^{n} v_i^2}$$

$$\|v\|_\infty := \max_{1\leq i\leq n} |v_i|$$

Proof Sketch. The stability follows from three key components:

1. **Lipschitz Continuity**: The weighting function's gradient bounds feature displacement sensitivity
2. **Gaussian Convolution**: Kernel integration smooths local perturbations
3. **Diagram Matching**: Optimal transport between B and B' controls mass redistribution

Full proof appears in (Adams et al. 2017, Section 5) using measure transportation theory. □

3.3 Advantages and Parameters

The Persistent Image offers several benefits:

- **Fixed Dimension:** The grid resolution r ensures consistent input size for ML models.
- **Stability:** Small perturbations in the diagram (under bottleneck distance) lead to small changes in the PI, preserving topological stability (Adams et al. 2017).

- **Interpretability:** Grid cells correspond to specific birth-death regions, linking features to the original topology.

Key parameters include the grid resolution r, bandwidth σ, and weighting exponent α. These are typically optimized via cross-validation.

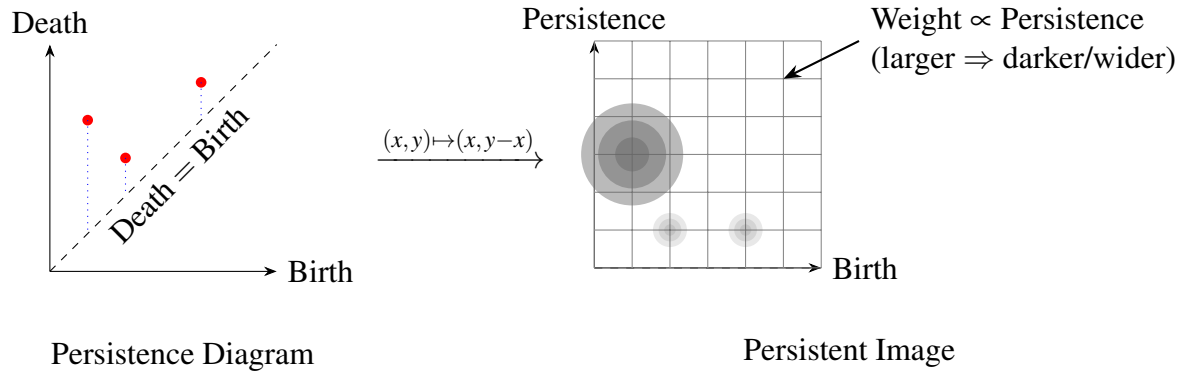

Figure 3: Transformation of a persistence diagram (left) into a Persistent Image (right).

4 Path Homology

Path homology is a generalization of simplicial homology that is defined for directed graphs. Unlike simplicial homology, which is defined on simplicial complexes, path homology is defined on directed graphs. This makes it a more natural choice for analyzing the topological properties of directed graphs. Path homology captures the directed paths in a graph and uses them to define homology groups, which provide information about the graph's structure.

4.1 Advantages Over Simplicial Homology

Traditional simplicial homology relies on cliques (complete subgraphs) to form simplices, restricting higher-dimensional features to densely connected regions. In contrast, path homology operates on directed paths, enabling the detection of structures formed by sequences of edges even when no cliques exist. This flexibility allows path homology to:

- **Capture Direction-Sensitive Features**: Directed cycles or paths can form nontrivial boundaries that alter homology groups compared to their undirected counterparts, a phenomenon not visible under simplicial homology on undirected clique complexes (Grigor'yan et al. 2012; Grigor'yan et al. 2020).
- **Reveal Higher-Dimensional Structures**: Chains of directed paths generate homology in higher degrees ($p \geq 2$) without requiring cliques. In particular, the boundary operator in a path complex is defined by removing interior vertices of a path, in contrast to

the alternating face sum used in simplicial complexes, enabling non-clique-based higher chains (Grigor'yan et al. 2012).

- **Distinguish Directed Topology**: Graphs with identical underlying undirected structures but differing edge directions can exhibit distinct path homologies or homotopy invariants (Grigor'yan et al. 2014), whereas simplicial homology on the clique complex would treat them identically (Grigor'yan et al. 2012).

4.2 Definitions

Let $G = (V, E)$ be a directed graph where V is a set of vertices and $E \subseteq V \times V$ is a set of directed edges.

Definition 4.1 (Directed p-Path)**.** A *p-path* in G is an ordered tuple

$$\gamma = (v_0, v_1, \ldots, v_p)$$

such that for each $i = 1, \ldots, p$, the ordered pair $(v_{i-1}, v_i) \in E$. These paths serve as the basic combinatorial objects for constructing our chain complex.

Definition 4.2 (Path Chain Complex of a Directed Graph)**.** Let $G = (V, E)$ be a directed graph and let $\Bbbk$ be a field. We construct two related chain complexes over $\Bbbk$:

1. **Universal Path Complex.** For each $p \geq 0$, let

$$\mathscr{U}_p(G) \;=\; \Bbbk\big\langle (v_0, \ldots, v_p) \mid v_i \in V \big\rangle$$

be the $\Bbbk$–vector space freely generated by all tuples of length p of vertices in G. Define the boundary map

$$d_p : \mathscr{U}_p(G) \longrightarrow \mathscr{U}_{p-1}(G), \quad d_p\big(v_0, \ldots, v_p\big) \;=\; \sum_{i=0}^{p} (-1)^i \, (v_0, \ldots, \widehat{v_i}, \ldots, v_p),$$

and set $d_0 \equiv 0$. One checks $d_{p-1} \circ d_p = 0$, giving a chain complex

$$\cdots \xrightarrow{d_{p+1}} \mathscr{U}_p(G) \xrightarrow{d_p} \mathscr{U}_{p-1}(G) \xrightarrow{d_{p-1}} \cdots \xrightarrow{d_1} \mathscr{U}_0(G) \to 0.$$

2. **Path complex.** We define the *path chain complex* $\Lambda_*(G)$ as the largest subchain complex of $\mathscr{U}_*(G)$ which elements are combinations of p-paths of G and is closed under the boundary operators.

$$\Lambda_0(G) = \mathscr{U}_0(G), \quad \Lambda_p(G) = \big\{ \gamma \in \mathscr{U}_p(G) \mid d_p(\gamma) \in \Lambda_{p-1}(G), \gamma - \text{combination of p-paths of } G \big\} \quad (p \geq 1)$$

$$\Lambda_0(G) = \mathscr{U}_0(G),$$

$$\Lambda_p(G) = \big\{\gamma \in \mathscr{U}_p(G) \mid \gamma \text{ is a } \Bbbk\text{–combination of } p\text{-paths in } G, \\ d_p(\gamma) \in \Lambda_{p-1}(G)\big\}, \quad p \geq 1.$$

Definition 4.3 (Path Homology)**.** The pth *path homology group* of G is defined as the homology of the chain complex $(\Lambda_*(G), d_*)$:

$$H_p(G) = \frac{\ker\big(d_p : \Lambda_p(G) \to \Lambda_{p-1}(G)\big)}{\operatorname{im}\big(d_{p+1} : \Lambda_{p+1}(G) \to \Lambda_p(G)\big)}.$$

4.3 Path Homology as a Generalization of Simplicial Homology

Given a simplicial complex S representing a shape, there is a natural ordering from lower- to higher-dimensional simplices determined by the subset relation (e.g., a point belonging to an edge, or an edge belonging to a triangle). A directed graph $D(S)$ can be constructed by taking the simplices as nodes and introducing a directed edge from simplex σ to simplex τ if $\sigma \subset \tau$.

Theorem 3 (Path Homology Recovers Simplicial Homology)**.** *Let S be a simplicial complex and $D(S)$ the digraph constructed as above. Then the path homology groups $H_*(D(S))$ are naturally isomorphic to the simplicial homology groups $H_*(S)$ of S (see Grigor'yan et al. 2014a). In this sense, path homology generalizes simplicial homology.*

Sketch of Proof. The directed graph $D(S)$ encodes the inclusion relations among the simplices of S. The natural ordering provided by the subset relation allows one to define a chain map between the path complex of $D(S)$ and the standard simplicial chain complex of S. One verifies that under this correspondence the boundary operators coincide (up to a canonical isomorphism), and thus the induced homology groups are isomorphic. □

4.4 Example: Calculating Path Homology

Consider the directed graph G with vertices A, B, C, D and edges

$$A \to B, \quad B \to C, \quad A \to D, \quad D \to C.$$

Figure 4 illustrates this graph.

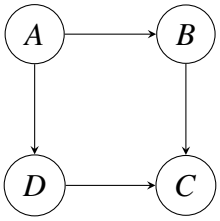

Figure 4: A directed graph G.

The corresponding chain groups for the path homology are:

- $\Lambda_0(G)$ is generated by the vertices $\{A,B,C,D\}$.
- $\Lambda_1(G)$ is generated by the directed edges $\{(A,B),(B,C),(A,D),(D,C)\}$.
- $\Lambda_2(G)$ is a biggest "allowed" subspace of $\Bbbk\big\langle\{(A,B,C),(A,D,C)\}\big\rangle$ (since only sequences $A\to B\to C$ and $A\to D\to C$ are in G).

