UNVEILING PATTERNS IN DATA: HARNESSING COMPUTATIONAL TOPOLOGY IN MACHINE LEARNING

by

Soham Mukherjee

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

May 2024

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Tamal Krishna Dey, Chair

Department of Computer Science

Dr. Saugata Basu

Department of Mathematics

Dr. Elena Grigorescu

Department of Computer Science

Dr. Aniket Bera

Department of Computer Science

Approved by:

Dr. Kihong Park

To my *baba*, Srijit Mukherjee.

ACKNOWLEDGMENTS

In the Fall of 2013, sitting in an undergraduate class on Introduction to C, I realized that I did not understand how to write an algorithm. Years passed, and I graduated with a Bachelor's degree in Electronics Engineering but the zeal in me to come up with algorithms never died. I decided to apply for the Ph.D. programme in Computer Science at Ohio State University.

Finishing the PhD program and becoming a competent researcher is undeniably a tough feat. However, I feel incredibly lucky to have been supported by an amazing group of people whose constant help and guidance made this whole journey possible. I want to express my deep gratitude to everyone who has had a significant impact on my academic path.

Firstly, I want to thank my advisor, Dr. Tamal K. Dey. His expertise, patience, and continuous encouragement have been crucial in shaping my PhD journey. I'm profoundly grateful for his invaluable insights and unwavering support. Even during my toughest moments, Dr. Dey's belief in my abilities kept me motivated. During this arduous journey, he believed in me at times when I even doubted myself. I know for sure that he will continue to inspire me in my future career.

To my family, especially my parents, *Srijit Mukherjee*, who didn't live long enough to see this finished thesis, *Banani Mukherjee* and my wife *Aishanee Sinha*, I am deeply thankful for their unwavering love, understanding, and support throughout my education. It was my father because of whom I took a liking towards mathematics, physical sciences, and above all, engineering. I distinctly remember doing grade 1 mathematics problems together and the fascinating electrical tools that appeared in front of me from the basement. Probably those were enough to persuade me to choose engineering as a major. My mother imparted the virtues of resilience, perfectionism, and determination to me through her guidance. I thank Aishanee for listening to all my rants and frustrations over late-night phone calls despite the 9:30-hour time difference. Thank you for being there at my best and at my worst. Their constant backing and sacrifices have been the foundation of my success, and I'll always be indebted to them.

My heartfelt thanks go to my roommates, Avishek Banerjee, Debanjan Nandi, and Vishaal Dey at OSU, and Arunashish Datta and Shovan Maity at Purdue, who supported me selflessly during the six and a half years of my PhD program. Their help got me through some of the hardest times. Debanjan Da, your guidance during the initial days, Vishaal, your delicious food, Avishek, your humor, and your coffee were invaluable to my research. Arunashish, your amazing and delicious food provided comfort, especially during the tough times of the COVID-19 pandemic. Shovan Da, quick trips across the USA with you gave me the necessary breather to continue along this arduous path. Shreyas Samaga, my half roommate, thank you for all the chais during our coding session.

I'm also immensely grateful to my thesis committee members, Dr. Saugata Basu, Dr. Elena Grigorescu, and Dr. Aniket Bera, for their time, expertise, and invaluable feedback. Their insightful comments significantly improved the quality of my work.

Thanks to my collaborators, colleagues, and labmates Sayan Mandal, Ryan Slechta, Tao Hou, Cheng Xin, Shreyas Samaga, Simon Zhang, Dr. Mustafa Hajij, Dr. Jayajit Das, and many others, for their contributions and collaboration. Their expertise, brainstorming sessions, and willingness to share resources greatly enriched my research project. I express my gratitude to Prof. Yusu Wang for her discussions, advice, and insights during the OSU days.

This thesis spans two universities OSU and Purdue. I'd like to thank some of my closest friends Rupam Kundu, Dhrubojyoti Roy, Semanti Mukhopadhyay, Soham Mandal, Barsha Bhowal, Preeti Mukherjee, Indulekha Guha, Arnab Mitra, Yudhajit Ray, Jie Yang, Suranjan Paul for their unwavering emotional support, motivation, and valuable discussions. Your friendship made this journey both memorable and enjoyable.

I want to thank my undergraduate advisor Prof. Mrinal Kanti Naskar for introducing me to research and motivating me to pursue higher studies.

Lastly, my deepest appreciation goes to all individuals who directly or indirectly contributed to this thesis. Your involvement and support played a crucial role in its successful completion.

