TopER: Topological Embeddings in Graph Representation Learning

Astrit Tola
Department of Mathematics
Florida State University
Tallahassee, FL 32306
atola@fsu.edu

Funmilola Mary Taiwo
Department of Statistics
University of Manitoba
Winnipeg, Manitoba, Canada
taiwom1@myumanitoba.ca

Cuneyt Gurcan Akcora
AI Institute
University of Central Florida
Orlando, FL, 32816
cuneyt.akcora@ucf.edu

Baris Coskunuzer
Department of Mathematical Sciences
University of Texas at Dallas
Richardson, TX 75080
coskunuz@utdallas.edu

Abstract

Graph embeddings play a critical role in graph representation learning, allowing machine learning models to explore and interpret graph-structured data. However, existing methods often rely on opaque, high-dimensional embeddings, limiting interpretability and practical visualization.

In this work, we introduce Topological Evolution Rate (TopER), a novel, low-dimensional embedding approach grounded in topological data analysis. TopER simplifies a key topological approach, Persistent Homology, by calculating the evolution rate of graph substructures, resulting in intuitive and interpretable visualizations of graph data. This approach not only enhances the exploration of graph datasets but also delivers competitive performance in graph clustering and classification tasks. Our TopER-based models achieve or surpass state-of-the-art results across molecular, biological, and social network datasets in tasks such as classification, clustering, and visualization.

> 🐍 pip install toper

1 Introduction

Graphs are a fundamental data structure utilized extensively to model complex interactions within various domains, such as social networks [LBKT08], molecular structures [YLY$^+$18], and transportation systems [DCS$^+$22]. Their inherent flexibility, however, introduces significant challenges when applied to machine learning (ML) tasks, primarily due to their irregular and high-dimensional nature. Graph data lacks inherent ordering and consistent dimensionality, making it challenging for traditional ML methods designed for vector spaces.

Graph Neural Networks (GNNs) have emerged as the state-of-the-art models for tackling graph machine learning tasks due to their ability to learn effectively from graph-structured data [KW17]. In the predominant paradigm of message-passing GNNs, the process begins by generating node embeddings [BHG$^+$21]. These embeddings can then be used in tasks such as node classification or link prediction. However, for graph-related tasks, such as molecular property prediction, the embeddings
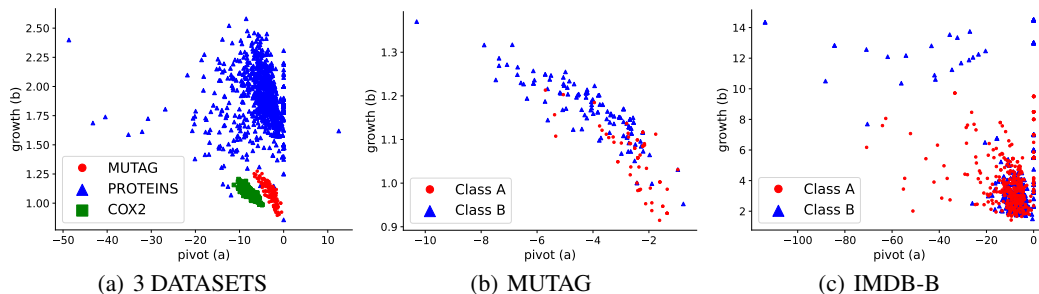
Figure 1: **TopER Visualizations.** Each data point represents an individual graph. On the left, TopER is applied to three benchmark compound datasets using closeness sublevel filtration. The middle panel zooms in on the red point cloud from the left, demonstrating TopER's effectiveness in distinguishing between classes within the MUTAG dataset. On the right, a TopER visualization for the IMDB-B dataset is displayed.

must be aggregated through a pooling layer to form graph-level representations [WKK⁺20]. This method is computationally intensive, mainly because generating and managing node embeddings as intermediate steps substantially increases the overall computational burden. Ideally, an approach would allow for the direct creation of graph embeddings, circumventing the need to generate node-level representations first. Furthermore, these graph embeddings must be both low-dimensional and interpretable to maximize their practical utility and efficiency in various applications.

Topological Data Analysis (TDA) is well-suited for directly constructing graph representations without costly node embeddings. Topology studies the shape of data, and TDA primarily focuses on the qualitative properties of space, such as continuity and connectivity [CA24]. A particularly effective technique in TDA is *Persistent Homology (PH)*, which tracks topological features, like connected components and cycles, across various scales via a process known as filtration. Filtration is adept at revealing both local and global structures within graphs. It proves exceptionally useful for comparing graphs of different sizes that maintain the same inherent structure, which may suggest similar properties in graph datasets. For instance, similar substructures in protein interaction networks across different species may indicate comparable biological functions. By focusing on data shape, PH proves invaluable in graph tasks that benefit from a graph-centric approach, offering insights that might not be as apparent when focusing on individual node analysis. This shift from a node-centric to a graph-centric perspective can dramatically improve the understanding and application of graph data in fields like bioinformatics and network analysis. However, the utility of Persistent Homology is limited by the high computational demands involved in extracting topological features during the filtration process, mainly due to its cubic time complexity [OPT⁺17]. This constraint reduces its practicality for large-scale graphs and has restricted the broader integration of PH in graph representation learning.

With this work, we take a significant step forward in addressing the challenges of topological graph representation learning and introduce *Topological Evolution Rate (TopER)*. This novel approach refines the Persistent Homology process to efficiently capture graph substructures, thereby mitigating the significant computational demands of calculating complex topological features. As graph representation learning aligns naturally with Topological Data Analysis, *TopER* excels in graph clustering and classification tasks, where it achieves the best rank in experiments. Furthermore, simplifying graph data into a low-dimensional space, *TopER* creates intuitive visualizations that reveal clusters, outliers, and other essential topological features, as demonstrated in Figure 1. As a result, *TopER* merges interpretability with efficiency in graph representation learning, providing an ideal balance that can scale to large graphs. To our knowledge, *TopER* is the first topology-based graph representation method that can create low-dimensional, efficient, and scalable graph representations.

Our contributions can be summarized as follows:

- **New Graph Representation:** We introduce *TopER*, a principled and computationally feasible graph representation designed to capture the structural evolution of a graph.

- **Topology-Inspired Design:** Rather than computing persistent diagrams or Betti numbers, *TopER* directly leverages the filtration process to provide efficient, low-dimensional summaries rooted in topological intuition.

2

- **Competitive Performance:** Experiments on benchmark graph classification and clustering tasks demonstrate that *TopER* achieves competitive results compared to more complex, state-of-the-art models, while offering superior interpretability.

- **Interpretable Visualizations:** *TopER* produces low-dimensional outputs that support intuitive visualization of clusters and structural outliers within and across graph datasets, enabling a form of visual model comparison often lacking in traditional embeddings.

- **Robustness Under Perturbations:** We provide a stability analysis showing that *TopER*'s representations are consistent under small changes to the filtration function, reinforcing its suitability for practical and comparative applications.

2   Background

2.1   Related Work

Graph-level Embedding Methods. Graph representation learning, including GNNs and Graph Pooling techniques, is a dynamic subfield of machine learning, focusing on transforming graph data into efficient, low-dimensional vector representations that encapsulate essential features of the data [Ham20, GHT$^+$19]. These representations facilitate a deeper analytical understanding of graphs, which is critical for various applications such as molecular graph property prediction [DTRF19]. GNNs have revolutionized the analysis of graph data, drawing parallels with the success of Convolutional Neural Networks in image processing [EPBM20]. GNNs utilize spectral and spatial approaches to graph convolutions based on the graph Laplacian and direct graph convolutions, respectively [BZSL14, DBV16, KW17]. Despite their success, GNNs often suffer from issues like over-smoothing and a lack of transparency, making them less ideal for applications requiring interpretability [Gün22].

Graph Pooling emerged as a key component in GNN architectures, drawing parallels to the role of pooling in Convolutional Neural Networks [EPBM20]. Pooling aims to deduce into meaningful graph embeddings through node aggregation (mean, max, and add pooling [XHLJ19]) or hierarchical pooling (node selection: Top-k [GJ19] and SAGPool [LLK19]; node clustering: DiffPool [YYM$^+$18] and MinCutPool [BGA19]). Both pooling groups have their challenges. Node aggregation methods may lose the structural information by treating each node equally, while the hierarchical pooling methods can be computationally heavy, and loss of information can occur if important nodes are discarded.

Topology-Inspired Representations. TDA provides a robust and computationally efficient framework to address the interpretability and over-smoothing issues present in GNNs [AAF19]. Persistent Homology, a key technique in TDA, has been applied successfully to graph data, demonstrating potential to match or even exceed the performance of traditional methods in classification and clustering tasks [HMR21, DCG$^+$22, HIL$^+$24, ISG23, CSA$^+$23, LSC$^+$23]. However, the computational intensity of PH limits its scalability [HKN19, ZYCW20, AKGC22].

Visualization Techniques. Graph embedding techniques, including spectral methods, random walk-based approaches, and deep learning-based models, transform graph data into vector representations to support tasks like visualization and machine learning [CZC18, GF18, Xu21]. Approaches such as Laplacian Eigenmaps and DeepWalk have been particularly effective in revealing clusters within graphs [BN01, PAS14]. However, these methods are predominantly applied to visualize a *single graph* in node classification tasks, focusing on cluster identification [WCZ16, MK20, TPPM23]. Furthermore, they often overlook domain-specific information, which can limit their effectiveness in more specialized applications [JZ20].

TopER addresses these challenges by combining the interpretative benefits of TDA with the analytical strength of modern graph machine learning. Distinct from current approaches, TopER employs a simplified filtration process to create embeddings that are both interpretable and computationally efficient. By extending the filtration to multiple functions, TopER stands out as one of the first methods to offer effective and interpretable visualizations of graph datasets, while also achieving superior performance in clustering and classification tasks.

2.2 Persistent Homology for Graphs

Topological Data Analysis (TDA) offers a powerful framework for graph representation learning [AAF19], with persistent homology (PH) being especially effective at capturing multi-scale topological features [CA24]. While PH typically involves filtrations, persistence diagrams, and vectorization, our model focuses on the filtration step, reformulating the evolution of topological features in a novel and efficient way.

In the crucial filtration step, PH decomposes a graph $\mathcal{G}$ into a nested sequence of subgraphs $\mathcal{G}_1 \subseteq \mathcal{G}_2 \subseteq \ldots \subseteq \mathcal{G}_n = \mathcal{G}$. For each $\mathcal{G}_i$, an abstract simplicial complex $\widehat{\mathcal{G}}_i$ is defined, forming a filtration of simplicial complexes. Clique complexes are typical choices, where each $(k+1)$-complete subgraph in $\mathcal{G}$ corresponds to a k-simplex [AAF19].

Filtration. Utilizing relevant filtration functions is essential to obtain effective filtrations. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a common approach is to define a node filtration function $f : \mathcal{V} \to \mathbb{R}$, which establishes a hierarchy among the nodes. By selecting a monotone increasing set of thresholds $\mathcal{I} = \{\epsilon_i\}_{i=1}^n$, this method generates subgraphs $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ where $\mathcal{V}_i = \{v \in \mathcal{V} \mid f(v) \leq \epsilon_i\}$ and $\mathcal{E}_i$ is the set of edges in $\mathcal{E}$ with endpoints in $\mathcal{V}_i$. This is called a sublevel filtration induced by f (See Figure 2). Also, superlevel filtrations can be constructed by defining $\mathcal{V}_i = \{v \in \mathcal{V} \mid f(v) \geq \epsilon_i\}$ for decreasing thresholds [AAF19].

