



TREPH: A Plug-In Topological Layer for Graph Neural Networks

Xue Ye ^{1,2,†} , Fang Sun ^{3,*,†} and Shiming Xiang ^{1,2} 

¹ National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

² School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China

³ School of Mathematical Sciences, Capital Normal University, Beijing 100048, China

* Correspondence: fsun@cnu.edu.cn

† These authors contributed equally to this work.

Abstract: Topological Data Analysis (TDA) is an approach to analyzing the shape of data using techniques from algebraic topology. The staple of TDA is Persistent Homology (PH). Recent years have seen a trend of combining PH and Graph Neural Networks (GNNs) in an end-to-end manner to capture topological features from graph data. Though effective, these methods are limited by the shortcomings of PH: incomplete topological information and irregular output format. Extended Persistent Homology (EPH), as a variant of PH, addresses these problems elegantly. In this paper, we propose a plug-in topological layer for GNNs, termed Topological Representation with Extended Persistent Homology (TREPH). Taking advantage of the uniformity of EPH, a novel aggregation mechanism is designed to collate topological features of different dimensions to the local positions determining their living processes. The proposed layer is provably differentiable and more expressive than PH-based representations, which in turn is strictly stronger than message-passing GNNs in expressive power. Experiments on real-world graph classification tasks demonstrate the competitiveness of TREPH compared with the state-of-the-art approaches.

Keywords: graph neural network; graph representation learning; topological data analysis; extended persistent homology



Citation: Ye, X.; Sun, F.; Xiang, S. TREPH: A Plug-In Topological Layer for Graph Neural Networks. *Entropy* **2023**, *25*, 331. <https://doi.org/10.3390/e25020331>

Academic Editor: Kinkar Chandra Das

Received: 11 January 2023

Revised: 4 February 2023

Accepted: 8 February 2023

Published: 10 February 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Graph-structured data are ubiquitous in various domains, including chemistry [1,2], physics [3,4], knowledge graphs [5,6], recommendation [7,8], and social science [9,10]. The analysis of graph data typically involves tasks such as graph classification, node classification, clustering, and link prediction, for which Graph Neural Networks (GNNs) [11] serve as a powerful tool. GNNs are deep learning-based models that take graph data as their input. Most GNNs are constructed by using an iterative message-passing framework [2,10,12–16], which updates a node representation by collecting information from its neighbors. Since messages are exchanged merely in the proximity of each node, these GNN structures are blind to certain global information of the graph. For example, they are unable to detect graph substructures such as cycles (loops) [17]. In fact, it can be shown that the expressive power of message-passing GNNs is bounded above by the one-dimensional Weisfeiler–Leman test [18]. The introduction of Topological Data Analysis (TDA) to the study of GNNs aims to enhance the effectiveness of GNNs by empowering them with global shape-awareness.

TDA is an emerging field that studies the shape of data through the lens of topology [19,20]. The most commonly used tool in TDA is Persistent Homology (PH). Intuitively, the procedure of applying PH can be visualized as scanning an object at multiple scales with respect to a predefined filtration process, which is essentially a process of “growing” the object. The homology functor, which is an algebraic topological construct, is applied to each scale and the inclusions in between, so as to extract the topological information

therein. The output of PH typically takes the form of Persistence Diagrams (PDs), which are multisets on the plane. The elements in these PDs encode persistence (i.e., moments of “birth” and “death”) of topological features (connected components, loops, and their analogies in higher dimensions) of the data studied. In the past two decades, TDA has found applications in a growing number of fields, including time series analysis [21], signal processing [22–24], image analysis [25,26], physics [27], biology [28,29], chemistry [30,31], material science [32–34], finance [35], and so on.

Recently, TDA has been introduced to the area of machine learning [36,37] to enhance the performance of both traditional machine learning methods and deep learning models. In particular, there has been an increased interest in combining TDA with neural networks for graph representation learning. For example, local topological features can be used as side information to rewire the graph and/or reweigh the messages passed between graph nodes during the convolution [38,39]. Although effectiveness is demonstrated well in these methods, the global topological features are still not fully utilized, which could be of great discriminative value. In [40,41], a filtration is built using the geometry of the graph, and the resulting PDs are fed into a neural network for both parameterized vectorization and task-specific prediction. Although the filtrations in these methods capture global topology, they are defined a priori and may not be the optimal choice for the specific task.

Another line of thought is constructing a learnable topological feature extraction layer either serving as a readout operation [42] or being a generic layer in GNNs to endow it with topology awareness [43]. In this manner, the learning signals can backpropagate through the calculation of TDA and adjust the filtrations as needed, enabling the models to be more flexible and adaptable to downstream graph-learning tasks. However, these methods are designed based on PH, which has certain shortcomings innately. One drawback lies in that the output PDs does not enjoy a uniform format, due to the presence of infinite coordinates. As a consequence, necessary compromises, such as tossing away the infinite parts or truncating the data artificially, must be made to enable subsequent use. Another drawback of PH is its inability to fully encode the topological features in graphs, the reason for which will be elaborated in Section 3.1.3.

The aforementioned issues of PH can be solved elegantly by Extended Persistent Homology (EPH) [44]. As a refined version of PH, EPH has an output always containing finite values and is able to extract strictly more topological information than PH. Recent years have seen efforts to integrate EPH with machine learning [38,41,45–47]. However, these studies do not incorporate EPH into a framework that is end-to-end trainable.

In this work, we propose a plug-in topological layer, called Topological Representation with Extended Persistent Homology (TREP). By virtue of EPH, the proposed layer not only retains all the merits held by the aforementioned PH-based layers, but also enjoys strictly stronger expressivity. Furthermore, our designed layer contains a novel aggregation mechanism that takes advantage of the uniformity of EPH for improved efficiency of feature learning. Our contributions in this work are summarized as follows:

- A plug-in topological layer named TREP is proposed for GNN, which utilizes EPH for effective extraction of topological features and can be conveniently inserted into any GNN architecture;
- TREP is proved to be differentiable and strictly more expressive than PH-based representations, which in turn is strictly stronger than message-passing GNNs in expressive power;
- By making use of the uniformity of EPH, a novel aggregation mechanism is designed to empower graph nodes with the ability to perform shape detection;
- Experiments on benchmark datasets for graph classification show the competitiveness of TREP, achieving state-of-the-art performances.

2. Related Work

2.1. Graph Neural Networks

The concept of Graph Neural Network [11] is initially outlined in [48] and further illustrated in [49,50]. Early works typically adopt a recurrent neural architecture to learn node representations, assuming that a node keeps exchanging messages with its neighbors until a fixed point is reached. More recently, it has become popular to stack multiple graph convolutional layers to learn high-level node representations [10,12–16]. Most of these models are developed on a message-passing framework [2], in which each node updates its hidden state by collecting messages from its neighbors. Though efficient, a recent study [18] shows that the expressive power of message-passing GNNs is bounded by the Weisfeiler–Leman test (i.e., WL[1]) [51]. In light of this, several works [52,53] design higher-order message-passing schemes to align GNNs with more discriminative WL[k] ($k \geq 2$) tests.

2.2. Topological Data Analysis

Topological data analysis is a field that investigates data by means of algebraic topology. The core concept of TDA is persistent homology, which can be seen as a multiscale representation of topological features. Since TDA is capable of extracting global, structural, and intrinsic properties of data at multiple scales, it is a desirable complement to geometry-based techniques in feature engineering. The synthesis of TDA and machine learning has become a rapidly developing research area [36,37,54]. Most works fall into three categories: vector-based, kernel-based, and neural network-integrated methods.

Vector-based methods focus on the appropriate vectorization of persistence diagrams. The resulting vectors can be used as the input of machine learning algorithms. Typical works involve statistical measurements [55], algebraic combinations [56], persistent entropy [57], Betti curves [58], persistence landscapes [59], and persistence images [60].

Kernel-based methods [61–66] aim at designing a kernel function that implicitly embeds PDs into a Reproducing Kernel Hilbert Space (RKHS). Since the kernel measures the inner product of two PDs embedded in the corresponding RKHS, it can be integrated into certain machine learning algorithms. Note that similarity measurements between PDs can also be modified into kernels [67].

More recently, it has been realized that neural networks could benefit from applying TDA to the relevant data. One of the most popular types of data for this application is graph. For instance, one could harness local topological information to rewire the graph and/or reweigh the messages passed between nodes [38,39]. To be precise, for each node, a filtration is defined on a neighborhood using structural information. The similarity between PDs serves as a measurement of the topological similarity between nodes. These methods, however, could fail to utilize the global topology of the graph, which may be critical for downstream tasks. Another approach [40,41,68] is building a filtration with global geometric information extracted from the graph and processing the resulting PDs using a neural network.

