Journal of Machine Learning Research 25 (2024) 1-36 Submitted 12/23; Revised 1/24; Published 12/24

Line Graph Vietoris-Rips Persistence Diagram for Topological Graph Representation Learning

Jaesun Shin J1991.SHIN@SAMSUNG.COM
Samsung SDS

Eunjoo Jeon EJ85.JEON@SAMSUNG.COM
Samsung SDS

Taewon Cho TAEWON08.CHO@SAMSUNG.COM
Samsung SDS

Namkyeong Cho[1] NAMKYEONG.CHO@GMAIL.COM
Center for Mathematical Machine Learning and its Applications(CM2LA), Department of Mathematics POSTECH

Youngjune Gwon GYJ.GWON@SAMSUNG.COM
Samsung SDS

Editor: Sayan Mukherjee

Abstract

While message passing graph neural networks result in informative node embeddings, they may suffer from describing the topological properties of graphs. To this end, node filtration has been widely used as an attempt to obtain the topological information of a graph using persistence diagrams. However, these attempts have faced the problem of losing node embedding information, which in turn prevents them from providing a more expressive graph representation. To tackle this issue, we shift our focus to edge filtration and introduce a novel edge filtration-based persistence diagram, named Topological Edge Diagram (TED), which is mathematically proven to preserve node embedding information as well as contain additional topological information. To implement TED, we propose a neural network based algorithm, named Line Graph Vietoris-Rips (LGVR) Persistence Diagram, that extracts edge information by transforming a graph into its line graph. Through LGVR, we propose two model frameworks that can be applied to any message passing GNNs, and prove that they are strictly more powerful than Weisfeiler-Lehman type colorings. Finally we empirically validate superior performance of our models on several graph classification and regression benchmarks.

Keywords: Graph Neural Network, Persistence Diagram, Topological Data Analysis, Weisfeiler-Lehman Test, Vietoris-Rips Filtration

1. Introduction

Recently, message passing graph neural networks and its variants have emerged as an effective method to learn graph representations (Morris et al. (2019); Chen et al. (2019); Balcilar et al. (2020); Kondor et al. (2018); Cai et al. (2021); Hamilton et al. (2017); Gilmer et al. (2017); Wu et al. (2020)). Since message passing GNNs are designed to learn node represen-

1. Work done while working at Samsung SDS

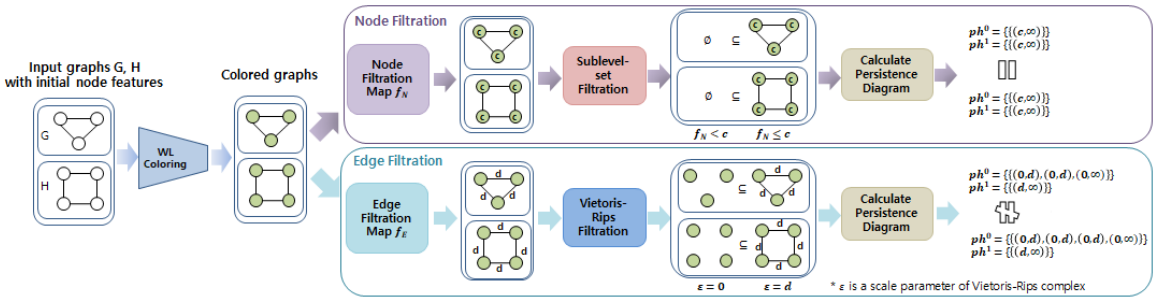

Figure 1: An overview of node filtration and edge filtration of WL coloring. For graphs G and H distinguishable by WL test, the edge filtration-based persistence diagram can also distinguish them while the node filtration-based one cannot.

tations, they can extract informative node embeddings by capturing localized information. However, they can hardly capture topological information of the entire graph (Chen et al. (2020); Hofer et al. (2020); Balcilar et al. (2021)). In this vein, various topological methods have been proposed (Zhang et al. (2018); Ying et al. (2018); Ranjan et al. (2020); Bouritsas et al. (2020)). In particular, node filtration has been widely used to extract topological information of graphs using persistence diagrams (Oudot (2017)), showing superior performance on graph benchmarks (Hofer et al. (2020); Horn et al. (2021); Hofer et al. (2017); Carriere et al. (2019); Rieck et al. (2019); Zhao and Wang (2019)). For example, Hofer et al. (2020) extracted topological information from the persistence diagram of sublevel sets of a node filtration map over node features. Moreover, Horn et al. (2021) proposed a multi-scale version of Hofer et al. (2020) using k node filtration maps. Theoretically, they claim that persistence diagrams based on node filtration map f_N can provide stronger expressivity than the Weisfeiler-Lehman (WL) test (Weisfeiler and Lehman (1968)), assuming that the input node features of f_N are all different for each node (Horn et al., 2021, Theorem 1, 2). However, even the WL test, which is known to be at least as powerful as message passing GNNs (Xu et al. (2018a)), cannot assign different features to different nodes. Furthermore, in cases where this assumption does not hold, there exist graphs that cannot be distinguished by node filtration-based persistence diagrams but can be distinguished by the WL test (Figure 1). In other words, node filtration-based persistence diagram cannot provide more powerful GNNs than the WL test in general.

We found out that such topological methods suffer from a loss of coloring information, due to the nature of node filtration, which extracts induced subgraphs (Figure 2). To address this, we shift our attention to edges. If we can capture information of two nodes in a single edge, we can extract more fruitful information from edge filtration by directly controlling the connectivity of graphs through edges while including all the node information. With this goal in mind, we propose a novel edge filtration-based persistence diagram, *Topologcial Edge Diagram (TED)*, which, to the best of our knowledge, is the first approach using edge-based filtration in topological graph representation learning. Specifically, TED is defined as a persistence diagram of Vietoris-Rips filtration (Hatcher (2002); Oudot (2017)), which is the well-known algebraic topological reconstruction technique, whose a set of graph nodes is represented as a point cloud and edge information as distances between points. In contrast

to node filtration, we prove that TED can preserve the expressive power of an arbitrary node coloring (Lemma 8). We further prove that TED can even increase the expressive powers of WL colorings thanks to its additional topological information (Theorem 9).

Next, we propose a novel neural-network-based algorithm, called *Line Graph Vietoris-Rips (LGVR) Persistence Diagram*, to implement our theoretical foundation. A key challenge is to assign unique features to edges that consist of different node features. To tackle this problem, we construct a map t_ϕ (Section 5.1) that transforms a colored graph into a colored line graph (Definition 11) through a neural network, which ensures the uniqueness of edge features. Thanks to this, we prove that LGVR has the same expressivity as TED and further analyze its theoretical expressivity. Through LGVR, we propose two types of topological model frameworks, $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$, that can be applied to any message passing GNNs $\mathcal{C}$ (Section 6.2). From a theoretical perspective, we analyze their theoretical expressivity when $\mathcal{C}$ is either GIN (Xu et al. (2018a)) or PPGN (Maron et al. (2019)), and prove that our models are strictly more powerful than $\mathcal{C}$ (Corollary 18).

In addition to proposing a theoretical framework, we performed experiments on several real-world datasets, including 7 classification and 12 regression tasks related to bioinformatics, social networks, and chemical compounds, to substantiate the superiority of our topological models (Section 7). We focus on three aspects. First, we test whether our models, $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$, which theoretically have more powerful representational power than the message passing GNN $\mathcal{C}$, also show better performances empirically. Our findings demonstrate that our topological models outperform the $\mathcal{C}$ by effectively comprehending diverse graph structures (Table 1, 3 and 4). Next, we conducted comparative experiments for our edge filtration-based methodology and the existing node filtration-based approach (Xu et al. (2018a)) to experimentally evaluate the representational capabilities between topological methodologies. In this vein, we compare their performances on both classification and regression tasks, and found that our approach shows superior performance compared to node filtration-based approach, which validates the superiority of edge filtration-based approach over node filtration-based one (Table 1, Figure 6 and Table 4). Finally, we observed that the performances of GNNs vary significantly based on data splits, even for the same dataset (Table 2). We speculate that this variation is due to the insufficient utilization of graph information in the training data, resulting from the limited representational powers of existing GNNs. Therefore, we measured the standard deviations of performances based on data split for the message passing GNN $\mathcal{C}$, node filtration-based methodology, and our models. As a result, our models exhibited the lowest standard deviation compared to other methods (Figure 7). From these results, we confirmed that our topological models, which can well reflect various features in graph representations due to strong expressive powers, enable more stable learning compared to the $\mathcal{C}$ and node filtration-based approach.

The main contributions of this paper can be summarized as follows:

1. We introduce a novel edge filtration-based persistence diagram, called *Topological Edge Diagram (TED)*. TED is mathematically proven to preserve node embedding information as well as provide additional topological information. As far as we know, our approach is the first to leverage edge-based filtration in topological graph representation learning.

2. We propose a novel neural-network-based algorithm, called *Line Graph Vietoris-Rips (LGVR) Persistence Diagram*, to implement our theoretical foundation. We prove that LGVR has the same expressivity as TED and further analyze its theoretical expressivity.

3. By applying the LGVR, we propose two model frameworks: $\mathcal{C}$-LGVR and $\mathcal{C}$-LGVR$^+$ that can be applied to any message passing GNN $\mathcal{C}$, and theoretically demonstrate the strong expressive powers of our models.

4. We perform experiments on several real-world datasets, including classification and regression tasks related to bioinformatics, social networks, and chemical compounds. Through these experiments, we demonstrate that our edge filtration-based models not only have strong experimental representational capabilities but also enable stable learning regardless of data split by encapsulating various graph properties in graph representations.

This paper is organized as follows. Section 2 provides a brief explanation of the prerequisite knowledge, including the Weisfeiler-Lehman test, GNN, and basics of persistence homology. We summarize some notations and conventions used throughout this paper in Section 3. In Section 4, we introduce our novel edge filtration-based persistence diagram, which we call *Topological Edge Diagram (TED)*, and analyze its theoretical expressive power. In Section 5, we propose a novel neural-network-based algorithm, named *Line Graph Vietoris-Rips (LGVR) Persistence Diagram*, to implement TED, and analyze its theoretical expressive power. In Section 6, we propose two model frameworks applying LGVR. Specifically, we first propose a simple mathematical technique that integrates the expressive powers of both coloring information and topological information induced by LGVR. Depending on the application of this integration technique, we propose two topological model frameworks, $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$, and analyze their theoretical expressive powers. In Section 7, we conduct experiments by focusing on experimental verification of our models. Section 8 concludes the paper with some future work directions.

2. Preliminaries

In this section, we briefly recall Weisfeiler-Lehman test, graph neural network, and some basic knowledge in algebraic topology related to persistence homology.

2.1 (Higher) Weisfeiler-Lehman Test

The *Weisfeiler-Lehman* test (WL test) is an algorithm which determines the graph isomorphism problem according to the histogram of colors on the vertices where the colors are iteratively aggregated by those of the neighborhood vertices. Precisely, for a finite graph $G = (V, E)$ with the initial coloring $X_0 : V \to \mathbb{Z}, v \mapsto 1$, the n-th coloring $X_n \in \mathrm{Hom}(V, \mathbb{Z})$ is given by $X_n(v) := \sum_{u \in \mathcal{N}(v)} X_{n-1}(u)$, where $\mathcal{N}(v)$ is a neighbor of v in G. Then two graphs G and G' are isomorphic only if their associated n-th colorings X_n and X'_n coincide for all $n \geq 1$. To distinguish two graphs of the same order $m = |V|$, it suffices to terminate the algorithm in the n-th iteration for some $n = O(m^k)$ Douglas (2011), and it is known that the algorithm is effective for a broad class of graphs Babai and Kucera (1979).

Higher-order variant of WL test, named k-WL, has been proposed to improve expressive power and apply color refinements iteratively on vertex tuples instead of single vertices. For a given $G = (V, E)$, the initial coloring is defined using the isomorphism type of each k-tuple of V. Specifically, two k-tuples $(v_1, \ldots, v_k)$ and $(w_1, \ldots, w_k)$ in V^k are assigned the same initial color if and only if for all $i, j \in \{1, \ldots, k\}$, (1) $v_i = v_j$ if and only if $w_i = w_j$, and (2) v_i is adjacent to v_j if and only if w_i is adjacent to w_j.

Given this initial coloring, it refines the colorings of k-tuples iteratively (similar to the WL test) until the histogram of coloring does not change further. In k-WL, the neighborhood of $\nu = (v_1, \ldots, v_k) \in V^k$ is set to $N_j(\nu) = \{(v_1, \ldots, v_{j-1}, u, v_{j+1}, \ldots, v_k) \,|\, u \in V\}$, where $j \in \{1, \ldots, k\}$ and $u \in V$. Then the coloring update rules are:

$$X_t(\nu) = HASH(X_{t-1}(\nu), \mathcal{N}(\nu, t-1)),$$

where $\mathcal{N}(\nu, t-1) = \{\{X_{t-1}(\nu') \,|\, \nu' \in N_j(\nu)\}\} \,|\, j \in \{1, \ldots, k\}$. We refer to (Cai et al. (1992); Grohe and Otto (2015); Morris et al. (2019)) for several results related to WL and k-WL.

2.2 Graph Neural Network

Graph neural network (GNN) computes the structure of a graph and its node features to learn a representation vector h_v of a vertex v. Modern GNNs use spatial methods based on a message passing scheme (Kipf and Welling (2016)). In short, the learning process of GNNs iteratively updates the node features from those of the neighboring nodes, which formally associates (1) the representation vectors $h_v^{(k)} \in \mathbb{R}^d$, and (2) the aggregation procedure $h_v^{(k)} = \varphi^k(h_v^{(k-1)}, f^k(\{\{h_u^{(k-1)} : u \in \{w \in V \mid (v, w) \in E\}\}\}))$, given by an *aggregation function* f^k that operates on multisets and a *combine function* φ^k. To extract the graph-level representation h_G, various pooling methods, also called readout operations, have been proposed to summarize the representation vectors $h_v^{(k)}$ of nodes $v \in V$ (Zhang et al. (2018); Ying et al. (2018); Ranjan et al. (2020); Hofer et al. (2020)). In terms of the expressive power of h_G, it is proven in Xu et al. (2018a) that GNNs are as powerful as the WL test under the assumption on the injectivity of f^k and φ^k for each k.

2.3 Simplicial Complexes, Persistence Homology, and Vietoris-Rips Filtration

In this section, we will introduce some basic knowledge in algebraic topology. Readers who are already familiar with algebraic topology may skip this section without hesitation.

2.3.1 Simplicial Complexes

In this subsection, we recall some basics of simplicial complexes. Briefly speaking, a simplex is the simplest geometric object, such as points, line segments, triangles, and their higher-dimensional analogs. Moreover, a simplicial complex is a set of simplices satisfying certain rules. Both are central topological concepts in algebraic topology in order to understand the shape and structure of complex spaces. Formal definitions of both objects are as follows:

Definition 1 *A* k-simplex *is a k-dimensional polytope which is the convex hull of affinely independent $k+1$ vertices. Moreover, a* simplicial complex $\mathcal{K}$ *is a set of simplices satisfying the following: (1) every face of a simplex in $\mathcal{K}$ is also in $\mathcal{K}$, and (2) for any $\sigma_1, \sigma_2 \in \mathcal{K}$ such*

that $\sigma_1 \cap \sigma_2 \neq \emptyset$, $\sigma_1 \cap \sigma_2$ is a face of both σ_1 and σ_2. Finally, the d-skeleton *of a simplicial complex $\mathcal{K}$ is the simplicial complex consisting of the set of all simplices in $\mathcal{K}$ of dimension d or less.*

Finally, a morphism, which is a map preserving structures of mathematical objects, between simplicial complexes can be defined as follows. It is easy to see that simplicial maps can be seen as an extension of graph maps from the perspective of simplicial complexes.