Similarly, one can use edge filtration functions $g : \mathcal{E} \to \mathbb{R}$ to define such a filtration. We define $\mathcal{E}_i = \{e_{jk} \in \mathcal{E} \mid g(e_{jk}) \leq \epsilon_i\}$, and $\mathcal{V}_i$ as all the endpoints of $\mathcal{E}_i$ to create a nested sequence $\{\mathcal{G}_i\}_{i=1}^n$. Especially for weighted graphs, this method is highly preferable as weights naturally define an edge filtration function. The common node filtration functions are degree, betweenness, centrality, heat kernel signatures [BK10], and node functions coming from the domain of the datasets (e.g., atomic number for molecular graphs). Common edge filtration functions are Ollivier and Forman Ricci curvatures [LLY11] and edge weights (e.g., transaction amounts for financial networks).

In addition to existing approaches, we introduce a new filtration function, *Popularity*, which extends the idea of degree-based ranking [New03]. While the degree function measures the number of direct neighbors a node has, *Popularity* captures the average degree of those neighbors. The underlying intuition is that, whereas degree reflects how many connections a node has, *Popularity* emphasizes node influence through association with high-degree nodes.

Formally, for each node v, we define popularity as: $\mathcal{P}(v) = \deg(v) + \frac{\sum_{u \in \mathcal{N}(v)} \deg(u)}{|\mathcal{N}(v)|}$ where $\deg(v)$ is the node's degree and $\mathcal{N}(v)$ is its set of neighbors. This function incorporates 2-neighborhood information by weighting high-degree neighbors more heavily, making it a refined version of the degree function.
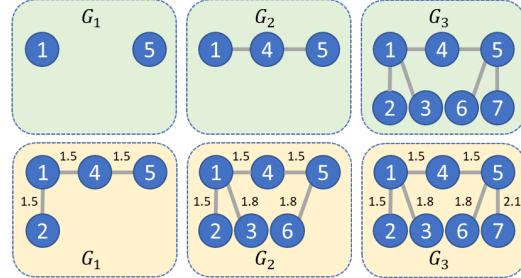

Figure 2: **Filtration.** For $\mathcal{G} = \mathcal{G}_3$ in both examples, the top figure illustrates superlevel filtration with node degree function for thresholds $\{1, 2, 3\}$. Similarly, the bottom figure illustrates sublevel filtration for edge weights with thresholds $\{1.5, 1.8, 2.1\}$.

3 TopER: Topological Evolution Rate

TopER is inspired by the foundational idea that topology can capture the shape of graph data, and that this shape can be observed through the evolution of a graph during the filtration process. In this sense, TopER is topology-inspired. However, TopER diverges from traditional PH in how it tracks the shape. We first achieve a computationally efficient alternative to persistent homology by simplifying its filtration-based perspective, and second, we develop a low-dimensional, interpretable representation of graphs that enables both effective classification and intuitive visualization across graph datasets.

Our reformulation reduces the computational overhead typically required for topological feature extraction. Unlike Persistent Homology, which extracts costly topological features, TopER summarizes the filtration process through two key parameters derived via regression: *filtration sequences* and *evolution*.

4

Filtration Sequences. We first decompose a graph $\mathcal{G}$ into a nested sequence of subgraphs (filtration graphs) $\mathcal{G}_1 \subseteq \mathcal{G}_2 \ldots \subseteq \mathcal{G}_n = \mathcal{G}$ by using a filtration function, such as node degree or closeness. Let $\mathcal{G}_i \subset \mathcal{G}$, $\mathcal{V}_i$ represent nodes in $\mathcal{G}_i$ and $\mathcal{E}_i$ represent the edges. Next, we compute $x_i = |\mathcal{V}_i|$ as the count of nodes, and $y_i = |\mathcal{E}_i|$ as the count of edges. Then, for each filtration graph $\mathcal{G}_i$, we obtain the pair $(x_i, y_i) \in \mathbb{R}^2$, which creates two monotone sequences, $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \cdots \leq y_n$. Hence, TopER yields two ordered sets $\mathcal{X}, \mathcal{Y}$ describing the evolution of the filtration graphs $\mathcal{G}_1 \subseteq \ldots \subseteq \mathcal{G}_n = \mathcal{G}$, $\mathcal{X} = (x_1, x_2, \ldots, x_n)$ and $\mathcal{Y} = (y_1, y_2, \ldots, y_n)$. Here, n corresponds to number of thresholds $\{\epsilon_i\}_{i=1}^n$ used in the filtration step. Consider the top row of Figure 2 where we have three filtration graphs (i.e., n=3); we have $\mathcal{X} = (2, 3, 7)$ for node counts and $\mathcal{Y} = (0, 2, 6)$ for edge counts.

Evolution. In the next step, PH would typically compute topological features on each filtration and create a persistence diagram to summarize the features. Not only is it costly, but the approach would require efforts to vectorize the persistence diagrams. We circumvent this computationally costly step and analyze how the number of edges $\{y_i\}$ relates to the number of nodes $\{x_i\}$ throughout the filtration sequence. We use line fitting to characterize this relationship as follows. Simple linear regression, often applied through the least
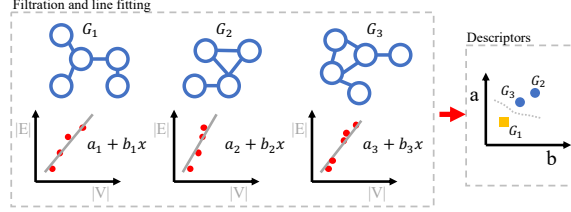

Figure 3: **TopER steps.** The filtration process on three different graphs using node or edge filtration. The graphs undergo filtration, and for each graph, a best-fit line is determined through the filtration data. The coefficients of these best-fit lines are then used as descriptors for the graphs.

squares method [JWH$^+$23], is a standard approach in regression analysis for fitting a linear equation to a set of data points $\{(x_i, y_i)\} \subset \mathbb{R}^2$. This method calculates the line $L(x) = a + bx$ that best fits the data by minimizing the loss function $\mathrm{E} = \sum_{i=1}^N [L(x_i) - y_i]^2$. The regression coefficients (a, b) capture the graph's structural evolution through filtration (see descriptor step in Figure 3). The full TopER method is outlined in Algorithm 1 in the Appendix.

With evolution on filtration sequences, we define the topological evolution rate of a graph as follows:

Definition 3.1 (Topological Evolution Rate (TopER)). Let $f : \mathcal{V} \to \mathbb{R}$ be a filtration function on graph $\mathcal{G}$ and $\mathcal{I} = \{\epsilon_i\}_{i=1}^n$ be the threshold set. Let $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ be the induced filtration. Let $x_i = |\mathcal{V}_i|$ and $y_i = |\mathcal{E}_i|$. Let $L(x) = a + bx$ be the best fitting line to $\{(x_i, y_i)\}_{i=1}^n$. Then, we define the *TopER vector* of $\mathcal{G}$ wrt. f as $\mathrm{TE}_f(\mathcal{G}, \mathcal{I}) = (\mathrm{a}, \mathrm{b})$. We call a the *pivot* and b the *growth* of $\mathcal{G}$.

Remark 3.2 (Why a Linear Fit?). Although one could consider higher-order polynomials, our experiments show that coefficients beyond the first degree are negligible (Table 15), revealing an essentially linear relationship. Due to space constraints, we defer a more detailed comparison of linear, polynomial, and higher-order fits to Appendix B, where we also provide visual examples of evolution rates (Figure 6) and fundamental pattern types (Figure 7).

Remark 3.3 (On the Name *TopER*). Although *TopER* does not compute classical topological invariants such as persistence diagrams or Betti numbers, its name is grounded in two topology-inspired principles. First, it mirrors the TDA notion of a *filtration*, a nested sequence of subgraphs $\{\mathcal{G}_i\}$ that progressively unveils structural features across scales. Second, it is directly tied to the evolution of the Euler characteristic $\chi(\mathcal{G}_i) = |\mathcal{V}_i| - |\mathcal{E}_i| = \beta_0(\mathcal{G}_i) - \beta_1(\mathcal{G}_i)$, where β_i denotes the ith Betti number. Equivalently, one could track $\{(x_i, y_i - x_i)\} = \{(|\mathcal{V}_i|, \chi(\mathcal{G}_i))\}$, but we use $\{(x_i, y_i)\}$ for notational simplicity. The resulting slope thus quantifies how topological complexity changes over the filtration, justifying the term *Topological Evolution Rate*.

We emphasize that the term "topological" reflects the conceptual roots of TopER in topological data analysis: the method uses a filtration to reveal structural patterns in the graph, analogous to the filtration process in persistent homology. While we do not compute full persistence diagrams or Betti numbers, the slope of the Euler characteristic across the filtration captures essential topological information, justifying the terminology.

3.1 Computational Complexity

The primary computational steps in TopER include constructing filtration graphs and performing regression on node and edge counts, which incur the following costs.

Analyzing each node and edge across n filtration thresholds typically requires $O(n \times (|\mathcal{V}| + |\mathcal{E}|))$ operations, where $\mathcal{V}$ and $\mathcal{E}$ denote the numbers of vertices and edges, respectively. The regression step involves fitting a line to the pairs (x_i, y_i) using the least squares method. The complexity of calculating the necessary sums for this regression is $O(n)$, and solving for the regression coefficients (slope and intercept) from these sums involves a constant amount of additional computation.

Thus, the overall complexity of TopER predominantly hinges on the graph filtration process, summing up to $O(n \times (|\mathcal{V}| + |\mathcal{E}|))$ where $|\mathcal{V}| \gg n$. As we will show in the next section, the runtime costs of TopER are notably low, making it practical and efficient for large-scale applications.

3.2 Stability Results

This section states our theorems on the stability of TopER. In the following, $\mathcal{W}_p(.,.)$ represents p-Wasserstein distance, and $\mathrm{PD}_k(\mathcal{X}, f)$ represents k^{th} persistence diagram of $\mathcal{X}$ with sublevel filtration with respect to f. Similarly, $\|.\|_p$ represents L^p-norm and $d_p(.,.)$ represents l_p-distance in $\mathbb{R}^m$. We fix a threshold set $\mathcal{I} = \{\epsilon_i\}_{i=1}^n$ for both functions to keep the exposition simple. Further, to keep the setting general, we use the pairs $\{(\beta_0(\epsilon_i), \beta_1(\epsilon_i))\}_{i=1}^n$ in $\mathbb{R}^2$ to fit the least squares line $y = a + bx$ defining $\mathrm{TE}(\mathcal{X}) = (a, b)$.

Theorem 3.4. *Let $\mathcal{X}$ be a compact metric space, and $f, g : \mathcal{X} \to \mathbb{R}$ be two filtration functions. Then, for some $C > 0$,*

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq C \cdot \mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}, f), \mathrm{PD}_k(\mathcal{X}, g)).$$

By combining the above result with the stability result for sublevel filtrations, we obtain the stability with respect to filtration functions as follows.

Corollary 3.5. *Let $\mathcal{X}$ be a compact metric space, and $f, g : \mathcal{X} \to \mathbb{R}$ be two filtration functions. Then, for some $C > 0$,*

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq C \cdot \|f - g\|_1$$

The following lemmas are essential for proving the theorem and corollary above.