The above-mentioned methods use fixed filtrations. A natural direction of improvement is to consider learnable ones instead. A learnable filter function is first proposed in GFL [42] to develop the readout operation. Under their framework, the learning signal can backpropagate through the computation of persistent homology, making it possible to learn a proper filter function for improved performance. Following this line, TOGL [43] proposes a generic topological layer with multiple learnable filter functions. In particular, the PDs computed from these filter functions are transformed and aggregated to graph nodes, producing node-level representations. Such design enables TOGL to be inserted into any GNN architecture for any graph-learning tasks, not limited to graph-level inference. Further, TOGL also proves the superiority of PH over message-passing GNNs in terms of expressivity.

The performance of TDA approaches that use PH, such as those appearing in [39,40,42,43,68], is hindered by two inherent deficiencies (see Section 3.1.3). First, the topological information extracted by PH is incomplete. Second, the infinite

coordinates in the PDs turn out to be problematic for certain tasks. For example, in [43], the infinite coordinates were truncated by a chosen upper bound before vectorization. The choice of the upper bound is artificial, twisting the information contained in the PDs. Moreover, each topological feature in PDs is associated only with the vertex whose addition results in its birth, ignoring the death association. This is because some (but not all) features do not die, and thus cannot be handled uniformly with others.

The notion of extended persistent homology arises as a completion of PH [44]. EPH is strictly more expressive than PH (see Theorem 2). The output of EPH enjoys uniformity, being composed solely of points with finite coordinates. Although EPH has found a variety of applications in the study of machine learning [38,41,45–47], an end-to-end trainable framework utilizing EPH has yet to be considered.

In this work, we fill this gap by proposing an EPH-enhanced topological layer for GNNs, called TREP. The performance of TREP benefits from both the superior expressivity of EPH and an effective aggregation mechanism made possible by the uniformity of EPH.

3. Methodology

In this section, we first briefly review the relevant notions of EPH. Then, we describe our TREP layer in detail. Finally, the differentiability and expressivity of our model are discussed. For a thorough treatment of EPH, we refer interested readers to [19].

Table 1 lists key symbols used in our paper.

Table 1. A summary of key symbols used in this paper.

Symbol	Meaning
G	An undirected graph.
V	The vertex set of G .
E	The edge set of G .
N	The cardinality of V .
$X \in \mathbb{R}^{N \times d}$	The input node features of G .
\mathbb{Z}	The ring of integers.
Ker	The kernel of a homomorphism.
Im	The image of a homomorphism.
f	A function $V \rightarrow \mathbb{R}$.
$G_a = (V_a, E_a)$	The sublevel subgraph of G at a .
$\overline{\mathbb{R}}$	The reversed real line.
\bar{a}	An element $a \in \mathbb{R}$ regarded as an element of $\overline{\mathbb{R}}$.
$\overline{V}, \overline{E}$	Copies of V, E .
\bar{v}, \bar{e}	Elements of $\overline{V}, \overline{E}$.
$G^{\bar{a}} = (V^{\bar{a}}, E^{\bar{a}})$	The superlevel subgraph of G at \bar{a} .
$\text{Ord}_0, \text{Ext}_0, \text{Ext}_1, \text{Rel}_1$	The four Persistence Diagrams for EPH.
\mathcal{F}	The Filtration module.
\mathcal{V}	The Vectorization module.
\mathcal{A}	The Aggregation module.
eph	The process of computing EPH for all the filter functions.
d_f	The number of filter functions.
d_v	The dimension of vectorized representations of PD points.

3.1. Preliminaries

3.1.1. Homology

Homology theory is the study of a class of algebraic invariants called the homology groups, capable of extracting topological information (shape) of an object. We will briefly recall the relevant definitions and properties of homology in what follows, and we refer the readers to [69] for a comprehensive treatment. For our purpose, we consider relative

simplicial homology over $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$, where \mathbb{Z} stands for the ring of integers, “/” indicates taking quotient, and $2\mathbb{Z} = \{2n|n \in \mathbb{Z}\}$.

Let $G = (V, E)$ be a finite graph with vertex set V and edge set E . Denote by $C_0(G)$ (resp. $C_1(G)$) the \mathbb{Z}_2 -vector space with basis V (resp. E). Given a subgraph $G' = (V', E')$ of G , we call (G, G') a graph pair. Define $C_0(G'), C_1(G')$ analogously and set $C_i(G, G') = C_i(G)/C_i(G'), i = 0, 1$. For $n \in \mathbb{Z} \setminus \{0, 1\}$, let $C_n(G, G') = 0$. Define the boundary homomorphism $\partial_1 : C_1(G) \rightarrow C_0(G)$ by sending each edge $e = \{v, w\}$ to $v + w$. Passing to quotients, we obtain a boundary homomorphism $\partial_1 : C_1(G, G') \rightarrow C_0(G, G')$. For $n \neq 1$, we let $\partial_n : C_n(G, G') \rightarrow C_{n-1}(G, G')$ be zero. The n -th (relative) homology group of (G, G') is then defined as $H_n(G, G') = \text{Ker}\partial_n/\text{Im}\partial_{n+1}$. When $G' = \emptyset$, we write $H_n(G)$ for $H_n(G, G')$.

The homology groups encode topological features of the pair (G, G') : the basis of $H_0(G, G')$ is in bijection with the components of G not intersecting G' , while the basis of $H_1(G, G')$ corresponds to the (independent) loops in the quotient graph G/G' , formed by collapsing the subgraph $G' \subseteq G$ to a point.

Given graph pairs (G_1, G'_1) and (G_2, G'_2) such that $G_1 \subseteq G_2, G'_1 \subseteq G'_2$, the inclusion $(G_1, G'_1) \hookrightarrow (G_2, G'_2)$ induces a canonical homomorphism $H_n(G_1, G'_1) \rightarrow H_n(G_2, G'_2)$ for each n .

The homology groups encode topological features of G : the basis of $H_0(G)$ is in bijection with the components of G , while the basis of $H_1(G)$ corresponds to (independent) loops. For this reason, we refer to components and loops of a graph as its 0-dimensional and 1-dimensional topological features, respectively.

3.1.2. Persistent Homology

The discrete nature of homology groups hinders their application in data analysis. One way to introduce continuity is to consider a filter function f . Let $G = (V, E)$ be a graph with vertex set V and edge set E , and $f : V \rightarrow \mathbb{R}$ be a function on V . For each $a \in \mathbb{R} \cup \{\infty\}$, define the sublevel subgraph $G_a = (V_a, E_a)$, where $V_a = f^{-1}(-\infty, a]$ and E_a denotes the set of edges in E joining vertices in V_a . As a increases, this produces the sublevel filtration, which is a family of nested graphs indexed by \mathbb{R} (see Figure 1a for an illustration). Regarding $a \in \mathbb{R}$ as a temporal parameter and the filtration as a process of expanding sublevel subgraphs, the theory of persistent homology reads the shape of this process using the homology groups of G_a 's.

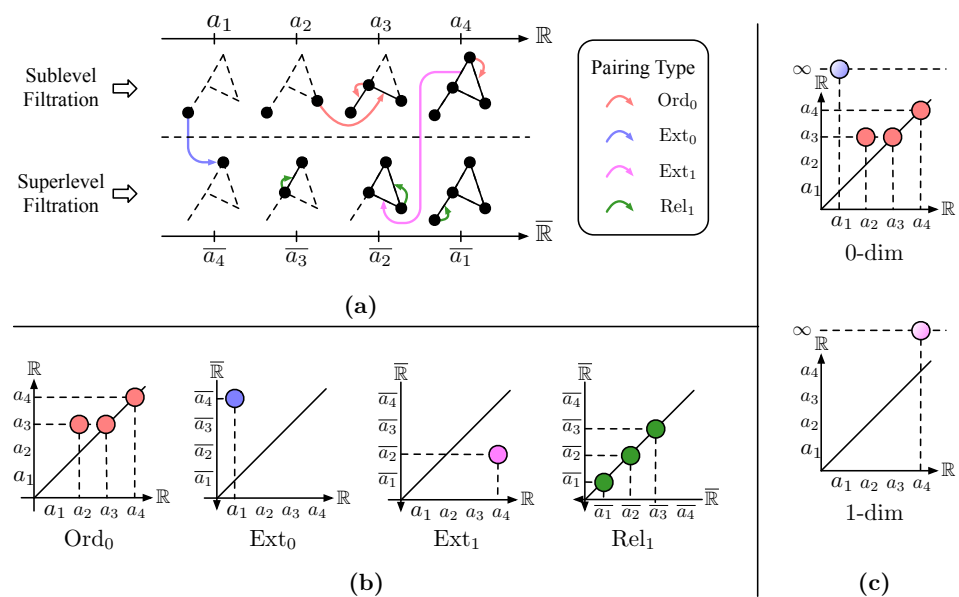


Figure 1. (a) Illustration of the sublevel and superlevel filtrations, as well as the pairings of extended persistent homology. (b,c) The persistence diagrams with respect to extended persistent homology and persistent homology, respectively.