Definition 2 *Let $\mathcal{K}$ and $\mathcal{L}$ be two simplicial complexes. A* simplicial map $f : \mathcal{K} \to \mathcal{L}$ *is a function from 0-simplices of $\mathcal{K}$ to 0-simplices of $\mathcal{L}$ that maps every simplex in $\mathcal{K}$ to a simplex in $\mathcal{L}$. Moreover, a simplicial map $f : \mathcal{K} \to \mathcal{L}$ is called a* simplicial isomorphism *if it is bijective and its inverse is also a simplicial map. If there exists a simplicial isomorphism between $\mathcal{K}$ and $\mathcal{L}$, we call $\mathcal{K}$ and $\mathcal{L}$ are isomorphic and denote it by $\mathcal{K} \cong \mathcal{L}$*

2.3.2 Homology and Betti Numbers

Homology is an abstract way of associating topological or algebraic spaces with a sequence of algebraic objects. In algebraic topology, this allows to encode the topological information of a space through a chain of vector spaces and linear maps. We refer to Hatcher (2002) for interested readers. In general, homology can be defined over an arbitrary field, but for simplicity, we restricted our attention to $\mathbb{Z}_2$. Furthermore, we will only deal with the homology classes whose algebraic objects are vector spaces.

Let $C_0, C_1, \dots$ be vector spaces over $\mathbb{Z}_2$, and let $\partial_n : C_n \to C_{n-1}$ be linear maps satisfying $\partial_{n+1} \circ \partial_n = 0$ for all $n \geq 0$, which we call *boundary operators*. A *chain complex* refers to the sequence

$$C_\bullet : \cdots \to C_{n+1} \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \to \cdots \xrightarrow{\partial_1} C_0 \to 0.$$

Let $\ker(\partial_n) = \{x \in C_n \mid \partial_n(x) = 0\}$ be the kernel of ∂_n, which we call *cycles*, and let $\mathrm{im}(\partial_n) = \{y \in C_{n-1} \mid \text{there exists } x \in C_n \text{ such that } \partial_n(x) = y\}$ be the image of ∂_n, which we call *boundaries*. Since $\partial_{n+1} \circ \partial_n = 0$ holds for all n, it is clear that $\mathrm{im}(\partial_{n+1}) \subseteq \ker(\partial_n)$ for all n. Since both $\mathrm{im}(\partial_{n+1})$ and $\ker(\partial_n)$ are vector spaces, we may form the quotient vector space

$$H_n(C_\bullet) := \ker(\partial_n)/\mathrm{im}(\partial_{n+1})$$

for all $n \geq 0$. We call $H_n(C_\bullet)$ the *n-th homology (group)* of $C_\bullet$, and the elements of $H_n(C_\bullet)$ are called *homology classes*. Moreover, the dimension of $H_n(C_\bullet)$ as a vector space, denoted by $\beta_k(C_\bullet)$, is called the *n-th Betti number* of $C_\bullet$.

2.3.3 Simplicial Homology

We will introduce *simplicial homology*, a type of homology that is widely used in algebraic topology and tailored to our purposes. As the name suggests, simplicial homology is derived from simplicial complexes.

Given a simplicial complex $\mathcal{K}$, we define a chain complex $C_\bullet(\mathcal{K})$ of $\mathbb{Z}_2$-vector spaces in the following way: Let $\mathcal{K}_n = \{\sigma_1, \dots, \sigma_k\}$ be the n-skeleton of $\mathcal{K}$. Then we define $C_n(\mathcal{K})$ to be the vector space over $\mathbb{Z}_2$ with $\mathcal{K}_n$ as a basis. Moreover, for each n-simplex

$\sigma_i = [v_0, \ldots, v_n] \in C_n(\mathcal{K})$, the boundary operator $\partial_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K})$ is defined by

$$\partial_n(\sigma_i) := \sum_{j=0}^{n} [v_0, \ldots, \hat{v_j}, \ldots, v_n],$$

where $\hat{v_j}$ means that v_j is omitted. By extending linearly to all of $C_n(\mathcal{K})$, the boundary operator ∂_n can be defined on $C_n(\mathcal{K})$.

It is easy to see that the boundary operator ∂_n satisfies $\partial_{n+1} \circ \partial_n = 0$. Hence vector spaces $C_n(\mathcal{K})$ with the boundary operators ∂_n form a chain complex. This allows us to form the homologies $H_n(C_\bullet(\mathcal{K}))$ of $C_\bullet(\mathcal{K})$, which we call *simplicial homology.*

2.3.4 Persistence Homology and Diagrams

Given a point cloud P sampled from the unknown manifold M, how can we determine the topological characteristics of M from P? The simplest way is to generate a suitable manifold K from P and compute its homology. However, homology is very sensitive to small changes, so there is a problem that the topological characteristics of K can be very different from those of M. *Persistence homology* (Oudot (2017)) addresses this problem by incorporating the scale ε, which varies from 0 to ∞ , into homology computations. As ε increases, the topological characteristics of a manifold induced by ε can vary: some topological information born at some ϵ_0 and die ϵ_1. Informally, the idea of persistence homology is to track all the birth and death of topological information with scale ε.

A persistence homology essentially tracks the evolution of homology classes in a filtration of simplicial complexes $\mathcal{K}$. Once a simplicial complex $\mathcal{K}$ admits a filtration

$$\mathcal{K}^\bullet : \emptyset = \mathcal{K}^{-\infty} \subseteq \cdots \subseteq \mathcal{K}^i \subseteq \cdots \subseteq \mathcal{K}^j \subseteq \cdots \subseteq \mathcal{K}^\infty = \mathcal{K},$$

then each inclusion $f_{i,j} : \mathcal{K}^i \hookrightarrow \mathcal{K}^j$ is a simplicial map so that it induces a linear map $H_n(f_{i,j})$ between the homologies of $H_n(\mathcal{K}^i)$ and $H_n(\mathcal{K}^j)$. Such indices i and j are referred to as 'time' in persistence homology theory. Fix a non-negative integer $n \geq 0$. For any $i < j$, one can see whether a homology class in $H_n(\mathcal{K}^i)$ are mapped to the same homology class in $H_n(\mathcal{K}^j)$ by $H_n(f_{i,j})$. If this happens, such a homology class is said to *persist from time i to j*. If not, such a class is said to have *died at some time between i and j*. If a homology class first appears at time i and disappears at time j, then we say that this class is *born at time i and dies at time j*, and appends (i, j) as an element of *n-th persistence homology.* By tracking all homology classes of $\{H_n(\mathcal{K}^t)\}_{t\in\mathbb{R}}$ and appending them as elements of n-th persistence homology, the n-th persistence homology can be seen as a multi-set of birth and death tuples. Now, regard the n-th persistence homology as a multi-set of points in $\mathbb{R}^2$. Then we call such a multi-set *n-th persistence diagram.*

Informally, n-th persistence homology (or diagram) tracks different topological features depending on n. For example, 0-th persistence homology tracks the birth and death of connected components, while 1-th persistence homology tracks those of circular holes. The general n-th persistence homology has information about the birth and death of n-dimensional holes. Through persistence homology information for each n, we can understand the characteristics of a given topological object.

2.3.5 Vietoris-Rips Complex and Filtration

Given a point cloud P with a distance matrix M and a scale ε, the *Vietoris-Rips complex* of ε is a type of simplicial complex constructed from P and M whose simplices are formed by connecting points in P that are within a certain distance ε of each other. In particular, given two scales $\varepsilon_1 < \varepsilon_2$, the Vietoris-Rips complex of ε_1 is contained in that of ε_2. Thus, by adjusting a scale ε, we can define a filtration of simplicial complexes, called *Vietoris-Rips filtration*. This makes it possible to analyze the persistence homologies of a point cloud. Here we will provide their definitions below.

Definition 3 *Let P be a finite point cloud, and let M be a non-negative symmetric matrix of size $|P| \times |P|$ with zero diagonals. The* Vietoris-Rips *complex of (P, M) with a scale $\varepsilon \in \mathbb{R}_{\geq 0}$, denoted as $\mathrm{VR}^{\varepsilon}(P, M)$, has one t-simplex per $(t+1)$-tuple of points $(u_0, \ldots, u_t)$ of P such that $M(u_i, u_j) \leq \varepsilon$ for all $i, j = 0, \ldots, t$. Moreover, the* Vietoris-Rips filtration *of (P, M) is the indexed family $\mathrm{VR}(P, M) = \{\mathrm{VR}^{\varepsilon}(P, M)\}_{\varepsilon \in \mathbb{R}}$. Finally, we denote the k-skeleton of $\mathrm{VR}^{\varepsilon}(P, M)$ as $\mathrm{VR}^{\varepsilon}_k(P, M)$, and let $\mathrm{VR}_k(P, M) = \{\mathrm{VR}^{\varepsilon}_k(P, M)\}_{\varepsilon \in \mathbb{R}}$ for $k \in \mathbb{Z}_{\geq 0}$.*

Finally, we will provide a short remark regarding the matrix M used in Definition 3. In general, when defining the Vietoris-Rips complex, the matrix M is constructed based on the distance between two points according to some metric. However, since the Vietoris-Rips complex can be defined based solely on the pairwise distances between points, we follow a general definition (Definition 3) with minimal conditions on the matrix M, independent of a metric. In other words, the matrix M in Definition 3 may not satisfy properties of metrics such as triangle inequality. For example, M may not satisfy the triangle inequality, that is, $M_{i,j} + M_{j,k} \not\leq M_{i,k}$ for some i, j, and k. However, since the existence of a metric does not affect the theoretical results that will be developed later in this paper, we will use the general version of the Vietoris-Rips complex based on such a matrix M.

3. Notations and conventions

In this section, we summarize some notations used throughout this paper. First of all, we use the following notations, which are commonly used, to distinguish between a set and a multi-set: we denote a set by $\{\cdots\}$ and a multi-set by $\{\{\cdots\}\}$. Moreover, the WL test refers to the 1-WL test, otherwise specified.

Let $(\mathcal{G}, \mathcal{C})$ be a space of graphs with node coloring $\mathcal{C}$, and let $\chi_{\mathcal{G}} \subseteq \mathbb{R}^N$ (or simply χ) be a space of node features of $(\mathcal{G}, \mathcal{C})$ containing $(0, \ldots, 0)$, where $N \in \mathbb{N}$. For a graph $G \in \mathcal{G}$, $V(G)$ denotes a set of nodes in G, $E(G)$ denotes a set of edges in G, and $E^{\mathcal{C}}(G)$ denotes a multi-set of colored edges in G, that is, $E^{\mathcal{C}}(G) = \{\{\ \{\{\mathcal{C}(u), \mathcal{C}(v)\}\} \mid \{\{u, v\}\} \in E(G)\}\}$.

Finally, since we frequently use the notation in Definition 3, we summarize it here again. Given a finite point cloud P and a non-negative symmetric matrix M of size $|P| \times |P|$ with zero diagonals, we denote the k-skeleton of $\mathrm{VR}^{\varepsilon}(P, M)$ as $\mathrm{VR}^{\varepsilon}_k(P, M)$, and put $\mathrm{VR}_k(P, M) = \{\mathrm{VR}^{\varepsilon}_k(P, M)\}_{\varepsilon \in \mathbb{R}}$ for $k \in \mathbb{Z}_{\geq 0}$.

4. Theoretical Framework: Topological Edge Diagram

In this section, we introduce a novel edge filtration-based persistence diagram, which we call *Topological Edge Diagram*. To begin with, we introduce the *edge filtration* map, and

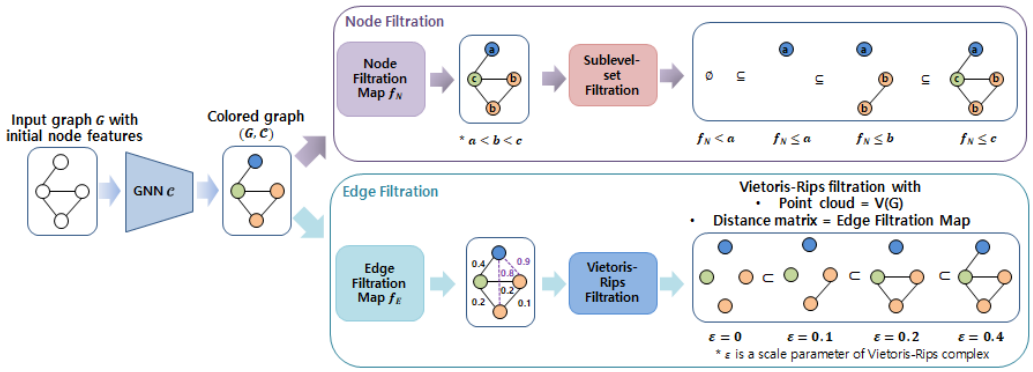

Figure 2: The difference in subgraph construction between two types of filtrations: node filtration and edge filtration.

then define the topological edge diagram by leveraging it. In the end, we will wrap up this section by analyzing its theoretical expressive power.

4.1 Definition of Topological Edge Diagram (TED)

To introduce our persistence diagram, we first introduce the *edge filtration* (map) which is essential in developing our theoretical framework. To grasp the distinction between node (Hofer et al. (2020); Horn et al. (2021)) and edge filtration, see Figure 2.

Definition 4 *Let $G \in \mathcal{G}$ be a graph, and let $\mathcal{C}$ be a node coloring of $\mathcal{G}$.*

1. *An* edge filtration (map) *$ef^{\mathcal{C}}$ of $\mathcal{G}$ with respect to $\mathcal{C}$ is defined to be a positive real-valued function $ef^{\mathcal{C}} : \bigcup_{G\in\mathcal{G}} E^{\mathcal{C}}(G) \to \mathbb{R}_{>0}$ such that $\sup\{ef^{\mathcal{C}}(x) \mid x \in \bigcup_{G\in\mathcal{G}} E^{\mathcal{C}}(G)\} < \infty$, where $E^{\mathcal{C}}(G) = \{\{\ \{\{\mathcal{C}(u), \mathcal{C}(v)\}\} \mid \{\{u, v\}\} \in E(G)\}\}$ is a multi-set of colored edges.*

2. *For an edge filtration $ef^{\mathcal{C}}$ and $i \ge 0$, $\mathrm{ph}^i_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ is the i-th persistence diagram of 1-skeleton of Vietoris-Rips filtration with point cloud $V(G)$ whose distance matrix $M(G)$ is defined as follows: for any $i, j = 1, \ldots, |V(G)|$ corresponding to nodes $u_i, u_j \in V(G)$,*

$$M(G)_{i,j} = \begin{cases} ef^{\mathcal{C}}(\{\{\mathcal{C}(u_i), \mathcal{C}(u_j)\}\}) & \text{if } \{\{\mathcal{C}(u_i), \mathcal{C}(u_j)\}\} \in E^{\mathcal{C}}(G), \\ \infty & \text{otherwise} \end{cases}$$

Remark 5 *In this remark, we will explain why the 1-skeleton of the Vietoris-Rips complex of $V(G)$ is a natural choice when defining the persistent homology $\mathrm{ph}^i_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ of the graph G (Definition 4-2). Before going on, we briefly explain the conditions that the filtration $\{X_\varepsilon\}_{\varepsilon>0}$ of a simplicial complex X must satisfy to define persistent homology: given a simplicial complex X, the filtration $\{X_\varepsilon\}_{\varepsilon>0}$ to define the persistence homology of X should satisfy the following condition: for any $\varepsilon > 0$, $X_\varepsilon \hookrightarrow X$ as simplicial complexes and $\lim_{\varepsilon\to\infty} X_\varepsilon = X$.*

In this perspective, we see how the dimension of the clique complex of G is related when defining its Vietoris-Rips filtration. More precisely, let $\mathrm{VR}^\varepsilon_d(V(G), M(G))$ be the d-skeleton of Vietoris-Rips complex of scale $\varepsilon > 0$ with point cloud $V(G)$ whose distance matrix is

$M(G)$ (Definition 4). Since a graph $G \in \mathcal{G}$ is 1-simplicial complex, it is easy to see the following:

1. *for any $\varepsilon > 0$, $\mathrm{VR}_1^\varepsilon(V(G), M(G)) \hookrightarrow G$ as simplicial complexes,*
2. *$\lim_{\varepsilon\to\infty} \mathrm{VR}_1^\varepsilon(V(G), M(G)) = G$, and*
3. *for any $d \geq 2$, there exists $\varepsilon > 0$ such that $\mathrm{VR}_d^\varepsilon(V(G), M(G)) \not\hookrightarrow G$ as simplicial complexes.*

In other words, 1-skeleton is the most natural choice for defining the persistence homology $\mathrm{ph}_{\mathrm{VR}}^i(G, ef^{\mathcal{C}})$ of the graph.