I acknowledge that this thesis wouldn't have been possible without the collective contributions of all those mentioned above. Thank you all for believing in me and for being an integral part of my academic journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Topological Data Analysis (TDA) with its roots embedded in the field of algebraic topology has successfully found its applications in computational biology, drug discovery, machine learning and in many diverse areas of science. One of its cornerstones, persistent homology, captures topological features latent in the data. Recent progress in TDA allows us to integrate these finer topological features into traditional machine learning and deep learning pipelines. However, the utilization of topological methods within a conventional deep learning framework remains relatively uncharted. This thesis presents four scenarios where computational topology tools are employed to advance machine learning.

The first one involves integrating persistent homology to explore high-dimensional cytometry data. The second one incorporates Extended persistence in a supervised graph classification framework and demonstrates leveraging TDA in cases where data naturally aligns with higher-order elements by extending graph neural networks to higher-order networks, applied specifically in non-manifold mesh classification. The third and fourth scenarios delve into enhancing graph neural networks through multiparameter persistence.

1. INTRODUCTION

Persistent homology is a mathematical technique in the field of algebraic topology used to analyze the shape and structure of datasets, particularly in the context of topological data analysis (TDA). It aims to extract and characterize the essential topological features that persist across different spatial scales within a dataset. At its core, persistent homology identifies and quantifies the topological features, such as connected components, holes, loops, and voids, present in a dataset. It does so by summarizing the evolution of these features in a robust, scale-invariant manner through *persistence diagrams* as the dataset is progressively filtered or transformed. Persistent homology has found diverse applications across various fields such as medical imaging, material science, gene expression analysis, robotics, image and signal processing, neuroscience due to its ability to capture topological features in complex data [1–10]. Consequently, progress has been made to include persistence diagrams for learning tasks [11–17]. However, the exploration of Topological Data Analysis (TDA) within the context of data naturally supported on higher-dimensional complexes, or the active integration of multiparameter persistence in a deep learning pipeline, remains largely unexplored. This dissertation aims to bridge the gap and delves into the realm of synergy between computation topology and machine learning.

Topological signatures given by persistence are stable, global, and scale invariant and show resilience to local perturbations [18]. It is this property of persistent homology that motivates us to use TDA in the analysis of high dimensional point cloud data coming from cytometry experiments that is difficult to interpret manually. Persistent homology has been applied to characterize shapes and shape-function relationships in a wide variety of biological systems including skin pattern formation in zebra fish [19], protein structure, gene expression [4], and pattern of neuronal firing in mouse hippocampus [20]. TDA has additionally previously been applied to identify immune parameters associated with transplant complications for patients undergoing allogeneic stem cell transplant using populations of immune cell types assayed via mass cytometry [21]. However, this work did not use persistent homology or expression levels of proteins in their analysis, leaving the shape of cytometry data uncharacterized. Another work focuses on the use of TDA as a data reduction method for single-cell

RNA sequencing data [22], but again does not attempt to characterize how topologies derived from point clouds differ among disparate data sources such as healthy and diseased individuals. We show empirically how TDA captures subtle topological features hidden in data, especially if those features are further masked by data transforms or significant donor-to-donor variations. Persistent homology identifies regions in cytometry datasets of varying density and identifies protruded structures that distinguish patients infected with COVID-19 and healthy controls. Our analysis of the PCDs indeed shows that *birth* time of cycles for the COVID-19 patient is much larger compared to that for the healthy individual indicating the presence of larger length scales in the PCD which is consistent with the presence of an 'elbow' shape in the PCD for the patient.

In Chapter 3, we delve into the integration of extended persistence into a supervised learning framework tailored for graph classification. Extended persistence, a technique derived from topological data analysis, proves instrumental in extracting comprehensive multiscale topological insights from a graph. This encompasses crucial information about connected components and cycles, elegantly represented by persistence barcodes. The global topological details, summarized by four types of bars and their corresponding cycle representatives incorporated into the model through a readout function computed via extended persistence. The entire model is end-to-end differentiable allowing us to learn graph filter functions but capturing more information than previous approaches did [14, 15].

On a parallel track, shape analysis emerges as a fundamental and formidable field with applications in computer vision, computer graphics, and geometric modeling. Conventional graph-based abstractions often fall short of capturing the intricate geometric and topological characteristics inherent in complex shapes. Beyond the scope of extended persistence, the chapter places a spotlight on Topological Deep Learning (TDL), a domain dedicated to developing deep learning models suited for data residing on topological domains like simplicial complexes, cell complexes, and hypergraphs. These domains extend their applicability to various scientific computations. In the realm of shape analysis, TDL presents an exciting frontier for pushing the boundaries of mesh processing.