The output of PH is specified by pairs of the form (v, e) , where $v \in V$ and $e \in E$. Each pair is interpreted as a component born when v is added during the filtration and dies (merges into an older component) with the addition of e . Such a feature (component) is called non-essential. Each unpaired vertex corresponds to a component born with the addition of this vertex and never dies. Similarly, an unpaired edge corresponds to a loop born when the edge is added and lives forever. We call a feature that never dies essential.

To utilize the output of PH in data analysis, one usually resorts to a representation called persistence diagrams (PD), which are multisets on the extended plane $\mathbb{R} \times (\mathbb{R} \cup \{\infty\})$. The 0-dimensional PD consists of two types of elements:

- For each pair (v, e) such that v (resp. e) is added at time a (resp. b), we place a point on the plane with coordinate (a, b) ;
- For each unpaired vertex v added at time a , we place a point (a, ∞) .

The 1-dimensional PD is constructed by placing for each e added at time a a point (a, ∞) . An example is given in Figure 1c.

3.1.3. Extended Persistent Homology

There are two drawbacks of PH. First, the output data do not enjoy a uniform format. The essential topological features are specified by one coordinate, while non-essential features are determined by two. One could mitigate this by either treating them separately [42] or truncating the second coordinate using an artificial finite upper bound [43,68]. However, there does not exist a natural way to handle both types within a uniform framework.

The second issue concerns completeness. The birth time of an essential feature corresponds to either the minimum of f in a component for H_0 or the maximum of f in a loop for H_1 . By symmetry, we should also consider the maximum in a component and the minimum in a loop. While the PDs of $-f$ contain this information, there is no indication of how the maxima and minima are matched. We shall see in the following how EPH remedies this problem.

The above drawbacks can be overcome with extended persistent homology. Denote by $\overline{\mathbb{R}}$ the set of real numbers with reversed order. For $b \in \mathbb{R}$, let \bar{b} be the corresponding element in $\overline{\mathbb{R}}$. Given a filter function f on $G = (V, E)$, for each $\bar{a} \in \overline{\mathbb{R}}$, we define the superlevel subgraph $G^{\bar{a}} = (V^{\bar{a}}, E^{\bar{a}})$, where $V^{\bar{a}} = f^{-1}[a, \infty)$ and $E^{\bar{a}}$ denotes the set of edges joining vertices in $V^{\bar{a}}$. These subgraphs form the superlevel filtration which expands in the direction of $\overline{\mathbb{R}}$. Essentially, EPH scans the graph bottom-up using the sublevel filtration and then top-down using the superlevel filtration (Figure 1a) and studies the topological features of this process.

The output of EPH is also a pairing [19], which we explain below. Denote \tilde{V}, \tilde{E} as copies of V, E . For $v \in V, e \in E$, let \tilde{v}, \tilde{e} be the corresponding element in \tilde{V} and \tilde{E} . The motivation behind this is to treat v and \tilde{v} as the same vertex scanned in the sublevel filtration and superlevel filtration, respectively. The same works for e and \tilde{e} . Elements of $V \cup E \cup \tilde{V} \cup \tilde{E}$ are called algebraic simplices, since they serve as the basis in the matrix reduction algorithm for computing EPH. Each algebraic simplex x is assigned a value $f(x) \in \mathbb{R} \cup \overline{\mathbb{R}}$, indicating its time of entrance into the corresponding filtration, in other words, sublevel filtration if $x \in V \cup E$ and superlevel filtration if $x \in \tilde{V} \cup \tilde{E}$.

The output of EPH is a bijection (pairing) between $V \cup \tilde{E}$ and $E \cup \tilde{V}$. A pair is of the form $(v, e), (v_1, \bar{v}_2), (e_1, \bar{e}_2)$ or (\bar{v}, \bar{e}) . Each pair (α, β) corresponds to a topological feature of a certain dimension persisting from $f(\alpha)$ to $f(\beta)$. For each of the four types of pairs, the collection of $(f(\alpha), f(\beta))$'s forms a multiset on the respective plane $(\mathbb{R} \times \mathbb{R}, \mathbb{R} \times \overline{\mathbb{R}}$ or $\overline{\mathbb{R}} \times \overline{\mathbb{R}})$. The four multisets are called persistence diagrams and are denoted by $\text{Ord}_0, \text{Ext}_0, \text{Ext}_1$ and Rel_1 , respectively (Figure 1b). We provide a topological interpretation of each:

- A pair (v, e) is associated with a component (0-dimensional). The points $(f(v), f(e)) \in \mathbb{R} \times \mathbb{R}$ form the diagram Ord_0 ;
- A pair (v_1, \bar{v}_2) is of dimension 0. Geometrically, this means v_1 and v_2 are where f obtains minimum and maximum in a component of G . The points $(f(v_1), f(\bar{v}_2)) \in \mathbb{R} \times \overline{\mathbb{R}}$ form the diagram Ext_0 ;

- A pair (e_1, \bar{e}_2) is of dimension 1. In this case, there is a loop whose maximum and minimum are obtained at e_1 and e_2 , respectively. The points $(f(e_1), f(\bar{e}_2)) \in \mathbb{R} \times \overline{\mathbb{R}}$ form the diagram Ext_1 ;
- A pair (\bar{v}, \bar{e}) is of dimension 1. It is not easy to illustrate the geometric meaning of such a pair directly. Fortunately, it can be interpreted as a 0-dimensional feature of $-f$ that persists from $-f(v)$ to $-f(e)$ by the symmetry of extended persistence (see [19] p.162). The points $(f(\bar{v}), f(\bar{e})) \in \overline{\mathbb{R}} \times \overline{\mathbb{R}}$ form the diagram Rel_1 .

The output of EPH is uniform: all algebraic simplices are paired and each point (feature) has two finite coordinates. EPH completes PH in the following sense. For a pair (G, f) , PH outputs two persistence diagrams of dimension 0 and 1, respectively. The 0-dimensional PD consists of Ord_0 and a point (a, ∞) for each (a, \bar{b}) in Ext_0 . The 1-dimensional PD consists of a point (a, ∞) for each (a, \bar{b}) in Ext_1 . By taking the PDs of $(G, \pm f)$ with respect to PH and using the symmetry of EPH, one could recover $\text{Ord}_0, \text{Rel}_1$ and individual numbers in the coordinates of $\text{Ext}_0, \text{Ext}_1$. Yet the pairing of those numbers is lost. Thus, EPH extracts strictly more information than PH. A concrete example demonstrating the distinguishing power of EPH over that of PH is presented in the proof of Theorem 2.

3.2. TrepH

The architecture of TREP H is shown in Figure 2. A graph $G = (V, E)$ with node features $X \in \mathbb{R}^{N \times d}$ is fed as its input. Through a Filtration module \mathcal{F} , d_f filter functions are learned. By applying EPH to each of the filter functions, we obtain the pairings and diagrams from different scanning perspectives. Then, a Vectorization module \mathcal{V} is responsible for transforming each point in the diagrams into a d_v -dimensional vector. These vectors, together with the pairings, are then fed into a novel Aggregation module \mathcal{A} to gather the vectorized topological features to their corresponding nodes on the graph. After the Aggregation, a residual connection and a fully connected (FC) layer are used to obtain the final output $X' \in \mathbb{R}^{N \times d'}$.

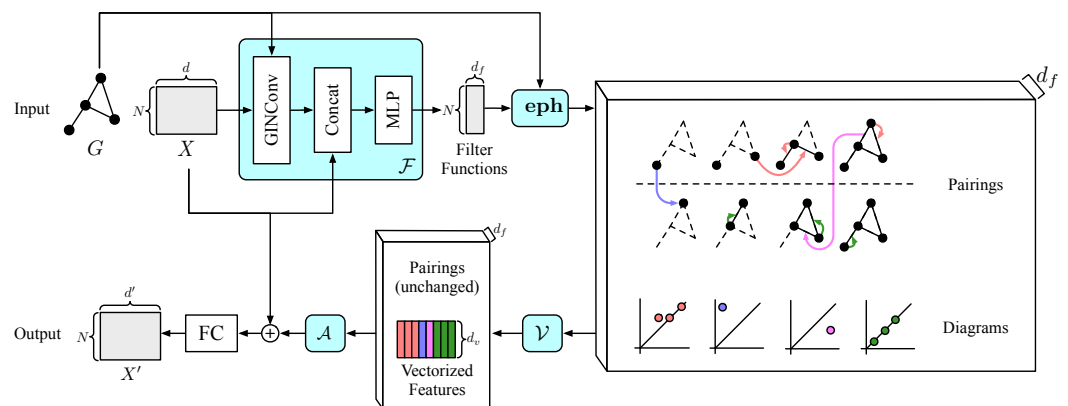


Figure 2. The architecture of TREP H.

The Filtration module \mathcal{F} is designed as a GIN- ϵ [18] graph convolutional layer followed by a two-layer MLP (multilayer perceptron). Skip connection by concatenation is used after the convolution. The sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ is adopted as the last activation, thus mapping values into range $[0, 1]$ for efficient learning in Vectorization.