Note that the edge filtration induces a novel persistence diagram by using the Vietoris-Rips filtration of graph. In particular, when the edge filtration is injective, we call this persistence diagram, *Topological Edge Diagram (TED).*

Definition 6 *Given a graph $G \in \mathcal{G}$ and an injective edge filtration $ef^{\mathcal{C}}$ with a node coloring $\mathcal{C}$, a* topological edge diagram (TED) *of $(G, ef^{\mathcal{C}})$, denoted by $\mathrm{TED}(G, ef^{\mathcal{C}})$, is defined to be the tuple of multi-sets $(\mathrm{ph}_{\mathrm{VR}}^0(G, ef^{\mathcal{C}}),\ \mathrm{ph}_{\mathrm{VR}}^1(G, ef^{\mathcal{C}}))$.*

4.2 Theoretical Expressivity of TED

It is clear that an injective edge filtration $ef^{\mathcal{C}}$ of the node coloring $\mathcal{C}$ includes all the pairwise node coloring information. Hence our question is whether TED also contains all of the node coloring information. For theoretical clarity, we first propose a weak assumption about the node coloring $\mathcal{C}$, called the *Degree Assumption.*

Definition 7 (Degree Assumption) *Let $\mathcal{C}$ be a node coloring of $\mathcal{G}$, that is, $\mathcal{C} : \bigcup_{G\in\mathcal{G}} V(G) \to \chi$. We say that C satisfies the* degree assumption *if the following holds: for any $u, v \in \bigcup_{G\in\mathcal{G}} V(G)$, if $\mathcal{C}(u) = \mathcal{C}(v)$, then $\deg(u) = \deg(v)$.*

We remark that the degree assumption is a very weak assumption related to node coloring, and almost all node colorings we know satisfy this assumption. For example, any node colorings of all message passing GNNs with arbitrary initial node colorings satisfy the Degree Assumption. Furthermore, the WL test, which is a classical graph isomorphism test, satisfies the Degree Assumption as demonstrated in the following example.

Now, we are ready to show the strong expressive power of TED. Specifically, our next result shows that TED can incorporate all of the node coloring information if any graph $G \in \mathcal{G}$ has no isolated nodes.

Lemma 8 *Let $G, H \in \mathcal{G}$ be graphs with no isolated nodes, and let $\mathcal{C}$ be a node coloring of $\mathcal{G}$ satisfying the degree assumption. If $\{\{\mathcal{C}(u) \mid u \in V(G)\}\} \neq \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$, then $\mathrm{TED}(G, ef^{\mathcal{C}}) \neq \mathrm{TED}(H, ef^{\mathcal{C}})$ for any injective edge filtration $ef^{\mathcal{C}}$ and any $k \geq 1$.*

Proof Since $\{\{\mathcal{C}(u) \mid u \in V(G)\}\} \neq \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$, there are three cases:

1. $|V(G)| \neq |V(H)|$,

2. $|V(G)| = |V(H)|$ and there exists $u \in V(G)$ such that $\mathcal{C}(u) \notin \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$,
3. $|V(G)| = |V(H)|$ and $\{\mathcal{C}(u) \mid u \in V(G)\} = \{\mathcal{C}(v) \mid v \in V(H)\}$ but $\{\{\mathcal{C}(u) \mid u \in V(G)\}\} \neq \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$.

Fix an injective edge filtration $ef^{\mathcal{C}}$. First of all, suppose the case (1). Since an edge filtration $ef^{\mathcal{C}}$ is positive, each element in $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ should be of the form $(0, t)$ for some $t \in (0, \infty]$. So the number of elements in the multi-set $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ is equal to the number of nodes in G. Hence $|V(G)| \neq |V(H)|$ implies that the cardinalities of $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ and $\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ are not equal as multi-sets so that $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}) \neq \mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$.

Next, suppose the case (2). Let $u \in V(G)$ be a node satisfying $\mathcal{C}(u) \notin \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$. Since G has no isolated nodes, there exists an edge $e_G \in E(G)$ whose one of the end points is u. Since $\mathcal{C}(u) \notin \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$, it is easy to see from the injectivity of $ef^{\mathcal{C}}$ that there does not exist an edge $e \in E(H)$ such that $ef^{\mathcal{C}}(e) = ef^{\mathcal{C}}(e_G)$.

Now we claim that either $(0, ef^{\mathcal{C}}(e_G)) \in \mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ or $(ef^{\mathcal{C}}(e_G), \infty) \in \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ holds but neither $(0, ef^{\mathcal{C}}(e_G)) \in \mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ nor $(ef^{\mathcal{C}}(e_G), \infty) \in \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ holds. Note that it is clear that both $(0, ef^{\mathcal{C}}(e_G)) \notin \mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ and $(ef^{\mathcal{C}}(e_G), \infty) \notin \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ hold since there is no edge $e \in E(H)$ satisfying $ef^{\mathcal{C}}(e) = ef^{\mathcal{C}}(e_G)$. Hence we need to show the first part of the claim. Note that the birth of a colored edge $e = \{\{\mathcal{C}(u), \mathcal{C}(v)\}\} \in E^{\mathcal{C}}(G)$ corresponds to either of the following two cases:

(i) there was no path between u and v before the birth of e, or

(ii) there was a path between u and v before the birth of e.

In case (i), the birth of e leads to the death of a connected component, which corresponds to $(0, ef^{\mathcal{C}}(e_G)) \in \mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$. Moreover, the second case (ii) implies that e creates a new cycle so that it corresponds to $(ef^{\mathcal{C}}(e_G), \infty) \in \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ since $\mathrm{ph}^i_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ is computed from the 1-skeleton of Vietoris-Rips complex with point cloud $V(G)$. This proves our claim. In other words, our claim shows that there is an element that is contained in $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}) \cup \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ but not in $\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}}) \cup \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}})$, we conclude that $(\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})) \neq (\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}}))$.

Finally, it remains to show the conclusion holds for the case (3). Since $\{\mathcal{C}(u) \mid u \in V(G)\} = \{\mathcal{C}(v) \mid v \in V(H)\}$, there exists an indexed coloring set $\{C_i\}_{i \in I}$ with an index set I such that for each $i \in I$, there exists $u \in V(G)$ and $v \in V(H)$ satisfying $\mathcal{C}(u) = \mathcal{C}(v) = C_i$. Towards contradiction, suppose not, that is, $\mathrm{ph}^k_{\mathrm{VR}}(G, ef^{\mathcal{C}}) = \mathrm{ph}^k_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ for any $k = 0, 1$. Again, by the claim in the proof of case (2), this implies that $E^{\mathcal{C}}(G) = E^{\mathcal{C}}(H)$. Hence, for all $i \in I$, we have

$$\sum_{\substack{u \in V(G), \\ \mathcal{C}(u) = C_i}} \deg(u) = \sum_{\substack{v \in V(H), \\ \mathcal{C}(v) = C_i}} \deg(v). \tag{1}$$

However, since the multi-set of node features of G and H are not the same, that is, $\{\{\mathcal{C}(u) \mid u \in V(G)\}\} \neq \{\{\mathcal{C}(v) \mid v \in V(H)\}\}$, there exists an index $i_0 \in I$ such that

$$|\{u \in V(G) \mid \mathcal{C}(u) = C_{i_0}\}| \neq |\{v \in V(H) \mid \mathcal{C}(v) = C_{i_0}\}|.$$

By the definition of I, both $\{u \in V(G) \mid \mathcal{C}(u) = C_{i_0}\}$ and $\{v \in V(H) \mid \mathcal{C}(v) = C_{i_0}\}$ are both non-empty, thus we put $\{u \in V(G) \mid \mathcal{C}(u) = C_{i_0}\} = \{u_1, \ldots, u_g\}$ and $\{v \in V(H) \mid \mathcal{C}(v) =$

$C_{i_0}\} = \{v_1, \ldots, v_h\}$ with $g \neq h$. Since all u_i and v_j have the same coloring, the degree assumption of $\mathcal{C}$ gives that

$$\deg(u_1) = \cdots = \deg(u_g) = \deg(v_1) = \cdots = \deg(v_h).$$

Moreover, such a degree value is positive since G and H have no isolated nodes. Hence we have

$$\sum_{\substack{u \in V(G), \\ \mathcal{C}(u) = C_{i_0}}} \deg(u) = g \cdot \deg(u_1) \neq h \cdot deg(v_1) = \sum_{\substack{v \in V(H), \\ \mathcal{C}(v) = C_{i_0}}} \deg(v),$$

which contradicts to Equation 1. Thus

$$(\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})) \neq (\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}})),$$

which concludes the proof. ■

Lemma 8 presents a general result applicable to any node coloring $\mathcal{C}$. Moving forward, we will narrow our focus to WL coloring. Since WL coloring satisfies the degree assumption, Lemma 8 implies that $\mathrm{TED}(\cdot, ef^{\mathcal{C}})$ is at least as powerful as the WL test for any edge filtration $ef^{\mathcal{C}}$, assuming non-isolated nodes. Our next theorem, which is the core of our theoretical framework, shows that a stronger result holds for WL coloring: TED is strictly more powerful than the WL test without assuming non-isolated nodes.

Theorem 9 *Let $G, H \in \mathcal{G}$, and let $\mathcal{C}$ be the stable WL coloring whose initial node colorings are all the same. Additionally, let $ef^{\mathcal{C}}$ be an arbitrary injective edge filtration. If G and H are distinguishable by WL test, then $\mathrm{TED}(G, ef^{\mathcal{C}}) \neq \mathrm{TED}(H, ef^{\mathcal{C}})$. Moreover, there exists a pair of non-isomorphic graphs $(G, H) \in \mathcal{G} \times \mathcal{G}$ such that $\mathrm{TED}(G, ef^{\mathcal{C}}) \neq \mathrm{TED}(H, ef^{\mathcal{C}})$ while the WL test cannot distinguish them.*

Proof Let G and H be two graphs that are distinguishable by the WL test, and fix an injective edge filtration $ef^{\mathcal{C}}$. We divide the graph $(G,\ H)$ into non-isolated components $(G^+,\ H^+)$ and isolated components $(G^0,\ H^0)$. More precisely, we have

$$\begin{aligned} V(G^0) &= \{u \in V(G) \mid \deg(u) = 0\}, \\ V(H^0) &= \{v \in V(H) \mid \deg(v) = 0\}, \\ V(G^+) &= \{u \in V(G) \mid \deg(u) > 0\}, \\ V(H^+) &= \{v \in V(H) \mid \deg(v) > 0\}. \end{aligned}$$

Then $V(G) = V(G^0) \sqcup V(G^+)$ and $V(H) = V(H^0) \sqcup V(H^+)$. If $|V(G)| \neq |V(H)|$, then $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}) \neq \mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ clearly holds. So we may assume that $|V(G)| = |V(H)|$. We divide it into two cases: (i) $|V(G^0)| \neq |V(H^0)|$ and (ii) $|V(G^0)| = |V(H^0)|$.

First, assume the case (i). For a contradiction, suppose not, that is, $\mathrm{ph}^k_{\mathrm{VR}}(G, ef^{\mathcal{C}}) = \mathrm{ph}^k_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ for all $k = 0, 1$. Since $V(G^0)$ and $V(H^0)$ are isolated, the multi-set of all non-essential points (that is, $(0, t)$ where $t \neq \infty$) of $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ (and $\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ respectively) and the multi-set $\mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ (and $\mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ respectively) should come

Figure 3: An example of a pair of non-isomorphic graphs that are not distinguishable by WL test.

from G^+ (and H^+ respectively). Hence $E^{\mathcal{C}}(G^+)$ and $E^{\mathcal{C}}(H^+)$ should be the same as multi-sets, which implies that

$$\sum_{\substack{u\in V(G^+),\\ \mathcal{C}(u)=C}} \deg(u) = \sum_{\substack{v\in V(H^+),\\ \mathcal{C}(v)=C}} \deg(v) \tag{2}$$

for any color C. However, since $|V(G^0)| \neq |V(H^0)|$, we have $|V(G^+)| \neq |V(H^+)|$. Thus there exists a color C_0 such that $|\{u \in V(G^+) \mid \mathcal{C}(u) = C_0\}| \neq |\{v \in V(H^+) \mid \mathcal{C}(v) = C_0\}|$. Since the WL test satisfies the degree assumption, all $u \in V(G^+)$ and $v \in V(H^+)$ whose WL color is C_0 should have the same positive degree, which we call it d. Hence

$$\begin{aligned}\sum_{\substack{u\in V(G^+),\\ \mathcal{C}(u)=C_0}} \deg(u) &= d \cdot |\{u \in V(G^+) \mid \mathcal{C}(u) = C_0\}| \\ &\neq d \cdot |\{v \in V(H^+) \mid \mathcal{C}(v) = C_0\}| = \sum_{\substack{v\in V(H^+),\\ \mathcal{C}(v)=C_0}} \deg(v),\end{aligned}$$

which contradicts to Equation 2. Hence

$$(\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})) \neq (\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}}))$$

for the case (i).

Now it remains to prove the case (ii) for the first statement. Since $|V(G^0)| = |V(H^0)|$ and all the isolated nodes of G (and H respectively) correspond to the essential points (that is, $(0, \infty)$) in $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ (and $\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ respectively), $\mathrm{ph}^k_{\mathrm{VR}}(G, ef^{\mathcal{C}}) \neq \mathrm{ph}^k_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ is equivalent to $\mathrm{ph}^k_{\mathrm{VR}}(G^+, ef^{\mathcal{C}}) \neq \mathrm{ph}^k_{\mathrm{VR}}(H^+, ef^{\mathcal{C}})$ for any $k = 0, 1$. Moreover, since

$$\{\mathcal{C}(u) \mid u \in V(G),\ \deg(u) = 0\} \cap \{\mathcal{C}(u) \mid u \in V(G),\ \deg(u) > 0\} = \emptyset,$$

the fact that WL test can distinguish G and H implies that it can also distinguish G^+ and H^+. In other words, G^+ and H^+ have no isolated nodes and can be distinguishable by WL test. Since WL test satisfies the degree assumption, Lemma 8 implies that $\mathrm{ph}^k_{\mathrm{VR}}(G^+, ef^{\mathcal{C}}) \neq \mathrm{ph}^k_{\mathrm{VR}}(H^+, ef^{\mathcal{C}})$ for some $k = 0, 1$. Thus $(\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(G, ef^{\mathcal{C}})) \neq (\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(H, ef^{\mathcal{C}}))$ holds as desired.