The real strength of TDL techniques lies in their ability to incorporate complex topological relationships, allowing us to identify shape features that are not captured by traditional

approaches. To further enhance the landscape, we introduce a novel architecture called the Combinatorial Complex Isomorphism Network (CCIN), leveraging higher-order networks. We demonstrate its efficacy on mesh-related tasks, with a special emphasis on the processing of 'non-manifold' meshes. The experimental results on non-manifold meshes show the effectiveness of CCIN, showcasing its high accuracy in handling these intricate structures.

In recent years, efforts have been made to successfully integrate persistent homology with machine learning models [12, 14, 15, 23–35]. In practical scenarios, data frequently requires a general bivariate filtration function $\mathcal{X} \to \mathbb{R}^2$ as opposed to a scalar function. The aim is to capture its topological information through 2-parameter persistence modules. However, delving into the realm of 2-parameter persistence modules reveals a significantly more intricate structure compared to their 1-parameter counterparts. In the 1-parameter case, the modules are completely characterized by what is called *barcode* or *persistence diagram* [36, 37]. Unfortunately, there is no such discrete complete invariant that can summarize 2-parameter persistence modules completely [38]. Given this limitation, building a useful vector representation from multiparameter persistence modules while capturing as much topological information as possible for machine learning models becomes an important but challenging problem. We propose to study 2-parameter persistence modules induced by bifiltration functions in Chapter 4. To incorporate these topological features revealed by the bifiltration functions into machine learning models, We introduce *Generalized Rank Invariant Landscape* (Gril), a new vector representation encoding richer information beyond fibered barcodes for 2-parameter persistence modules, based on the idea of *generalized rank invariant* [39] and its computation by zigzag persistence [40]. The construction of Gril can be viewed as a generalization of persistence landscape [16, 24], hence has more discriminating power. Moreover, we propose an efficient algorithm to compute (Gril), demonstrate its use on synthetic and benchmark graph datasets, and compare the results with previous vector representations of 1-parameter and 2-parameter persistence modules. Specifically, we present results indicating that GNNs improve when augmented with Gril features for graph classification tasks.

Recently, There have been many works on vectorization of a 2-parameter persistence module using incomplete invariants such as *rank invariant* or equivalently *sliced barcodes*, multiparameter persistence images [35], multiparameter persistence landscapes [24], multi-

parameter persistence kernel [23], vectorization of signed barcodes [41] or *generalized rank invariant* [42]. The authors, in all these works, show that 2-parameter persistence methods perform better than 1-parameter persistence methods on many graph and time-series datasets, suggesting that an $\mathbb{R}^2$-valued filter function, indeed, captures richer topological information than a scalar filter function. However, in all these works, the bifiltration function is fixed *a-priori*. Similar to 1-parameter filtration learning, learning the filter function, rather than it, can prove to be more informative and beneficial for the task at hand. In Chapter 5 we make GRIL differentiable and leverage its power in an end-to-end pipeline to predict the biological activity of synthesized drug molecules. Our differentiable framework allows us to learn the bifiltration function in a data-driven fashion. We empirically demonstrate that the learned function is more robust, emphasizing the necessity of a differentiable end-to-end pipeline for 2-parameter persistence.

Several passages from these chapters are also drawn verbatim from [42, 43].

2. DETERMINING CLINICALLY RELEVANT FEATURES IN CYTOMETRY DATA USING PERSISTENT HOMOLOGY

Cytometry data contain information about the abundance of proteins in single cells and are widely used to determine mechanisms and biomarkers that underlie infectious diseases and cancer. Recent advances in flow and mass cytometry techniques enable measurement of abundances of over 40 proteins in a single cell [44, 45]. Thus, in the space spanned by protein abundance values measured in cytometry experiments, a cytometry dataset is represented by a point cloud composed of thousands of points where each point corresponds to a single cell. Abundances of proteins or chemically modified forms (e.g., phosphorylated forms) of proteins in single immune cells change due to infection of the host by pathogens (e.g., a virus) or due to the presence of tumors which usually result in changes in the 'shape' of point cloud data measured in cytometry experiments [46–48]. Cytometry data analysis techniques commonly rely on Boolean gating and calculation of relative proportions of resulting populations as a method to compare datasets across control/healthy and experimental/diseased conditions. In recent years, state-of-the-art analyses based on sophisticated machine learning algorithms capable of mitigating batch effects, ad hoc gating assumptions, and donor-donor variability have been developed [49, 50]. However, these methods are not designed to quantitatively characterize shape features (e.g., connected clusters, cycles) in high dimensional cytometry datasets that can contain valuable information regarding unique co-dependencies of specific proteins in diseased individuals compared to healthy subjects.