Lemma 3.6. *[ST20] Let $\mathcal{X}$ be a compact metric space, and $f, g : \mathcal{X} \to \mathbb{R}$ be two filtration functions. Then, for any $p \geq 1$, we have $\mathcal{W}_p(\mathrm{PD}_k(\mathcal{X}, f), \mathrm{PD}_k(\mathcal{X}, g)) \leq \|f - g\|_p$*

The next lemma is on the stability of Betti curves by [DG23] [Proposition 1].

Lemma 3.7. *[DG23] Let $\beta_k(\mathcal{X})$ is the kth Betti function obtained from the persistence module $\mathrm{PM}_k(\mathcal{X})$.*

$$\|\beta_k(\mathcal{X}) - \beta_k(\mathcal{Y})\|_1 \leq 2\mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}), \mathrm{PD}_k(\mathcal{Y}))$$

By adapting the above results to the graph setting, when two metric graphs $\mathcal{G}_1, \mathcal{G}_2$ are close in the Gromov-Hausdorff sense, one can obtain a similar stability result for the filtrations of $\mathcal{G}_i$ induced by the same filtration function. Due to space limitations, details and the proofs are given in Appendix C.

4 Experiments

We evaluate the performance of TopER in

classification, clustering and visualization. Our Python implementation is available at https://github.com/AstritTola/TopER.

4.1 Experimental Setup

Datasets. We conduct experiments on nine benchmark datasets for graph classification. These are (i)

Table 1: Characteristics of the benchmark graph classification datasets.

Datasets	#Graphs	$|\mathcal{V}|$	$|\mathcal{E}|$	Classes
BZR	405	35.75	38.36	2
COX2	467	41.22	43.45	2
MUTAG	188	17.93	19.79	2
PROTEINS	1113	39.06	72.82	2
IMDB-B	1000	19.77	96.53	2
IMDB-M	1500	13.00	65.94	3
REDDIT-B	2000	429.63	497.75	2
REDDIT-5K	4999	508.52	594.87	5
OGBG-MOLHIV	41127	243.4	2266.1	2

the molecule graphs of BZR, and COX2 [MV09]; (ii) the biological graphs of MUTAG and PROTEINS [KM12]; and (iii) the social graphs of IMDB-Binary (IMDB-B), IMDB-Multi (IMDB-M), REDDIT-Binary (REDDIT-B), and REDDIT-Multi-5K (REDDIT-5K) [YV15]. Finally, the OGBG-MOLHIV is a large molecular property prediction dataset, part of the open graph benchmark (OGB) datasets [HFZ+20]. Data statistics are given in Table 1.

Table 2: **Graph Classification.** Accuracy results on eight benchmark datasets. Best results are in **bold blue**, second-best are <u>underlined</u>. The final column shows each model's average deviation from the best per dataset.

Model	BZR	COX2	MUTAG	PROTEINS	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	Avg.↓
DiffPool [YYM+18]	83.93±4.41	79.66±2.64	79.22±1.02	73.63±3.60	68.60±3.10	45.70±3.40	79.00±1.10	–	8.06
P-WL-C [RBB19]	–	–	90.51±1.34	75.27±0.38	–	–	–	–	2.08
SAGPool [LLK19]	82.95±4.95	79.45±2.98	76.78±2.12	71.86±0.97	74.87±4.09	49.33±4.90	84.70±4.40	–	6.61
Top-k [GJ19]	79.40±1.20	80.30±4.21	67.61±3.36	69.60±3.50	73.17±4.84	48.80±3.19	79.40±7.40	–	9.70
1-GIN (GFL) [HGR+20]	–	–	–	74.10±3.40	74.50±4.60	49.70±2.90	90.20±2.8	55.70±2.90	2.09
6 GNNs [EPBM20]	–	–	80.42±2.07	75.80±3.70	71.20±3.90	49.10±3.50	89.90±1.90	56.10±1.60	4.13
MinCutPool [BGA19]	82.64±5.05	80.07±3.85	79.17±1.64	<u>76.62±2.58</u>	70.77±4.89	49.00±2.83	87.20±5.00	–	5.82
DMP [BCL21]	–	–	84.00±8.60	75.30±3.30	73.80±4.50	50.90±2.50	86.20±6.80	51.90±2.10	4.20
FC-V [ORB21]	85.61±0.59	81.01±0.88	87.31±0.66	74.54±0.48	73.84±0.36	46.80±0.37	89.41±0.24	52.36±0.37	4.01
SubMix [YSK22]	86.34±2.00	<u>84.68±3.70</u>	80.99±0.60	67.80±2.00	70.30±1.40	46.47±2.50	–	–	6.15
G-Mix [HJLH22]	84.15±2.30	83.83±2.10	81.96±0.60	66.28±1.10	69.40±1.10	46.40±2.70	–	–	6.91
RGCL [LWZ+22]	84.54±1.67	79.31±0.68	87.66±1.01	75.03±0.43	71.85±4.84	49.31±0.42	90.34±0.58	56.38±0.40	3.56
AutoGCL [YWH+22]	86.27±0.71	79.31±0.70	88.64±1.08	75.80±0.36	72.32±0.93	50.60±0.80	88.58±1.49	**56.75±0.18**	3.08
FF-GCN [PAMF23]	<u>89.00±5.00</u>	78.00±8.00	71.00±4.00	62.00±1.00	63.00±8.00	–	–	–	11.53
WWLS [FHSK23]	88.02±0.61	81.58±0.91	88.30±1.23	75.35±0.74	**75.08±0.31**	<u>51.61±0.62</u>	–	–	2.26
EPIC [HLAK24]	88.78±2.30	**85.53±1.60**	82.44±0.70	69.06±1.00	71.70±1.00	47.93±1.30	–	–	4.67
EMP [CSA+23]	–	–	88.79±0.63	72.78±0.54	74.44±0.45	48.01±0.42	<u>91.03±0.22</u>	54.41±0.32	2.97
MP-HSM [LSC+23]	–	77.10±3.00	85.60±5.30	74.60±2.10	<u>74.80±2.50</u>	47.90±3.20	–	–	4.67
TopoGCL [CFG24]	87.17±0.83	81.45±0.55	90.09±0.93	**77.30±0.89**	74.67±0.32	**52.81±0.31**	90.40±0.53	–	<u>1.76</u>
PGOT [QTLL24]	87.32±3.90	82.98±5.21	**92.63±2.58**	73.21±2.59	62.90±3.05	51.33±1.76	–	–	3.85
RePHINE [ISG23]	–	–	–	71.25±1.60	69.40±3.78	–	–	–	5.86
GPSE [CLL+24]	80.49±4.18	78.37±2.62	87.19±8.66	72.15±3.66	69.30±3.61	47.40±5.40	80.40±3.40	–	7.27
TopER	**90.13±4.14**	82.01±4.59	<u>90.99±6.64</u>	74.58±3.92	73.20±3.43	50.00±4.02	**92.70±2.38**	<u>56.51±2.22</u>	**1.60**

Hardware. We ran experiments on a single machine with 12th Generation Intel Core i7-1270P vPro Processor (E-cores up to 3.50 GHz, P-cores up to 4.80 GHz), and 32GB of RAM (LPDDR5-6400MHz).

Runtime. TopER is highly scalable and can be applied to a 100K node graph in less than 2 minutes (see Figure 5). Our small network experiments took about two days in a shared resource setting, whereas the OGBG-MOLHIV experiments took 7.85 hours. One of the most demanding datasets, REDDIT-5K, requires 2.91 hours to calculate all node and edge functions. The runtime of our methods is dominated by the computation of node functions such as closeness and Riccis blue(see Appendix A.6). Using approximate values for centrality metrics instead could greatly decrease computation time [BP07]. Since this is not our current focus, we leave it as future work.

Model Setup and Metrics. We employ a rigorous experimental setup to ensure a fair comparison and the selection of the best graph classification model. We begin by applying BatchNormalization to the input features to maintain consistent scaling. We employ a 90/10 train-test split, adopt the StratifiedkFold strategy, and present the average accuracy from ten-fold cross-validation across all our models. We employ accuracy as the evaluation metric, a widely utilized performance measure within graph classification tasks [EPBM20].

Filtration Functions. In TopER, we use both node and edge filtrations (Definition 3.1). Alongside popularity, we apply degree, closeness, and degree centrality [EC22] as node filtration functions and Forman- and Ollivier-Ricci functions [LLY11] as edge filtration functions. We also use atomic weight as a node function for molecular and biological datasets (BZR, COX2, and MUTAG), and node attributes (PROTEINS). We utilized the t-test to assess the statistical significance of each function and applied the Lasso method for regularization. Functions were retained in the model only if they achieved p-values less than 0.05 in the t-test and had non-zero coefficients in the Lasso model [JWH+23]. This approach ensures that the selected filtration functions contribute statistically significant and regularized features to the model. Incorporating additional filtration functions can enhance TopER's ability to analyze graphs from diverse perspectives. However, as we will next illustrate in Table 5, TopER demonstrates strong performance even in its most basic form using the simple and scalable node degree function. This balance of performance and simplicity suits our scalability philosophy; we avoid complex and costly schemes for learning dataset-specific activation functions and homogenize the filtration step in all datasets.

Classifier. We utilize a Multilayer Perceptron (MLP) in our graph classification task. The hyperparameters are detailed in Appendix A.7.

4.2 Graph Classification Results

Baselines. We compare our method with 22 state-of-the-art and recent models in graph classification, including variants of graph neural networks: six GNNs including GCN, DGCNN, Diffpool, ECC, GIN, GraphSAGE which are compared in [EPBM20] (best results of these six GNNs are given in the *6 GNNs row*), FF-GCN [PAMF23]; topological methods: DMP [BCL21], FC-V [ORB21], WWLS [FHSK23], MP-HSM [LSC+23] and EMP [CSA+23]; GNNs enhanced with data augmentation methods: SubMix [YSK22], G-Mix [HJLH22], and EPIC [HLAK24]; GNNs enhanced with contrastive learning methods: RGCL [LWZ+22], AutoGCL [YWH+22], TopoGCL [CFG24]; prototype-based methods: PGOT [QTLL24]; pooling methods: Top-k [GJ19], SAGPool [LLK19], DiffPool [YYM+18], MinCutPool [BGA19] and structural encoder: GPSE [CLL+24].

Table 2 shows the accuracy results for the given models. We use the reported results in the corresponding references for each model. "−" entries in the table mean the reference did not report any result for that dataset. In [EPBM20], the authors compare the six most common GNNs on the graph classification task (see the GNNs row). The last column summarizes each model's overall performance. We report the average of the differences between each model's performance and the best performance in the column across all datasets. If a model's performance is missing for a dataset, it is excluded from the average computation for the model.

Out of eight datasets, TopER achieves the best results in two and ranks second in two other datasets. For the remaining four datasets, TopER's performance is within 4% of the SOTA results. *For overall performance, TopER outperforms all other models with an average deviation of 1.60% from the best performances.* The closest competitor is TopoGCL, which has an average deviation of 1.76%.

OGBG-MOLHIV results. To evaluate our model's performance on large datasets, we compare it with recently published models on the OGBG-MOLHIV dataset, as shown in Table 3. The performances of these models are listed in chronological order based on their publication dates, with baseline performances reported from [CPWC23, YCL+21] or the respective model's references. In Appendix A.1, we give further details for TopER performance and the contribution of each function on this dataset. TopER achieves the second-best result on the MOLHIV dataset, while the top-performing model requires learning a significantly larger model with 119.5 million parameters.