Next, we will prove the second statement. Let G be the graph on the left of Figure 3 and H the graph on the right. It is clear that the WL test cannot distinguish them since stable WL colorings of all nodes are identical and the number of nodes of G and H are the same. However, for any injective edge filtration $ef^{\mathcal{C}}$, the number of non-essential elements

of $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}})$ is 5 while the number of non-essential elements of $\mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$ is 4. Thus $\mathrm{ph}^0_{\mathrm{VR}}(G, ef^{\mathcal{C}}) \neq \mathrm{ph}^0_{\mathrm{VR}}(H, ef^{\mathcal{C}})$, that is, $(\mathrm{ph}^0_{\mathrm{VR}}(\cdot, ef^{\mathcal{C}}),\ \mathrm{ph}^1_{\mathrm{VR}}(\cdot, ef^{\mathcal{C}}))$ can distinguish G and H as desired. ■

5. Algorithm: Line Graph Vietoris-Rips Persistence Diagram

In this section, we propose a novel neural-network-based algorithm, named *Line Graph Vietoris-Rips Persistence Diagram*, to implement TED. First, we construct the map t_ϕ that transforms a colored graph into a *colored line graph* (Definition 11) in Section 5.1. Next, through t_ϕ, we propose an algorithm to construct an injective edge filtration, which is the core of our Line Graph Vietoris-Rips Persistence Diagram, in Section 5.2. Finally, we analyze its theoretical expressivity in Section 5.3.

5.1 Construction of the map $t_\phi : (\mathcal{G}, \mathcal{C}) \to (L_{\mathcal{G}}, \mathcal{C}^\phi)$

First of all, we briefly recall *line graph* L_G of a graph $G \in \mathcal{G}$. In graph theory, a line graph is a type of graph where the vertices correspond to the edges of a given graph G, and two vertices in the line graph are connected by an edge if and only if their corresponding edges in G share a common endpoint. Formally, it is defined as follows:

Definition 10 *Let $G = (V(G), E(G)) \in \mathcal{G}$ be a graph. Its line graph $L_G = (V(L_G), E(L_G))$ is a graph such that (1)) each node in $V(L_G)$ represents an edge in $E(G)$, and (2) for any $\{u_1, u_2\} \neq \{v_1, v_2\} \in E(G)$ with $u_1, u_2, v_1, v_2 \in V(G)$, their corresponding nodes in $V(L_G)$ are adjacent if and only if $u_i = v_j$ for some $i = 1, 2$ and $j = 1, 2$.*

Throughout this paper, we will write the node of the line graph L_G corresponding to $\{\{u, v\}\} \in E(G)$ as $l_{\{\{u,v\}\}}$. Moreover, we denote $L_{\mathcal{G}} := \{L_G \mid G \in \mathcal{G}\}$. Now we first propose a *colored line graph*, which is a coloring version of a line graph (Definition 10). This is defined using the node coloring $\mathcal{C}$ of a given colored graph $(G, \mathcal{C})$.

Definition 11 *Let $(G, \mathcal{C})$ be a graph with node coloring $\mathcal{C}$. A* colored line graph $(L_G, \mathcal{C}^h)$ *of $(G, \mathcal{C})$ with respect to h is the line graph L_G of G with the node coloring $\mathcal{C}^h$ such that for any $l_{\{\{u,v\}\}} \in V(L_G)$, $\mathcal{C}^h(l_{\{\{u,v\}\}}) = h(\{\{\mathcal{C}(u), \mathcal{C}(v)\}\})$, where $l_{\{\{u,v\}\}}$ is a node in L_G corresponding to $\{\{u, v\}\} \in E(G)$ and h is a hash map on $\{\ \{\{\mathcal{C}(u), \mathcal{C}(v)\}\} \in E^{\mathcal{C}}(G) \mid u, v \in V(G), G \in \mathcal{G}\}$.*

In order to elaborate our algorithm, *Line Graph Vietoris-Rips (LGVR) Persistence Diagram*, we first construct a map t_ϕ that transforms $(G, \mathcal{C}) \in (\mathcal{G}, \mathcal{C})$ into a colored line graph $(L_G, \mathcal{C}^\phi) \in (L_{\mathcal{G}}, \mathcal{C}^\phi)$. Recall that χ denotes a space of node features of $(\mathcal{G}, \mathcal{C})$ containing $(0, \ldots, 0)$, where $N \in \mathbb{N}$. Let $\mathcal{M}_\chi(2) = \{\{\{x, y\}\} \mid x, y \in \chi\}$, and let m be a multi-layer perceptron with learnable parameters. Now, define a map

$$\phi : \mathcal{M}_\chi(2) \to \mathbb{R}^{2N},\ \{\{x, y\}\} \mapsto (x + \eta \cdot m(x) + y + \eta \cdot m(y),\ |x + \eta \cdot m(x) - y - \eta \cdot m(y)|),$$

where η is a learnable scalar parameter. Then we can define the map $t_\phi : (\mathcal{G}, \mathcal{C}) \to (L_{\mathcal{G}}, \mathcal{C}^\phi)$ as follows: for a given $(G, \mathcal{C}) \in (\mathcal{G}, \mathcal{C})$, let $l_{\{\{u,v\}\}}$ be the node in L_G corresponding to

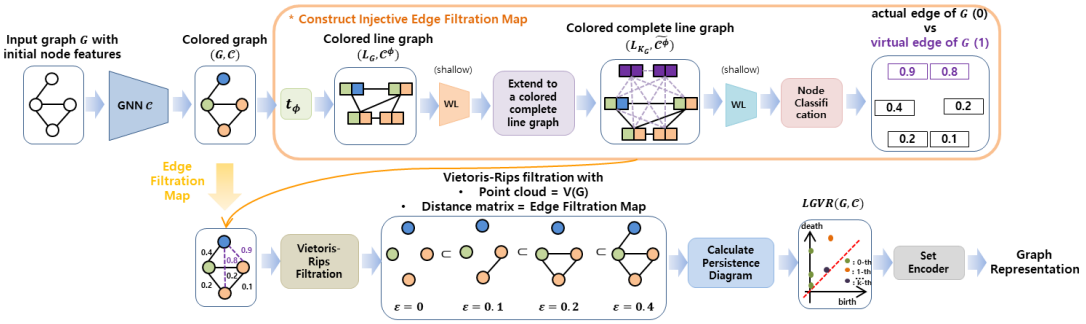

Figure 4: An overall framework of Algorithm 1. Note that the map t_ϕ transforms a colored graph $(G, \mathcal{C})$ into a colored line graph $(L_G, \mathcal{C}^\phi)$ as described in **Construction of the map** t_ϕ.

$\{\{u, v\}\} \in E(G)$. Since $\{\{\mathcal{C}(u), \mathcal{C}(v)\}\} \in E^{\mathcal{C}}(G) \subseteq \mathcal{M}_\chi(2)$, we can assign a coloring $\phi(\{\{\mathcal{C}(u), \mathcal{C}(v)\}\})$ to a node $l_{\{\{u,v\}\}} \in V(L_G)$, that is,

$$\mathcal{C}^\phi(l_{\{\{u,v\}\}}) := \phi(\{\{\mathcal{C}(u), \mathcal{C}(v)\}\}).$$

Since each edge in G has a corresponding node in L_G, t_ϕ can be well-defined by assigning colors to all the nodes in L_G via ϕ.

5.2 Construction of Line Graph Vietoris-Rips (LGVR) Persistence Diagram

In this section, we will elaborate on a neural network-based algorithm, named *Line Graph Vietoris-Rips (LGVR) Persistence Diagram*, with the same expressivity as TED. Pseudocode for the construction of LGVR is described in Algorithm 1 (See Figure 4 for the overall framework). Here, we briefly explain it.

The core of LGVR is to construct an injective edge filtration matrix $A_G^{\mathcal{C}}$ that contains distance information between nodes in a colored graph $(G, \mathcal{C})$ (See orange box in Figure 4). To do this, we first convert $(G, \mathcal{C})$ into a colored line graph $(L_G, \mathcal{C}^\phi)$ through a map t_ϕ. Now that the edge information of G has been converted into the node information of L_G, we perform the binary node classification task (actual vs virtual) so that the nodes of L_G corresponding to the actual edges of G have a value of 0. However, since all nodes in L_G correspond to actual edges in G, they are all trained to have values sufficiently close to 0 during node classification, which can hinder the acquisition of rich information. To address this, we extend $(L_G, \mathcal{C}^\phi)$ into a colored complete line graph $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$, and perform the task on $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$. In this way, we extract meaningful edge information by adding virtual edges to G and training to distinguish actual edges from virtual ones (Appendix A). Based on the extracted edge information of G, we construct the matrix $A_G^{\mathcal{C}}$. Finally, $\mathrm{LGVR}(G, \mathcal{C})$ is defined as the persistence diagram of $\{\mathrm{VR}_1^\varepsilon(V(G), A_G^{\mathcal{C}})\}_{\varepsilon \in [0, 0.5]}$ (See Appendix B for $\varepsilon \in [0, 0.5]$). Similar to conventional GNNs, LGVR is trained in an end-to-end fashion with task-specific loss $\mathcal{L}_{\text{task}}$ (for example, cross entropy) and the LGVR loss $\mathcal{L}_{\text{LGVR}}$ in Algorithm 1.

We conclude this section with the complexity analysis of LGVR. Since the complexity of LGVR is dominated by the calculation of persistence homology, we will focus on explaining the computational complexity of dimensions 0 and 1. In short, they can be computed efficiently with a worst-case complexity of $\mathcal{O}(m\alpha(m)))$ for a graph with m sorted edges

Algorithm 1 Line Graph Vietoris-Rips (LGVR) Persistence Diagram of $(G, \mathcal{C})$

Input:
Labeled graph dataset $(G, y_G) \in (\mathcal{G}, y_{\mathcal{G}})$
Initialize:
$\mathcal{C}(G|\theta^{\mathcal{C}}) \leftarrow$ GNN with weight $\theta^{\mathcal{C}}$
$t_\phi((G,\mathcal{C})|\theta^{t_\phi}) \leftarrow$ a map with weight θ^{t_ϕ} in **Construction of the map** t_ϕ
$P_1(G|\theta^{P_1}) \leftarrow$ (shallow) node-level GINs with weights θ^{P_1}
$P_2(G|\theta^{P_2}) \leftarrow$ (shallow) $[0,1]$-valued node-level GINs with weights θ^{P_2}
$S(\cdot|\theta^S) \leftarrow$ a set encoder with weight θ^S ▷ See Appendix C
$m_{\text{task}}(\cdot|\theta^{m_{\text{task}}}) \leftarrow$ a task-specific MLP with weight $\theta^{m_{\text{task}}}$
for T = 1 to total iteration **do**
Sample a mini-batch $\{(G_1, y_{G_1}), \ldots, (G_n, y_{G_n})\}$ of size n from $(\mathcal{G}, y_{\mathcal{G}})$
for t = 1 to n **do**
$(G_t, \mathcal{C}) \leftarrow \mathcal{C}(G_t|\theta^{\mathcal{C}})$ and construct the edge filtration matrix $A^{\mathcal{C}}_{G_t}$ in **Supplementary**
$\text{LGVR}(G_t, \mathcal{C}) \leftarrow (\text{ph}^0(\{\text{VR}_1^\varepsilon(V(G_t),\ A^{\mathcal{C}}_{G_t})\}_{\varepsilon\in[0,0.5]}), \text{ph}^1(\{\text{VR}_1^\varepsilon(V(G_t), A^{\mathcal{C}}_{G_t})\}_{\varepsilon\in[0,0.5]}))$
end for
Update all the weights $\theta^{\mathcal{C}}$, θ^{t_ϕ}, θ^{P_1}, θ^{P_2}, θ^S, $\theta^{m_{\text{task}}}$ by minimizing the loss

$$\frac{1}{n}\cdot\sum_{t=1}^{n}(\mathcal{L}_{\text{task}}(y_{G_t}, m_{\text{task}}(S(\text{LGVR}(G_t,\mathcal{C})|\theta^S)|\theta^{m_{\text{task}}})) + \lambda\cdot\mathcal{L}_{\text{LGVR}}(G_t)),$$

where λ is a hyperparameter, $\mathcal{L}_{\text{task}}$ is the task-specific loss, and

$$\mathcal{L}_{\text{LGVR}}(G_t) = \frac{1}{|V(L_{K_{G_t}})|}\cdot\sum_{l_{\{\{u_x,u_y\}\}}\in V(L_{K_{G_t}})}(\widetilde{\mathcal{C}^\phi}(l_{\{\{u_x,u_y\}\}}) - 1_{l_{\{\{u_x,u_y\}\}}\notin V(L_{G_t})})^2.$$

end for

Supplementary - Construction of Edge Filtration Matrix $A^{\mathcal{C}}_G$ of $(G,\mathcal{C})$ of size $|V(G)|\times|V(G)|$

$(L_G, \mathcal{C}^\phi) \leftarrow P_1(t_\phi((G,\mathcal{C})|\theta^{t_\phi})|\theta^{P_1})$
$K_G \leftarrow$ the complete graph of $V(G)$
$\pi_G \leftarrow$ the natural inclusion map: $V(L_G) = E(G) \hookrightarrow E(K_G) = V(L_{K_G})$
for $v \in V(L_{K_G})$ **do**

$$\widetilde{\mathcal{C}_0^\phi}(v) \leftarrow \begin{cases}\mathcal{C}^\phi(u) & \text{if } v = \pi_G(u) \text{ for some } u \in V(L_G),\\ (0,\ldots,0) & \text{otherwise.}\end{cases}$$

end for
Extend $(L_G, \mathcal{C}^\phi)$ to $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$, and then $(L_{K_G}, \widetilde{\mathcal{C}^\phi}) \leftarrow P_2((L_{K_G}, \widetilde{\mathcal{C}_0^\phi})|\theta^{P_2})$ ▷ See Appendix A
for $i, j \in \{1, \ldots, |V(G)|\}$ corresponding to $u_i, u_j \in V(G)$ with $l_{\{\{u_i,u_j\}\}} \in V(L_{K_G})$, **do**

$$(A^{\mathcal{C}}_G)_{i,j} \leftarrow \begin{cases}0 & \text{if } i = j,\\ \widetilde{\mathcal{C}^\phi}(l_{\{\{u_i,u_j\}\}}) & \text{otherwise}\end{cases}$$

▷ Edge Filtration Map $\widetilde{\mathcal{C}^\phi}(\cdot)$ and Matrix $A^{\mathcal{C}}_G$ of $(G,\mathcal{C})$
end for

(Edelsbrunner and Harer (2022)), where $\alpha(\cdot)$ refers to the inverse Ackermann function, which can be considered practically constant for all purposes. Therefore, the computation of persistence homology is mainly affected by the complexity of sorting all edges, which is $\mathcal{O}(m\log(m)))$.

5.3 Theoretical Expressivity of LGVR

The most crucial point in LGVR is whether the edge filtration in Algorithm 1 can be injective. In this regard, we will first state and prove the most essential lemma that demonstrates the injectivity of the edge filtration used in LGVR as follows.

Lemma 12 *Assume that χ is countable, and let $\mathcal{M}_\chi(2) = \{\{\{x, y\}\} \mid x, y \in \chi\}$. Then there exists an injective map*

$$\phi : \mathcal{M}_\chi(2) \to \tilde{\chi} \oplus \tilde{\chi} \hookrightarrow \mathbb{R}^{2N},$$

where $\tilde{\chi} \hookrightarrow \mathbb{R}^N$ and $\tilde{\chi}$ is countable.