The boundary maps are defined as follows. For a directed edge (u,v), we have:

$$d_1(u,v) = v-u,$$

so that

$$\begin{aligned} d_1(A,B) &= B-A,\\ d_1(B,C) &= C-B,\\ d_1(A,D) &= D-A,\\ d_1(D,C) &= C-D. \end{aligned}$$

For a 2-path (u,v,w), the boundary is given by:

$$d_2(u,v,w) = (v,w)-(u,w)+(u,v).$$

Thus, we obtain:

$$\begin{aligned} d_2(A,B,C) &= (B,C)-(A,C)+(A,B),\\ d_2(A,D,C) &= (D,C)-(A,C)+(A,D). \end{aligned}$$

Since the edge $A\to C$ is not present in G, $\Lambda_2(G)$ is generated by $\{(A,B,C)-(A,D,C)\}$ as:

$$\begin{aligned} d_2\big((A,B,C)-(A,D,C)\big) &= d_2(A,B,C) \;-\; d_2(A,D,C)\\ &= \big[(B,C)-(A,C)+(A,B)\big]-\big[(D,C)-(A,C)+(A,D)\big]\\ &= (A,B)+(B,C) \;-\; (A,D)-(D,C)\in\Lambda_1(G). \end{aligned}$$

- $H_0(G)\cong\mathbb{K}$ because the graph is connected.

- $H_1(G) = \frac{\ker d_1}{\operatorname{im} d_2}$. A typical cycle in $\Lambda_1(G)$ is given by

$$(A,B) + (B,C) - (A,D) - (D,C),$$

hence:

$$H_1(G) = 0.$$

- For $p \geq 2$, we have $H_p(G) = 0$.

This example illustrates the procedure for computing path homology and highlights the challenge of determining basis of chain groups in the path complex.

5 Persistent Path Homology

Persistent Path Homology (PPH) generalizes classical persistent homology to directed graphs by tracking homological features induced by directed paths rather than simplices. This approach preserves edge orientation, enabling the detection of asymmetric topological structures such as directed cycles and flow-like patterns. PPH produces persistence diagrams that reflect the underlying directed connectivity of networks (Grigor'yan et al. 2012; Chowdhury and Mémoli 2018; Dey, T. Li, and Y. Wang 2020).

5.1 Filtration Construction

Given a directed graph $G = (V,E)$ and a nonnegative weight function $w : E \to [0,\infty)$, we define a filtration of subgraphs:

$$G_{\varepsilon_1} \subseteq G_{\varepsilon_2} \subseteq \cdots \subseteq G_{\varepsilon_m} = G,$$

where

$$G_{\varepsilon_i} = (V, E_{\varepsilon_i}), \quad E_{\varepsilon_i} = \{\, e \in E \mid w(e) \leq \varepsilon_i \,\}.$$

We enumerate all directed paths $\pi_1, \ldots, \pi_N$ in order of their birth times:

$$t_1 \leq t_2 \leq \cdots \leq t_N, \quad t_i = \max\{w(e) : e \text{ on } \pi_i\}.$$

Each path π_i enters the filtration when its largest-weight edge appears.

5.2 Boundary Matrix Construction

Let $\{\pi_i\}_{i=1}^N$ be the set of valid directed paths and F the set of forbidden paths (those that do not exist in G but are generated during boundary computations). Define the boundary matrix:

$$\partial \in \mathbb{Z}_2^{(N+|F|)\times N},$$

with:

- Columns indexed by valid paths $\pi_1, \ldots, \pi_N$.
- Rows indexed by all potential faces—both valid and forbidden.

For a path $\pi_i = (v_0, \ldots, v_p)$, the boundary is

$$\partial \pi_i = \sum_{k=0}^{p} (-1)^k \pi_{i,[k]},$$

where $\pi_{i,[k]}$ omits vertex v_k. Set

$$\partial_{j,i} = 1 \quad \text{iff} \quad \pi_j = \pi_{i,[k]}.$$

If $\pi_{i,[k]}$ is forbidden, the corresponding row remains isolated, preventing invalid homology classes from forming.

5.3 Persistence Computation

We reduce the boundary matrix ∂ using a standard persistent homology algorithm (e.g., PHAT (Bauer, Kerber, Reininghaus, and Wagner 2017)) to identify persistence pairs:

This yields persistence diagrams by recording the birth and death times t_i and t_j of features in each homological dimension. These diagrams can then be transformed into vector representations, such as persistence images, for downstream tasks.

5.4 Comparison with Simplicial Persistent Homology

- **Face Validity:** Simplicial PH assumes all faces of a simplex are present; PPH verifies each directed subpath individually.
- **Dimensions:** A directed path of length $p+1$ corresponds to a p-dimensional feature in $H_p(G)$.
- **Forbidden Faces:** Invalid paths rows are placed at the bottom of a boundary matrix, avoiding considering false features.

6 Link Prediction on Graphs

Link prediction on graphs is fundamentally a **binary classification task** that estimates the likelihood of unobserved edges between node pairs. Formally, given a graph $G = (V, E)$ with adjacency matrix A, the goal is to classify pairs $(i, j) \notin E$ as either edges ($A_{ij} = 1$) or non-edges ($A_{ij} = 0$). This problem arises in three key scenarios:

- **Missing Edge Recovery:** Detecting erroneously omitted connections (e.g., unrecorded protein interactions).
- **Temporal Forecasting:** Predicting future relationships (e.g., social connections or citations).
- **Network Completion:** Inferring missing links in partially observed networks.

6.1 Modeling Approaches

Methods for link prediction fall into two categories:

Heuristic (Similarity-Based) Methods These unsupervised approaches compute similarity scores s_{ij} between nodes as proxy measures for classification. Common metrics include:

- Local similarity: Common neighbors, Jaccard index, Adamic–Adar
- Global similarity: Katz index, SimRank

While efficient, these scores lack learned discriminative power and often serve as features for downstream classifiers.

Learning-Based Classification Supervised models treat link prediction as a binary classification problem:

- **Features:** Handcrafted topological metrics or learned representations (e.g., Node2Vec, GNN embeddings)
- **Classifier:** Logistic regression, neural networks, or pairwise scoring functions (e.g., dot product of embeddings)
- **Training:** Observed edges as positive class, with negative examples sampled from non-edges

The standard evaluation metric is the **Area Under the ROC Curve (AUC)**, which measures the probability that a randomly chosen true edge ranks above a randomly chosen non-edge.

6.2 Link Prediction in Directed Graphs

Link prediction in digraphs aims to estimate the probability that a directed edge (u, v) will appear between two nodes in a graph, i.e. to predict $A_{uv} \in \{0, 1\}$ for ordered pair (u, v) not in E. In directed graphs, unlike undirected ones, $A_{uv} \neq A_{vu}$, which invalidates many classical symmetry-based heuristics and necessitates specialized methods.

7 Node2Vec for Directed Graph Representation Learning

Node2Vec (Grover and Leskovec 2016) constructs continuous vector representations for graph nodes, much like Word2Vec produces word embeddings from text. In Word2Vec, similar words co-occur in contexts; in Node2Vec, similar nodes co-occur on random walks "sentences") through the graph.

7.1 Algorithm Overview

Let $\mathscr{C} = \emptyset$ be our corpus of directed walks. We then train a skip-gram model (Mikolov et al. 2013) to maximize the likelihood of observing each node's sampled neighborhood $N(u)$.

Input: $G = (V, E)$, embed. dim. d, walks per node r, walk length l, return p, in–out q
Output: $X \in \mathbb{R}^{|V| \times d}$
procedure LEARNFEATURES
    Precompute $\pi_{v,x}$ for all $(v, x) \in E$
    for each $u \in V$ **do**
        for $i = 1$ to r **do**
            $W \leftarrow [u]$ ▷ init walk
            for $j = 1$ to $l - 1$ **do**
                $v \leftarrow W[-1], \quad x \sim P(\cdot \mid v)$
                Append x to W
            end for
            Add W to $\mathscr{C}$
        end for
    end for
    Train skip-gram on $\mathscr{C}$ to maximize

$$\sum_{u \in V} \log P\big(N(u) \mid f(u)\big) \quad \text{(skip-gram likelihood)}$$

end procedure

7.2 Integration with Graph Neural Networks

Having obtained Node2Vec embeddings, we initialize our GNN's node features with these vectors before message passing. Empirical studies show that this initialization often yields higher AUC on link prediction and node classification than training from scratch (Gupta et al. 2021; Liao et al. 2023).

8 Basic Graph Neural Network Models

Before extending to directed architectures, we first review three foundational GNNs that operate on undirected graphs.

8.1 Graph Convolutional Network (GCN)

Kipf & Welling (Kipf and Welling 2017) proposed the GCN layer as a spectral-spatial convolution that aggregates normalized neighbor features. Given adjacency A, let $\hat{A} = A + I$ and $\hat{D} = \mathrm{diag}(\hat{A}\,\mathbf{1})$. At layer l, with node features $H^{(l)}$ and weight $W^{(l)}$,

$$H^{(l+1)} = \sigma\Big(\hat{D}^{-\frac{1}{2}}\,\hat{A}\,\hat{D}^{-\frac{1}{2}}\,H^{(l)}\,W^{(l)}\Big).$$

This formulation ensures symmetric normalization and enables efficient batch processing (Kipf and Welling 2017).

8.2 GraphSAGE

Hamilton et al. (Hamilton, Ying, and Leskovec 2017) introduced GraphSAGE, which generalizes GNNs to inductive settings via neighbor sampling and learnable aggregators. For each node v,

$$h^{(l)}_{\mathcal{N}(v)} = \mathrm{AGG}\big(\{\,h^{(l)}_u : u \in \mathcal{N}(v)\}\big), \quad h^{(l+1)}_v = \sigma\Big(W^{(l)}\big[h^{(l)}_v \,\|\, h^{(l)}_{\mathcal{N}(v)}\big]\Big).$$

Here AGG can be mean, LSTM, or pooling, enabling flexibility and scalability on large graphs (Hamilton, Ying, and Leskovec 2017).