TopER vs. PH. TopER consistently outperforms Persistent Homology methods in both accuracy and computational efficiency. As shown in Table 4, we compare against the best PH results reported in [Cai21], which evaluates 16 combinations of four filtration functions and four vectorization techniques per dataset. *TopER achieves higher accuracy on all six benchmarks.* In terms of runtime, *TopER is over 10 times faster than PH* on large graphs like Reddit-5K, while maintaining high performance (see Table 8 and Appendix A.2).

Table 3: AUC results for OGBG-MOLHIV dataset.

Model	AUC
GIN-VN [XHLJ19]	$77.80_{\pm1.82}$
HGK-WL [TGL+19]	$79.05_{\pm1.30}$
WWL [BGL+20]	$75.58_{\pm1.40}$
PNA [CCB+20]	$79.05_{\pm1.32}$
DGN [BPL+21]	$79.70_{\pm0.97}$
GraphSNN [WW22]	$79.72_{\pm1.83}$
GCN-GNorm [CLX+21]	$78.83_{\pm1.00}$
Graphormer [YCL+21]	$\mathbf{80.51_{\pm0.53}}$
Cy2C-GCN [CPWC23]	$78.02_{\pm0.60}$
GAWL [NV23]	$78.34_{\pm0.39}$
LLM-GIN [ZZM24]	$79.22_{\pm\text{NA}}$
GMoE-GIN [WJY+23]	$76.90_{\pm0.90}$
TopER	$80.21_{\pm0.15}$

Table 4: Accuracy results for TopER vs. Persistent Homology in graph classification tasks.

	BZR	COX2	PROTEINS	IMDB-B	IMDB-M	RED-5K
PH	$88.4_{\pm0.6}$	$\mathbf{82.0_{\pm0.6}}$	$74.0_{\pm0.4}$	$69.5_{\pm0.5}$	$46.5_{\pm0.3}$	$54.1_{\pm0.1}$
TopER	$\mathbf{90.1_{\pm4.1}}$	$\mathbf{82.0_{\pm4.6}}$	$\mathbf{74.6_{\pm3.9}}$	$\mathbf{73.2_{\pm3.4}}$	$\mathbf{50.0_{\pm4.0}}$	$\mathbf{56.5_{\pm2.2}}$

Ablation Studies. We have conducted **three ablation studies**. In the first one, we evaluated *the individual performance of each function* as well as their combined effect on classification. As shown in Table 5, the common filtration functions we employ from TDA exhibit strong individual performance. Moreover, when combined, they synergistically enhance overall performance. This is not surprising, as different filtration functions, such as atomic weight or Ricci curvature, generate distinct hierarchies and node-edge distributions, resulting in diverse connectivity patterns throughout the filtration sequence. This diversity is analogous to viewing an object from multiple angles. Hence,

integrating these complementary perspectives improves performance by offering a richer and more varied representation of the graph structure, allowing the model to capture more intricate features. The other two ablation studies are provided in the Appendix A.4, which examines *the effect of the number of thresholds* on TopER's performance, while Appendix A.5 analyzes *the impact of the number of filtration functions* used.

Table 5: **Ablation Study.** Individual and altogether performances of filtration functions with TopER.

Datasets	Degree-cent.	Popularity	Closeness	Degree	F. Ricci	O. Ricci	Atom weight	TopER
BZR	$82.22_{\pm2.13}$	$82.20_{\pm3.42}$	$81.48_{\pm1.99}$	$\underline{82.73_{\pm2.12}}$	$80.75_{\pm1.73}$	$80.99_{\pm1.48}$	$82.23_{\pm2.12}$	$\mathbf{90.13_{\pm4.14}}$
COX2	$\underline{75.38_{\pm3.96}}$	$69.21_{\pm8.19}$	$67.90_{\pm7.96}$	$73.88_{\pm5.02}$	$70.46_{\pm7.28}$	$73.03_{\pm4.21}$	$69.82_{\pm8.27}$	$\mathbf{82.01_{\pm4.59}}$
MUTAG	$76.61_{\pm7.87}$	$77.66_{\pm6.12}$	$80.88_{\pm4.79}$	$74.97_{\pm6.40}$	$80.85_{\pm9.25}$	$\underline{82.46_{\pm7.84}}$	$73.45_{\pm8.01}$	$\mathbf{90.99_{\pm6.64}}$
PROTEINS	$67.66_{\pm3.16}$	$70.71_{\pm4.41}$	$69.01_{\pm4.24}$	$69.01_{\pm3.48}$	$72.96_{\pm3.47}$	$71.25_{\pm2.66}$	$\underline{73.59_{\pm3.33}}$	$\mathbf{74.58_{\pm3.92}}$
IMDB-B	$73.00_{\pm4.49}$	$71.90_{\pm3.48}$	$72.60_{\pm4.20}$	$\underline{73.10_{\pm4.18}}$	$69.80_{\pm2.44}$	$66.40_{\pm3.35}$	-	$\mathbf{73.20_{\pm3.43}}$
IMDB-M	$\underline{48.47_{\pm3.90}}$	$47.87_{\pm3.07}$	$48.33_{\pm3.49}$	$47.93_{\pm2.88}$	$48.13_{\pm4.11}$	$43.60_{\pm3.17}$	-	$\mathbf{50.00_{\pm4.02}}$
REDDIT-B	$76.70_{\pm3.69}$	$79.35_{\pm3.46}$	$78.10_{\pm3.23}$	$\underline{79.55_{\pm2.20}}$	$72.35_{\pm2.91}$	$68.20_{\pm2.28}$	-	$\mathbf{92.70_{\pm2.38}}$
REDDIT-5K	$42.85_{\pm1.74}$	$\underline{50.87_{\pm2.63}}$	$50.03_{\pm1.49}$	$47.01_{\pm1.89}$	$50.27_{\pm1.92}$	$45.81_{\pm2.08}$	-	$\mathbf{56.51_{\pm2.22}}$

4.3 Graph Clustering Results

We employ cluster quality metrics to assess the embeddings of graphs sourced from all datasets in Table 2. The embeddings are labeled with their respective dataset memberships, and we assume that good embeddings will have graphs of the same dataset clustered together. We evaluate embeddings based on three widely used clustering metrics: Silhouette (SILH), Calinski-Harabasz (CH), and Davies-Bouldin (DB) [GBC21]. Table 6 compares the clustering performance of TopER and Spectral Zoo [JZ20], which is, to our knowledge, the only model that allows low-dimensional graph embeddings. Detailed results are provided in Appendix A.3. The findings demonstrate that the embeddings generated by TopER outperform those created by Spectral Zoo. This is evident from the superior cluster quality metrics observed for five out of eight datasets in the case of Silhouette and CH, and for all eight datasets in the case of DB.

Table 6: **Clustering Performances.** Comparison of Spectral Zoo vs. TopER. The detailed results are given in Appendix A.3.

Metric	Method	BZR	COX2	MUTAG	PROT.	IMDB-B	IMDB-M	REDD-B	REDD-5K
Silh ↑	Spec. Zoo	0.050	0.049	**0.344**	0.050	**0.097**	**-0.024**	0.108	-0.121
	TopER	**0.249**	**0.414**	0.258	**0.086**	0.064	-0.032	**0.196**	**-0.067**
CH ↑	Spec. Zoo	3.51	6.13	**120.73**	38.77	**85.24**	**30.98**	269.94	119.81
	TopER	**42.58**	**26.00**	72.52	**151.64**	60.52	11.77	**446.12**	**1209.95**
DB ↓	Spec. Zoo	7.25	6.07	0.95	4.55	2.78	10.73	2.20	25.74
	TopER	**1.93**	**2.29**	**0.88**	**1.54**	**2.19**	**6.87**	**1.32**	**2.78**

4.4 Graph Visualization

In the case of a single filtration function, TopER creates $2D$ graph embeddings (a, b) that can be visualized with ease (see Figure 1). Traditional dimensionality reduction techniques such as PCA can be used to visualize point cloud data, but accurately depicting graph data has historically been a significant challenge [GBGA20]. To our knowledge, until TopER, the only model that allowed graph visualization was the GraphZoo [JZ20].

TopER creates highly interpretable graph visualizations. To recall, the pair (a, b) represents the coefficients of the best-fitting function $L(x) = a + bx$, where a is the pivot (y-intercept) and b is the growth (the slope). Specifically, the pivot a reflects graph con-
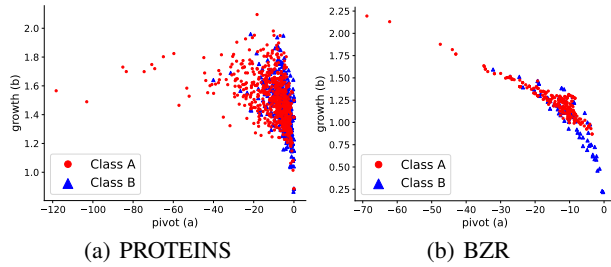

(a) PROTEINS   (b) BZR

Figure 4: TopER visualizations of the PROTEINS dataset with O.Ricci edge filtration, and the BZR dataset with degree centrality node filtration. Each point corresponds to an individual graph.

nectivity, while b reflects the growth rate of edges/nodes for the filtration function. In particular,

a higher value of a corresponds to a more interconnected graph. As we demonstrate in Appendix Figure 7, graph connectivity and community structure can be analyzed using three types of pivot behavior. In the following, we illustrate how these quantities can be employed to interpret our two-dimensional representations of the graph datasets.

In Figure 1(b) of the MUTAG dataset, class B has a higher growth rate and smaller pivot than the red class. This shows that the class is growing faster than class A with respect to the closeness function in the MUTAG dataset, i.e., the graph has a low diameter. Similarly, in contrast, in the PROTEINS dataset (Fig. 4(a)), the growth rates are similar for both classes ($\sim 1.5 - 1.7$), but the pivot (initial graph size) is smaller in class A. This implies that class A has fewer edges in relation to the number of nodes. Such patterns, as described in Sec. B.4, can reveal key insights into graph topology. In a similar vein, TopER visualizations can be used for anomaly detection. For example, in Fig. 1(a), an outlier PROTEINS graph alone has a positive pivot and appears as the rightmost data point.

More importantly, **TopER homogenizes graph representations, allowing us to compare graphs across datasets**, which may open new paths in training graph foundation models. To our knowledge, TopER is unique in directly producing interpretable 2D embeddings for cross-dataset visualization without relying on learned high-dimensional encodings or opaque projections, unlike GPSE [CLL$^+$24] and GFSE [CZW$^+$25], which rely on learned high-dimensional embeddings. For example, Figure 1(a) visualizes graphs of three datasets on the same panel, where we see that Mutag and COX2 differ in their pivot only. The similarity is not surprising; MUTAG and COX2 are datasets of molecular graphs where nodes are atoms and edges are chemical bonds. As the molecules in both datasets have similar types of atoms and bond configurations (e.g., ring structures), TopER captures these similarities, leading to similar embeddings.