Topological Data Analysis (TDA) aims to capture the underlying shape of a given dataset by describing its topological properties. Unlike geometry, topological features (e.g., the hole in a doughnut) are invariant under continuous deformation such as rotation, bending, twisting but not tearing and gluing. One of the tools by which TDA describes topological features latent in data is persistent homology [51, 52]. For example, for a point cloud data, persistent homology captures the birth and death of topological features (e.g., 'holes') in a dataset after building a scaffold called a simplicial complex out of the input points. This exercise provides details regarding topological features that 'persist' over a range of scale and thus contain information regarding the shape topology at different length scales. Persistent homology

has been applied to characterize shapes and shape-function relationships in a wide variety of biological systems including skin pattern formation in zebra fish [19], protein structure, and pattern of neuronal firing in mouse hippocampus [20]. TDA has additionally previously been applied to identify immune parameters associated with transplant complications for patients undergoing allogenic stem cell transplant using populations of immune cell types assayed via mass cytometry [21]. However, this work did not use persistent homology or expression levels of proteins in their analysis, leaving the shape of cytometry data uncharacterized. Another work focuses on the use of TDA as a data reduction method for single-cell RNA sequencing data [22], but again do not attempt to characterize how topologies derived from point clouds differ among disparate data sources such as healthy and diseased individuals.

The challenges of directly applying current persistence methodologies to cytometry data to characterize distinguishing features between healthy and diseased states are the following:

1. Features that separate healthy from diseased state can pertain to the change in density of points in a region in point cloud data - therefore, the information of local density should be incorporated in persistent homology methods, in particular in the filtration step that brings in sequentially the simplices connecting the points. In commonly used Rips filtration [53], the density of points is not included.

2. There can be shape changes giving a different length scale in the point cloud data, such as formation of an elbow, in a diseased condition.

3. There can be systematic differences between healthy and diseased states across batch effects and donor-donor variations. Topological features should capture these global differences being oblivious to the local variations caused by measurement noise.

We address the above challenges by developing an appropriate filtration function to compute persistence and applying the method to characterize distinguishing features of non-naïve CD8+ T cells between healthy and SARS-CoV-2 infected patients.

Most of the materials in this chapter are based on [43].

2.1 Background

Topological signatures given by persistence are stable, global, scale invariant and show resilience to local perturbations [18]. It is this property of persistent homology that motivates us to use TDA in distinguishing clinically relevant features in flow cytometry data in COVID-19 patients.

2.1.1 Persistent Homology:

Persistent homology builds on an algebraic structure called homology groups graded by its dimension i and denoted by H_{i} . Intuitively, they describe the shape of the data by 'connectivity' at different levels. For example, H_0 describes the number of connected components, H_1 describes the number of holes, and, H_2 describes the number of enclosed voids apparently present in the shape that the dataset represents. Three and higher dimensional homology groups capture analogous higher (≥ 3) dimensional features. A point cloud data (henceforth abbreviated as PCD) itself does not have much of a 'connected structure'. So, a scaffold called a *simplicial complex* is built on top of it. This simplicial complex, in general, is made out of simplices of various dimensions such as vertices, edges, triangles, tetrahedra, and other higher dimensional analogues. Given a growing sequence of such complexes called *filtrations*, a persistence algorithm tracks information regarding the homology groups across this sequence. In our case, these complexes can be restricted only to vertices and edges. With the restriction that both vertices of an edge appear before the edge, we get a nested sequence of graphs

$$G_0 \subset G_1 \subset G_2 \subset \ldots G_n$$

as the filtration. Fig 2.1 shows such a *filtration*.

2.1.2 Persistence Diagram:

Appearance ('birth') and disappearance ('death') of topological features, that is, cycles whose classes constitute the homology groups, can be captured by persistence algorithms [51, 54]. These 'birth' and 'death' events are represented as points in the so-called *persistence*