8.3 Graph Attention Network (GAT)

Veličković et al. (Veličković et al. 2018) incorporated self-attention into GNNs to learn attention coefficients over neighbors. For each edge $u \to v$,

$$e_{vu} = \mathrm{LeakyReLU}\big(\mathbf{a}^\top[W\,h_v \,\|\, W\,h_u]\big), \quad \alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k\in\mathcal{N}(v)}\exp(e_{vk})},$$

and the update is

$$h^{(l+1)}_v = \sigma\Big(\sum_{u\in\mathcal{N}(v)} \alpha_{vu}\,W\,h_u\Big).$$

Multi-head attention further stabilizes learning and improves expressivity (Veličković et al. 2018).

9 Graph Neural Networks and Topological Awareness

Graph Neural Networks (GNNs) excel at learning from graph-structured data via local message passing. However, this locality limits their expressive power to that of the 1-dimensional Weisfeiler–Lehman test, making them blind to global cycles or holes and unable to distinguish graphs with identical local patterns but different topology.

9.1 Example 1: Distinct Betti Numbers

Both graphs below share identical degree sequences and local substructures, yet their first Betti numbers (β_1) differ.

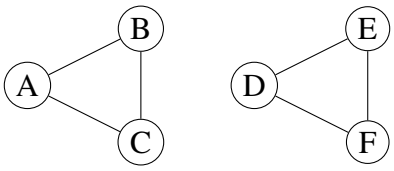

Figure 5: Graph G_1: two disconnected triangles ($\beta_1 = 2$).

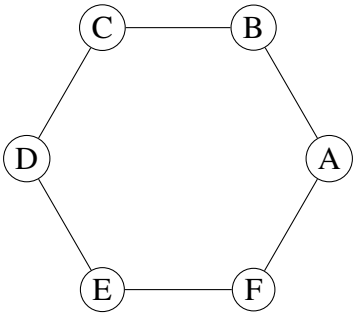

Figure 6: Graph G_2: a single hexagon ($\beta_1 = 1$).

These two examples show that purely local GNNs cannot tell apart G_1 and G_2, even though $\beta_1(G_1) = 2$ and $\beta_1(G_2) = 1$.

9.2 Example 2: Symmetric Link Prediction Failure

Start with a regular hexagon on vertices A, B, C, D, E, F. Remove edges (A, F) and (C, D), then add the chord (D, E). All vertices still share identical local features, yet the geodesic distance between A and C is 2 while between A and D it is 3. A GNN with only local aggregation would predict the same link probability for both pairs.

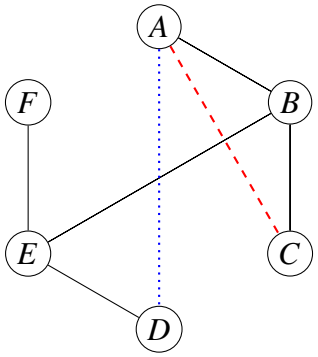

Figure 7: Graph H: red dashed is (A, C) (dist. 2), blue dotted is (A, D) (dist. 3).

9.3 Bridging to Topological Features

These examples illustrate that local message passing alone can't capture fundamental topological or metric differences. We therefore propose to integrate a Persistent Homology pipeline into GNN link prediction frameworks.

9.4 Prior Works: Integrating Persistent Homology with GNNs

Persistent homology (PH) has been grafted onto GNNs along two complementary paradigms: learnable filtrations with differentiable persistence, and explicit topological layers that inject global cycle information.

1. Learnable Filtrations & Differentiable Persistence Hofer *et al.* first introduced "Graph Filtration Learning," where a parameterized function maps node (or edge) features to filtration values and backpropagates through a differentiable persistence layer (e.g., PersLay) to learn task-specific topological signatures (Hofer et al. 2020). This end-to-end framework aligns PH invariants with graph- and node-level objectives.

2. Topological Graph Layers (TOGL) Horn *et al.* proposed the Topological Graph Layer (TOGL), which computes PH on vertex-induced filtrations within a GNN and concatenates persistence-based descriptors (e.g., persistence images or silhouettes) to the learned embeddings, strictly increasing expressive power beyond 1-WL GNNs (Horn et al. 2022b). TOGL has shown superior performance on both synthetic benchmarks sensitive to cycle structure and real-world graph classification tasks.

10 Directed Graph Neural Networks

As opposed to basic GNNs that ignore orientation on the edges, directed GNNs (dirGNNs) explicitly model edge direction in their message-passing, enabling them to capture source-to-target semantics and asymmetric influence patterns that undirected GNNs miss.

10.1 Directed Graph Convolutional Networks (dirGCN)

To distinguish incoming from outgoing information, dirGCN uses two separate convolution paths. Let

$$A_{\text{in}} = A^{\top}, \quad A_{\text{out}} = A,$$

and $\hat{A}_{\text{dir}} = A_{\text{dir}} + I$, $\hat{D}_{\text{dir}} = \text{diag}(\hat{A}_{\text{dir}}\mathbf{1})$. At layer l, with node features $H^{(l)}$ and learnable weights $W_{\text{in}}, W_{\text{out}}$,

$$H^{(l+1)} = \sigma\Big(\hat{D}_{\text{in}}^{-1/2}\hat{A}_{\text{in}}\hat{D}_{\text{in}}^{-1/2}H^{(l)}W_{\text{in}} + \hat{D}_{\text{out}}^{-1/2}\hat{A}_{\text{out}}\hat{D}_{\text{out}}^{-1/2}H^{(l)}W_{\text{out}}\Big). \tag{4}$$

This preserves directional flows while maintaining computational efficiency comparable to the standard GCN model (Kipf and Welling 2017).

10.2 Directed GraphSAGE (dirGraphSage)

Next, dirGraphSAGE augments neighbor sampling with separate in- and out-aggregations. For each node v,

$$h_{\mathcal{N}(v)}^{(l)} = \text{AGG}_{\text{in}}\big(\{h_u^{(l)} : u \in \mathcal{N}_{\text{in}}(v)\}\big) \,\big\|\, \text{AGG}_{\text{out}}\big(\{h_u^{(l)} : u \in \mathcal{N}_{\text{out}}(v)\}\big),$$

and then

$$h_v^{(l+1)} = \sigma\big(W^{(l)}[h_v^{(l)} \,\|\, h_{\mathcal{N}(v)}^{(l)}]\big).$$

Separating in- and out-neighbors lets the model learn asymmetric structural roles, useful in recommendation or influence modeling (Hamilton, Ying, and Leskovec 2017).

10.3 Directed Graph Attention Networks (dirGATconv)

Finally, dirGATconv extends the attention mechanism with direction-specific parameters. For each neighbor u of v,

$$\alpha_{vu}^{\text{dir}} = \frac{\exp\big(\text{LeakyReLU}(\mathbf{a}_{\text{dir}}^{\top}[Wh_v \,\|\, Wh_u])\big)}{\sum_{k\in\mathcal{N}_{\text{dir}}(v)}\exp\big(\text{LeakyReLU}(\mathbf{a}_{\text{dir}}^{\top}[Wh_v \,\|\, Wh_k])\big)},$$

with $\text{dir} \in \{\text{in}, \text{out}\}$. The update is then

$$h_v^{(l+1)} = \sigma\Big(\sum_{u\in\mathcal{N}_{\text{in}}(v)}\alpha_{vu}^{\text{in}}Wh_u + \sum_{u\in\mathcal{N}_{\text{out}}(v)}\alpha_{vu}^{\text{out}}Wh_u\Big).$$

This lets the network dynamically attend to correct direction-specific influences (Veličković et al. 2018).

11 Proposed method: Local Persisten Path Homology

Our proposed directed edge invariant, LPPH designed for link prediction in directed graphs. It is computed by constructing a descriptive edge-centric subgraph filtration (partially inspired by the vertex-centered approach in Minghua Wang et al. “Persistent Local Homology in Graph

Learning" (M. Wang et al. 2024), but instead of vertex-centered, we will have edge-centered subgraph) defined by the ordered pair of nodes (edge candidate) and then performing persistent image calculations on this filtration. Aside from persistent image hyperparameters, it will also have a "filtration depth" and "maximal feature dimension" as hyperparameter.

11.1 Model Components

1. **Edge-Centric Filtration:** Extract a local subgraph around (u,v) via alternating forward/backward expansions (Sec. 11.2), assigning each path a birth time.
2. **Persistent Diagram:** Compute the persistence pairs (b,d) of path homology classes up to dimension `max_dim`, capturing how directed cycles and flows emerge and vanish as the subgraph grows.
3. **Persistent Image:** Transform the diagram into a stable grid-based feature vector (the "persistent image"), which preserves both the lifespan $(d-b)$ and scale (b,d) information.

11.2 Edge-Centric Filtration Construction

We consider a target edge (u,v) in a directed graph and construct a direction-aware subgraph that captures asymmetric connectivity patterns crucial for persistent path homology. Our method iteratively expands a subgraph around (u,v) via four alternating phases, where each cycle corresponds to a single expansion step until a predefined maximum depth (`max_depth`) is reached.

Algorithm Outline: For each target edge (u,v):

1. **Initialization:**
   - Begin with a subgraph containing the nodes u and v, but exclude the target edge (u,v) (as the model should not see it to prevent data leakage).
   - Initialize four node sets corresponding to the expansion fronts:

$$\begin{aligned} N_0 &: \text{ Forward frontier from } u, \\ N_1 &: \text{ Backward frontier toward } u, \\ N_2 &: \text{ Forward frontier from } v, \\ N_3 &: \text{ Backward frontier toward } v. \end{aligned}$$

2. **Phased Expansion:** For each expansion step t (with $0 \leq t <$ `max_depth`), use modulo arithmetic to select the phase:

- **Phase 0 (Forward from u):** For every node in N_0, add outgoing edges (u,w) not already in the subgraph (and excluding (u,v)). Update N_0 with newly reached nodes.
- **Phase 1 (Backward to u):** For every node in N_1, add incoming edges (w,u), then update N_1 accordingly.
- **Phase 2 (Forward from v):** For every node in N_2, add outgoing edges (v,w) and update N_2.
- **Phase 3 (Backward to v):** For every node in N_3, add incoming edges (w,v) and update N_3.