For each filter function generated by \mathcal{F} , EPH can be calculated by using the matrix reduction algorithm described in [19]. Denote by **eph** the process of mapping all the filter functions to their diagrams and pairings. The output consists of $4d_f$ PDs $\{\mathcal{D}^{(k)} \mid 1 \leq k \leq d_f, \mathcal{D} \in \{\text{Ord}_0, \text{Rel}_1, \text{Ext}_0, \text{Ext}_1\}\}$, and their corresponding pairings $\{(a_p^b, \alpha_p^d) \mid p \in \cup_{\mathcal{D}, k} \mathcal{D}^{(k)}\}$.

To make better use of the points in the diagrams, a Vectorization module \mathcal{V} is designed to map each point $p \in \cup_{\mathcal{D}, k} \mathcal{D}^{(k)}$ to a high-dimensional vector representation $x_p \in \mathbb{R}^{d_v}$.

More concretely, for each type of diagram, a total of d_v coordinate functions are learned. In this work, we choose the rational hat structure element [40] as our coordinate function:

$$s(p) = \frac{1}{1 + \|p - c\|_1} - \frac{1}{1 + \|r\| - \|p - c\|_1}, \tag{1}$$

where $p \in \mathbb{R}^2$ is a point in the diagrams, and $c \in \mathbb{R}^2, r \in \mathbb{R}$ are the learnable parameters.

The main idea behind our Aggregation module \mathcal{A} is associating each point p in the diagrams to the geometric locations (nodes) marking its birth and death. The vectorized topological feature x_p is then aggregated to those locations correspondingly. Given the pairing (α_p^b, α_p^d) of p , the problem translates into defining a locating function $\mathbf{v} : V \cup \bar{V} \cup E \cup \bar{E} \rightarrow V$. To this end, the sublevel filtration can be regarded as a process of incremental addition of nodes, one at a time, and edges joining existing nodes and the new one. Thus, each element $x \in V \cup E$ can be associated with a node $\mathbf{v}(x)$ marking its location of entrance. Similarly, each $x \in \bar{V} \cup \bar{E}$ can be assigned an \bar{v} marking its entrance into the superlevel filtration, and we define $\mathbf{v}(x) = \bar{v}$ in this case. Hence, we construct \mathcal{A} by aggregating each x_p to $\mathbf{v}(\alpha_p^b)$ and $\mathbf{v}(\alpha_p^d)$, thus empowering graph nodes with the awareness of certain “shapes” in the graph. The details are shown in Algorithm 1, where we also employ the batch normalization [70] and the ReLU activation ($\text{ReLU}(x) = \max(x, 0)$) for efficient learning.

Algorithm 1: Aggregation.

Input: The pairings $\{(\alpha_p^b, \alpha_p^d)\}$ and the vector representations $\{x_p\}$ of all the PD points $p \in \cup_{1 \leq k \leq d_f, \mathcal{D} \in \{\text{Ord}_0, \text{Rel}_1, \text{Ext}_0, \text{Ext}_1\}} \mathcal{D}^{(k)}$, where d_f is the number of filter functions and $\text{Ord}_0, \text{Rel}_1, \text{Ext}_0, \text{Ext}_1$ are the EPH diagrams.

The locating function $\mathbf{v}(\cdot)$.

Output: The aggregated node representations $Z \in \mathbb{R}^{N \times d}$.

- 1 Initialization: for $v \in V, 1 \leq k \leq d_f$ and $S \in \{V, \bar{V}, E, \bar{E}\}$, initialize $Z_v^{k,S}$ as a d_v -dimensional vector filled with zeros;
 - 2 **for** p in $\cup_{\mathcal{D}, k} \mathcal{D}^{(k)}$ **do**
 - 3 Denote by $\mathcal{D}^{(k)}$ the diagram that contains p ;
 - 4 Denote by $S_b(S_d)$ the set that contains $\alpha_p^b(\alpha_p^d)$;
 - 5 $Z_{\mathbf{v}(\alpha_p^b)}^{k, S_b} += x_p; Z_{\mathbf{v}(\alpha_p^d)}^{k, S_d} += x_p$;
 - 6 **end**
 - 7 $\tilde{Z} = \text{stack}\{\text{concat}_{k,S}(Z_v^{k,S}) | v \in V\} \in \mathbb{R}^{N \times 4d_f d_v}$;
 - 8 $Z = \text{ReLU}(\text{BatchNorm}(\text{Linear}(\tilde{Z}))) \in \mathbb{R}^{N \times d}$;
 - 9 **return** Z
-

3.3. Differentiability and Expressive Power

The following theorem ensures the differentiability of the TREPH layer. Thus, TREPH is end-to-end-trainable with gradient-based optimization.

Theorem 1. *If the filter functions involved depend differentiably on a set of parameters Θ , then the output of the TREPH layer is differentiable with respect to Θ wherever the filter functions are injective (one-to-one).*

Proof. Suppose the filter functions $\mathcal{F}(G, X) = \mathcal{F}(G, X, \Theta)$ depend differentiably on a set of parameters Θ . For Θ_0 such that $\mathcal{F}(G, X, \Theta_0)$ consists of injective filter functions, we would like the composition $\mathcal{A} \circ \mathcal{V} \circ \mathbf{eph}(G, \mathcal{F}(G, X, \Theta))$ to be differentiable with respect to Θ at Θ_0 .

It is not hard to see that $\mathcal{F}(G, X, \cdot)$ and $\mathcal{A} \circ \mathcal{V}$ are differentiable (the latter with respect to the coordinates of points on the PDs). The desired result for PH in place of EPH is proved in [42] (Lemma 1).

The key idea for their proof is the following observation. Given Θ_0 such that $\mathcal{F}(G, X, \Theta_0)$ consists of injective functions, for Θ in a small enough neighborhood of Θ_0 , the order of vertices with respect to the filter functions is invariant, so are the pairing (between vertices and edges) and the vertices associated to each pair (i.e., marking the entrance of that vertex/edge). Since each pair determines a point on the PD whose coordinates are values of filter functions on the vertex associated with the pair, differentiability follows. The above argument remains valid for EPH, with pairing now between algebraic simplices. The rest of the proof for [42] Lemma 1 carries over. \square

It is shown that PH-enhanced GNNs have strictly stronger expressivity than standard GNNs in terms of the Weisfeiler–Leman test (WL[1]) ([43] Theorem 2). The same can be said for EPH since EPH extracts strictly more information than PH.

To compare the expressivity of PH and EPH, we consider the task of distinguishing pairs of the form (G, f) where f is a real-valued function defined on the node set of G . By allowing for arbitrary functions (instead of functions generated by the WL algorithm [51]), we take into account the actual input of PH/EPH layers in GNNs.

Theorem 2. *EPH is strictly more expressive than PH. That is, there exist pairs $(G_1, f_1), (G_2, f_2)$ such that both $(G_1, f_1), (G_2, f_2)$ and $(G_1, -f_1), (G_2, -f_2)$ have identical PDs with respect to PH. Yet the PDs of (G_1, f_1) and (G_2, f_2) with respect to EPH differ.*

Proof. Consider two pairs (G_1, f_1) and (G_2, f_2) illustrated in Figure 3a. The PDs of (G_1, f_1) and (G_2, f_2) with respect to PH coincide (Figure 3b). The PDs of $(G_1, -f_1)$ and $(G_2, -f_2)$ with respect to PH are also identical (Figure 3c). Yet (G_1, f_1) and (G_2, f_2) have distinct Ext_0 and Ext_1 , as illustrated in Figure 3d,e. Here is a brief explanation of what transpires. For both (G_1, f_1) and (G_2, f_2) , the minima of the two components are 1 and 2, while the maxima are 4 and 5. Thus, the infinite part of the 0-dimensional PDs cannot distinguish them. However, for (G_1, f_1) , the maximum 5 is matched with the minimum 1, and 4 is matched with 2. For (G_2, f_2) , the matchings are (5,2) and (4,1). This matching is captured by Ext_0 , resulting in different PDs. The case for Ext_1 can be explained analogously. \square

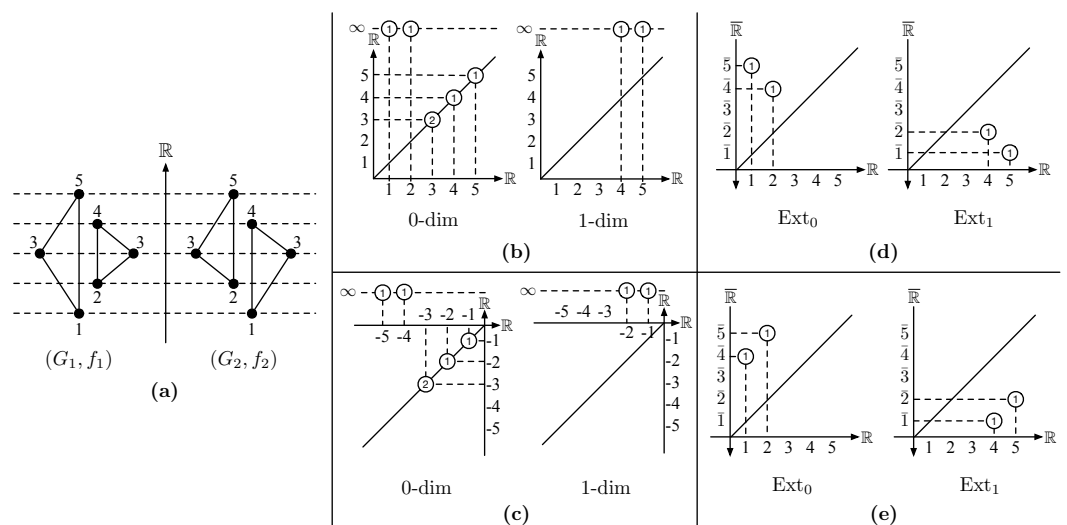


Figure 3. (a) Two graphs G_1 and G_2 are presented side by side, equipped with filter functions f_1 and f_2 , respectively. (b) PDs of both (G_1, f_1) and (G_2, f_2) with respect to PH. The numbers in circles represent multiplicity. (c) PDs of both $(G_1, -f_1)$ and $(G_2, -f_2)$ with respect to PH. (d) $\text{Ext}_0, \text{Ext}_1$ of (G_1, f_1) . (e) $\text{Ext}_0, \text{Ext}_1$ of (G_2, f_2) .