These examples highlight that, in many practical settings, TopER's interpretability outweighs modest performance differences compared to more expressive or data-driven models. As shown in Figure 4, the pivot–growth representation captures fine-grained structural variations, such as changes in edge density, community organization, and filtration behavior, through simple geometric patterns that can be directly visualized and interpreted. This enables users to pinpoint the topological mechanisms responsible for observed differences between classes or datasets. Beyond classification, such interpretability makes TopER particularly well-suited for applications where structural understanding is critical. Its training-free and computationally efficient design further allows deployment in large-scale or data-scarce environments, as well as integration into hybrid pipelines where TopER embeddings serve as interpretable anchors for downstream learning models. Thus, while deep or spectral approaches may achieve marginally higher benchmark performance in some cases, TopER offers a complementary framework that prioritizes clarity, scalability, and theoretical grounding in topological structure.

Limitations. While our approach is designed to be efficient and broadly applicable, its performance can vary depending on the choice of filtration function, which may require domain knowledge in certain applications. In practice, we found the method to be robust across a variety of datasets, and further refinements to filtration strategies could enhance adaptability to new domains.

5   Conclusion

We have introduced a novel graph embedding method, *TopER*, leveraging Persistent Homology from Topological Data Analysis. *TopER* demonstrates strong performance in graph classification tasks, rivaling SOTA models. Furthermore, it naturally generates effective 2D visualizations of graph datasets, facilitating the identification of clusters and outliers. For future research, one promising direction is to extend *TopER* to temporal graph learning tasks, enabling the capture of dynamic graph trajectories that reflect evolving user behaviors over time. Another avenue is the integration of *TopER* embeddings into graph foundation models, where the homogenization of graph structures could enhance the learning of transferable representations across different domains.

References

[AAF19] Mehmet Emin Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: methods and applications. *Appl. Netw. Sci.*, 4(1):61:1–28, 2019.

[AKGC22] Cuneyt Gurcan Akcora, Murat Kantarcioglu, Yulia R. Gel, and Baris Coskunuzer. Reduction algorithms for persistence diagrams of networks: Coraltda and prunit. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[BCL21] Cristian Bodnar, Catalina Cangea, and Pietro Liò. Deep graph mapper: Seeing graphs through the neural lens. *Frontiers Big Data*, 4:680535, 2021.

[BGA19] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut pooling in graph neural networks. *arXiv*, https://arxiv.org/abs/1907.00481, 2019.

[BGL+20] Karsten M. Borgwardt, M. Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *Found. Trends Mach. Learn.*, 13(5-6), 2020.

[BHG+21] Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 599–608. PMLR, 2021.

[BK10] Michael M. Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 1704–1711. IEEE Computer Society, 2010.

[BN01] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 585–591. MIT Press, 2001.

[BP07] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *Int. J. Bifurc. Chaos*, 17(7):2303–2318, 2007.

[BPL+21] Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 748–758. PMLR, 2021.

[BZSL14] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[CA24] Baris Coskunuzer and Cüneyt Gürcan Akçora. Topological methods in machine learning: A tutorial for practitioners. *arXiv*, http://arxiv.org/abs/2409.02901, 2024.

[Cai21] Chen Cai. Sanity check for persistence diagrams. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning (GTRL)*. OpenReview.net, 2021.

[CCB+20] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[CFG24] Yuzhou Chen, José Frías, and Yulia R. Gel. Topogcl: Topological graph contrastive learning. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 11453–11461. AAAI Press, 2024.

[CHM12] Nikolai Chernov, Qiang Huang, and Hongwei Ma. Does the best-fitting curve always exist? *International Scholarly Research Notices*, 2012:895178, 2012.

[CLL+24] Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampásek. Graph positional and structural encoder. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[CLX+21] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1204–1215. PMLR, 2021.

[CPWC23] Yun Young Choi, Sun Woo Park, Youngho Woo, and U Jin Choi. Cycle to clique (cy2c) graph neural network: A sight to see beyond neighborhood aggregation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[CSA+23] Yuzhou Chen, Ignacio Segovia-Dominguez, Cuneyt Gurcan Akcora, Zhiwei Zhen, Murat Kantarcioglu, Yulia R. Gel, and Baris Coskunuzer. EMP: effective multidimensional persistence for graph representation learning. In *Learning on Graphs Conference, 27-30 November 2023, Virtual Event*, volume 231 of *Proceedings of Machine Learning Research*, page 24. PMLR, 2023.

[CZC18] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018.

[CZW+25] Jialin Chen, Haolan Zuo, Haoyu Peter Wang, Siqi Miao, Pan Li, and Rex Ying. Towards A universal graph structural encoder. *arXiv*, http://arxiv.org/abs/2504.10917, 2025.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.

[DCG+22] Andac Demir, Baris Coskunuzer, Yulia R. Gel, Ignacio Segovia-Dominguez, Yuzhou Chen, and Bulent Kiziltan. Todd: Topological compound fingerprinting in computer-aided drug discovery. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[DCS+22] Youxiang Duan, Ning Chen, Shigen Shen, Peiying Zhang, Youyang Qu, and Shui Yu. FDSA-STG: fully dynamic self-attention spatio-temporal graph networks for intelligent traffic flow prediction. *IEEE Trans. Veh. Technol.*, 71(9):9250–9260, 2022.

[DG23] Paweł Dłotko and Davide Gurnari. Euler characteristic curves and profiles: A stable shape invariant for big data problems. *GigaScience*, 12:giad094, 2023.

[DTRF19] Xiaowen Dong, Dorina Thanou, Michael G. Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Process. Mag.*, 36(3):44–63, 2019.

[EC22] Tim S. Evans and Bingsheng Chen. Linking the network centrality measures closeness and degree. *Communications Physics*, 5(1):172, July 2022.

[EPBM20] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[FHSK23] Zhongxi Fang, Jianming Huang, Xun Su, and Hiroyuki Kasai. Wasserstein graph distance based on l1-approximated tree edit distance between weisfeiler-lehman subtrees. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 7539–7549. AAAI Press, 2023.

[GBC21] Marek Gagolewski, Maciej Bartoszuk, and Anna Cena. Are cluster validity measures (in) valid? *Inf. Sci.*, 581:620–636, 2021.

[GBGA20] Loann Giovannangeli, Romain Bourqui, Romain Giot, and David Auber. Toward automatic comparison of visualization techniques: Application to graph visualization. *Vis. Informatics*, 4(2):86–98, 2020.

[GF18] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.*, 151:78–94, 2018.

[GHT+19] Xiang Gao, Wei Hu, Jiaxiang Tang, Jiaying Liu, and Zongming Guo. Optimized skeleton-based action recognition via sparsified graph regression. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, pages 601–610. ACM, 2019.

[GJ19] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2083–2092. PMLR, 2019.

[Gün22] Stephan Günnemann. Graph neural networks: Adversarial robustness. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 149–176. Springer Nature Singapore, Singapore, 2022.

[Ham20] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.

[HFZ+20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[HGR+20] Christoph D. Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4314–4323. PMLR, 2020.

[HIL+24] Yasuaki Hiraoka, Yusuke Imoto, Théo Lacombe, Killian Meehan, and Toshiaki Yachimura. Topological node2vec: Enhanced graph embedding via persistent homology. *J. Mach. Learn. Res.*, 25:134:1–26, 2024.

[HJLH22] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 8230–8248. PMLR, 2022.

[HK02] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2):026107, January 2002.

[HKN19] Christoph D. Hofer, Roland Kwitt, and Marc Niethammer. Learning representations of persistence barcodes. *J. Mach. Learn. Res.*, 20:126:1–45, 2019.

[HLAK24] Jaeseung Heo, Seungbeom Lee, Sungsoo Ahn, and Dongwoo Kim. EPIC: graph augmentation with edit path interpolation via learnable cost. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 4116–4126. ijcai.org, 2024.

[HMR21] Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers Artif. Intell.*, 4:681108, 2021.

[ISG23] Johanna Immonen, Amauri H. Souza, and Vikas Garg. Going beyond persistent homology using persistent homology. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[JWH+23] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Springer International Publishing, Cham, 2023.

[JZ20] Shengmin Jin and Reza Zafarani. The spectral zoo of networks: Embedding and visualizing networks with spectral moments. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1426–1434. ACM, 2020.

[KM12] Nils M. Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[KW17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[LBKT08] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 462–470. ACM, 2008.

[LLK19] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3734–3743. PMLR, 2019.

[LLY11] Yong Lin, Linyuan Lu, and Shing-Tung Yau. Ricci curvature of graphs. *Tohoku Mathematical Journal*, 63(4):605–627, 2011.

[LSC+23] David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Bakke Botnan, and Steve Oudot. Stable vectorization of multiparameter persistent homology using signed barcodes as measures. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[LWZ+22] Sihang Li, Xiang Wang, An Zhang, Yingxin Wu, Xiangnan He, and Tat-Seng Chua. Let invariant rationale discovery inspire graph contrastive learning. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 13052–13065. PMLR, 2022.

[MK20] Costas Mavromatis and George Karypis. Graph infoclust: Leveraging cluster-level node information for unsupervised graph representation learning. *arXiv*, http://arxiv.org/abs/2009.06946, 2020.

[MV09]   Pierre Mahé and Jean-Philippe Vert. Graph kernels based on tree patterns for molecules. *Mach. Learn.*, 75(1):3–35, 2009.

[New03] Mark E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45(2):167–256, 2003.

[NV23]   Giannis Nikolentzos and Michalis Vazirgiannis. Graph alignment kernels using weisfeiler and leman hierarchies. In *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pages 2019–2034. PMLR, 2023.

[OPT+17] Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Sci.*, 6(1):17, 2017.

[ORB21] Leslie O'Bray, Bastian Rieck, and Karsten M. Borgwardt. Filtration curves for graph representation. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1267–1275. ACM, 2021.

[PAMF23] Daniele Paliotta, Mathieu Alain, Bálint Máté, and François Fleuret. Graph neural networks go forward-forward. In *NeurIPS 2023 Workshop on Graph Learning Frontiers*, 2023. Workshop Poster.

[PAS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM, 2014.

[QTLL24] Chen Qian, Huayi Tang, Hong Liang, and Yong Liu. Reimagining graph classification from a prototype view with optimal transport: Algorithm and theorem. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 2444–2454. ACM, 2024.

[RBB19] Bastian Rieck, Christian Bock, and Karsten M. Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5448–5458. PMLR, 2019.

[SOG09] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Comput. Graph. Forum*, 28(5):1383–1392, 2009.

[ST20]   Primož Skraba and Katharine Turner. Wasserstein stability for persistence diagrams. *arXiv*, http://arxiv.org/abs/2006.16824, 2020.

[TGL+19] Matteo Togninalli, M. Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten M. Borgwardt. Wasserstein weisfeiler-lehman graph kernels. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6436–6446, 2019.

[TPPM23] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *J. Mach. Learn. Res.*, 24:127:1–21, 2023.

[WCZ16] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1225–1234. ACM, 2016.

[WJY⁺23] Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Kompella, and Zhangyang Wang. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[WKK⁺20] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

[WW22] Asiri Wijesinghe and Qing Wang. A new perspective on "how graph neural networks go beyond weisfeiler-lehman?". In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Xu21] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Rev.*, 63(4):825–853, 2021.

[YCL⁺21] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28877–28888, 2021.

[YLY⁺18] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6412–6422, 2018.

[YSK22] Jaemin Yoo, Sooyeon Shim, and U Kang. Model-agnostic augmentation for accurate graph classification. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 1281–1291. ACM, 2022.

[YV15] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1365–1374. ACM, 2015.

[YWH⁺22] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 8892–8900. AAAI Press, 2022.