3. **Path Generation:** Every new edge introduced at step t is recorded as a 1-path. Then, for $2 \leq p \leq$ `max_dim`, extend the paths by appending or prepending nodes that are adjacent in the subgraph:

$$\mathscr{P}_k = \{(v_0,\ldots,v_{k-1},v_k) \mid (v_{k-1},v_k) \in E\} \cup \{(v_{-1},v_0,\ldots,v_{k-1}) \mid (v_{-1},v_0) \in E\}.$$

   By calculating all paths during the filtration computation we essentially compute path ordering for the boundary matrix for the persistent homology.

4. **Filtration Time Assignment:** Set the filtration time for each element σ as

$$t(\sigma) = \begin{cases} 0, & \text{if } \sigma \in \{u,v\}, \\ t+1, & \text{if the element is introduced during expansion step } t. \end{cases}$$

The Python implementation mirrors this process by maintaining separate node sets (e.g., `nodes_0`, `nodes_1`, `nodes_2`, `nodes_3`) and iterating over steps using a modulo-4 cycle to alternate between the four phases. This method allows to collect the neighborhood subgraph that has all necessary information about the probability of a given oriented edge in a descriptive way. We differentiate different type of edges and prioritize links that are closer to the target nodes (features connected to them will be "born" earlier hence having longer lifespan/persistance).

Illustration of the Filtration Process: Figure 8 provides an example of the filtration for a target edge (u,v) with `max_depth`=13. In the figure, the dashed line represents the target edge (which is excluded from the subgraph), and colored arrows indicate the expansion phase and step.

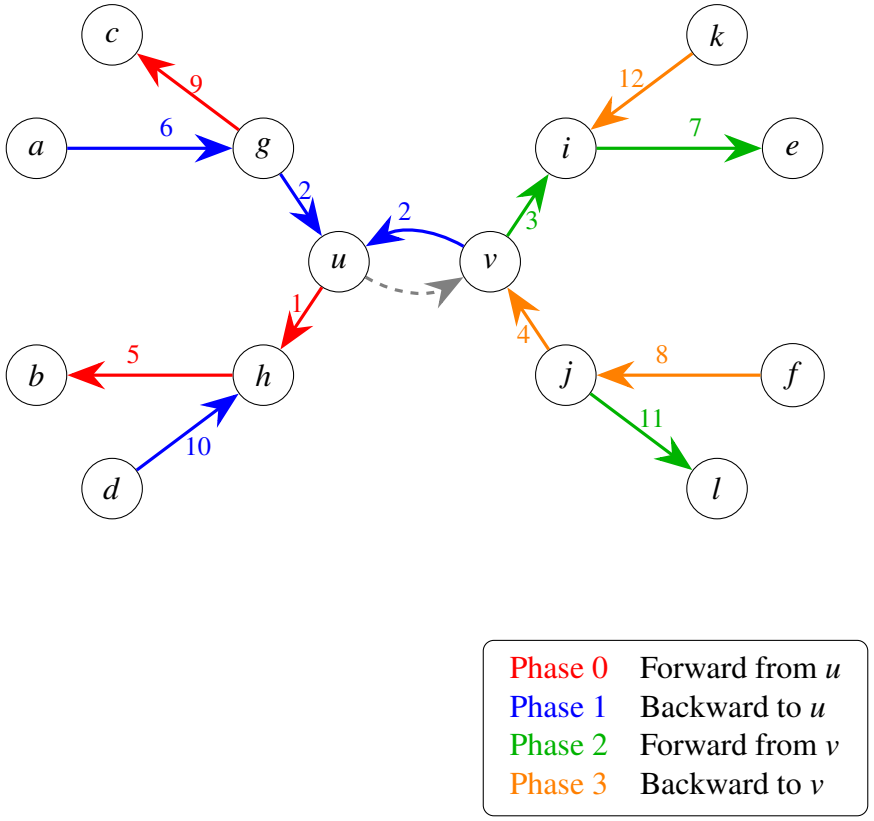

Figure 8: Edge-centric filtration process for target edge (u, v) with `max_depth`=13. The dashed line represents the excluded target edge, while colored arrows indicate the expansion phase and step.

11.3 Example: LPPH Calculation

We demonstrate a complete LPPH calculation for an edge-centric filtration in figure 8 up to `max_depth`=4 and `max_dim`=1 (homologies will be of dimension 0, 1), showing each step.

First, we need to calculate the boundary matrix. The paths and their appearance time are as follows (as we will only look at homologies up to the second dimension, we will not track paths longer than 3 edges):

Table 1: Paths and associated filtration times

Path	**Filtration Time (ε)**
(u)	0 (initial node)
(v)	0 (initial node)
(h)	1
(u,h)	1
(g)	2
(g,u)	2
(v,u)	2
(g,u,h)	2
(i)	3
(v,i)	3
(j)	4
(j,v)	4
(j,v,u)	4
(j,v,u,h)	4

The boundary operators in matrix in $\mathbf{Z}_2$ is as follows (dduring boundary operation computing we get two "forbidden" paths with dimension $<= 1$ which we add to the end of the table):

time		0		1		2				3		4		
	paths	(u)	(v)	(h)	(u,h)	(g)	(g,u)	(v,u)	(g,u,h)	(i)	(v,i)	(j)	(j,v)	(j,v,u)
0	(u)	0	0	0	1	0	1	1	0	0	0	0	0	0
0	(v)	0	0	0	0	0	0	1	0	0	1	0	1	0
1	(h)	0	0	0	1	0	0	0	0	0	0	0	0	0
1	(u,h)	0	0	0	0	0	0	0	1	0	0	0	0	0
2	(g)	0	0	0	0	0	1	0	0	0	0	0	0	0
2	(g,u)	0	0	0	0	0	0	0	1	0	0	0	0	0
2	(v,u)	0	0	0	0	0	0	0	0	0	0	0	0	1
2	(g,u,h)	0	0	0	0	0	0	0	0	0	0	0	0	0
3	(i)	0	0	0	0	0	0	0	0	0	1	0	0	0
3	(v,i)	0	0	0	0	0	0	0	0	0	0	0	0	0
4	(j)	0	0	0	0	0	0	0	0	0	0	0	1	0
4	(j,v)	0	0	0	0	0	0	0	0	0	0	0	0	1
4	(j,v,u)	0	0	0	0	0	0	0	0	0	0	0	0	0
4	(g,v,u,h)	0	0	0	0	0	0	0	0	0	0	0	0	0
∞	(g,h)	0	0	0	0	0	0	0	1	0	0	0	0	0
∞	(j,u)	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 2: Boundary Matrix

11.4 Matrix Reduction Process

We now apply the standard column-reduction algorithm over $\mathbf{Z}_2$ to the boundary matrix in Table 2. Recall alg. 1:

- We process columns in nondecreasing order of their filtration times.
- For each column c, we compute

$$\text{low}(c) \;=\; \max\{\, i : \partial_{i,j} = 1 \}\,,$$

  and then, as long as there exists a prior column $k < c$ with $\text{low}(k) = \text{low}(c)$, we add column k to column c (mod 2).
- If after this loop column c becomes entirely zero, it indicates that column c has reduced out a previous pivot; we record a death pairing between the class born at position $\text{low}(c)$ and the time of column c. If column c never has a pivot (i.e. remains zero throughout), it represents a homology class born at the filtration time of c that never dies (persists to ∞).

Persistence pairs. In our edge-centric filtration up to `max_depth` $= 4$:

- Two H_0 classes are born at time 0, corresponding to the isolated nodes u and v. - When the edge (v,u) appears at $t = 2$, one of those two classes dies (they merge into a single component). - The remaining connected component never dies within the filtration; to ensure it is represented in the persistent image, we assign its death time to `max_depth` $+ 1$.

No nontrivial H_1 classes appear, since no cycles form by depth 4.

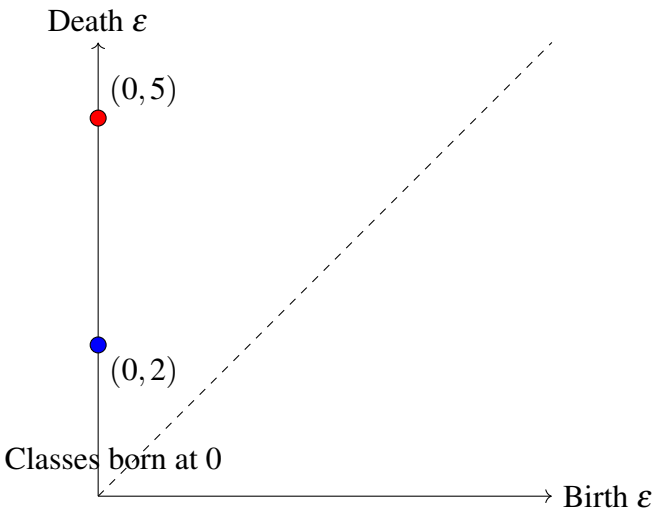

Figure 9: H_0 persistence diagram for the worked example (one class dies at $\varepsilon = 2$, the other persists and is capped at $\varepsilon = 5$ for the persistent image).