4. Experiments

4.1. Datasets

Our model is evaluated on six graph classification benchmark datasets: REDDIT-BINARY, IMDB-BINARY, IMDB-MULTI, COX2, DHFR, and NCI1 [71]. The first three datasets are derived from social networks, while the rest are molecules collected from medical or biological domains. A statistical summary of these datasets is shown in Table 2.

Table 2. A summary of the datasets used in this paper.

Datasets	REDDIT-B	IMDB-B	IMDB-M	COX2	DHFR	NCI1
Graphs	2000	1000	1500	467	467	4110
Classes	2	2	3	2	2	2
Avg. #Nodes	429.63	19.77	13.00	41.22	42.43	29.87
Avg. #Edges	497.75	96.53	65.94	43.45	44.54	32.30

Ten-fold cross-validation splitting is performed on each dataset. In each fold, 10% of the training set is reserved for validation. For fair comparisons with baselines, experiments in Sections 4.2.1 and 4.3 are conducted on ten folds of the datasets, and we report the mean and standard deviation of the testing accuracies across these folds. In Sections 4.2.2, 4.4 and 4.5, where we focus on factors that affect the performance of our layer, a fixed fold of each dataset is used for experiments under the consideration of limited resources.

4.2. Structure-Based Experiments

The experiments in this section are performed in a purely structure-based context, i.e., the input node features are set as uninformative (vectors of all ones) for all datasets. The main purpose of such design is to examine the practical capacity of TREPH in structural capturing compared with classical message-passing GNNs. To be specific, we first investigate the effects of inserting TREPH into GCNs [10], GATs [16], and GINs [18] and then study the influences of different positions where TREPH is placed.

4.2.1. TREPH in GNNs

Three groups of comparative experiments are conducted on six datasets. In each group, a four-layer GNN of a certain type (4-GCN, 4-GAT, or 4-GIN) is used as the baseline, and the other model is designed by substituting TREPH for the first layer of the baseline (3-GCN-1-TREPH, 3-GAT-1-TREPH, or 3-GIN-1-TREPH). Table 3 shows that the models equipped with TREPH generally perform better than the baselines of the same group (TREPH shows superiority in 14 out of 18 comparisons). Such a phenomenon indicates the relative advantages of TREPH over message-passing GNNs in terms of structural learning. Note that in GAT cases, where we experience severe gradient-vanishing problems on some of the datasets, by applying TREPH for the first layer, these training issues can be fixed to some extent. The number of parameters for each model on each dataset is listed in Table 4. Similar to the GNN baselines, the number of parameters of TREPH-equipped models is independent of the graph sizes. A little increase of the parameters in IMDB-MULTI comes from the classifier, which is caused by the increase of classes to predict.

Table 3. Graph classification results in the study of inserting TREPHE into GNNs under a purely structure-based setting. The best results within each group are emphasized in **bold** font.

Method	REDDIT-B	IMDB-B	IMDB-M	COX2	DHFR	NCI1
4-GCN	92.9 ± 1.9	65.1 ± 5.1	39.9 ± 3.5	77.1 ± 8.5	75.7 ± 4.3	74.7 ± 1.8
3-GCN-1-TREPHE	92.6 ± 1.4	71.6 ± 3.9	48.1 ± 2.6	79.2 ± 2.6	77.9 ± 2.6	76.7 ± 1.6
4-GAT	50.0 ± 0.0	50.0 ± 0.0	31.7 ± 4.4	72.4 ± 17.1	61.0 ± 0.5	50.0 ± 0.1
3-GAT-1-TREPHE	90.1 ± 3.8	70.9 ± 4.4	48.3 ± 2.8	75.8 ± 7.3	77.9 ± 4.3	76.5 ± 1.6
4-GIN	91.2 ± 2.4	71.1 ± 3.7	45.3 ± 4.2	82.7 ± 2.2	80.7 ± 2.6	77.2 ± 1.3
3-GIN-1-TREPHE	90.1 ± 1.1	72.0 ± 3.8	47.4 ± 3.4	80.1 ± 2.9	79.6 ± 4.1	77.6 ± 1.6

Table 4. Number of parameters for GNN baselines and TREPHE-equipped models on six datasets.

Method	REDDIT-B	IMDB-B	IMDB-M	COX2	DHFR	NCI1
4-GCN	19,298	19,298	19,331	19,298	19,298	19,298
3-GCN-1-TREPHE	102,891	102,891	102,924	102,891	102,891	102,891
4-GAT	19,810	19,810	19,843	19,810	19,810	19,810
3-GAT-1-TREPHE	103,275	103,275	103,308	103,275	103,275	103,275
4-GIN	36,454	36,454	36,487	36,454	36,454	36,454
3-GIN-1-TREPHE	115,758	115,758	115,791	115,758	115,758	115,758

4.2.2. Study of TREPHE Positions

TREPHE is a plug-in layer, which means that it can be inserted into any position of any GNN architecture. To investigate the influences of placing TREPHE at different positions, three groups of experiments are implemented. In each group, we consider substituting TREPHE for the first, second, third, or fourth layer in certain four-layer GNNs. The results are displayed in Table 5. It can be seen that the best position to place TREPHE differs from case to case. For GINs, the second layer seems to be a good choice for TREPHE in general. However, such placement is not appropriate for GATs due to the training issues mentioned in Section 4.2.1. In these cases, the first layer is a more reliable place for TREPHE to avoid degradation of the model. For GCNs, there seems to be no uniform standard on which position is preferable. The best position depends on the dataset being processed.

Table 5. Graph classification results in the study of placing TREPHE at different positions in GNNs. The best results within each group are emphasized in **bold** font.

Method	Pos	REDDIT-B	IMDB-B	IMDB-M	COX2	DHFR	NCI1
3-GCN-1-TREPHE	1	94.0	77.0	49.3	78.7	80.0	76.6
	2	94.5	79.0	48.0	83.0	82.7	78.4
	3	94.5	77.0	49.3	78.7	82.7	77.6
	4	96.0	78.0	49.3	63.8	86.7	78.4
3-GAT-1-TREPHE	1	91.5	76.0	46.0	78.7	81.3	76.2
	2	51.0	57.0	37.3	78.7	61.3	60.6
	3	50.0	50.0	33.3	78.7	38.7	50.1
	4	50.0	50.0	33.3	78.7	61.3	49.9
3-GIN-1-TREPHE	1	90.0	77.0	47.3	80.9	81.3	78.8
	2	94.5	79.0	50.0	89.4	88.0	79.8
	3	91.0	78.0	48.7	83.0	88.0	79.1
	4	94.0	78.0	50.0	85.1	77.3	80.8

4.3. Comparison with State-of-the-Art Methods

There are nine state-of-the-art baseline methods considered in our work for comparison. WL and WL-OA [72] are two kernel methods inspired by the Weisfeiler–Leman

test. PATCHY-SAN [73], DGCNN [74], and SAT [75] are three end-to-end-trainable neural networks that operate on graphs. SV [66], PersLay [41], GFL [42], and TOGL [43] are four TDA-based approaches. Note that all TDA-based approaches (including ours) did not utilize the node labels/attributes possibly provided by the datasets, while other methods did. To align with previous TDA-based methods, we employ a learnable embedding layer to generate input from node degrees for all datasets except for REDDIT-BINARY, whose input is set as uninformative.

Table 6 shows the graph classification results of each method on six benchmark datasets. It can be seen that our model performs comparably with state-of-the-art methods, and in particular, it outperforms all TDA-based approaches. On social networks, which do not have node labels/attributes, our model exhibits superiority over other methods. On molecular datasets, for which node labels/attributes are available, our model still outperforms others on COX2 and DHFR, despite the fact that we do not utilize that information during training and inference. The only exception is NCI1. We conjecture that the node labels of NCI1 might contain critical information for effective prediction, which is somewhat complementary to the structural information. Nevertheless, our method has a clear advantage over those TDA-based approaches, proving the power of TREPHE in extracting discriminative topological features.