[YYM⁺18] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4805–4815, 2018.

[ZYCW20] Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2896–2906. PMLR, 2020.

[ZZM24] Zhiqiang Zhong, Kuangyu Zhou, and Davide Mottin. Benchmarking large language models for molecule prediction tasks. *arXiv*, https://arxiv.org/abs/2403.05075, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction accurately reflect the theoretical, methodological, and experimental contributions presented in the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in a dedicated paragraph, noting areas such as dependence on the filtration function and computational scalability to very large graphs.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 4 states the full set of assumptions for each theorem, and the complete proofs are provided in the Appendix.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The main paper and appendix include detailed descriptions of experimental setups, and an anonymous code repository is provided to ensure full reproducibility.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is provided via an anonymous link in the supplementary materials, along with detailed instructions for reproducing all key experiments.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Training/test splits, model configurations, optimizer settings, and hyperparameter values are all specified either in the main text or in the appendix.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All reported results include standard deviations across multiple runs, with details on how error bars were computed described in the experimental section.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The appendix provides details about the computational environment used for all experiments, including GPU type and training time.

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The research complies fully with the NeurIPS Code of Ethics and does not involve sensitive data, human subjects, or high-risk applications.

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The Broader Impact section discusses potential benefits of interpretability and efficiency in scientific applications, while noting ethical considerations related to graph data usage.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The models and datasets used in the paper do not present high risk for misuse and thus do not require additional safeguards.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and external codebases are cited properly and used under their respective open-source licenses.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper introduces a new method and accompanying code, which are thoroughly documented and made available via anonymous link.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing or any research involving human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The research does not involve human participants and does not require IRB approval.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: No large language models were used in the methodology or experiments; any language assistance was limited to standard editing support.

Appendix

A   Further Experimental Details

A.1   OGBG-MOLHIV Results

For the OGBG-MOLHIV dataset, we further evaluated the improvements of TopER with the addition of new filtration functions. Table 7 provides the performance of each TopER$-i$, where i represents number of filtration functions used in the model, i.e., TopER-i uses $\{(a_1, b_1, \ldots, a_i, b_i)\}$ as graph embedding where (a_i, b_i) is the pivot and growth for function f_i. We used XG-Boost to rank the importance of filtration functions first, and the functions are added iteratively with this ranking. We fixed max-

Table 7: Results for OGBG-MOLHIV of each TopER$-i$.

Method	Added Function	Valid. AUC	Test AUC
TopER-1	degree-centrality	$72.76_{\pm 0.23}$	$74.44_{\pm 0.20}$
TopER-2	atomic weight	$71.89_{\pm 0.12}$	$74.25_{\pm 0.16}$
TopER-3	O. Ricci	$70.11_{\pm 0.28}$	$76.79_{\pm 0.24}$
TopER-4	F. Ricci	$71.76_{\pm 0.18}$	$78.15_{\pm 0.15}$
TopER-5	degree	$71.79_{\pm 0.35}$	$79.26_{\pm 0.14}$
TopER-6	popularity	$72.27_{\pm 0.29}$	$79.88_{\pm 0.24}$
TopER-7	closeness	$71.30_{\pm 0.18}$	$\mathbf{80.21_{\pm 0.15}}$

imum tree depth $= 3$, learning rates $= 0.035$, subsample ratios $= 0.95$, the number of estimators $= 1000$, and the regularization parameter lambda $= 45$, where the objective function is rank:pairwise, with log loss as the evaluation metric. The seed is set to be 16.

A.2   Time Experiments for TopER vs. PH

To compare the time efficiency and performance of TopER and persistent homology (PH),

We conducted experiments using the same filtration function, the sublevel degree filtration. For PH, we applied Betti vectorization. Our results, summarized below, show that TopER is significantly faster than PH. Although both methods use the same filtration function, a key distinction lies in their embeddings: TopER generates 2D embeddings, whereas PH produces a vector with dimensionality equal to the number of thresholds in the filtration. Despite the considerable difference in dimensionality, TopER's performance with 2D embeddings remains comparable to that of PH.

Figure 5 shows that TopER scales efficiently with graph size, maintaining low runtime even with 100 filtration steps and high node degree. It processes graphs with up to 100,000 nodes in just over a minute, demonstrating its suitability for large-scale applications.
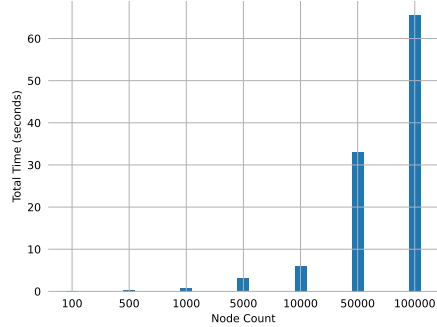

Figure 5: **Scalability.** TopER run time for synthetic power law graphs [HK02] with node degree filtration. The mean node degree is 30, and 100 filtration steps are used.

A.3   Clustering Performances

In Table 9, we showcase our clustering performance across eight benchmark graph classification datasets using three widely adopted clustering metrics: Silhouette, Calinski-Harabasz, and Davies-Bouldin. These metrics serve as evaluative measures for assessing the efficacy of clustering algorithms

Table 8: Comparison of TopER-1 (only one filtration function) and PH in terms of time and accuracy across different datasets.

Dataset	TopER-1		PH		# Thresholds
	Time	Accuracy	Time	Accuracy	
BZR	1.14 s	82.73 ± 2.12	5.99 s	83.70 ± 3.51	4
IMDB-B	3.27 s	73.10 ± 4.18	319.95 s	71.00 ± 4.07	65
REDDIT-B	107.65 s	79.55 ± 2.20	9173.37 s	84.50 ± 2.51	501

Table 9: The clustering performances of Spectral Embeddings and TopER with different metrics. Best performances are given in **blue**.

Silhouette Scores (↑)

Method	BZR	COX2	MUTAG	PROT.	IMDB-B	IMDB-M	REDD-B	REDD-5K
Spec Zoo	0.050	0.049	**0.344**	0.050	**0.097**	**-0.024**	0.108	-0.121
degree	-0.108	**0.414**	0.258	0.048	0.030	-0.032	0.049	-0.169
popularity	**0.249**	-0.015	0.134	-0.000	0.008	-0.159	**0.196**	-0.173
closeness	0.019	0.036	0.036	**0.086**	nan	nan	0.087	-0.185
degree	0.084	0.030	0.017	0.065	0.056	-0.075	0.034	**-0.067**

Calinski-Harabasz scores (↑)

Method	BZR	COX2	MUTAG	PROT.	IMDB-B	IMDB-M	REDD-B	REDD-5K
Spec Zoo	3.51	6.13	**120.73**	38.77	**85.24**	**30.98**	269.94	119.81
degree	0.42	1.06	11.29	130.07	60.52	3.92	97.85	**1209.95**
popularity	13.85	**26.00**	36.13	77.22	12.89	11.77	**446.12**	619.37
closeness	**42.58**	1.02	40.04	73.51	10.17	0.30	188.10	689.27
F.Ricci	4.92	0.48	11.82	**151.64**	11.68	1.03	92.14	454.34

Davies-Bouldin scores (↓)

Method	BZR	COX2	MUTAG	PROT.	IMDB-B	IMDB-M	REDD-B	REDD-5K
Spec Zoo	7.25	6.07	0.95	4.55	2.78	10.73	2.20	25.74
degree	9.84	**2.29**	**0.88**	1.95	4.92	46.46	2.32	3.27
popularity	4.16	37.87	1.62	2.11	25.25	**6.87**	**1.32**	3.46
closeness	**1.93**	26.44	1.41	2.25	4.99	37.51	1.95	**3.09**
F.Ricci	4.19	7.20	1.27	**1.54**	**2.19**	10.35	1.83	5.41

in partitioning datasets into meaningful clusters. They gauge the degree of similarity or dissimilarity within and between clusters, offering insights into the quality of clustering outcomes. For precise definitions of Silhouette, Calinski-Harabasz, and Davies-Bouldin metrics, as well as additional details on clustering measures, refer to [GBC21].

A.4 Number of Thresholds

In our experiments, we utilized a large number of thresholds to capture finer-grained information, as the model is computationally efficient and the additional cost of increasing the number of thresholds is minimal. Furthermore, in Table 10, we evaluated the model's performance with fewer thresholds and observed that it remains robust and highly effective even in such scenarios.

Table 10: The accuracy results of TopER with different numbers of thresholds.

# Thresholds	PROTEINS	REDDIT-B	REDDIT-5K
10	$72.78_{\pm 4.04}$	$90.55_{\pm 1.96}$	$55.99_{\pm 1.97}$
20	$74.31_{\pm 3.23}$	$91.20_{\pm 1.66}$	$55.91_{\pm 2.14}$
50	$74.76_{\pm 4.55}$	$92.05_{\pm 1.96}$	$55.39_{\pm 2.10}$
100	$73.85_{\pm 3.67}$	$92.85_{\pm 1.18}$	$55.51_{\pm 2.61}$
200	$75.47_{\pm 3.06}$	$93.15_{\pm 2.10}$	$56.51_{\pm 2.04}$
500	$74.58_{\pm 3.92}$	$92.70_{\pm 2.38}$	$56.51_{\pm 3.22}$

A.5 Combining Filtration Functions

To assess the impact of embedding dimensions, we conducted new experiments evaluating the performance of the TopER model by progressively adding each filtration function step by step. This analysis provides insights into how the inclusion of additional filtration functions influences the model's performance. In Table 11, the TopER-n model represents the TopER utilizing n-filtration functions (2n features).

Table 11: Performance improvements achieved by integrating filtration functions into the TopER model. Here, TopER-n denotes the TopER model with n filtration functions.

Dataset	TopER-1	TopER-2	TopER-3	TopER-4
BZR	82.48 ± 1.98	84.70 ± 2.84	85.66 ± 5.00	86.68 ± 3.81
COX2	78.81 ± 1.94	79.26 ± 4.86	79.04 ± 7.49	80.30 ± 3.91
MUTAG	86.14 ± 6.38	88.33 ± 3.88	86.75 ± 4.78	88.30 ± 4.63
PROTEINS	74.03 ± 2.71	74.67 ± 2.73	75.21 ± 3.39	75.65 ± 3.87
IMDB-B	73.00 ± 4.40	74.20 ± 4.26	74.50 ± 3.50	74.70 ± 3.95
IMDB-M	48.73 ± 4.33	49.80 ± 2.94	49.73 ± 4.18	49.87 ± 4.00
REDDIT-B	81.95 ± 2.74	90.45 ± 2.55	91.05 ± 2.62	91.50 ± 2.01
REDDIT-5K	50.21 ± 1.41	54.11 ± 2.43	56.19 ± 2.40	56.33 ± 2.74

A.6 TopER filtrations runtimes and substitute

Filtration Timing Results. In Table 12, we report the computation times (in seconds) for the TOPER filtration functions across various datasets. We also include the timings for the Heat Kernel Signature (HKS) [SOG09], which can serve as an efficient substitute for the Ollivier–Ricci curvature in certain scenarios due to its faster computation while preserving relevant structural information about the graphs. The table below summarizes the observed computation times for each filtration type across multiple benchmark datasets.

Table 12: Computation times (in seconds) of different filtration functions across datasets.