Computing persistent images to our calculated persistent diagrams (H_0 diagram is in fig. 9 and H_1 diagram has no features) and concatenating them we complete the worked example of LPPH via matrix reduction up to dimension 1 and depth 4.

12 Complexity Analysis

In this section we provide a detailed account of both the time and space complexity of the routine `compute_edge_feat(graph, edge, max_depth, pixel_size, max_dim, sigma)`. We measure complexity in terms of:

- V: number of vertices in the graph,
- E: number of edges in the graph,
- Δ: maximum degree of any vertex,
- k: filtration depth parameter (`max_depth`),
- d: homology dimension parameter (`max_dim`),
- p: approximate side length of the persistence-image grid, given by

$$p \approx \frac{k}{\text{pixel\_size}}.$$

12.1 Overview of Algorithmic Stages

The computation proceeds in four main stages:

1. *Edge Filtration*: Collect all simple paths (and singleton nodes) emanating from the target edge up to length d, over k breadth-first expansion steps.
2. *Boundary Matrix Construction*: Assemble a sparse boundary matrix with one column per path and one row per *face* of a path. "Forbidden" face-paths (those not in the original path list) are appended as extra rows.
3. *Persistence Pairing*: Reduce this (rectangular) matrix via PHAT (Bauer, Kerber, Reininghaus, and Wagner 2017) to compute birth–death pairs in each dimension $\leq d$.
4. *Persistence Image Generation*: Convolve each diagram with a Gaussian and rasterize onto a $p \times p$ grid for each homology dimension.

12.2 Time Complexity

12.2.1 1. Edge Filtration

As before,

$$T_{\text{filt}} = O\big(k \cdot (E + \Delta^d)\big),$$

since we scan all edges over k steps and enumerate up to $O(\Delta^d)$ simple paths of length $\leq d$.

12.2.2 2. Boundary Matrix Construction

Let

$$N \;=\; |\{\text{paths of length} \le d\}| \;=\; O(\Delta^d).$$

For each of the N original paths of length $\ell \le d$, we list its ℓ faces (subpaths of length $\ell-1$). If a face-path has not appeared among the originals, it becomes a "forbidden path" and is assigned a new row index. Denote by

$$F \;=\; \big|\{\text{forbidden paths}\}\big|.$$

In the worst case each face of each path is new, so

$$F \;=\; O\big(\sum_{\ell=1}^{d} \ell\,\Delta^{\ell}\big) = O\big(d\,\Delta^d\big).$$

Thus the boundary matrix has

$$\text{rows} \;=\; N+F = O\big(d\,\Delta^d\big), \quad \text{cols} \;=\; N = O\big(\Delta^d\big), \quad \text{nonzeros} \;=\; O(dN).$$

Building it costs

$$T_{\text{boundary}} = O\big(\text{nonzeros}\big) = O\big(d\,\Delta^d\big).$$

12.2.3 3. Persistence Pairing

We reduce an $(N+F)\times N$ sparse matrix with $O(dN)$ nonzeros. While worst-case reduction is $O(N^3)$, in sparse practice one observes

$$T_{\text{pairing}} = O\big(d\,N^2\big) = O\big(d\,\Delta^{2d}\big).$$

12.2.4 4. Persistence Image Generation

As before,

$$T_{\text{image}} = O\big(N\,p^2\big) = O\big(\Delta^d\,p^2\big),$$

plus $O(d\,p^2)$ overhead for multiple dimensions.

Total Time. Collecting dominant terms,

$$T_{\text{total}} = O\big(kE \;+\; k\Delta^d \;+\; d\Delta^d \;+\; d\Delta^{2d} \;+\; \Delta^d\,p^2\big) = O\big(kE + k\Delta^d + d\Delta^{2d} + \Delta^d p^2\big),$$

showing exponential dependence on d and polynomial dependence on (V,E,k,p).

12.3 Space Complexity

$$
\begin{aligned}
S_{\text{paths}} &= O(N) = O(\Delta^d), \\
S_{\text{matrix}} &= O(\text{rows} + \text{nonzeros}) = O(d\,\Delta^d + d\,\Delta^d) = O(d\,\Delta^d), \\
S_{\text{image}} &= O(d\,p^2).
\end{aligned}
$$

Hence

$$
S_{\text{total}} = O\big(d\,\Delta^d + d\,p^2\big).
$$

12.4 Practical Considerations

- **Boundary-matrix shape:** The addition of $F = O(d\,\Delta^d)$ forbidden-path rows yields a tall matrix, increasing both storage and the nonzero count but not the number of columns.
- **Dimensionality bottleneck:** Δ^d (and hence $d\,\Delta^d$) grows rapidly; capping d or pruning is essential.
- **Sparse vs. worst-case:** In real graphs neighborhoods are smaller, so practical runtimes are often much lower.

13 Incorporating LPPH into Directed GNNs

To leverage global topological information alongside directional message passing, we augment a dirGNN encoder with LPPH features via a skip connection into the link decoder. Concretely, for each candidate edge (u,v) we compute an LPPH vector $\phi_{u,v} \in \mathbb{R}^k$ and concatenate it with the learned node embeddings before decoding.

Algorithm 2 Link Prediction with dirGNN + LPPH

Input:

- Directed graph $G = (V, E)$ and node features $X \in \mathbb{R}^{|V| \times d}$ (e.g., node2vec).
- LPPH parameters: `max_dim`, `max_depth`, image resolution r.
- dirGNN hyperparameter: number of layers L.

Output: Trained link-prediction model $\mathscr{M}$.
procedure PREPAREDATA
    Sample negative edges E_{neg} of size $|E|$ disjoint from E
    Randomly split $E \cup E_{\text{neg}}$ into E_{train} and E_{test} (stratified by edge existence)
    Remove E_{test} from G to form $G_{\text{train}} = (V, E_{\text{train}})$ (to avoid data leakage)
    return $G_{\text{train}}, E_{\text{train}}, E_{\text{test}}, E_{\text{neg}}$
end procedure
procedure COMPUTELPPH($G_{\text{train}}, \mathscr{E}$)
    for all $(u, v) \in \mathscr{E}$ **do**
        Build edge-centric filtration $\mathscr{F}_{u \to v}$ on G_{train}
        Compute LPPH vectors ($\phi_{u,v} \in \mathbb{R}^k$) up to `max_dim`, `max_depth`
    end for
    return $\{\phi_{u,v}\}$
end procedure
procedure TRAIN
    $(G_{\text{train}}, E_{\text{train}}, E_{\text{test}}, E_{\text{neg}})$ = PREPAREDATA
    Precompute $\{\phi_{u,v}\}$ = COMPUTELPPH($G_{\text{train}}, E_{\text{train}} \cup E_{\text{test}}$)
    Initialize: $H \leftarrow \text{dirGNN}_{\text{enc}}(G_{\text{train}}, X, L)$ ▷ Encodes with L layers optimizer, MLP decoder, sigmoid σ
    for epoch = 1 to N_{epochs} **do**
        $\mathscr{L} = 0$
        for all $(u, v, y) \in \{(e, 1) \mid e \in E_{\text{train}} \setminus E_{\text{neg}}\} \cup \{(e, 0) \mid e \in E_{\text{train}} \cap E_{\text{neg}}\}$ **do**
            $\hat{y} = \sigma(\text{MLP}_{\text{dec}}([H[u] \| H[v] \| \phi_{u,v}]))$
            $\mathscr{L} += \text{BCE}(y, \hat{y})$
        end for
        optimizer.step($\nabla \mathscr{L}$)
    end for
end procedure
procedure EVALUATE
    Compute AUC-ROC on E_{test} using G_{train} embeddings and $\phi_{u,v}$
end procedure

14 Testing and Results

We evaluated the dirGNN+LPPH model on 11 real-world directed graphs. The dataset consists of a social network where nodes represent users and edges represent directed interactions between users. We compared against baseline dirGNN variants without topological features, demonstrating improvements on 10/11 datasets.

15 Datasets

We evaluate our model on 11 directed networks spanning five domains. Each dataset represents unique directional relationships in real-world systems.

15.1 Network Classification and Descriptions

- **Social Interaction Networks**
  - **Bison**: Directed dominance interactions within a bison herd.
  - **Highschool**: Temporal face-to-face contacts among students.
  - **Japanese Macaques**: Dominance behaviour in a colony of 62 adult female Japanese macaques
- **Biological Networks**
  - **Caenorhabditis elegans (neural)**: A weighted directed network representing the neural connectome of *C. elegans*
  - **Figeys**: Protein–protein interaction network in human cells from a mass-spectrometry study
  - **Stelzl**: Directed yeast two-hybrid protein interaction network in human cells
- **Ecological Systems**
  - **Florida Ecosystem Dry**: Trophic interactions under seasonal dry conditions.
  - **Little Rock Lake**: Freshwater food web of Little Rock Lake, Wisconsin
- **Human Organizational Networks**
  - **Congress Votes**: Directed voting similarity network in the U.S. Congress.
  - **Physicians**: Patient-referral network among doctors.
- **Transportation Networks**
  - **Air Traffic Control**: Airports and service centers connected by preferred-route recommendations from the NFDC

15.2 Dataset Statistics

Table 3: Directed network dataset statistics.

Dataset	Nodes	Edges	Domain
Bison	26	314	Social
Highschool	70	366	Social
Caenorhabditis	297	4296	Biological
Congress Vote	219	764	Organizational
Florida Ecosystem Dry	128	2137	Ecological
Japanese Macaques	62	1187	Social
Little Rock Lake	183	2494	Ecological
Physicians	241	1098	Organizational
Figeys	2239	6452	Biological
Stelzl	1706	6207	Biological
Air Traffic Control	1226	2615	Transportation

This comprehensive collection enables rigorous evaluation of our model's ability to handle different directional relationship types across domains.