Table 6. Graph classification comparison results with state-of-the-art methods. The best results overall are emphasized in **bold** font. The best results in the group of TDA-based approaches are underlined. The results of all baseline methods except SAT are quoted from their papers, where “n/a” means “not available”. The experiments of SAT are implemented by us using the code of its paper, where “OOM” means “Out Of Memory”.

Method	REDDIT-B	IMDB-B	IMDB-M	COX2	DHFR	NCI1
<i>Kernel Methods</i>						
WL [72]	78.0 ± 0.6	71.2 ± 0.5	50.3 ± 0.7	79.7 ± 1.3	81.7 ± 0.8	85.6 ± 0.4
WL-OA [72]	87.6 ± 0.3	74.0 ± 0.7	50.0 ± 0.5	81.1 ± 0.9	82.4 ± 1.0	86.0 ± 0.2
<i>Neural Networks</i>						
PATCHY-SAN [73]	86.3 ± 1.6	71.0 ± 2.3	45.2 ± 2.8	n/a	n/a	78.6 ± 1.9
DGCNN [74]	n/a	70.0 ± 0.9	47.8 ± 0.9	n/a	n/a	74.4 ± 0.5
SAT [75]	OOM	73.8 ± 4.6	49.9 ± 3.5	79.2 ± 5.0	80.2 ± 4.1	75.3 ± 1.6
<i>TDA-based Approaches</i>						
SV [66]	87.8 ± 0.3	74.2 ± 0.9	49.9 ± 0.3	78.4 ± 0.4	78.8 ± 0.7	71.3 ± 0.4
PersLay [41]	n/a	72.6	52.2	81.6	80.9	74.0
GFL [42]	90.2 ± 2.8	74.5 ± 4.6	49.7 ± 2.9	n/a	n/a	71.2 ± 2.1
TOGL [43]	90.1 ± 0.8	74.3 ± 3.6	52.0 ± 4.0	n/a	n/a	75.8 ± 1.8
TREPHE	91.5 ± 2.2	75.7 ± 3.6	52.3 ± 1.6	82.2 ± 3.0	82.9 ± 3.0	78.4 ± 1.5

4.4. Ablation Study

Two variant models are designed to study the effect of replacing EPH with PH. In fact, we go a step further and compare the EPH of (G, f) to the PH of (G, f) and $(G, -f)$ combined. As remarked near the end of Section 3.1.3, this replacement amounts to the following two operations:

- Each point (a, b) of Ext_0 is broken into a point (a, ∞) for the 0-dimensional PD of f and a point $(-b, \infty)$ for the 0-dimensional PD of $-f$;
- Each point (c, d) of Ext_1 is broken into a point (c, ∞) for the 1-dimensional PD of f and a point $(-d, \infty)$ for the 1-dimensional PD of $-f$.

Note that the information contained in Ord_0 and Rel_1 is preserved perfectly by the replacement.

The first variant model TREPHE-noext treats infinite points on the PDs separately from finite ones. To be precise, it regards a point (on one of the PDs) of the form (a, ∞) as $a \in \mathbb{R}$, vectorizes accordingly, and assigns the vector to the algebraic simplex where the topological

feature corresponding to the point is born. The finite portion of the 0-dimensional PDs of f and $-f$ are identical to Ord_0 and Rel_1 of f , respectively (modulo symmetry for Rel_1), and are treated in the same way as TREPH. Note that there are no finite points in the 1-dimensional PDs.

The second variant model TREPH-noext-trunc truncates infinite coordinates by replacing each point (a, ∞) on the PDs of f (respectively, $-f$) by $(a, \max f)$ (respectively, $(a, \max -f)$). Each resulting PD consists solely of finite planar points, which can be vectorized uniformly. Each vector obtained is aggregated to the algebraic simplex corresponding to the birth of that feature, ignoring the simplex of death. This is because truncated points have no simplex of death.

Experimental results comparing the performance of TREPH with that of its variants on six datasets are displayed in Figure 4. It can be seen that TREPH compares favorably with both of its variants on all six datasets. This demonstrates the superiority of the expressivity of EPH.

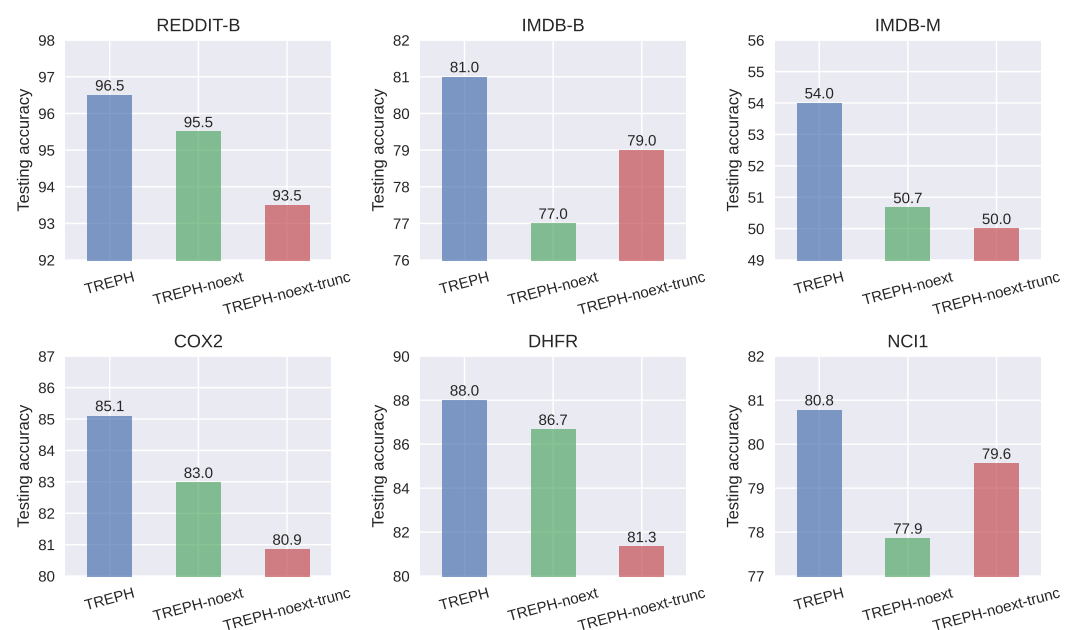


Figure 4. Ablation study of TREPH on six datasets.

4.5. Analysis of Hyperparameters

Two types of hyperparameters are analyzed. One is the number of filter functions d_f , the other is the dimension of vectorized representations d_v . Figure 5 shows the experimental results on six datasets under different choices of hyperparameters. On at least four datasets (REDDIT-BINARY, COX2, DHFR, and NCI1) out of six, there is an evident trend that larger d_f 's tend to provide better performances, indicating that increasing the number of topological features extracted could enhance the capability of our model. Similarly, experiments on the same four datasets show that increasing d_v results in better performances. The remaining two datasets (IMDB-B, IMDB-M) are significantly denser (having a larger edge-to-node ratio) than the others. We speculate that this results in their vulnerability to overfitting. The speculation is validated by the experiments on these datasets, as the increase of two independent hyperparameters leads to poorer performance.

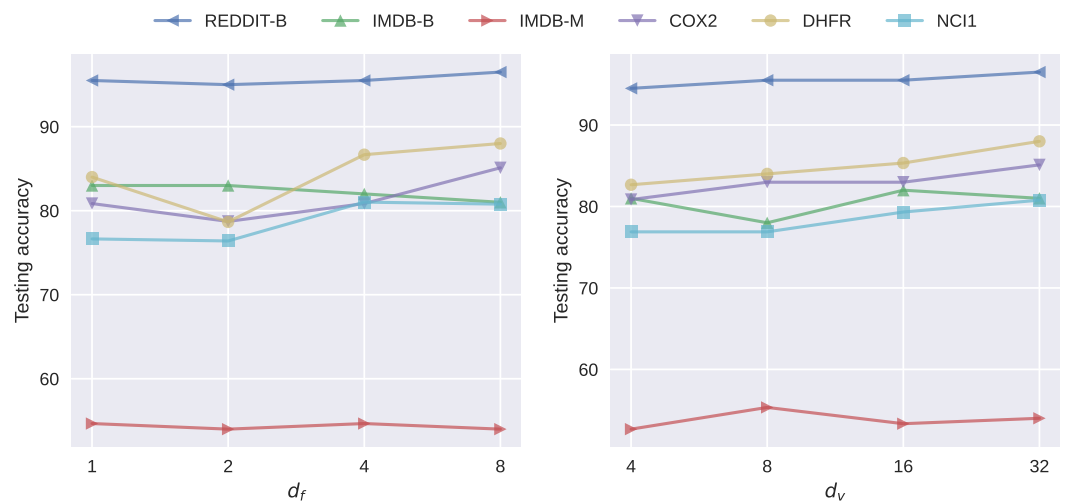


Figure 5. Analysis of the hyperparameters d_f and d_v in TREP.