Proof Since χ is countable, it is easy to see that there exists an injective map $f : \chi \to \mathbb{R}$ and infinitely many $\varepsilon \in \mathbb{R}$ such that for any $d \in \mathbb{N}$,

$$\Big(\bigcup_{X \in \mathcal{M}_\chi(d)} \{ \sum_{\substack{x \in X, \\ rsgn(x) = \pm 1}} rsgn(x) \cdot \varepsilon \cdot (f(x), \ldots, f(x)) \in \mathbb{R}^N \} \Big) \cap \chi = \{\vec{0}\}. \tag{3}$$

By fixing a desired f and ε, we put $\tilde{\chi} = \{x + \varepsilon \cdot (f(x), \ldots, f(x)) \in \mathbb{R}^N \mid x \in \chi\}$. It is trivial that $\tilde{\chi}$ is countable.

Now consider the map $\phi : \mathcal{M}_\chi(2) \to \tilde{\chi} \oplus \tilde{\chi}$,

$$\{\{x, y\}\} \mapsto (x + \varepsilon \cdot f^N(x) + y + \varepsilon \cdot f^N(y),\ |x + \varepsilon \cdot f^N(x) - y - \varepsilon \cdot f^N(y)|),$$

where $f^N(x) = (f(x), \ldots, f(x)) \in \mathbb{R}^N$. Note that ϕ is well-defined since it is order-invariant. Hence it remains to show that ϕ is injective. For the injectivity, assume that

$$\begin{aligned} &(x_1 + \varepsilon \cdot f^N(x_1) + y_1 + \varepsilon \cdot f^N(y_1),\ |x_1 + \varepsilon \cdot f^N(x_1) - y_1 - \varepsilon \cdot f^N(y_1)|) \\ &= (x_2 + \varepsilon \cdot f^N(x_2) + y_2 + \varepsilon \cdot f^N(y_2),\ |x_2 + \varepsilon \cdot f^N(x_2) - y_2 - \varepsilon \cdot f^N(y_2)|) \end{aligned} \tag{4}$$

for some $x_1, x_2, y_1, y_2 \in \chi$. We need to show that $\{\{x_1, y_1\}\} = \{\{x_2, y_2\}\}$ holds under (4). Note that the first equality of (4) implies that $x_1 + y_1 - x_2 - y_2 = \varepsilon \cdot (f^N(x_2) + f^N(y_2) - f^N(x_1) - f^N(y_1))$. By (3), both sides should be 0. Hence $f^N(x_1) + f^N(y_1) = f^N(x_2) + f^N(y_2)$, which implies that

$$f(x_1) + f(y_1) = f(x_2) + f(y_2). \tag{5}$$

Moreover, the second equality of (4) implies that $x_1 + \varepsilon \cdot f^N(x_1) - y_1 - \varepsilon \cdot f^N(y_1) = \delta \cdot (x_2 + \varepsilon \cdot f^N(x_2) - y_2 - \varepsilon \cdot f^N(y_2))$, where $\delta = (\delta_1, \ldots, \delta_N)$ and

$$\delta_i = \begin{cases} 1 & \text{if the signs of } i\text{-th component of } x_1 + \varepsilon \cdot f^N(x_1) - y_1 - \varepsilon \cdot f^N(y_1) \\ & \text{and } x_2 + \varepsilon \cdot f^N(x_2) - y_2 - \varepsilon \cdot f^N(y_2) \text{ are the same,} \\ -1 & \text{otherwise} \end{cases}$$

for any $i = 1, \ldots, N$. Again, the same argument as the first equality implies that

$$f(x_1) - f(y_1) = \delta \cdot (f(x_2) - f(y_2)), \quad \text{that is,} \quad |f(x_1) - f(y_1)| = |f(x_2) - f(y_2)|. \tag{6}$$

By interchanging x_i and y_i, if necessary, we may assume that $f(x_1) \geq f(y_1)$ and $f(x_2) \geq f(y_2)$. Then both equations (5) and (6) imply that $f(x_1) = f(x_2)$ and $f(y_1) = f(y_2)$. Since f is injective, we conclude that $x_1 = x_2$ and $y_1 = y_2$, that is, $\{\{x_1, y_1\}\} = \{\{x_2, y_2\}\}$, which implies the injectivity of ϕ as desired. ■

Lemma 12 shows that under a countable universe, the map ϕ in Section 5.1 is injective. Moreover, since the subsequent steps also maintain the injectivity, this implies that the edge filtration map $\{\{u_i, u_j\}\} \mapsto \widetilde{\mathcal{C}^{\phi}}(l_{\{\{u_i,u_j\}\}})$ in Algorithm 1 is inejctive Based on this, we show that LGVR satisfies the essential property (Lemma 8, Theorem 9) of TED, which implies that LGVR has the same expressivity as TED.

Theorem 13 *Assume that χ is countable. Then the following statements hold:*

1. *Let $\mathcal{C}$ be an arbitrary node coloring of $\mathcal{G}$ satisfying the degree assumption. For any $G, H \in \mathcal{G}$ with no isolated nodes, if $\{\{\mathcal{C}(v) \mid v \in V(G)\}\} \neq \{\{\mathcal{C}(u) \mid u \in V(H)\}\}$, then $\mathrm{LGVR}(G, \mathcal{C}) \neq \mathrm{LGVR}(H, \mathcal{C})$.*
2. *Let $\mathcal{C}$ be the stable WL coloring whose initial node colorings are all the same. If $G, H \in \mathcal{G}$ are distinguishable by $\mathcal{C}$, then we have $\mathrm{LGVR}(G, \mathcal{C}) \neq \mathrm{LGVR}(H, \mathcal{C})$. Moreover, there exists a pair of graphs $(G, H) \in \mathcal{G} \times \mathcal{G}$ such that $\mathrm{LGVR}(G, \mathcal{C}) \neq \mathrm{LGVR}(H, \mathcal{C})$.*

Proof First, we prove (1). Lemma 12 and the universal approximation theorem of multilayer perceptrons (Hornik (1991, 1992)) imply that our map

$$\phi : \mathcal{M}_{\chi}(2) \to \mathbb{R}^{2N}, \quad \{\{x, y\}\} \mapsto (x + \varepsilon \cdot m^*_{\mathcal{G}}(x) + y + \varepsilon \cdot m^*_{\mathcal{G}}(y),\ |x + \varepsilon \cdot m^*_{\mathcal{G}}(x) - y - \varepsilon \cdot m^*_{\mathcal{G}}(y)|)$$

in Section 5.1 induces the injective edge filtration of $\mathcal{G}$ with respect to $\mathcal{C}$. Moreover, since the WL message passing scheme does not decrease the expressive power in distinguishing node colorings of line graphs, the edge filtration of the LGVR that derives the positive symmetric matrix $A^{\mathcal{G}}_G$ is injective. Finally, since G and H are graphs with no isolated nodes and the coloring $\mathcal{C}$ satisfies the degree assumption, the desired result is a direct consequence of Lemma 8.

Next assume that $\mathcal{C}$ is the stable WL coloring whose initial node colorings are all the same. As mentioned in the proof above, we know that the edge filtration of the LGVR that derives the positive symmetric matrix $A^{\mathcal{C}}_G$ is injective. Hence the statement (2) can be proved similarly as the proof of Theorem 9. ■

6. Model Framework

From Theorem 13, we can infer two important results regarding LGVR: (1) LGVR can preserve arbitrary node coloring information, and (2) especially for WL type coloring, LGVR has stronger expressive powers, which enables the construction of more powerful topological GNNs. In this section, we will delve deeper into the construction of topological GNNs. Before providing a framework for applying LGVR to GNN, we first remark one important aspect of LGVR. From a theoretical perspective, LGVR can guarantee the expressive powers of node colorings due to the injectivity of edge filtration. However, from a practical perspective, if specific node information plays a crucial role in determining the characteristics of a graph due to its rich node coloring information, it is likely that LGVR may not extract a graph representation that properly reflects this since LGVR can only indirectly use the node coloring information by converting it into other (topological) information.

To bridge this gap between theoretical and practical perspectives, we first propose a simple mathematical technique that integrates the expressive powers of both coloring information and topological information induced by LGVR under a countable universe in Section 6.1. Depending on the application of this technique, we propose two topological model frameworks in Section 6.2 and analyze their expressivities in Section 6.3.

6.1 Integration Technique

In this subsection, we will present a general mathematical method for constructing an embedding space that integrates the expressive power of each representation under a countable universe. Proposition 15 and Corollary 16 propose a framework that preserves all expressive powers of representations by finding $|I|$ integrated embedding maps f_i. This integration technique will be used for the construction of one of our model frameworks in Section 6.2 in order to integrate the coloring information of $\mathcal{C}$ and the topological information of LGVR simultaneously.

First, we state and prove a useful lemma that is crucial to prove Proposition 15.

Lemma 14 *Given $m \in \mathbb{N}$ and countable spaces $\{\chi_i\}_{i\in I}$, where I is an index set satisfying $|I| < \infty$, there exists a family of functions $\{f_i : \chi_i \to \mathbb{R}^m\}_{i\in I}$ so that*

$$\bigcap_{i\in I}\{f_i(x_i) \mid x_i \in \chi_i\} = \emptyset.$$

Proof First, we prove the case when $|I| = 2$. We claim that given any functions $f_1 : \chi_1 \to \mathbb{R}^m$ and $f_2 : \chi_2 \to \mathbb{R}^m$, there exists $\varepsilon \in \mathbb{R}^m$ such that $\{f_1(x_1) \mid x_1 \in \chi_1\} \cap \{f_2(x_2) + \varepsilon \mid x_2 \in \chi_2\} = \emptyset$. Choose any two functions $f_1 : \chi_1 \to \mathbb{R}^m$ and $f_2 : \chi_2 \to \mathbb{R}^m$, and consider the set

$$DF := \{f_1(x_1) - f_2(x_2) \mid x_1 \in \chi_1 \text{ and } x_2 \in \chi_2\}.$$

Since χ_1 and χ_2 are countable, so is DF. Hence there exists infinitely many $\varepsilon \in \mathbb{R}^m$ such that $\varepsilon \notin DF$, which proves the claim. Now, by replacing f_2 by $f_2 + \varepsilon$ for some $\varepsilon \notin DF$, it is clear that f_1 and f_2 satisfy the desired property.

For general I with $|I| < \infty$, the same arguments implies the existence of infinitely many $\varepsilon_i \in \mathbb{R}^m$, $i \in I$ satisfying

$$\bigcap_{i\in I}\{f_i(x_i) + \varepsilon_i \mid x_i \in \chi_i\} = \emptyset$$

since the countable union of countable sets are countable. Hence by replacing f_i by $f_i + \varepsilon$, the result holds. ∎

Recall that $\mathcal{M}_\chi(d)$ is a space of multi-sets of χ whose cardinality is d, and let $\mathcal{M}_\chi = \bigcup_{d\in\mathbb{N}} \mathcal{M}_\chi(d)$. With this notation in mind, we can prove the following result.

Proposition 15 *Given $m \in \mathbb{R}$ and countable spaces $\{\chi_i\}_{i\in I}$, where I is an index set satisfying $|I| < \infty$, there exists a family of functions $\{f_i : \chi_i \to \mathbb{R}^m\}_{i\in I}$ so that*

$$g(X_1, \ldots, X_{|I|}) = \sum_{i\in I}\sum_{x\in X_i} f_i(x)$$

is unique for each $(X_1, \ldots, X_{|I|}) \in \mathcal{M}_{\chi_1} \times \cdots \times \mathcal{M}_{\chi_{|I|}}$ of bounded sizes.

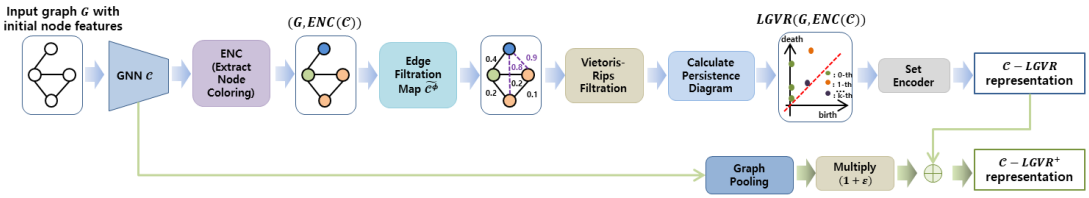

Figure 5: Two model frameworks: $\mathcal{C}$-LGVR and $\mathcal{C}$-LGVR$^+$. Note that ENC component refers to the process of extracting node coloring from the message passing graph neural network (or coloring) $\mathcal{C}$ as described in Section 6.2 and 6.3.

Proof We first prove the case when $|I| = 2$. Since χ_1 and χ_2 are countable, there exists $\tilde{f}_1 : \chi_1 \to \mathbb{R}^m$ and $\tilde{f}_2 : \chi_2 \to \mathbb{R}^m$ so that $\{\tilde{f}_1(x_1) \mid x_1 \in \chi_1\} \cap \{\tilde{f}_2(x_2) \mid x_2 \in \chi_2\} = \emptyset$ by Lemma 14. This implies that for each $(X_1, X_2) \in \mathcal{M}_{\chi_1} \times \mathcal{M}_{\chi_2}$, we have

$$\{\{\tilde{f}_1(x_1) \mid x_1 \in X_1\}\} \cap \{\{\tilde{f}_2(x_2) \mid x_2 \in X_2\}\} = \emptyset. \tag{7}$$

Let $\chi := \tilde{f}_1(\chi_1) \cup \tilde{f}_2(\chi_2)$. Since both χ_1 and χ_2 are countable, χ is also countable. Hence (Xu et al., 2018a, Lemma 5) implies that there exists a function $f : \chi \to \mathbb{R}^n$ so that $g(X) = \sum_{x \in X} f(x)$ is unique for each $X \in \mathcal{M}_\chi$ of bounded size. Now, we put $f_1 = f \circ \tilde{f}_1$ and $f_2 = f \circ \tilde{f}_2$. By Equation (7), every $X \in \mathcal{M}_\chi$ of bounded size can be uniquely decomposed by $\{\{\tilde{f}_1(x_1) \mid x_1 \in X_1\}\} \sqcup \{\{\tilde{f}_2(x_2) \mid x_2 \in X_2\}\}$ for some $(X_1, X_2) \in \mathcal{M}_{\chi_1} \times \mathcal{M}_{\chi_2}$ of bounded sizes. Hence $g(X_1, X_2) = \sum_{i=1}^{2} \sum_{x \in X_i} f_i(x)$ is unique for each $(X_1, X_2) \in \mathcal{M}_{\chi_1} \times \mathcal{M}_{\chi_2}$ of bounded sizes as desired.

As in the proof of Lemma 14, the same argument used in $|I| = 2$ can be extended to any finite number of countable spaces without difficulty by using the fact that countable union of countable sets is countable. ■

From another point of view, we can reformulate the problem of finding the functions f_i in Proposition 15 into a simpler one of finding $|I| - 1$ scalar values. Since this can be easily derived from the proof of Proposition 15, we will omit the proof.

Corollary 16 *Let $m \in \mathbb{R}$ and let I be an index set satisfying $|I| < \infty$. Given countable spaces $\{\chi_i\}_{i \in I}$ and a family of functions $\{f_i : \chi_i \to \mathbb{R}^m\}_{i \in I}$, there exists infinitely many values $\varepsilon_2, \ldots, \varepsilon_{|I|} \in \mathbb{R}$ such that*

$$g(X_1, \ldots, X_{|I|}) = \sum_{x \in X_1} f_1(x) + \sum_{i=2}^{|I|} (1 + \varepsilon_i) \cdot \sum_{x \in X_i} f_i(x)$$

is unique for each $(X_1, \ldots, X_{|I|}) \in \mathcal{M}_{\chi_1} \times \cdots \times \mathcal{M}_{\chi_{|I|}}$ of bounded sizes.