15.3 Analysis of LPPH Parameter Effectiveness

Across our eleven directed networks, the best dirGNN + LPPH variant outperforms its non-topological baseline on 10/11 datasets, with an average AUC-ROC uplift of +8.3% (range: +0.4%...+27.8%).

Scale-Aware Depth Selection Comparing Table 3 (network size/density) with the chosen filtration depths ($\varepsilon^* = 5$ vs. 9 in Table 4):

- *Dense/small graphs* (e.g., Bison: 26 nodes/314 edges; C. elegans: 297 nodes/4296 edges) consistently favor $\varepsilon = 5$, avoiding noisy long-path features in tightly connected clusters.
- *Sparse/large graphs* (e.g., Highschool: 70 nodes/366 edges temporal; Air Traffic: 1226 nodes/2615 edges; Stelzl: 1706 nodes/6207 edges) uniformly use $\varepsilon = 9$, leveraging extended paths to capture long-range dependencies.

Domain-Specific Gains

- **Social networks** exhibit mixed outcomes: Bison sees a slight drop (-1.4%), whereas Highschool jumps +27.8%, highlighting how temporal contact patterns may or may not benefit from path enrichment.

- **Biological networks** (C. elegans, Figeys, Stelzl) obtain steady moderate gains (+1.4...+8.4%), with immediate synaptic or interaction motifs best captured at shallow depths.
- **Ecological webs** (Florida Ecosystem, Little Rock Lake) show marginal improvements (+0.5...+1.0%), suggesting local trophic links already suffice for accurate prediction.
- **Organizational/transportation** graphs (Congress Vote, Physicians, Air Traffic) enjoy clear benefits (+5...+11%), reflecting the importance of multi-step referral or route chains.

Architecture Synergy

- **dirGAT** leads on 6/11 datasets and records the largest single-dataset gain (+27.8% on Highschool), indicating its attention mechanism effectively highlights the most informative topological features.
- **dirGraphSAGE** wins on 2/11 (Congress Vote, Japanese Macaques), showing that flexible neighbor aggregation can sometimes better integrate path signals in medium-sized graphs.
- **dirGCN** tops only on Little Rock Lake—a highly structured food web—implying that uniform averaging with modest topological enrichment is ideal for near-hierarchical networks.

These findings suggest practitioners should:

1. Tune filtration depth in accordance with graph density: use $\varepsilon = 5$ for dense clusters, $\varepsilon = 9$ for sparse/topologically extended networks.
2. Favor attention-based encoders (dirGAT) when combining LPPH with GNNs, especially on large or asymmetric datasets.
3. Expect only marginal improvements in very dense domains; focus on minimal depths and simpler architectures there.

Table 4: Best LPPH parameter (5 or 9) for each dataset and model, with mean AUC scores. Best models are marked with bold text.

Dataset	Model	Edge depth	With LPPH	Without LPPH
bison	dirGCN	5	0.7583 ± 0.031	0.7690 ± 0.055
	dirSage	5	0.7885 ± 0.047	**0.8131 ± 0.029**
	dirGAT	9	0.7873 ± 0.025	0.7605 ± 0.092
highschool	dirGCN	9	0.8177 ± 0.052	0.6398 ± 0.114
	dirSage	9	0.8120 ± 0.063	0.7149 ± 0.104
	dirGAT	9	**0.8252 ± 0.054**	0.7224 ± 0.092
caenorhabditis	dirGCN	5	**0.9296 ± 0.007**	0.8834 ± 0.035
	dirSage	5	0.9099 ± 0.016	0.8927 ± 0.013
	dirGAT	5	0.9193 ± 0.006	0.9014 ± 0.008
congress_vote	dirGCN	5	0.8341 ± 0.038	0.8218 ± 0.041
	dirSage	9	**0.8689 ± 0.038**	0.8470 ± 0.031
	dirGAT	9	0.8639 ± 0.043	0.7713 ± 0.055
florida_ecosystem_dry	dirGCN	5	0.9445 ± 0.009	0.9455 ± 0.010
	dirSage	5	0.9440 ± 0.010	0.9430 ± 0.011
	dirGAT	5	**0.9498 ± 0.011**	0.9402 ± 0.010
japanese_macaques	dirGCN	5	0.8821 ± 0.023	0.8828 ± 0.020
	dirSage	9	**0.8976 ± 0.024**	0.8839 ± 0.023
	dirGAT	5	0.8820 ± 0.024	0.8765 ± 0.025
little_rock_lake	dirGCN	5	**0.9918 ± 0.003**	0.9904 ± 0.004
	dirSage	5	0.9893 ± 0.008	0.9857 ± 0.006
	dirGAT	5	0.9882 ± 0.010	0.9865 ± 0.007
physicians	dirGCN	9	0.8670 ± 0.041	0.8813 ± 0.029
	dirSage	9	0.8995 ± 0.021	0.8859 ± 0.021
	dirGAT	5	**0.9123 ± 0.020**	0.8872 ± 0.023
figeys	dirGCN	9	0.9709 ± 0.002	0.9679 ± 0.004
	dirSage	5	0.9723 ± 0.002	0.9348 ± 0.008
	dirGAT	5	**0.9745 ± 0.003**	0.9613 ± 0.005
stelzl	dirGCN	9	0.8616 ± 0.015	0.8513 ± 0.010
	dirSage	9	0.8653 ± 0.017	0.7705 ± 0.017
	dirGAT	9	**0.8878 ± 0.012**	0.7565 ± 0.012
air_traffic_control	dirGCN	9	0.6723 ± 0.038	0.5793 ± 0.024
	dirSage	9	0.6920 ± 0.038	0.6126 ± 0.020
	dirGAT	9	**0.7430 ± 0.030**	0.7014 ± 0.039

16 Comparative Analysis of LPPH as a Standalone Classifier

Our experiments reveal a counterintuitive result: training an XGBoost classifier solely on LPPH vectors outperforms both standalone dirGNNs and their LPPH-augmented hybrids on 8 of 11

datasets (Table 5). This suggests that the persistent path homology features themselves capture the core structural signals needed for link prediction, often more effectively than learned message passing.

Table 5: Best LPPH parameter (5 or 9) for each dataset and model, with mean AUC scores. Best models are marked with bold text.

Dataset	Model	Edge depth	mean AUC-ROC
bison	best dirGNN+LPPH	5	0.8131 ± 0.029
	LPPH+XGBoost	5 (max_dim = 3)	**0.8583 ± 0.046**
highschool	best dirGNN+LPPH	9	0.8252 ± 0.054
	LPPH+XGBoost	13	**0.8689 ± 0.033**
caenorhabditis	best dirGNN+LPPH	5	**0.9296 ± 0.007**
	LPPH+XGBoost	5 (max_dim = 3)	0.9265 ± 0.009
congress_vote	best dirGNN+LPPH	5	**0.8689 ± 0.038**
	LPPH+XGBoost	9	0.8637 ± 0.037
florida_ecosystem_dry	best dirGNN+LPPH	5	0.9498 ± 0.011
	LPPH+XGBoost	17	**0.9514 ± 0.013**
japanese_macaques	best dirGNN+LPPH	5	**0.8976 ± 0.024**
	LPPH+XGBoost	13	0.8854 ± 0.016
little_rock_lake	best dirGNN+LPPH	5	0.9918 ± 0.003
	LPPH+XGBoost	13	**0.9944 ± 0.002**
physicians	best dirGNN+LPPH	9	0.9123 ± 0.020
	LPPH+XGBoost	17	**0.9204 ± 0.020**
figeys	best dirGNN+LPPH	9	0.9745 ± 0.003
	LPPH+XGBoost	13	**0.9819 ± 0.003**
stelzl	best dirGNN+LPPH	9	0.8878 ± 0.012
	LPPH+XGBoost	13	**0.9652 ± 0.003**
air_traffic_control	best dirGNN+LPPH	9	0.7430 ± 0.030
	LPPH+XGBoost	9	**0.8508 ± 0.011**

16.1 Explanatory Factors

1. Feature Richness vs. Model Complexity LPPH vectors encode multi-scale cycle and path structures in a fixed-length representation. Tree-based learners like XGBoost can exploit these high-dimensional patterns without the overhead of learning graph convolutions, leading to stronger generalization on many datasets.

2. Overfitting in GNNs On smaller or simpler graphs (e.g. Bison, Japanese Macaques), dirGNNs can overfit the limited data, whereas XGBoost on LPPH features—with its regularized boosting—remains robust.

3. Capturing Long-Range Dependencies Persistent path homology naturally encodes long paths up to depth ε^*. In sparse graphs (e.g. Air Traffic Control), this explicit multi-hop information outperforms GNNs whose receptive field requires stacking many layers (and thus risks oversmoothing).

4. Hyperparameter Alignment The optimal LPPH depths for XGBoost (often deeper than for GNNs) suggest that the two methods prioritize different aspects of the same topological signal. Practitioners can therefore tune ε independently of GNN depth to match the classifier's needs.

16.2 Implications

Our findings challenge the notion that complex GNN architectures are always required for link prediction on directed graphs. In particular, LPPH demonstrates that carefully designed topological features alone can outperform or match deep models when:

- **Edge directionality matters:** Asymmetric roles (e.g. authority vs. hub) are naturally encoded by path-based homology.
- **Local structure is discriminative:** When neighborhood connectivity already separates link vs. non-link, LPPH highlights those patterns without extra message passing.
- **Node attributes are scarce or noisy:** In purely structural settings, LPPH vectors provide rich, stable descriptors where GNNs lack informative node signals.