4.6. Implementation Details

The input node dimension, the number of filter functions d_f , and the length of vector representation d_v are set as 64, 8, and 32, respectively. The node representations are average-pooled before being passed to a two-layer MLP with hidden dimension 32 for final prediction. Cross-entropy loss is used for training. The experiments are implemented in PyTorch [76] and optimized with the Adam algorithm [77]. The learning rate is set as 0.001 initially and halved if the validation accuracy does not improve after 10 epochs. We stop training when the learning rate is less than 10^{-5} or the number of epochs reaches 200.

4.7. Complexity Analysis

The original matrix reduction algorithm for computing EPH [19] runs in $O(n^3)$ with $n = |V \cup E|$ in the worst case. In our experiments, a CUDA (<https://developer.nvidia.com/cuda-toolkit>) programmed variant is designed for speedup. Timing experiments are conducted on the dataset REDDIT-BINARY (see Figure 6), which shows that our algorithm scales well in practice.

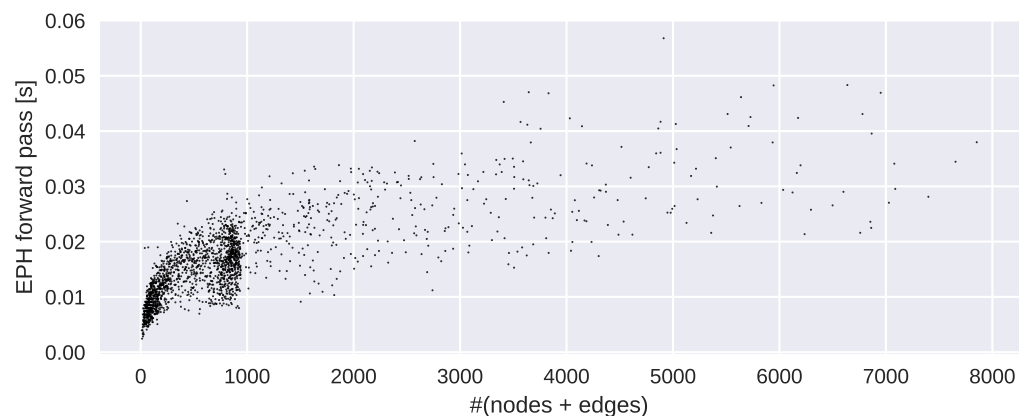


Figure 6. Timing experiments on REDDIT-BINARY.

We remark that the computational efficiency of the TREP layer could be further improved by optimizations of EPH computation. For example, one could resort to neural algorithm execution. In fact, a GNN framework for approximating the PDs of EPH is proposed in [78]. Their method, however, cannot be directly applied to our model, since the pairings of EPH cannot be obtained from their algorithm. A neural approximation that outputs the pairings of EPH is left as a subject for future research.

5. Conclusions

In this paper, we propose TREP, which is a plug-in topological layer for GNN. By virtue of the uniformity of EPH, a novel aggregation mechanism is designed to correlate topological features to their corresponding nodes on the graph, thus empowering graph nodes with the ability of shape detection. We have proved that the proposed layer is differentiable and strictly more expressive than PH-based representations, which in turn is stronger than message-passing GNNs in expressivity. Experiments on real-world graph classification tasks demonstrate the effectiveness and competitiveness of our model compared with the state-of-the-art methods.

For future research, we could consider speeding up the computation of our layer by developing a neural approximation algorithm for computing the pairings of EPH. In addition, we plan to explore the application of our proposed layer in graph learning tasks beyond graph classification, such as node classification and link prediction.

Author Contributions: Conceptualization, F.S.; methodology, X.Y. and F.S.; software, X.Y.; validation, X.Y.; formal analysis, X.Y.; visualization, X.Y.; resources, S.X.; writing—original draft preparation, X.Y. and F.S.; writing—review and editing, S.X.; supervision, S.X.; funding acquisition, S.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China under Grant No. 2018AAA0100400, and the National Natural Science Foundation of China under Grant 62076242.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in TUDatasets at <https://chrsmrrs.github.io/datasets/> (accessed on 10 January 2023), reference number [71].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GNN	Graph Neural Network
TDA	Topological Data Analysis
PH	Persistent Homology
EPH	Extended Persistent Homology
PD	Persistence Diagram
TREP	Topological Representation with Extended Persistent Homology

References

1. Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R.P. Convolutional Networks on Graphs for Learning Molecular Fingerprints. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2224–2232.
2. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1263–1272.
3. Cranmer, M.D.; Xu, R.; Battaglia, P.; Ho, S. Learning Symbolic Physics with Graph Networks. *arXiv* **2019**, arXiv:1909.05862.
4. Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; Battaglia, P. Learning to Simulate Complex Physics with Graph Networks. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; Volume 119, pp. 8459–8468.
5. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference; Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 10843, pp. 593–607.
6. Chami, I.; Wolf, A.; Juan, D.C.; Sala, F.; Ravi, S.; Ré, C. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. *arXiv* **2020**, arXiv:2005.00545.
7. Monti, F.; Bronstein, M.; Bresson, X. Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3697–3707.
8. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 974–983.

9. Hamilton, W.L.; Zhang, J.; Danescu-Niculescu-Mizil, C.; Jurafsky, D.; Leskovec, J. Loyalty in Online Communities. In Proceedings of the Eleventh International AAAI Conference on Web and Social Media, Montreal, QC, Canada, 15–18 May 2017; pp. 540–543.
10. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
11. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [[CrossRef](#)] [[PubMed](#)]
12. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3837–3845.
13. Atwood, J.; Towsley, D. Diffusion-convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 1993–2001.
14. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1024–1034.
15. Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; Bronstein, M.M. Geometric deep learning on graphs and manifolds using mixture model CNNs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5115–5124.
16. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
17. Bouritsas, G.; Frasca, F.; Zafeiriou, S.P.; Bronstein, M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 657–668. [[CrossRef](#)]
18. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
19. Edelsbrunner, H.; Harer, J. *Computational Topology: An Introduction*; American Mathematical Society: Providence, RI, USA, 2010.
20. Dey, T.K.; Wang, Y. *Computational Topology for Data Analysis*; Cambridge University Press: Cambridge, UK, 2022.
21. Yu, D.; Zhou, X.; Pan, Y.; Niu, Z.; Yuan, X.; Sun, H. University Academic Performance Development Prediction Based on TDA. *Entropy* **2023**, *25*, 24. [[CrossRef](#)]
22. Emrani, S.; Gentimis, T.; Krim, H. Persistent Homology of Delay Embeddings and its Application to Wheeze Detection. *IEEE Signal Process. Lett.* **2014**, *21*, 459–463. [[CrossRef](#)]
23. Perea, J.A.; Harer, J. Sliding windows and persistence: An application of topological methods to signal analysis. *Found. Comput. Math.* **2015**, *15*, 799–838. [[CrossRef](#)]
24. Erden, F.; Cetin, A.E. Period Estimation of an Almost Periodic Signal Using Persistent Homology With Application to Respiratory Rate Measurement. *IEEE Signal Process. Lett.* **2017**, *24*, 958–962. [[CrossRef](#)]
25. Qaiser, T.; Tsang, Y.W.; Taniyama, D.; Sakamoto, N.; Nakane, K.; Epstein, D.; Rajpoot, N. Fast and accurate tumor segmentation of histology images using persistent homology and deep convolutional features. *Med. Image Anal.* **2019**, *55*, 1–14. [[CrossRef](#)]
26. Rieck, B.; Yates, T.; Bock, C.; Borgwardt, K.; Wolf, G.; Turk-Browne, N.; Krishnaswamy, S. Uncovering the Topology of Time-Varying fMRI Data using Cubical Persistence. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6900–6912.
27. Mezher, R.; Arayro, J.; Hascoet, N.; Chinesta, F. Study of Concentrated Short Fiber Suspensions in Flows, Using Topological Data Analysis. *Entropy* **2021**, *23*, 1229. [[CrossRef](#)]
28. Yao, Y.; Sun, J.; Huang, X.; Bowman, G.R.; Singh, G.; Lesnick, M.; Guibas, L.J.; Pande, V.S.; Carlsson, G. Topological methods for exploring low-density states in biomolecular folding pathways. *J. Chem. Phys.* **2009**, *130*, 144115. [[CrossRef](#)]
29. Wang, B.; Wei, G.W. Object-oriented persistent homology. *J. Comput. Phys.* **2016**, *305*, 276–299. [[CrossRef](#)]
30. Lee, Y.; Barthel, S.D.; Dlotko, P.; Moosavi, S.M.; Hess, K.; Smit, B. Quantifying similarity of pore-geometry in nanoporous materials. *Nat. Commun.* **2017**, *8*, 15396. [[CrossRef](#)]
31. Smith, A.D.; Dlotko, P.; Zavala, V.M. Topological data analysis: Concepts, computation, and applications in chemical engineering. *Comput. Chem. Eng.* **2021**, *146*, 107202. [[CrossRef](#)]
32. Kovacev-Nikolic, V.; Bubenik, P.; Nikolić, D.; Heo, G. Using persistent homology and dynamical distances to analyze protein binding. *Stat. Appl. Genet. Mol. Biol.* **2016**, *15*, 19–38. [[CrossRef](#)] [[PubMed](#)]
33. Nakamura, T.; Hiraoka, Y.; Hirata, A.; Escolar, E.G.; Nishiura, Y. Persistent homology and many-body atomic structure for medium-range order in the glass. *Nanotechnology* **2015**, *26*, 304001. [[CrossRef](#)] [[PubMed](#)]
34. Buchet, M.; Hiraoka, Y.; Obayashi, I. Persistent homology and materials informatics. In *Nanoinformatics*; Springer: Singapore, 2018; pp. 75–95.
35. Yen, P.T.W.; Xia, K.; Cheong, S.A. Understanding Changes in the Topology and Geometry of Financial Market Correlations during a Market Crash. *Entropy* **2021**, *23*, 1211. [[CrossRef](#)] [[PubMed](#)]
36. Pun, C.S.; Xia, K.; Lee, S.X. Persistent-homology-based machine learning and its applications—A survey. *arXiv* **2018**, arXiv:1811.00252.
37. Hensel, F.; Moor, M.; Rieck, B. A survey of topological machine learning methods. *Front. Artif. Intell.* **2021**, *4*, 681108. [[CrossRef](#)]
38. Zhao, Q.; Ye, Z.; Chen, C.; Wang, Y. Persistence enhanced graph neural network. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020; Volume 108, pp. 2896–2906.
39. Chen, Y.; Coskunuzer, B.; Gel, Y.R. Topological relational learning on graphs. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 27029–27042.
40. Hofer, C.D.; Kwitt, R.; Niethammer, M. Learning representations of persistence barcodes. *J. Mach. Learn. Res.* **2019**, *20*, 1–45.