6.2 Model Frameworks: $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$

We propose two model frameworks, $\mathcal{C}$-LGVR and $\mathcal{C}$-LGVR$^+$, depending on the application of the integration technique in Section 6.1. For both model frameworks, $\mathcal{C}$ indicates a message passing graph neural network.

6.2.1 $\mathcal{C}$-LGVR

Note that the output of $\mathcal{C}$ for a graph G may or may not be node coloring, depending on $\mathcal{C}$. For example, if $\mathcal{C}$ is GIN (Xu et al. (2018a)), then its output is the node coloring. However, when $\mathcal{C}$ is PPGN (Maron et al. (2019)), it provides all node tuple colorings as its output, which is not the node coloring exactly. However, since LGVR relies on the node coloring information of $\mathcal{C}$ as input, we first extract the node colorings from $\mathcal{C}$ in order to bridge this gap. We denote the process of extracting node coloring from $\mathcal{C}$ as $\mathrm{ENC}(\mathcal{C})$, and we refer to Section 6.3 for specific examples.

After taking node coloring of G from $\mathrm{ENC}(\mathcal{C})$, we extract the LGVR persistence diagram $\mathrm{LGVR}(G, \mathrm{ENC}(\mathcal{C}))$ (Algorithm 1). Finally, $\mathrm{LGVR}(G, \mathrm{ENC}(\mathcal{C}))$ is encoded via a set encoder to extract a graph representation (Figure 5).

6.2.2 $\mathcal{C}$-LVGR$^+$

$\mathcal{C}$-LVGR$^+$ differs from $\mathcal{C}$-LVGR in the way it extracts the graph representation: $\mathcal{C}$-LVGR$^+$ uses not only the $\mathcal{C}$-LGVR representation but also the pooling output of $\mathcal{C}$ (Figure 5). Specifically, $\mathcal{C}$-LGVR$^+$ uses

$$(\mathcal{C}\text{-LGVR representation}) + (1+\varepsilon) \cdot (\text{Graph Pooling of } \mathcal{C}) \tag{8}$$

as the graph representation vector, where ε is a learnable parameter in Corollary 16, to leverage the expressive powers of both representations (See Remark 17).

Remark 17 *We will briefly explain how Corollary 16 is related to the fact that $\mathcal{C}$-LVGR$^+$ preserves the expressive powers of both coloring information of $\mathcal{C}$ and topological information of LGVR. Note that each component of the output of $\mathcal{C}$-LVGR$^+$ (Equation 8) can be written simply as follows: given a graph G,*

1. *($\mathcal{C}$-LVGR$^+$ representation) $= MLP(\sum_{x_0 \in \mathrm{ph}^0(G)} S_0(x_0), \sum_{x_1 \in \mathrm{ph}^1(G)} S_1(x_1))$, where $\mathrm{ph}^i(G)$ is the i-th persistence diagram of the Vietoris-Rips filtration of G (which is a multi-set) and S_i is a universal set encoder for each $i = 0, 1$, and*
2. *(Graph Pooling of $\mathcal{C}$) $= \sum_{u \in V(G)} \mathcal{C}(u)$.*

Since $\mathrm{ph}^0(G)$, $\mathrm{ph}^1(G)$, and $\{\{\mathcal{C}(u) \mid u \in V(G)\}\}$ are all multi-sets of bounded sizes, the universalities of MLP (Hornik (1991, 1992)) and S_i (Appendix C) guarantee the existence of f_i in Corollary 16. Hence this implies that $\mathcal{C}$-LVGR$^+$ extracts different representation vectors for each triple of multi-sets $(\mathrm{ph}^0(G), \mathrm{ph}^1(G), \{\{\mathcal{C}(u) \mid u \in V(G)\}\})$ by Corollary 16. In other words, $\mathcal{C}$-LVGR$^+$ integrates the coloring information of $\mathcal{C}$, which corresponds to $\{\{\mathcal{C}(u) \mid u \in V(G)\}\}$, and the topological information of LGVR, which corresponds to $(\mathrm{ph}^0(G), \mathrm{ph}^1(G))$.

6.2.3 Training Loss of $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$

For both $\mathcal{C}$-LGVR and $\mathcal{C}$-LGVR$^+$, we train them in an end-to-end fashion by minimizing the task-specific loss $\mathcal{L}_{\text{task}}$ and the LGVR loss $\mathcal{L}_{\text{LGVR}}$ in Algorithm 1 simultaneously, that is, for a set of all learnable parameters θ and a hyperparameter λ,

$$\theta^* = \mathrm{argmin}_\theta \sum_{G \in \mathcal{G}, y_G \in \mathcal{Y}_\mathcal{G}} \{\mathcal{L}_{\text{task}}(G, y_G; \theta) + \lambda \cdot \mathcal{L}_{\text{LGVR}}(G; \theta)\},$$

where $\mathcal{Y}_{\mathcal{G}}$ is the set of ground truth of $\mathcal{G}$ and $y_G \in \mathcal{Y}_{\mathcal{G}}$ is the ground truth of a graph $G \in \mathcal{G}$.

6.3 Theoretical Expressivities of GIN-LGVR, GIN-LVGR$^+$, and PPGN-LVGR$^+$

We analyze the expressive powers of three specific models (which will be used in Section 7), GIN-LGVR, GIN-LVGR$^+$, and PPGN-LVGR$^+$, with the model frameworks in Section 6.2. As described in Section 6.2.1, we first introduce the simplest form of ENC$(\cdot)$ to be used in GIN (Xu et al. (2018a)) and PPGN (Maron et al. (2019)), respectively (Figure 5).

ENC for GIN. Since GIN provides node colorings as output, ENC is set to the identity:

$$\mathrm{ENC}(Y) := Y \in \mathbb{R}^{n \times d},$$

where $Y \in \mathbb{R}^{n \times d}$ is the GIN output, n is the number of graph nodes and d is the dimension of node colorings.

ENC for PPGN. Since PPGN provides all node tuple colorings as output, ENC is set up to extract diagonal elements of the node tuple matrix of PPGN:

$$\mathrm{ENC}(Y) := \mathrm{diag}(Y) \in \mathbb{R}^{n \times d},$$

where $Y \in \mathbb{R}^{n \times n \times d}$ is the PPGN output, n is the number of graph nodes and d is the dimension of node tuple colorings.

Finally, we will conclude this section with a theoretical analysis of our models as follows.

Corollary 18 *Assume that χ is countable. Then the following statements hold:*

1. *Assume that either all initial node colorings are the same or $\mathcal{G}$ contains no graphs with isolated nodes. Then* GIN-LGVR *is strictly more powerful than WL test.*
2. GIN-LVGR$^+$ *is strictly more powerful than WL test.*
3. PPGN-LVGR$^+$ *is at least as powerful as* 3*-WL test.*

Proof First (1) is a direct consequence of Theorem 13 and (Xu et al., 2018a, Theorem 3), and (3) follows from Corollary 16 and Remark 17. Hence it remains to show (2). First, Corollary 16 and Remark 17 again imply that GIN-LVGR$^+$ is at least as powerful as the WL test. To prove the 'strictly powerful' part, we need to show that there exists a pair of graphs G and H that GIN-LVGR$^+$ can distinguish but the WL test cannot. Again, it is easy to see that G and H in Figure 3 work as a desired example, which concludes the proof. ■

7. Experiments

In this section, we evaluate the performance of our models, GIN-LGVR, GIN-LVGR$^+$, and PPGN-LVGR$^+$, on several graph classification and regression benchmark datasets. We measure the performance improvements of our models compared to the baseline message passing GNNs $\mathcal{C}$, GIN and PPGN, to demonstrate that our edge filtration-based approach helps $\mathcal{C}$ to achieve a substantial gain in predictive performance. Furthermore, we compare our models with node filtration-based ones (Hofer et al. (2020)) to validate the superiority of edge filtration over node filtration. In short, we focus on experimental verification of the following claims:

Claim 1. Superior performances of our model frameworks, $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$, that outperform the message passing GNN $\mathcal{C}$.

Claim 2. Superiority of edge filtration-based approach over node filtration-based one.

Claim 3. Experimental validation of our integration technique (Section 6.1; Corollary 16) that integrates the pooling information of $\mathcal{C}$ and the LGVR information.

7.1 Experimental Setup

We conduct experiments on three models: GIN-LGVR, GIN-LVGR$^+$, and PPGN-LVGR$^+$. Both networks, GIN (Xu et al. (2018a)) and PPGN (Maron et al. (2019)), are constructed in their most basic form: they both consist of three message passing layers, with a hidden dimension of 64 for GIN and 400 for PPGN. To minimize the impacts of other techniques, we refrain from using the jumping knowledge network scheme (Xu et al. (2018b)). Finally, to encode LGVR diagrams, which are a tuple of multi-sets, we use a set encoder. To sufficiently leverage their topological information, we set our set encoder by combining Deep Set (Zaheer et al. (2017)) and Set Transformer (Lee et al. (2019)) (See Appendix C for details). We run all experiments on a single DGX-A100 GPU. Our code is publicly available at `https://github.com/samsungsds-research-papers/LGVR`.

7.2 Datasets

We evaluate our methods on two different tasks: graph classification and graph regression. For classification, we test our method on 7 benchmark graph datasets: 5 bioinformatics datasets (MUTAG, PTC, PROTEINS, NCI1, NCI109) that represent chemical compounds or protein substructures, and two social network datasets (IMDB-B, IMDB-M) (Yanardag and Vishwanathan (2015)). For the regression task, we experiment on a standard graph benchmark QM9 dataset (Ramakrishnan et al. (2014); Ruddigkeit et al. (2012); Wu et al. (2018)). It is made up of 134k small organic molecules of varying sizes from 4 to 29 atoms, and the task is to predict 12 real-valued physical quantities for each molecule graph. Further details can be found in Appendix D.1.

7.3 Baseline and Comparison Models

To argue the superiority of our method, we essentially compare our models $\mathcal{C}$-LGVR and $\mathcal{C}$-LVGR$^+$ to the same message passing graph neural network $\mathcal{C}$: we compare GIN-LGVR and GIN-LVGR$^+$ to GIN, and PPGN-LVGR$^+$ to PPGN. To demonstrate the superiority of edge filtration over node filtration, we further compare the performance of both methods for GIN. For fairness in comparison, both models are constructed with the same GIN architecture and extract topological information in a single scale.

Specifically, we conduct experiments using (1) GFL (Hofer et al. (2020)), which is a single-scale version of node filtration, and (2) GFL$^+$ (which we call) that integrates graph pooling and GFL by using the integration technique in Section 6.1. All hyperparameters were carried out on the same set based on $\mathcal{C}$ (Appendix D.2): the learning rate is set to $\{5 \cdot 10^{-3}, 10^{-3}, 5 \cdot 10^{-4}, 10^{-4}, 5 \cdot 10^{-5}\}$ for GIN while it is set to $\{10^{-4}, 5 \cdot 10^{-5}\}$ for PPGN.

The decay rate is set to $\{0.5, 0.75, 1.0\}$ with Adam optimizer (Kingma and Ba (2014)), and we implement all the models by tuning hyperparameters based on the validation score.

7.4 Graph Classification Results

We test our models on datasets from the domains of bioinformatics and social networks. Since there is no separate test set for these datasets, for a fair comparison, we follow the standard 10-fold cross-validation based on the same split used by Zhang et al. (2018) and report the results according to the evaluation protocol described by Maron et al. (2019):

$$\max_{i\in\{1,\dots,t\}} \frac{1}{10}\cdot\sum_{k=1}^{10} P_{k,i}, \tag{9}$$

where t is the total epoch and $P_{k,i}$ is the k-fold validation accuracy at i-th epoch.

7.4.1 Performance Analysis

Table 1 presents the performances of our models (GIN-LGVR, GIN-LVGR$^+$, PPGN-LVGR$^+$) and comparison models (GIN, GFL, GFL$^+$, PPGN). First of all, we experimentally verify **Claim 1, 2, 3**. As shown in Table 1, our models show the best performances across all datasets for the message passing GNN $\mathcal{C}$. This empirically demonstrates that our approach helps the GNN $\mathcal{C}$ to achieve substantial gains in predictive performance, which validates **Claim 1**. Moreover, we found that our approach based on edge filtration shows superior performance compared to node filtration-based methods (GFL, GFL$^+$) by utilizing both nodes and edges to extract more informative representations. This demonstrates the superiority of our edge filtration over node filtration (Figure 6), which validates **Claim 2**. Finally, we remark that GIN-LVGR$^+$ generally shows better performances than GIN-LGVR and outperforms GIN on all datasets: GIN-LVGR$^+$ achieves the best performance on 5 out of 7 datasets. This empirically demonstrates that our theoretical framework (Corollary 16) which integrates the pooling output of $\mathcal{C}$ and the LGVR output without losing information, works well in practice, which validates **Claim 3**.

In addition to **Claim 1, 2, 3**, Table 1 provides an interesting finding: the improvement percentages of our models compared to GIN are inversely proportional to the amount of initial node information. Specifically, as shown in Figure 6, our models show relatively larger improvement percentages on datasets such as MUTAG, PROTEINS, and IMDB with less initial node information (see the feature column in Table 5), compared to PTC, NCI1, and NCI109. This is because having an initial node with rich information enables GNNs to extract a sufficiently expressive graph representation using only the node information so that the performance improvement of additional topological information from LGVR decreases. Therefore, this implies that our LGVR is more effective for datasets with less initial node information, which are relatively difficult to analyze.

For a comprehensive analysis, we further analyze the classification results with $\mathcal{C} = \text{GIN}$ from two aspects in the next subsection. Here we will briefly explain them. First, we found that model performances varied significantly by the data split (Table 2, Figure 7). Hence we further analyze the standard deviation of the 10-fold performances in Table 1 to determine how stable each model is in training regardless of the data split. We found that our models,

Table 1: Graph classification results (with mean accuracy)

	Model / Dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	IMDB-B	IMDB-M
$\mathcal{C}$=GIN	GIN	78.89	59.71	68.11	70.19	69.34	73.8	43.8
	GFL	86.11	60.0	73.06	71.14	70.53	68.7	44.67
	GFL$^+$	79.44	59.11	69.55	71.16	70.12	71.9	44.33
	GIN-LGVR	**86.67**	60.29	**73.42**	69.22	69.39	72.5	45.47
	GIN-LGVR$^+$	85.0	**61.76**	69.55	**71.61**	**70.68**	**74.0**	**45.8**
$\mathcal{C}$=PPGN	PPGN	88.88	64.7	76.39	81.21	81.77	72.2	44.73
	PPGN-LGVR$^+$	**91.11**	**66.47**	**76.76**	**83.04**	**81.88**	**73.5**	**51.0**

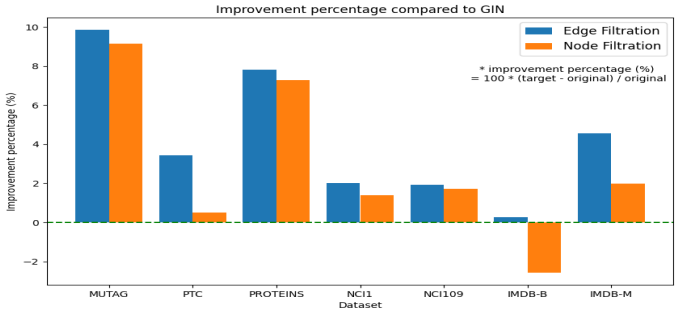

Figure 6: Improvement percentages of two filtrations compared to GIN. The performances of two filtrations are calculated as $\max\{P_{\text{GIN-LGVR}}, P_{\text{GIN-LVGR}^+}\}$ for edge filtration, and $\max\{P_{\text{GFL}}, P_{\text{GFL}^+}\}$ for node filtration, where $P_\bullet$ is the model performance of Table 1.