Beyond serving as a competitive baseline, LPPH offers a *diagnostic tool*: if XGBoost on LPPH alone rivals or outperforms a GNN, it reveals that the link-prediction signal is primarily topological—and suggests where GNN architectures could be simplified or augmented to capture those same invariants.

17 Comparing to State-of-The-Art models

17.1 Model "SRTGCN"

The Social Ranking Theory-based Graph Convolutional Network (SRTGCN) (Liao et al. 2023) addresses directed link prediction through three key mechanisms: (1) social hierarchy modeling, (2) directional neighborhood partitioning, and (3) rank-aware feature aggregation. We outline its core components below.

17.1.1 Social Hierarchy Modeling

SRTGCN operationalizes social ranking theory through triad analysis of directed connections. For each target link (v_i, v_j), neighbors are categorized into three sets based on connection reciprocity and path directionality:

- **Same-ranked set** (E): Nodes connected via reciprocated edges (bidirectional links)
- **Higher-ranked set** (H): Nodes connected through unreciprocated out-links (lower→higher status)
- **Lower-ranked set** (L): Nodes connected via unreciprocated in-links (higher→lower status)

This partitioning enables differential treatment of social relationships during information propagation.

17.1.2 Multi-Perspective Neighborhood Aggregation

SRTGCN maintains three separate representation matrices that evolve across L graph convolution layers:

$$\begin{aligned}
\mathbf{H}^{E(l)} &= \sigma\left(\widetilde{\mathbf{D}}_R^{-1}\widetilde{\mathbf{R}}\mathbf{H}^{E(l-1)}\mathbf{W}^{E(l-1)}\right) \\
\mathbf{H}^{H(l)} &= \sigma\left(\widetilde{\mathbf{D}}_T^{-1}\widetilde{\mathbf{T}}\mathbf{H}^{H(l-1)}\mathbf{W}^{H(l-1)}\right) \\
\mathbf{H}^{L(l)} &= \sigma\left(\widetilde{\mathbf{D}}_S^{-1}\widetilde{\mathbf{S}}\mathbf{H}^{L(l-1)}\mathbf{W}^{L(l-1)}\right)
\end{aligned} \tag{5}$$

Where $\widetilde{\mathbf{R}}$, $\widetilde{\mathbf{T}}$, and $\widetilde{\mathbf{S}}$ represent normalized adjacency matrices for reciprocated, outgoing unreciprocated, and incoming unreciprocated connections respectively. This architecture preserves directional semantics through:

- Separate weight matrices ($\mathbf{W}^E$, $\mathbf{W}^H$, $\mathbf{W}^L$) for each relationship type
- Layer-wise concatenation of set-specific representations
- Adaptive normalization coefficients ($\widetilde{\mathbf{D}}$) per connection type

17.1.3 Architectural Pipeline

The complete framework operates through three stages:

Neighbor Subgraph Extraction For target link (v_i, v_j), extracts h-hop directional neighborhood G_{ij}^h containing:

$$V_{ij}^h = \{v_i, v_j\} \cup \bigcup_{k=1}^{h} (\Gamma^k(v_i) \cup \Gamma^k(v_j)) \tag{6}$$

Node Information Construction Combines structural features with learned embeddings:

- Structural labels via Weisfeiler-Lehman relabeling
- Node2vec embeddings for positional awareness
- Concatenated adjacency-directional matrices $\mathbf{A}_{ij}^h \oplus \mathbf{X}_{ij}^h$

Hierarchical Learning

1. Multi-layer graph convolution with Eqs. (6-7)
2. SortPooling for invariant node ordering
3. 1D convolution + dense layers for final prediction

This architecture enables simultaneous learning of local structural patterns and global social hierarchies through directional message passing. The separation of reciprocated/unreciprocated connections provides critical signal for predicting asymmetric relationships that baseline GCN architectures fail to capture.

17.1.4 Comparing Performance with Ours LPPH model

Table 6: Best LPPH parameter (5, 9, 13 or 17) for each dataset, with mean AUC scores. Best models are marked with bold text. Results for SRTGCN were taken from their paper (Liao et al. 2023)

Dataset	Model	Edge depth	mean AUC-ROC
Twitter	SRTGCN		0.9942
	LPPH+XGBoost	13	**0.9962 ± 0.0005**
Google+	SRTGCN		0.9876
	LPPH+XGBoost	5	**0.99878 ± 0.0004**
digg	SRTGCN		0.8817
	LPPH+XGBoost	5	**0.9155 ± 0.0013**
Slashdot	SRTGCN		0.92553
	LPPH+XGBoost	5	**0.9649 ± 0.0009**
Physiscians	SRTGCN		0.9213
	LPPH+XGBoost	17	**0.9246 ± 0.0149**
adolescent	SRTGCN		**0.9053**
	LPPH+XGBoost	17	0.8716 ± 0.0047
cora	SRTGCN		**0.9686**
	LPPH+XGBoost	13	0.9405 ± 0.023
Dblp	SRTGCN		0.9657
	LPPH+XGBoost	13	**0.9759 ± 0.0006**

Dataset Overview

- **Twitter**: Social network of user-to-follow relationships, capturing directed "follower" connections between 23,370 users. Contains 33,101 directed edges, representing asymmetric follow interactions.
- **Google+**: Network of user-defined social circles (groups) on Google+, with 23,628 users and 39,242 directed edges indicating membership or interaction within circles.
- **Digg**: Interaction network from the social news platform Digg, where 30,398 users engage via 87,627 directed replies to posts or comments.
- **Slashdot**: Technology-focused discussion network with 51,083 users and 140,778 directed reply interactions, reflecting threaded conversations on articles or comments.
- **Physicians**: Directed advice-seeking network among 241 physicians in a medical community, modeling innovation diffusion (e.g., adoption of new practices) through 1,098 advice links.

- **Adolescent**: Directed friendship network derived from surveys of 2,539 high school students, with 12,969 edges representing reported friendships (not necessarily reciprocal).
- **Cora**: Academic citation network of 23,166 machine learning and computer science papers, linked by 91,500 directed citations between publications.
- **DBLP**: Citation network of 12,591 computer science publications from the DBLP database, connected by 49,743 directed citation edges.

17.2 Model "IRW/DRW indexes"

This (J. Li et al. 2020) approach enhances standard similarity indices by explicitly leveraging the structural importance of reciprocal links in three steps.

17.2.1 Empirical Significance of Reciprocal Links

To assess whether reciprocal links participate in closing triads more frequently than by chance, *Reciprocal Null Models* (RNMs) are constructed by randomizing edge directions while preserving both the total number of directed edges $|E|$ and the exact number of reciprocal pairs $|E^r|$. For each triad type with $i \in \{0,1,2,3\}$ reciprocal edges, the statistic

$$Z_i = \frac{N_i^D - \langle N_i^{\text{null}} \rangle}{\text{std}(N_i^{\text{null}})}$$

is computed, where N_i^D denotes the count in the original network and $\langle N_i^{\text{null}} \rangle$, $\text{std}(N_i^{\text{null}})$ are the mean and standard deviation over 1000 RNMs. Large positive values of Z_i indicate that triads with one or more reciprocal edges appear far more often than in randomized counterparts.

17.2.2 Indirect Reciprocity–Aware Weighting (IRW)

Let

$$\rho = \frac{|E^r|}{|E|}$$

represent the global reciprocity (the fraction of edges that are reciprocated). Each directed edge $(x \to y)$ with original adjacency $a_{xy} \in \{0,1\}$ is assigned the weight

$$w_{xy} = a_{xy} + \rho \, \frac{a_{yx}}{k_y^{\text{out}}},$$

where k_y^{out} is the out-degree of node y. This formulation provides a boost of ρ / k_y^{out} whenever the reverse link exists. To constrain w_{xy} to the interval $[0,1]$, the transformation

$$w_{xy} \mapsto \exp\bigl(-w_{xy}/\rho\bigr)$$

is applied, which compresses larger values toward 1 while preserving small weights near 0.

17.2.3 Direct Reciprocity–Aware Weighting (DRW)

Direct Reciprocity–Aware Weights are obtained by augmenting any IRW-based similarity score s_{xy}^{IRW} as follows:

$$s_{xy}^{\mathrm{DRW}} = s_{xy}^{\mathrm{IRW}} + \rho \, \frac{s_{yx}^{\mathrm{IRW}}}{k_y^{\mathrm{out}}},$$

thereby reinforcing the prediction of $x \to y$ when the reverse direction $y \to x$ already has a high IRW score.

17.2.4 Integration into Similarity Indices

For each of the four classic directed metrics—Directed Common Neighbors (DCN), Directed Adamic–Adar (DAA), Directed Resource Allocation (DRA), and the Bifan motif index—edges a_{ij} are replaced by weights w_{ij} to produce IRW variants:

$$\text{IRW-DCN}_{xy} = \sum_z w_{xz}\, w_{zy}, \quad \text{IRW-DAA}_{xy} = \sum_z \frac{w_{xz}\, w_{zy}}{\log\bigl(1 + \sum_u w_{zu}\bigr)},$$

$$\text{IRW-DRA}_{xy} = \sum_z \frac{w_{xz}\, w_{zy}}{\sum_u w_{zu}}, \quad \text{IRW-Bifan}_{xy} = \sum_{z,z'} w_{xz}\, w_{zz'}\, w_{z'y}.$$

These metrics were chosen for their complementary characteristics: DCN captures weighted common neighbors, DAA penalizes high-degree hubs logarithmically, DRA distributes resources inversely by neighbor strength, and Bifan accounts for paths of length three via two intermediaries. Each IRW score is then upgraded to its DRW counterpart using the formula described above.