41. Carriere, M.; Chazal, F.; Ike, Y.; Lacombe, T.; Royer, M.; Umeda, Y. PersLay: A neural network layer for persistence diagrams and new graph topological signatures. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020; Volume 108, pp. 2786–2796.
42. Hofer, C.; Graf, F.; Rieck, B.; Niethammer, M.; Kwitt, R. Graph filtration learning. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; Volume 119, pp. 4314–4323.
43. Horn, M.; De Brouwer, E.; Moor, M.; Moreau, Y.; Rieck, B.; Borgwardt, K. Topological graph neural networks. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
44. Cohen-Steiner, D.; Edelsbrunner, H.; Harer, J. Extending Persistence Using Poincaré and Lefschetz Duality. *Found. Comput. Math.* **2009**, *9*, 79–103. [[CrossRef](#)]
45. Yan, Z.; Ma, T.; Gao, L.; Tang, Z.; Chen, C. Link prediction with persistent homology: An interactive view. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021; Volume 139, pp. 11659–11669.
46. Zhao, Q.; Wang, Y. Learning metrics for persistence-based summaries and applications for graph classification. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32, pp. 9855–9866.
47. Royer, M.; Chazal, F.; Levrard, C.; Umeda, Y.; Ike, Y. ATOL: Measure Vectorization for Automatic Topologically-Oriented Learning. In Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, San Diego, CA, USA, 13–15 April 2021; Volume 130, pp. 1000–1008.
48. Gori, M.; Monfardini, G.; Scarselli, F. A new model for learning in graph domains. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 2, pp. 729–734.
49. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80. [[CrossRef](#)]
50. Gallicchio, C.; Micheli, A. Graph echo state networks. In Proceedings of the 2010 International Joint Conference on Neural Networks, Barcelona, Spain, 18–23 July 2010; pp. 1–8.
51. Weisfeiler, B.; Leman, A.A. The reduction of a graph to canonical form and the algebra which appears therein. *NTI Ser.* **1968**, *2*, 12–16.
52. Maron, H.; Ben-Hamu, H.; Serviansky, H.; Lipman, Y. Provably powerful graph networks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 2153–2164.
53. Chen, Z.; Villar, S.; Chen, L.; Bruna, J. On the equivalence between graph isomorphism testing and function approximation with GNNs. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 15868–15876.
54. Chazal, F.; Michel, B. An Introduction to Topological Data Analysis: Fundamental and Practical Aspects for Data Scientists. *Front. Artif. Intell.* **2021**, *4*, 667963. [[CrossRef](#)] [[PubMed](#)]
55. Cang, Z.; Mu, L.; Wu, K.; Opron, K.; Xia, K.; Wei, G.W. A topological approach for protein classification. *Comput. Math. Biophys.* **2015**, *3*, 140–162. [[CrossRef](#)]
56. Adcock, A.; Carlsson, E.; Carlsson, G. The ring of algebraic functions on persistence bar codes. *Homol. Homotopy Appl.* **2016**, *18*, 381–402. [[CrossRef](#)]
57. Atienza, N.; Gonzalez-Diaz, R.; Rucco, M. Persistent entropy for separating topological features from noise in Vietoris-rips complexes. *J. Intell. Inf. Syst.* **2019**, *52*, 637–655. [[CrossRef](#)]
58. Rieck, B.; Sadlo, F.; Leitte, H. Topological Machine Learning with Persistence Indicator Functions. In *Topological Methods in Data Analysis and Visualization V*; Mathematics and Visualization; Springer: Cham, Switzerland, 2020; pp. 87–101.
59. Bubenik, P. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.* **2015**, *16*, 77–102.
60. Adams, H.; Emerson, T.; Kirby, M.; Neville, R.; Peterson, C.; Shipman, P.; Chepushtanova, S.; Hanson, E.; Motta, F.; Ziegelmeier, L. Persistence images: A stable vector representation of persistent homology. *J. Mach. Learn. Res.* **2017**, *18*, 1–35.
61. Mileyko, Y.; Mukherjee, S.; Harer, J. Probability measures on the space of persistence diagrams. *Inverse Probl.* **2011**, *27*, 124007. [[CrossRef](#)]
62. Reininghaus, J.; Huber, S.; Bauer, U.; Kwitt, R. A Stable Multi-Scale Kernel for Topological Machine Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4741–4748.
63. Kusano, G.; Hiraoka, Y.; Fukumizu, K. Persistence weighted Gaussian kernel for topological data analysis. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 2004–2013.
64. Chazal, F.; Fasy, B.; Lecci, F.; Michel, B.; Rinaldo, A.; Rinaldo, A.; Wasserman, L. Robust topological inference: Distance to a measure and kernel distance. *J. Mach. Learn. Res.* **2017**, *18*, 5845–5884.
65. Le, T.; Yamada, M. Persistence Fisher Kernel: A Riemannian Manifold Kernel for Persistence Diagrams. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 10028–10039.
66. Tran, Q.H.; Vo, V.T.; Hasegawa, Y. Scale-variant topological information for characterizing the structure of complex networks. *Phys. Rev. E* **2019**, *100*, 032308. [[CrossRef](#)] [[PubMed](#)]
67. Chen, Y.; Garcia, E.K.; Gupta, M.R.; Rahimi, A.; Cazzanti, L. Similarity-based classification: Concepts and algorithms. *J. Mach. Learn. Res.* **2009**, *10*, 747–776.
68. Hofer, C.; Kwitt, R.; Niethammer, M.; Uhl, A. Deep learning with topological signatures. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1634–1644.
69. Hatcher, A. *Algebraic Topology*; Cambridge University Press: Cambridge, UK, 2002.

70. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 448–456.
71. Morris, C.; Kriege, N.M.; Bause, F.; Kersting, K.; Mutzel, P.; Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv* **2020**, arXiv:2007.08663.
72. Borgwardt, K.; Ghisu, E.; Llinares-López, F.; O’Bray, L.; Rieck, B. *Graph Kernels: State-of-the-Art and Future Challenges*; Now Foundations and Trends: Boston, MA, USA, 2020; Volume 13, pp. 531–712.
73. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning convolutional neural networks for graphs. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 2014–2023.
74. Zhang, M.; Cui, Z.; Neumann, M.; Chen, Y. An End-to-End Deep Learning Architecture for Graph Classification. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32, pp. 4438–4445.
75. Chen, D.; O’Bray, L.; Borgwardt, K. Structure-Aware Transformer for Graph Representation Learning. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MA, USA, 17–23 July 2022; Volume 162, pp. 3469–3489.
76. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035.
77. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
78. Yan, Z.; Ma, T.; Gao, L.; Tang, Z.; Wang, Y.; Chen, C. Neural approximation of graph topological features. In Proceedings of the Thirty-Sixth Conference on Neural Information Processing Systems, New Orleans, LA, USA, 28 November–8 December 2022.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.