GIN-LGVR and GIN-LVGR$^+$, show low standard deviations compared to GIN, GFL, and GFL$^+$ (Figure 7). This implies that GIN-LVGR and GIN-LVGR$^+$, which use both nodes and edges to reflect various characteristics in graph representations, would have enabled more stable learning compared to GIN, GFL, and GFL$^+$ that only use nodes.

Next, we compare the models using performance metric different from Equation 9. Metrics are indicators to determine the perspective from which models are compared, hence we measure performances based on another commonly used metric (Equation 10) to compare models from a different perspective. Concerning this metric, our models, GIN-LGVR and GIN-LVGR$^+$, still exhibit the best performance compared to others (See Table 3).

7.4.2 Additional Analysis on Graph Classification with $\mathcal{C}$ = GIN

Here, we will provide additional analysis on the 10-fold classification results of each dataset for five GIN type models in Section 7.4.1: GIN, GFL, GFL$^+$, GIN-LVGR, and GIN-LVGR$^+$. Before conducting the analysis, we fix some notation. Let i_0 be the epoch that computes the performance (Equation 9), that is, Equation (9) $= \frac{1}{10} \cdot \sum_{k=1}^{10} P_{k,i_0}$. For a notational convenience, we denote P_{k,i_0} by P_k.

Analysis of P_k for each model. In Section 7.4.1, we use the average of $\{P_k\}_{k=1,\dots,10}$ as the final performance metric. However, as can be seen in Table 2, the performance P_k varies significantly for each fold k. Therefore, we measure the standard deviation of the 10-fold performances in Table 1 to determine how stable each model (GIN, GFL, GFL$^+$, GIN-LVGR, and GIN-LVGR$^+$) is in training regardless of the data split.

Table 2: Minimum and maximum values of P_k for each model and dataset.

Dataset / Model		GIN	GFL	GFL$^+$	GIN-LVGR	GIN-LVGR$^+$
MUTAG	min	61.11	72.22	66.67	77.78	72.22
	max	94.44	100.0	100.0	100.0	100.0
PTC	min	50.0	47.06	44.12	44.12	55.88
	max	85.35	70.59	76.47	70.59	73.53
PROTEINS	min	59.46	63.06	58.56	63.96	58.56
	max	76.58	81.08	77.48	85.59	82.88
NCI1	min	66.91	66.67	67.15	65.94	67.15
	max	74.21	74.69	74.69	71.29	74.7
NCI109	min	66.02	64.8	67.23	65.78	67.48
	max	72.09	74.51	74.02	73.54	75.97
IMDB-B	min	65.0	64.0	61.0	69.0	72.0
	max	79.0	72.0	81.0	82.0	79.0
IMDB-M	min	39.33	39.33	32.67	42.67	40.0
	max	50.0	49.33	56.67	48.0	54.0

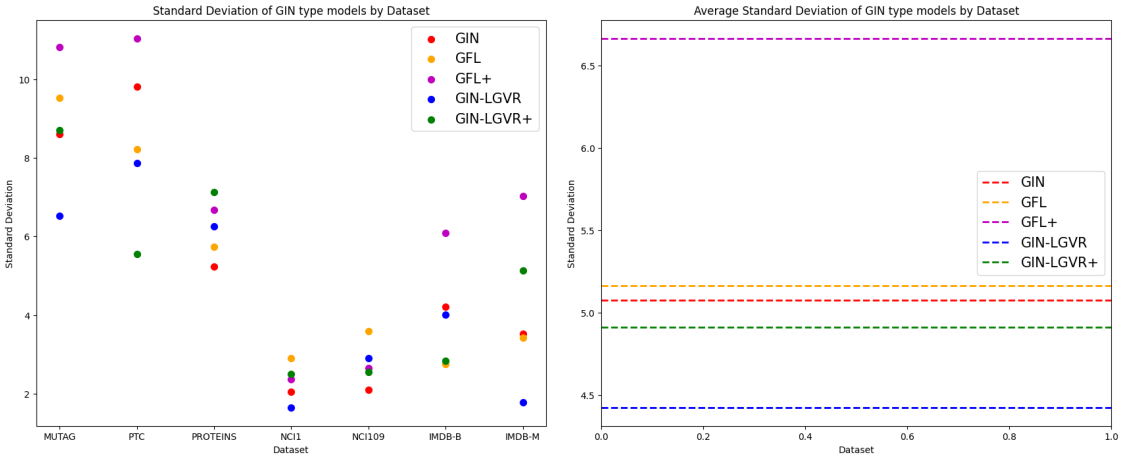

Figure 7: Standard deviations of GIN type models: GIN, GFL, GFL$^+$, GIN-LVGR, and GIN-LVGR$^+$. (Left) Each point represents the standard deviation of 10-fold performances (Table 1) for each model and dataset. (Right) The horizontal dashed line represents the average of the overall standard deviation for each model.

The standard deviation results for $\{P_k\}_{k=1,\ldots,10}$ for each model and dataset are summarized in Figure 7. Here, we find that GIN-LVGR and GIN-LVGR$^+$ show relatively low standard deviations compared to other models. More specifically, each model shows the following average of the overall standard deviation: 4.42 (for GIN-LVGR), 4.91 (for GIN-LVGR$^+$), 5.08 (for GIN), 5.16 (for GFL), and 6.66 (for GFL$^+$). Since graphs are composed of nodes and edges, the importance of nodes and edges can vary depending on the data. From this perspective, we claim that these results stem from the characteristics of GIN-LVGR and GIN-LVGR$^+$, which leverage both nodes and edges. In other words, this result implies that GIN-LVGR and GIN-LVGR$^+$, which leverage both nodes and edges to reflect various features in graph representations, would have enabled more stable learning compared to other models (GIN, GFL, and GFL$^+$) that rely solely on nodes.

Table 3: Graph classification results (with Equation 10)

Model / Dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	IMDB-B	IMDB-M
GIN	88.33	70.29	74.5	72.31	71.97	77.3	50.0
GFL	92.22	71.47	76.84	73.23	72.88	75.2	49.6
GFL$^+$	91.11	69.41	73.96	73.28	72.62	77.4	51.0
GIN-LGVR	**92.78**	70.29	**77.3**	72.34	71.77	76.7	50.07
GIN-LGVR$^+$	91.11	**72.64**	74.68	**73.94**	**73.45**	**77.8**	**52.0**

Other Performance Metric: Mean of the maximum accuracy per fold. As mentioned in Section 7.4.1, since there was no separate test set in graph classification benchmarks, we performed 10-fold cross-validation and used Equation 9 as our performance metric. The reason why we use this metric is that it is best suited to evaluate model performance through cross-validation. Specifically, the key aspect of cross-validation is to provide performance estimation and hyperparameters (for example, learning rate, decay rate, epoch, etc) without a test set. However, to reduce the hyperparameter search space, the epoch is typically set to a fixed value, which may result in reduced reliability of performance estimation due to inappropriate epoch settings. To address this, we used Equation 9 as our final performance metric. Equation 9 calculates performance estimation for all epochs, preventing a decrease in performance reliability due to inappropriate epoch settings and providing the optimal epoch, which is consistent with the purpose of cross-validation.

However, in addition to Equation 9, the average of the maximum validation performance for each fold is also frequently used:

$$\frac{1}{10} \cdot \sum_{k=1}^{10} (\max_{i \in \{1,\dots,t\}} P_{k,i}), \tag{10}$$

where t is the total epochs and $P_{k,i}$ is the k-fold validation accuracy at i-th epoch. Therefore, we also provide the classification results measured by Equation 10 in Table 3.

Note that there is a significant difference between the values of Table 1 with Equation 9 and those of Table 3 with Equation 10. Moreover, there is a slight variation in the rankings between models for some datasets. In particular, GFL$^+$ showed a significant ranking improvement for some datasets (MUTAG, IMDB-B, IMDB-M, for example) when measured by Equation 10, due to its high variations as can be seen in Table 2 and Figure 7. However, there is no change in the model that shows the best performance for each dataset. In other words, even when using Equation 10 as the performance metric, we confirm that our models, GIN-LVGR and GIN-LVGR$^+$, still show the best performance for all datasets.

7.5 Graph Regression Results

We evaluate our models on the QM9 dataset, which involves predicting 12 numeric quantities for a given molecular graph. The dataset is split into 80% train, 10% validation, and 10% test. Finally, we use the same network from the classification experiments and compare the performances of models by training a single network to predict all 12 quantities simultaneously.

Table 4 presents the test mean absolute error for both our models (GIN-LGVR, GIN-LVGR$^+$, PPGN-LVGR$^+$) and comparison models (GIN, GFL, GFL$^+$, PPGN). These results

Table 4: Graph regression results on the QM9 dataset (with mean absolute error).

	$\mathcal{C}$=GIN					$\mathcal{C}$=PPGN	
Target / Model	GIN	GFL	GFL$^+$	**GIN-LGVR**	**GIN-LGVR$^+$**	PPGN	**PPGN-LGVR$^+$**
μ	0.729	0.917	**0.655**	1.058	0.661	0.231	**0.09**
α	3.435	4.818	2.985	3.542	**2.76**	0.382	**0.19**
ϵ_{homo}	0.00628	0.01595	**0.00581**	0.01528	0.00683	0.00276	**0.00178**
ϵ_{lumo}	0.00957	0.02356	0.00987	0.02952	**0.00911**	0.00287	**0.0019**
Δ_ϵ	0.01023	0.01737	0.01003	0.03253	**0.00931**	0.00406	**0.00253**
$\langle R^2 \rangle$	124.05	175.23	121.33	174.96	**113.1**	16.07	**3.47**
$ZPVE$	0.00719	0.02354	0.00393	0.00534	**0.0026**	0.00064	**0.00032**
U_0	17.477	18.938	18.121	17.458	**15.705**	0.234	**0.216**
U	17.477	18.881	18.12	17.807	**15.706**	0.234	**0.215**
H	17.476	18.923	18.118	17.626	**15.705**	0.229	**0.217**
G	17.477	18.889	18.121	17.72	**15.708**	0.238	**0.216**
C_v	1.361	3.515	1.258	1.993	**1.163**	0.184	**0.081**

provide several interesting findings. First, GIN-LVGR$^+$ and PPGN-LVGR$^+$, show the best performances for almost all 12 quantities, which empirically demonstrates the superior performance of our model for regression task as well. This validates **Claim 1**. Next, regardless of filtration types, both GFL and GIN-LGVR, which do not use pooling information, show a decrease in performance compared to GIN. As mentioned in the classification experiment, we remark that such decreases in performances stem from the fact that the initial node features of molecule graphs in QM9 are highly informative (Appendix D.1.3). However, we found that GIN-LVGR$^+$ outperforms GIN and GIN-LVGR for all 12 quantities. This empirically validates the effectiveness of our integration framework technique (Section 6.1; Corollary 16), which validates **Claim 3**. Finally, to verify **Claim 2**, we compare our models with node filtration methodologies for the GIN type and show that GIN-LVGR$^+$ outperforms node filtration-based models (GFL, GFL$^+$) on almost all tasks (10 out of 12 tasks). Through these results, we empirically demonstrate the superiority of our edge filtration-based approach over node filtration-based one for regression tasks as well (**Claim 2**).

8. Conclusion

We propose a novel edge filtration-based persistence diagram, named Topological Edge Diagram (TED), which can incorporate topological information into any message passing graph neural networks. We mathematically prove that TED can preserve the node embedding information as well as contain additional topological information. We further prove that TED can even strictly increase the expressivity of the WL test. To implement our theoretical foundation, we propose a novel neural network-based algorithm, called Line Graph Vietoris-Rips (LGVR) Persistence Diagram and prove that LGVR has the same expressivity as TED. To evaluate the performance of LGVR, we propose two model frameworks ($\mathcal{C}$-LVGR and $\mathcal{C}$-LVGR$^+$) that can be applied to any message passing GNNs $\mathcal{C}$, and prove that they are strictly more powerful than the WL test. Through our model frameworks, we empirically demonstrate the superior performances of our approach. The downside is that since the ENC only extracts node coloring information, applying the $\mathcal{C}$-LVGR would result in losing all non-node coloring information. Although we address this problem by introducing the integration technique in Section 6.1, we believe that investigating ENC to prevent this information loss would be an interesting future work.

Acknowledgments

The authors would like to thank the Action Editor and anonymous reviewers for their careful reading of the paper and useful suggestions to help improve the exposition of the paper.

Appendix A. Purpose of extension to $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$ in Algorithm 1

In this section, we will explain the reason why we extend a colored line graph $(L_G, \mathcal{C}^\phi)$ to a colored complete line graph $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$. In short, it is done to extract rich topological information about graph G by performing binary node classification on the line graph to distinguish actual edges and virtual edges of G.

Suppose that we perform node classification on $(L_G, \mathcal{C}^\phi)$ instead of $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$ with the loss $\mathcal{L}_{\text{LGVR}}$ in Algorithm 1. Then all the nodes will end up being trained to have a value of 0 since all the nodes in L_G correspond to actual edges in G. This means that all edge values of G will be sufficiently close to 0 (that is, they are all similar), which can act as a bottleneck in obtaining rich topological information, as the persistence of all homology classes of $\text{ph}^i_{\text{VR}}(G, ef^{\mathcal{C}})$ becomes excessively short. To avoid this issue, we add virtual edges to G using the complete graph K_G and perform the node classification on $(L_{K_G}, \widetilde{\mathcal{C}^\phi})$ instead of $(L_G, \mathcal{C}^\phi)$.

Appendix B. A remark on the range $[0, 0.5]$ of ε for $\{\text{VR}_1^\varepsilon(V(G), A_G^{\mathcal{C}})\}_{\varepsilon\in[0,0.5]}$

In this section, we explain why the range of ε is set to $[0, 0.5]$ in the definition of $\text{LGVR}(G, \mathcal{C})$: $(\text{ph}^0(\{\text{VR}_k^\varepsilon(V(G),\ A_G^{\mathcal{C}})\}_{\varepsilon\in[0,0.5]}), \ldots, \text{ph}^k(\{\text{VR}_k^\varepsilon(V(G), A_G^{\mathcal{C}})\}_{\varepsilon\in[0,0.5]}))$. Since we assign scalar values to edges through $A_G^{\mathcal{C}}$ and use them as distances between nodes, the range of ε determines which edges are significant and used to extract the topological information of the graph. Hence, to set an appropriate range for ε, it is necessary to interpret the values of the distance matrix $A_G^{\mathcal{C}}$, which contains information about the edges. The values in $A_G^{\mathcal{C}}$ range from 0 to 1, and meaningful (or actual) edges have a value close to 0, while meaningless (or virtual) edges have a value close to 1. Therefore, we determine that meaningful edges have values of 0.5 or less, so we use this to extract the topological information of the graph by setting the range of ε to $[0, 0.5]$.

Appendix C. Our Set Encoder

To extract the representation of LGVR Diagram, we use both Deep Set (Zaheer et al. (2017)) and Set Transformer (Lee et al. (2019)). Deep Set has a simple architecture and has been theoretically proven to be a universal approximator of set functions under a countable universe ((Zaheer et al., 2017, Theorem 2)). However, Deep Set operates independently on each element of the set, which leads to the disadvantage of discarding all information related to interactions between elements. Inspired by (Vaswani et al. (2017)), Lee *et al.*Lee et al. (2019) proposed the Set Transformer, to reflect higher-order interactions between elements in a set.