Filtration	BZR	COX2	MUTAG	PROTEINS	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K
Degree Centrality	0.86	1.67	0.40	13.82	13.32	13.03	115.09	447.72
Popularity	0.73	1.36	0.58	5.81	13.21	15.60	111.11	414.04
Closeness	2.29	4.40	0.77	15.97	5.85	6.63	399.2	1274.00
Forman Ricci	0.80	1.56	0.58	4.63	8.29	7.38	258.32	693.18
Ollivier Ricci	130.04	121.42	48.98	313.21	277.66	428.32	1291.93	6640.95
Degree	1.14	1.16	0.61	2.62	3.27	4.22	107.65	363.32
Weight	2.25	2.80	0.90	12.89	–	–	–	–
HKS	2.07	1.94	0.56	16.90	15.54	15.72	470.01	1007.50

We evaluate the performance of our TopER model against state-of-the-art (SOTA) methods on benchmark graph classification datasets, including BZR, COX2, MUTAG, PROTEINS, IMDB-B, IMDB-M, REDDIT-B, and REDDIT-5K. Table 13 reports classification accuracy (mean ± standard deviation) across multiple runs. TopER generally achieves competitive results compared to SOTA. Ablation studies show the impact of Ricci curvature and Heat Kernel Signatures (HKS) on model performance. Notably, TopER without Ricci but with HKS recovers most of the performance lost when Ricci is removed, suggesting that HKS can serve as a viable replacement for Ollivier-Ricci curvature in capturing structural information.

Table 13: Graph classification accuracy (mean ± std) for different models across benchmark datasets. Highest scores per dataset are **bold blue**, second-highest are <u>underlined blue</u>.

Model	BZR	COX2	MUTAG	PROTEINS	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K
SOTA	89.00±5.00	**85.53±1.60**	**92.63±2.58**	**77.30±0.89**	**75.08±0.31**	**52.81±0.31**	91.03±0.22	<u>56.75±0.18</u>
TopER	**90.13±4.14**	82.01±4.59	90.99±6.64	74.58±3.92	73.20±3.43	50.00±4.02	<u>92.70±2.38</u>	56.51±2.22
TopER w/o O. R.	87.00±4.30	77.96±8.38	87.78±7.84	74.04±3.86	73.50±3.53	50.00±5.44	91.90±2.63	56.37±1.89
TopER w/o O. R.& w/ HKS	<u>89.63±3.65</u>	81.58±3.54	<u>92.08±4.23</u>	<u>75.20±3.59</u>	75.00±3.49	<u>50.67±5.58</u>	**92.75±2.47**	**57.33±2.02**

A.7 Hyperparameters

Our proposed MLP algorithm is constructed with a single hidden layer. The output layer's activation function is set to log softmax, and the loss function we used is Negative Log Likelihood Loss. The learning rate is chosen between 0.01 and 0.001. Subsequently, we investigate the impact of the number of neurons in the hidden layer, considering values from the set {16, 64, 128}. The optimizer is set to be Adam, and the number of epochs is 500. To prevent large weights and overfitting, we apply L2 regularization coefficients of 1e-3, 1e-4. The activation function for the hidden layer varies between ReLU, GeLU, and ELU. Lastly, we consider the cases of adding or not a batch normalization

---

Algorithm 1 TopER: Topological Evolution Rate

Input: Graph $\mathcal{G}$, Filtration function $f : \mathcal{V} \to \mathbb{R}$, Threshold set $\mathcal{I} = \{\epsilon_i\}_{i=0}^n$
Output: TopER vector $\mathcal{T}_f(\mathcal{G}, \mathcal{I})$
Initialize lists $\mathcal{X} = [], \mathcal{Y} = []$
for $i = 1$ **to** n **do**
   $\mathcal{G}_i \leftarrow$ Induced subgraph of $\mathcal{G}$ where $\mathcal{V}_i \subseteq f^{-1}([\epsilon_0, \epsilon_i])$
   $x_i \leftarrow |\mathcal{V}_i|$
   $y_i \leftarrow |\mathcal{E}_i|$
   Append x_i to $\mathcal{X}$
   Append y_i to $\mathcal{Y}$
end for
Fit a line $\mathcal{L}(x) = a + bx$ to pairs (x_i, y_i) from $\mathcal{X}$ and $\mathcal{Y}$ using least squares
Extract coefficients a and b
Return (a, b) as the TopER vector $\mathcal{T}_f(\mathcal{G}, \mathcal{I})$

---

layer to the output of the hidden layer and setting dropout values to be 0.0 or 0.5. In Table 14, we provide the details for each dataset. The last column shows the number of TopER features used for each dataset after the feature selection step.

Table 14: Employed hyperparameters for each dataset.

Dataset	Neurons	Dropout	Batch Norm.	Decay	Learning rate	Activation	TopER Dim.
BZR	64	0.5	True	1e-4	0.001	gelu	26
COX2	128	0	True	1e-4	0.01	relu	26
MUTAG	16	0.5	False	1e-3	0.01	gelu	20
PROTEINS	64	0.5	True	1e-3	0.01	elu	26
IMDB-B	128	0	False	1e-3	0.001	relu	20
IMDB-M	16	0	False	1e-3	0.01	elu	20
REDDIT-B	64	0.5	False	1e-3	0.01	relu	24
REDDIT-5K	128	0	False	1e-3	0.01	elu	14

B   More on TopER

B.1   Refining the point set

While we have described the main steps of TopER in Section 3, due to the repetitions of the points in $\mathcal{A} = \{(x_i, y_i)\} \subset \mathbb{R}^2$, there are some choices to be made before defining the set $\mathcal{A}$ (i.e., $\mathcal{X}$ and $\mathcal{Y}$) to get the best fitting function $L : \mathcal{X} \to \mathcal{Y}$. The main reason is that the set $\{(x_i, y_i)\}_{i=1}^N$ can contain repetitions of x-values ($x_i = x_{i+1}$), repetitions of y-values ($y_i = y_{i+1}$) or repetitions of both ($(x_i, y_i) = (x_{i+1}, y_{i+1})$) depending on the filtration function, the threshold set $\mathcal{I}$, and the graph $\mathcal{G}$.

For the filtrations induced by *node filtration functions*, the number of edges can not change unless the number of nodes changes, i.e., $x_i = x_{i+1} \Rightarrow y_i = y_{i+1}$. Hence, with this elimination, we still allow keeping y-values the same while x-values are increasing. This means there can be horizontal jumps in $\mathcal{A}_u$. In this paper, to eliminate all horizontal jumps for filtrations with node functions, we eliminate all repetitions of y-values from $\mathcal{A}_u$. In particular, we remove all the points with the same $\widehat{y}$-value and add a point with a mean of x-values. In other words, if $y_i = y_{i+1} = \cdots = y_{i+k} = \widehat{y}$, we define $\widehat{x} = \text{mean}\{x_i, x_{i+1}, \ldots, x_{i+k}\}$. Then, we replace (k+1) points $\{(x_i, \widehat{y}), (x_{i+1}, \widehat{y}), \ldots, (x_{i+k}, \widehat{y})\}$ with one point $(\widehat{x}, \widehat{y})$ in $\mathcal{A}_u$. This process eliminates all repetitions and horizontal jumps in $\mathcal{A}$, and we define our best-fitting line on this refined set.

B.2   TopER with Alternative Quantities

While we use the most general quantities for a graph—the count of vertices and edges—in our algorithm, depending on the problem, there might be other induced quantities (x_i, y_i) for a given subgraph $\mathcal{G}_i$ which can give better vectors. To keep the line-fitting approach meaningful in our model, as long as the sequences $\{x_i\}$ and $\{y_i\}$ are monotone like our node-edge counts above, for a given

dataset in a domain (e.g., biochemistry, finance), one can use other domain-related quantities induced by substructure $\mathcal{G}_i$ as a (x_i, y_i) pair to obtain a TopER vector.
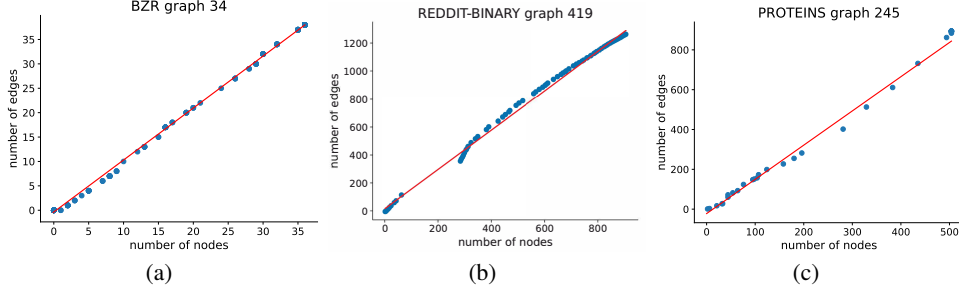

Figure 6: **Linear Fit.** TopER summarizes the growth behavior in the graph induced by filtration with a linear fit.

B.3 Linear or Higher Order Fitting

In our experiments, we observe that linear fitting captures the growth information for node-edge pair $\{(x_i, y_i)\}$ well (See Figure 6), and quadratic fit and linear fit stay very close to each other. However, if one decides to use other quantities as described above and loses the monotonicity of the sequences $\{x_i\}$ and $\{y_i\}$, trying higher order fits (e.g., $y = ax^2 + bx + c$) can be more meaningful. In Table 15, we present the average of the coefficients of quadratic terms when we use quadratic fit for the datasets, i.e., if we fit $y = a + bx + cx^2$ polynomial, we observe that the quadratic term cx^2 is mostly negligible, and tends to be a linear fit.

Table 15: Average of x^2 coefficient across datasets for quadratic fitting.

Dataset	BZR	COX2	MUTAG	REDDIT-5k
Average of x^2 Coefficient	4.71×10^{-5}	6.61×10^{-4}	1.16×10^{-2}	1.78×10^{-5}

B.4 Interpreting TopER

Our approach involves accurately modeling the evolution of a graph throughout the filtration process. One can easily identify clusters for each class and outliers in the other datasets given in Figure 1(a) and make inferences about the different clusters and outliers. Furthermore, when the pivot a_f is positive or negative, it can be interpreted as graph density behavior in the filtration sequence (See Figure 7).

C   Proofs of Stability Theorems

In this part, we prove the stability results for our TopER.

Lemma 3.6. [ST20] Let $\mathcal{X}$ be a compact metric space, and $f, g : \mathcal{X} \to \mathbb{R}$ be two filtration functions. Then, for any $p \geq 1$, we have $\mathcal{W}_p(\mathrm{PD}_k(\mathcal{X}, f), \mathrm{PD}_k(\mathcal{X}, g)) \leq \|f - g\|_p$

The next lemma is on the stability of Betti curves by [DG23] [Proposition 1].

Lemma 3.7. [DG23] Let $\beta_k(\mathcal{X})$ is the kth Betti function obtained from the persistence module $\mathrm{PM}_k(\mathcal{X})$.
$$\|\beta_k(\mathcal{X}) - \beta_k(\mathcal{Y})\|_1 \leq 2\mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}), \mathrm{PD}_k(\mathcal{Y}))$$

Now, we are ready to prove our stability result.