17.2.5 Comparing Performance with Ours LPPH model

Table 7: Best LPPH parameter (5, 9, 13 or 17) for each dataset, with mean AUC scores. Best models are marked with bold text. Results for IRW/DRW indexes were taken from their paper (J. Li et al. 2020)

Dataset	Model	Edge depth	mean AUC-ROC
Adolescent	best IRW/DRW index		**0.883**
	LPPH+XGBoost	17	0.8716 ± 0.005
Highschool	best IRW/DRW index		**0.921**
	LPPH+XGBoost	13	0.8654 ± 0.035
Political blogs	best IRW/DRW index		0.963
	LPPH+XGBoost	5 (max_dim = 3)	**0.9715 ± 0.001**
Email	best IRW/DRW index		0.960
	LPPH+XGBoost	9	**0.9817 ± 0.004**
Air traffic control	best IRW/DRW index		0.845
	LPPH+XGBoost	9	**0.8489 ± 0.020**
US airports	best IRW/DRW index		**0.981**
	LPPH+XGBoost	5	0.9753 ± 0.001
Celegans	best IRW/DRW index		0.888
	LPPH+XGBoost	13	**0.9341 ± 0.011**
Figeys	best IRW/DRW index		**0.984**
	LPPH+XGBoost	13	0.9823 ± 0.002

Dataset Overview

- **Adolescent health (ADO)**: Directed friendship network constructed from surveys of 701 students, capturing 7,886 reported social ties.
- **High-school (HIG)**: Friendship network among 1,153 male students in an Illinois high school, containing 73,180 directed social connections.
- **Political blogs (PB)**: Hyperlink network between 1,222 US political blogs, connected by 19,025 directed hyperlink references.
- **Email (EMA)**: Email communication network among 1,005 users at a European research institution, mapped through 39,264 directed email exchanges.
- **Air traffic control (ATC)**: Network of 1,562 airports and service centers linked by 3,168 directed preferred flight routes.
- **US airports (USA)**: Commercial flight network between 500 major U.S. airports, comprising 57,109 directed scheduled flight connections.

- ***C. elegans* (CELE)**: Metabolic network of the nematode C. elegans, representing 279 metabolites connected by 5,233 directed biochemical reactions.
- **Figeys (FIG)**: Regulatory interaction network among 2,039 human proteins, connected by 15,612 directed protein-protein regulatory links.

18 Conclusion

This work introduces the Local Persistent Path Homology (LPPH) vector as a novel topological feature for directed link prediction and demonstrates its practical and theoretical advantages. Our contributions include:

- **Directional Topological Features:** LPPH explicitly encodes asymmetric structures, addressing the limitations of traditional GNNs that struggle with directional asymmetry and topological awareness.
- **DirGNN Improvement:** On 11 diverse directed graphs, LPPH boosted link prediction performance in 10/11 graphs with a hybrid approuch and then outperformed best hybrid models in 8/11 datasets as a standalone boosting classifier on LPPH.
- **State Of The Art Competitiveness:** Compared to two sota methods, XGBoost on LPPH showed competitive performance as it outperformed SRTGN in 6/8 graphs and IRW/DRW indexes in 4/8 graphs.
- **Hyperparameter Intuitiveness:** The $1+4k$ depth formula, derived from a four-phase expansion cycle, provides clear guidelines for selecting filtration depth according to graph scale and density.
- **Parallelized Computation Framework:** We provide open-source parralelized implementation with boundary matrix computation during filtration construction with PHAT (Bauer, Kerber, Reininghaus, and Wagner 2017) boundary matrix reduction, enabling efficient LPPH computation on large graphs. The full code of testing and data is available at `https://github.com/antonsokol57/LPPH`.

Future Work

Building on these findings, we outline several promising directions for future research:

- **Learnable Filtration:** Building on (Hofer et al. 2020), we propose developing adaptive filtration functions that may learn optimal edge weights for through gradient-based optimization

- **Reciprocal-Link Enhanced Filtration:** Inspired by (J. Li et al. 2020; Liao et al. 2023), modifying the expansion phases with reciprocal edges could better capture reciprocity patterns in a neighbor subgraph, possibly boosting performance in link prediction as it can make LPPH more descriptive

- **Adaptive SOTA Hybridization:** Create ensemble models combining LPPH with SRT-GCN's social hierarchy features and IRW/DRW's reciprocity indices or with other SOTA model for performance improvement like with dirGNNs example from the thesis

- **Temporal and Domain Extensions:** Adapt LPPH for temporal directed graphs through time-aware filtrations (as something like in (Myers et al. 2023)) to tackle little-studied machine learning on dynamic graphs area

Our results establish LPPH as a powerful and interpretable component for directed graph learning, opening avenues for both theoretical exploration and practical applications across networked domains.

References

Adams, Henry et al. (2017). "Persistence images: A stable vector representation of persistent homology". In: *Journal of Machine Learning Research* 18.1, pp. 218–252.

Bauer, Ulrich, Michael Kerber, and Jan Reininghaus (2014). "Clear and compress: Computing persistent homology in chunks". In: *Topological Methods in Data Analysis and Visualization III*. Springer, pp. 103–117. DOI: 10.1007/978-3-319-04099-8_7.

Bauer, Ulrich, Michael Kerber, Jan Reininghaus, and Hubert Wagner (2017). "Phat–Persistent homology algorithms toolbox". In: *Journal of Symbolic Computation* 78, pp. 76–90. DOI: 10.1016/j.jsc.2016.03.008.

Chen, Chao and Michael Kerber (2011). "Persistent homology computation with a twist". In: *Proceedings of the 27th European Workshop on Computational Geometry*, pp. 197–200.

Chen, Chao, Xiuyan Ni, et al. (2021). "Topological Regularization for Graph Neural Networks". In: *AAAI*, pp. 3434–3441.

Chowdhury, Samir and Facundo Mémoli (2018). "Persistent path homology of directed networks". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, pp. 1152–1169.

Dey, Tamal K., Tianqi Li, and Yusu Wang (2020). "An Efficient Algorithm for 1-Dimensional (Persistent) Path Homology". In: *36th International Symposium on Computational Geometry (SoCG 2020)*. Vol. 164. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 36:1–36:15. DOI: 10.4230/LIPIcs.SoCG.2020.36.

Edelsbrunner, Herbert and John Harer (2010). *Computational Topology - an Introduction*. American Mathematical Society, pp. I–XII, 1–241. ISBN: 978-0-8218-4925-5.

Giusti, Chad et al. (2015). "Clique topology reveals intrinsic geometric structure in neural correlations". In: *Proc. Nat. Acad. Sci. USA* 112.44, pp. 13455–13460. DOI: 10.1073/pnas.1506407112. eprint: http://www.pnas.org/content/112/44/13455.full.pdf. URL: http://www.pnas.org/content/112/44/13455.abstract.

Grigor'yan, Alexander et al. (2012). "Homologies of path complexes and digraphs". In: *arXiv preprint arXiv:1207.2834*.

— (2014). "Homotopy theory for digraphs". In: *Pure and Applied Mathematics Quarterly* 10.4, pp. 619–674.

— (2020). "Path complexes and their homologies". In: *Journal of Mathematical Sciences* 248.5, pp. 564–599.

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable Feature Learning for Networks". In: *KDD*, pp. 855–864.

Gupta, Chitrank et al. (2021). "Integrating Transductive And Inductive Embeddings Improves Link Prediction Accuracy". In: *arXiv preprint arXiv:2108.10108*.

Hamilton, William L., Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30, pp. 1024–1034.

Hiraoka, Yasuaki et al. (2016). "Hierarchical structures of amorphous solids characterized by persistent homology". In: *Proceedings of the National Academy of Sciences* 113.26, pp. 7035–7040. DOI: 10.1073/pnas.1520877113. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1520877113. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1520877113.

Hofer, Christoph et al. (July 2020). "Graph filtration learning". In: *International Conference on Machine Learning*. 13–18 July, pp. 4314–4323.

Horn, Max et al. (2022a). "Multi-Scale Topological Analysis of Neural Networks". In: *arXiv preprint arXiv:2207.13603*.

— (2022b). "Topological Graph Neural Networks". In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=oxxUMeFwEHd.

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Poster. URL: https://openreview.net/forum?id=SJU4ayYgl.

Lee, Hyekyung and Moo K. Chung (2023). "Directional Topological Learning for Brain Functional Networks". In: *Medical Image Analysis* 84, p. 102726.

Li, Jinsong et al. (2020). "Link prediction in directed networks utilizing the role of reciprocal links". In: *IEEE Access* 8, pp. 28668–28680.

Liao, Xiangwen et al. (2023). “SRTGCN: Social Ranking Theory-based Graph Convolutional Network for Directed Link Prediction”. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1234–1245.

Mikolov, Tomas et al. (2013). “Distributed representations of words and phrases and their compositionality”. In: *NeurIPS* 26.

Morris, Christopher et al. (2024). “Position: Future Directions in the Theory of Graph Machine Learning”. In: *Proceedings of the 41st International Conference on Machine Learning (ICML)*. Vol. 235. Vienna, Austria: PMLR.

Myers, Audun et al. (2023). “Temporal network analysis using zigzag persistence”. In: *EPJ Data Sci.* 12.1, p. 6. DOI: 10.1140/EPJDS/S13688-023-00379-5. URL: https://doi.org/10.1140/epjds/s13688-023-00379-5.

Veličković, Petar et al. (2018). “Graph Attention Networks”. In: *6th International Conference on Learning Representations (ICLR)*.

Wang, Minghua et al. (2024). “Persistent Local Homology in Graph Learning”. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: https://openreview.net/forum?id=qunyX9WYr6.

Zhao, Qi et al. (2022). “Boosting Graph Pooling with Persistent Homology”. In: *ICML*, pp. 12145–12155.