Our set encoder is constructed by combining these two models: Deep Set and Set Transformer. Briefly speaking, we first apply Deep Set and then pass the results to Set

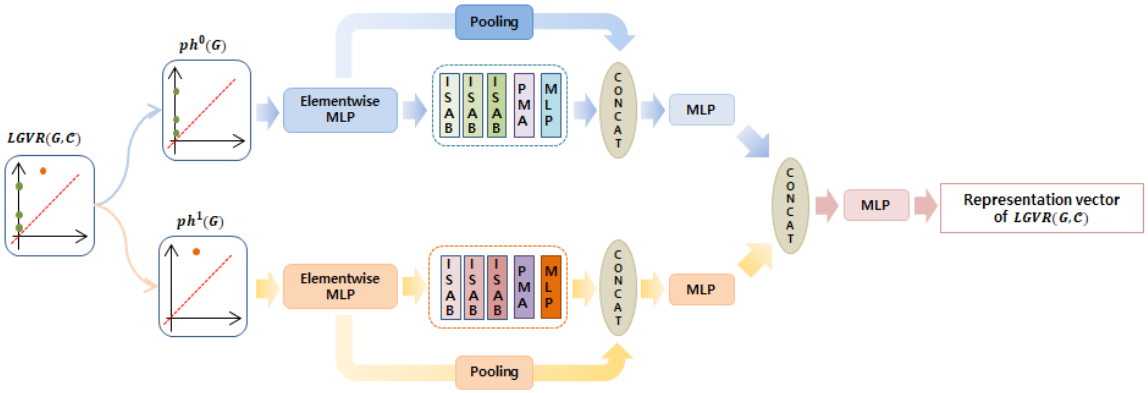

Figure 8: An architecture of our set encoder. Note that Elementwise MLP means performing MLP on each element in the multi-set, and ISAB and PMA respectively denote Induced Set Attention Block and Pooling by Multihead Attention (Lee et al. (2019)). After passing the 0-th and 1-st persistence diagrams through Elementwise MLPs, their outputs are not only used as inputs for Set Transformers (Lee et al. (2019)) but also used to extract the representation vectors of Deep Set (Zaheer et al. (2017)) through pooling (for example, sum). Through this skip connection scheme, our set encoder takes on a form that combines Deep Set and Set Transformer.

Transformer. We expect to reflect better element-wise interactions within the set by passing higher-quality feature vectors through the Deep Set to Set Transformer.

Here we will describe our set encoder architecture in detail. For notational convenience, let

$$\mathrm{ph}^i(G) := \mathrm{ph}^i(\{\mathrm{VR}_1^\varepsilon(V(G),\ A_G^{\mathcal{C}})\}_{\varepsilon\in[0,0.5]})$$

for $i = 0, 1$. Since multi-sets $ph^0(G)$ and $ph^1(G)$ are independent of each other, we construct each set encoder and concatenate them. First, we construct the element-wise encoding part of Deep Set using multi-layer perceptrons MLP_0^0 and MLP_1^0 with two layers: for $i = 0, 1$,

$$\mathrm{DE}_i(G) := \mathrm{MLP}_i^0(\mathrm{ph}^i(G)),$$

where

$$\mathrm{ph}^i(G) = \{\{x_1, \ldots, x_g\}\}, \text{ and}$$
$$\mathrm{MLP}_i(\mathrm{ph}^i(G)) = \mathrm{MLP}_i(\{\{x_1, \ldots, x_g\}\}) := \{\{\mathrm{MLP}_i(x_1), \ldots, \mathrm{MLP}_i(x_g)\}\}$$

Next, we construct Set Transformers ST_0 and ST_1 taking multi-sets $\mathrm{DE}_0(G)$ and $\mathrm{DE}_1(G)$ as their inputs. For each $i = 0, 1$, the encoder of each ST_i is composed of three layers of Induced Set Attention Block (ISAB) ISAB_i^j, $j = 0, 1, 2$, and the decoder is composed of a single layer of Pooling by Multi-head Attention (PMA) PMA_i and fully connected layer FC_i: for each $i = 0, 1$,

$$(\text{Encoder of } \mathrm{ST}_i)(\mathrm{DE}_i(G)) := \mathrm{ISAB}_i^2(\mathrm{ISAB}_i^1(\mathrm{ISAB}_i^0(\mathrm{DE}_i(G)))),$$
$$\mathrm{ST}_i(\mathrm{DE}_i(G)) := \mathrm{FC}_i(\mathrm{PMA}_i(\text{Encoder of } \mathrm{ST}_i)(\mathrm{DE}_i(G))).$$

Table 5: Details of datasets for graph classification task. Note that features (5th column) refer to the number of classes of the initial node feature. When the feature is NA, it indicates that there is no initial node feature so the degree is used as the initial node feature. Moreover, #classes (6th column) means the number of classes of labels.

Dataset	#graphs	avg. #nodes	avg. #edges	features	#classes
MUTAG	188	17.9	19.79	7	2
PTC	344	14.29	14.69	22	2
PROTEINS	1113	39.06	72.82	3	2
NCI1	4110	29.87	32.3	37	2
NCI109	4127	4	29.6	38	2
IMDB-B	1000	19.77	96.53	NA	2
IMDB-M	1500	13	65.94	NA	3

Now, for each $i = 0, 1$, we define a new vector $\mathrm{DST}_i(G)$ by concatenating the result of Deep Set, $\sum_{x \in \mathrm{DE}_i(G)} x$, with the result of Set Transformer, $\mathrm{ST}_i(\mathrm{DE}_i(G))$, and then passing it through an additional multi-layer perceptrons MLP_i^1: for each $i = 0, 1$,

$$\mathrm{DST}_i(G) := \mathrm{MLP}_i^1([\sum_{x \in \mathrm{DE}_i(G)} x \mid \mathrm{ST}_i(\mathrm{DE}_i(G))]).$$

Finally, we extract the embedding vector for the LGVR diagram by concatenating $\mathrm{DST}_0(G)$ and $\mathrm{DST}_1(G)$, and passing it through the last multi-layer perceptrons MLP^2:

$$\mathrm{SetEncoder}(G) := \mathrm{MLP}^2([\mathrm{DST}_0(G) \mid \mathrm{DST}_1(G)]).$$

By combining the Deep Set and Set Transformer as described above, we construct our set encoder. Note that Figure 8 depicts the overall architecture of our set encoder.

Appendix D. Experimental Details

In this section, we explain some details of datasets and hyperparameter settings used in the experiments (Section 7).

D.1 Details of Datasets

We give detailed descriptions of datasets used in our experiments: MUTAG, PTC, PROTEINS, NCI1, NCI109, IMDB-B, IMDB-M, and QM9.

D.1.1 Bioinformatics datasets

In bioinformatic graphs, the nodes have categorical input features, and we test 5 datasets in our experiments: MUTAG, PTC, PROTEINS, NCI1, and NCI109. MUTAG is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds. PTC is consisted of 344 chemical compounds that reports the carcinogenicity for male and female rats. Moreover, PROTEINS is a dataset whose nodes are secondary structure elements and there is an edge between two nodes if they are neighborhoods in the amino-acid sequence or in 3D space. Finally, NCI1 and NCI109, made publicly available by the National Cancer Institute, are

Table 6: A brief description of each regression target on QM9 dataset.

Target	Description
μ	Dipole moment
α	Isotropic polarizability
ϵ_{homo}	Highest occupied molecular orbital energy
ϵ_{lumo}	Lowest unoccupied molecular orbital energy
Δ_ϵ	Gap between ϵ_{homo} and ϵ_{lumo}
$\langle R^2 \rangle$	Electronic spatial extent
$ZPVE$	Zero point vibrational energy
U_0	Internal energy at 0K
U	Internal energy at 298.15K
H	Enthalpy at 298.15K
G	Free energy at 298.15K
C_v	Heat capacity at 298.15K

two subsets of balanced datasets of chemical compounds screened for ability to suppress or inhibit the growth of a panel of human tumor cell lines. Statistics for these datasets are summarized in Table 5.

D.1.2 Social network datasets

In social network graphs, no features are provided for nodes so we set all node features to be the same (thus, node features are uninformative). We test two social network datasets in our experiments: IMDB-B (binary) and IMDB-M (multi-class). Both datasets are movie collaboration datasets. Each graph provides information about actors/actresses and genres of different movies on IMDB. For each graph, nodes correspond to actors/actresses and an edge is drawn between actors/actresses who appear in the same movie. Each graph corresponds to a specific genre label, and the task is to classify which genre a given graph belongs to. IMDB-B consists of collaboration graphs on *Action* and *Romance* genres, and IMDB-M is a multi-class version of IMDB-B derived from *Comedy*, *Romance*, and *Sci-Fi* genres. Statistics for these datasets are summarized in Table 5.

D.1.3 QM9 dataset

QM9 is the dataset consisting of 134k small organic molecules of varying sizes from 4 to 29 atoms. Each graph is represented by an adjacency matrix and input node features, which can be obtained from the pytorch-geometric library Fey and Lenssen (2019). Note that the input node features are of dimension 18, which contain information about the distance between atoms, categorical data on the edges, etc.

This dataset has three characteristics: (1) nodes correspond to atoms and edges correspond to close atom pairs, (2) edges are purely distance-based, and (3) it only provides the coordinates of atoms and atomic numbers as node features. Moreover, the number of classes of initial node features is 18, and the task is to predict 12 real-valued physical quantities for each molecule graph. We provide a brief description of each regression target in Table 6. A more detailed explanation of QM9 dataset can be found in Pinheiro et al. (2020).

Table 7: GIN Type: Details of hyperparameter settings used in our experiments for graph classification and regression tasks.

	GIN				GIN-LVGR				GIN-LVGR$^+$				GFL				GFL$^+$			
Dataset	LR	DR	BS	Ep	LR	DR	BS	Ep	LR	DR	BS	Ep	LR	DR	BS	Ep	LR	DR	BS	Ep
MUTAG	10^{-3}	0.75	5	500	$5*10^{-4}$	0.75	5	500	$5*10^{-4}$	0.75	5	500	$5*10^{-4}$	0.75	5	500	10^{-3}	0.75	5	500
PTC	10^{-3}	0.75	5	400	10^{-3}	0.75	5	400	$5*10^{-4}$	0.75	5	400	$5*10^{-4}$	0.75	5	400	$5*10^{-4}$	0.75	5	400
PROTEINS	10^{-3}	0.75	5	400	10^{-3}	0.75	5	400	10^{-3}	0.75	5	400	$5*10^{-4}$	0.75	5	400	10^{-3}	0.75	5	400
NCI1	$5*10^{-4}$	0.75	5	200	$5*10^{-4}$	0.75	5	200	$5*10^{-4}$	0.75	5	200	10^{-3}	0.75	5	200	10^{-3}	0.75	5	200
NCI109	$5*10^{-4}$	0.75	5	250	$5*10^{-4}$	0.75	5	250	10^{-3}	0.75	5	250	$5*10^{-4}$	0.75	5	250	$5*10^{-4}$	0.75	5	250
IMDB-B	10^{-3}	0.75	5	150	10^{-4}	0.75	5	150	10^{-3}	0.75	5	150	10^{-3}	0.75	5	150	10^{-3}	0.75	5	150
IMDB-M	10^{-3}	0.75	5	150	10^{-4}	0.75	5	150	10^{-3}	0.75	5	150	10^{-3}	0.75	5	150	10^{-3}	0.75	5	150
QM9	$5*10^{-3}$	0.8	64	300	$5*10^{-5}$	0.8	64	300	10^{-3}	0.8	64	300	$5*10^{-4}$	0.8	64	300	$5*10^{-3}$	0.8	64	300

Table 8: PPGN Type: Details of hyperparameter settings used in our experiments for graph classification and regression tasks.

	PPGN				PPGN-LVGR$^+$			
Dataset	LR	DR	BS	Ep	LR	DR	BS	Ep
MUTAG	10^{-4}	1.0	5	500	10^{-4}	1.0	5	500
PTC	10^{-4}	1.0	5	400	$5*10^{-5}$	1.0	5	400
PROTEINS	10^{-3}	0.5	5	400	$5*10^{-5}$	0.5	5	400
NCI1	10^{-4}	0.75	5	200	$5*10^{-5}$	0.75	5	200
NCI109	10^{-4}	0.75	5	250	$5*10^{-5}$	0.75	5	250
IMDB-B	$5*10^{-5}$	0.75	5	150	$5*10^{-5}$	0.75	5	150
IMDB-M	10^{-4}	0.75	5	150	$5*10^{-5}$	0.75	5	150
QM9	10^{-4}	0.8	64	300	10^{-4}	0.8	64	300

D.2 Details of Hyperparameter Settings

The hyperparameters we tune for each dataset are: (1) learning rate (LR), (2) decay rate (DR), (3) batch size (BS), and (4) the number of epochs (Ep). For classification, the search space for each hyperparameter is as follows: when $\mathcal{C}$ is GIN, the learning rate is set to $\{10^{-3}, 5 \cdot 10^{-4}, 10^{-4}\}$ and when $\mathcal{C}$ is PPGN, they are set to $\{10^{-4}, 5 \cdot 10^{-5}\}$. Moreover, for regression, in the case of PPGN, it maintains the same setting as classification, whereas in the case of GIN, the learning rate is set to $\{5 \cdot 10^{-3}, 10^{-3}, 5 \cdot 10^{-4}, 10^{-4}, 5 \cdot 10^{-5}\}$. In both cases, we use the Adam optimizer (Kingma and Ba (2014)) and decay the learning rate by $\{0.5, 0.75, 1.0\}$ every 20 epochs.

Among the search space for hyperparameter, we summarize the hyperparameter settings for each dataset used in our experiments in Table 7 and 8.

References

László Babai and Ludik Kucera. Canonical labelling of graphs in linear average time. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 39–46. IEEE, 1979.

Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gauzere, Sébastien Adam, and Paul Honeine. Bridging the gap between spectral and spatial domains in graph neural networks. *arXiv preprint arXiv:2003.11702*, 2020.

Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. *arXiv preprint arXiv:2106.04319*, 2021.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

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 *International Conference on Machine Learning*, pages 1204–1215. PMLR, 2021.

Mathieu Carriere, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. A general neural network architecture for persistence diagrams and graph classification. 2019.

Fengwen Chen, Shirui Pan, Jing Jiang, Huan Huo, and Guodong Long. Dagcn: dual attention graph convolutional networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Brendan L Douglas. The weisfeiler-lehman method and graph isomorphism testing. *arXiv preprint arXiv:1101.5211*, 2011.

Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2022.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

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

Martin Grohe and Martin Otto. Pebble games and linear equations. *The Journal of Symbolic Logic*, 80(3):797–844, 2015.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. *arXiv preprint arXiv:1707.04041*, 2017.

Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *International Conference on Machine Learning*, pages 4314–4323. PMLR, 2020.

Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. *arXiv preprint arXiv:2102.07835*, 2021.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Kurt Hornik. Multilayer feed-forward networks are universal approximators. *Artificial neural networks: Approximation and learning theory*, 1992.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *arXiv preprint arXiv:1905.11136*, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

Steve Y Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209. American Mathematical Soc., 2017.

Gabriel A Pinheiro, Johnatan Mucelini, Marinalva D Soares, Ronaldo C Prati, Juarez LF Da Silva, and Marcos G Quiles. Machine learning prediction of nine molecular properties based on the smiles representation of the qm9 quantum-chemistry dataset. *The Journal of Physical Chemistry A*, 124(47):9854–9866, 2020.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1): 1–7, 2014.

Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning*, pages 5448–5458. PMLR, 2019.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

BJ Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction, nauchno–technicheskaja informatsia, 9 (1968), 12–16, 1968.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

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

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018b.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. *arXiv preprint arXiv:1904.12189*, 2019.