Theorem 3.4 Let $\mathcal{X}$ be a compact metric space, and $f, g : \mathcal{X} \to \mathbb{R}$ be two filtration functions. Then, for some $C > 0$,
$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq \mathrm{C} \cdot \mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}, f), \mathrm{PD}_k(\mathcal{X}, g))$$
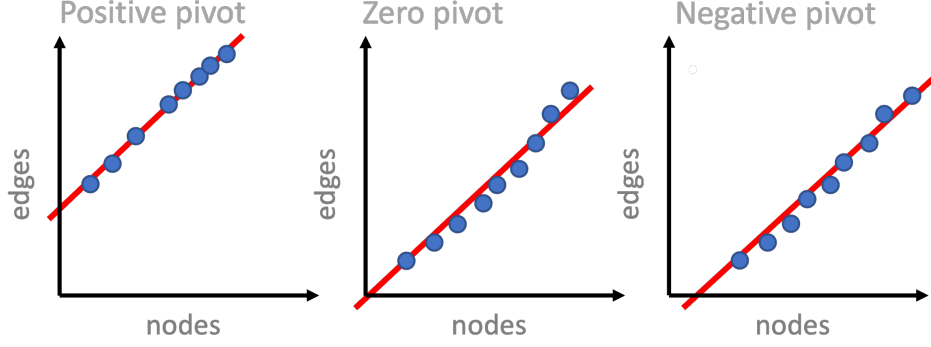
Figure 7: **Pivot Behavior.** A graph can exhibit three distinct pivot behaviors. Positive pivot graphs display a cluster of vertices that are closely interconnected and appear early in the filtration process. On the other hand, negative pivot graphs feature loosely connected nodes where the edges enter the filtration at a later stage. Graphs with zero pivot are usually quasi-complete graphs.

Proof. We will utilize the stability theorems from topological data analysis given above.

First, we employ the stability of Betti curves by Lemma 3.7.

$$\|\beta_k(\mathcal{X}) - \beta_k(\mathcal{Y})\|_1 \leq 2\mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}), \mathrm{PD}_k(\mathcal{Y})) \tag{1}$$

Hence to obtain $\mathrm{TE}_f(\mathcal{X}) = (\mathrm{a_f}, \mathrm{b_f})$, we fit least squares line $y = a_f + b_f x$ to the set of N points in $\mathbb{R}^2$, i.e., $\mathcal{Z}_f = \{(\beta_0^f(\epsilon_i), \beta_1^f(\epsilon_i))\}_{i=1}^N$. Similarly, we obtain $\mathrm{TE}_g(\mathcal{X}) = (\mathrm{a_g}, \mathrm{b_g})$ by fitting least squares line to $\mathcal{Z}_g = \{(\beta_0^g(\epsilon_i), \beta_1^g(\epsilon_i))\}_{i=1}^N$. By Equation (1), we have

$$\mathbf{D}_H(\mathcal{Z}_f, \mathcal{Z}_g) \leq 4\mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}), \mathrm{PD}_k(\mathcal{Y})) \tag{2}$$

where $\mathbf{D}_H(\mathcal{Z}_f, \mathcal{Z}_g)$ represent Hausdorff distance between the point clouds $\mathcal{Z}_f$ and $\mathcal{Z}_g$ in $\mathbb{R}^2$.

Now, by the stability of least squares fit with respect to Hausdorff distance ([CHM12] [Theorem 3.1]), we have

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq \mathrm{C} \cdot \mathbf{D}_H(\mathcal{Z}_f, \mathcal{Z}_g) \tag{3}$$

Hence, when we combine Equations (2) and (3), we have

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq \mathrm{C} \cdot \mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}), \mathrm{PD}_k(\mathcal{Y}))$$

The proof follows. □

By combining the above result with Lemma 3.6, we obtain the following corollary.

Corollary 3.5 Let $\mathcal{X}$ be a compact metric space, and $f, g : \mathcal{X} \to \mathbb{R}$ be two filtration functions. Then, for some $C > 0$,

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq \mathrm{C} \cdot \|\mathrm{f} - \mathrm{g}\|_1$$

Proof. By Lemma 3.6, we have

$$\mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}, \mathrm{f}), \mathrm{PD}_k(\mathcal{X}, \mathrm{g})) \leq \|\mathrm{f} - \mathrm{g}\|_1 \tag{4}$$

By Theorem 3.4, we have

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq \mathrm{C} \cdot \mathcal{W}_1(\mathrm{PD}_k(\mathcal{X}, \mathrm{f}), \mathrm{PD}_k(\mathcal{X}, \mathrm{g})) \tag{5}$$

By combining Equations (4) and (5), we conclude

$$\|\mathrm{TE}_f(\mathcal{X}) - \mathrm{TE}_g(\mathcal{X})\|_1 \leq \widehat{\mathrm{C}} \cdot \|\mathrm{f} - \mathrm{g}\|_1$$

The proof follows. □

D   Synthetic Experiments

This section describes experiments performed on the Erdos-Renyi synthetic graph. We conducted two experiments by applying TopER on this graph and Principal Component Analysis (PCA). The goal is to compare the performance, runtime, and interpretability of the two models. From the plots shown in Figures 8, 9, and 10, we see that TopER is interpretable compared to when PCA is used.
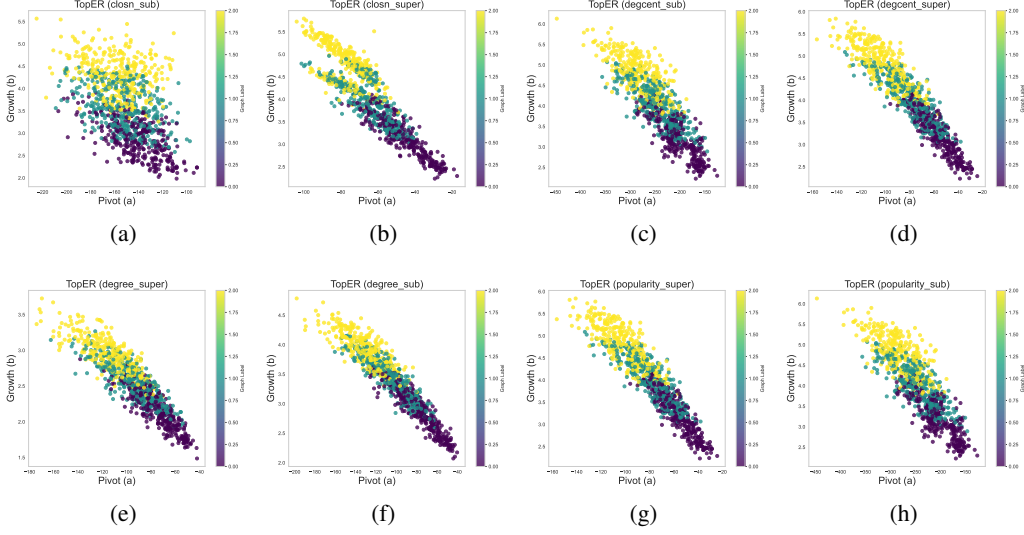

Figure 8: TopER plots showing pivot vs growth for each function—degree centrality, closeness, degree, and popularity when threshold=50.
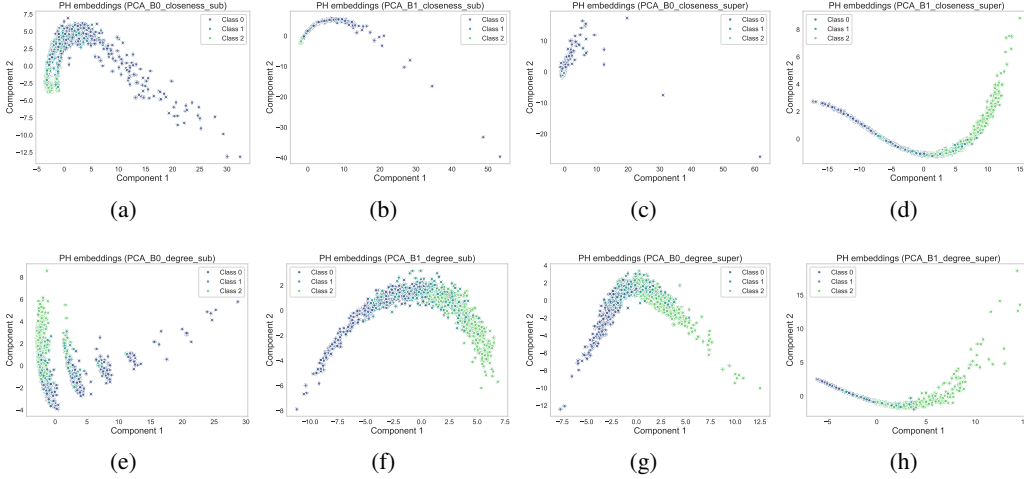

Figure 9: Persistent Homology PCA plots embeddings (Betti 0 and Betti 1) showing component 1 vs component 2 for each function—closeness and degree when threshold=50 and the respective filtrations —sublevel and superlevel.

E   Broader Impact

This work advances the field of graph representation learning by introducing a topological approach that is both interpretable and scalable. By leveraging the structural insights of persistent homology without incurring its prohibitive computational costs, *TopER* enables more efficient and insightful analysis of complex graph-structured data. This has the potential to benefit a range of scientific and
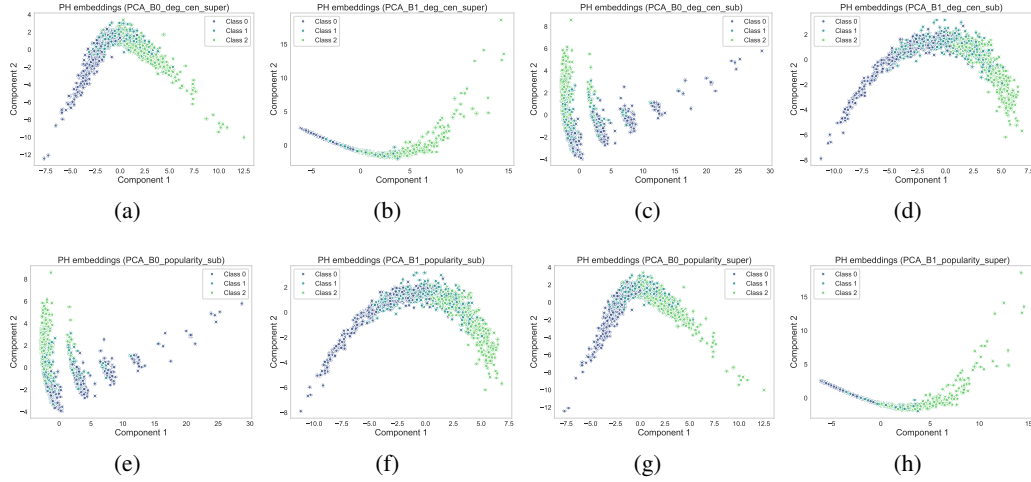
Figure 10: Persistent Homology PCA plots embeddings (Betti 0 and Betti 1) showing component 1 vs component 2 for each function—degree centrality and popularity when threshold=50 and the respective filtrations —sublevel and superlevel.

industrial domains where graph data is prevalent, including bioinformatics, social network analysis, and infrastructure monitoring. In particular, the ability to generate low-dimensional and interpretable embeddings could assist researchers in visual analytics, pattern discovery, and model debugging. At the same time, we acknowledge that the use of graph representations—especially in social networks or biological datasets—may carry ethical concerns around data privacy, representational bias, or unintended consequences of automated decision-making. While *TopER* itself is an unsupervised and domain-agnostic method, its application must be governed by domain-specific ethical considerations. To support responsible use, we emphasize interpretability and transparency in our design, and we release our code and visualizations to promote reproducibility and community oversight.