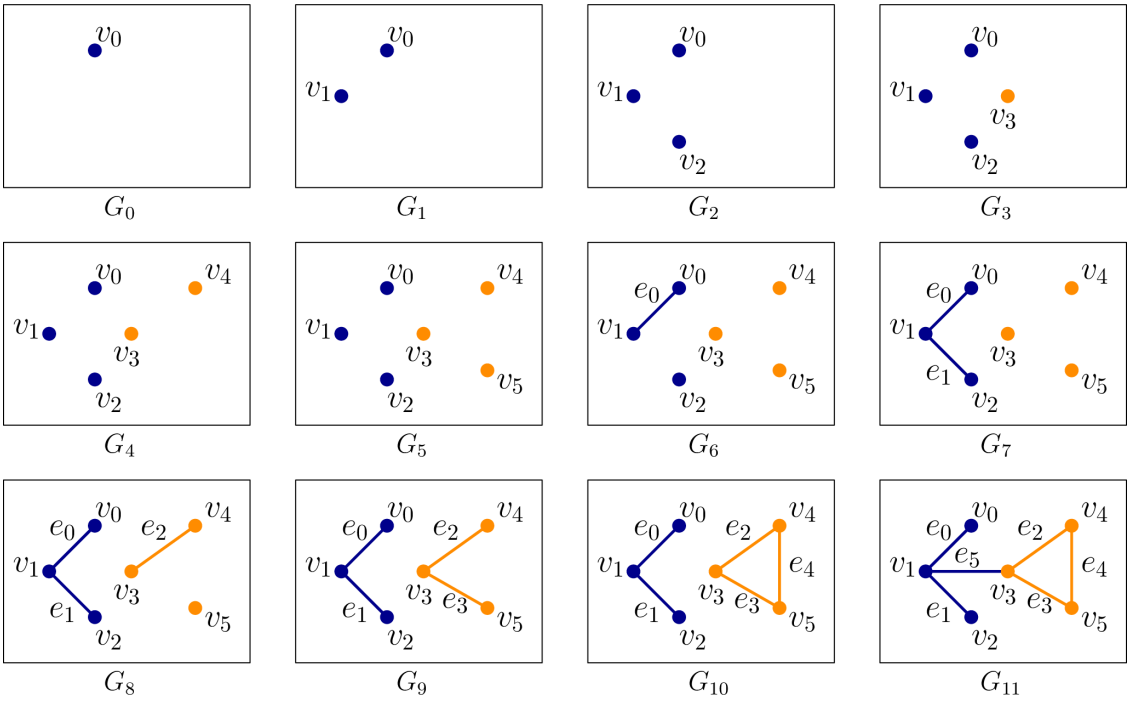

Figure 2.1. An example of *filtration* for a graph. The nested sequence of graphs $G_0 \subset G_1 \subset \dots G_{11}$ forms a filtration of the final graph G_{11}. Each vertex v_i creates a new component in the nested sequence, and edges e_0, e_1, e_2, e_5 merge two components whereas e_4 creates a cycle (yellow).

diagram. If a topological feature is born at filtration step b and dies at step d, we represent this by *persistence pair* (b, d) with persistence $d - b$. The pair (b, d) becomes a point in the persistence diagram with the 'birth' as x-axis and 'death' as y-axis. This 2D plot summarizes topological features latent in the data. In the example-filtration of Fig 2.1, a new component gets 'born' when a vertex v_i appears in the filtration for the first time. When an edge is introduced, one of the two things can happen–either two components are joined, or a cycle is created. In the first case, a 'death' happens for 0-th homology group H_0, and in the second case, a 'birth' happens for the 1-st homology group H_1. For example, when e_0 comes in the filtration (G_6), it merges two components created by v_0 and v_1. By convention, we choose to kill the component that got created later in the filtration and thus we let the component created by v_1 die. We obtain a persistence pairing $(1, 6)$ since edge e_0 at filtration step 6 kills the component created by v_1 at step 1. Similarly, we obtain pairs

$(2,7),(4,8),(5,9)$, and $(3,11)$. These points, tracking the 'birth' and 'death' of components, produce the persistence diagram for the 0-th homology group H_0 and hence we refer to it as the H_0-persistence diagram. Note that the edge e_4 creates a cycle (yellow) that never dies. In such cases, *i.e.* when a topological feature never dies, we pair it with ∞. For the edge e_4, we obtain a persistence pair $(10,\infty)$. But, this feature concerns the 1-st homology group H_1 and thus it becomes a point in the persistence diagram for H_1 which we refer to as H_1-persistence diagram. One way to leverage the above framework for studying a function is to assign function values to vertices and edges and construct a filtration by ordering them according to these assigned values. For such cases the persistence pairs take the form (b,d) where b is the value at which a feature is born and d is the value at which it dies. The function values that induce the filtration (Fig 2.2) are chosen to capture two features of the input PCD–(i) the density variations, and (ii) the *anisotropy* of the features, that is, how elongated it is in a certain direction, henceforth termed as *length scale* 'of the feature' or collectively 'of the data'. In particular, length scales refer to the prominence of protrusions such as 'elbows' in COVID-19 data.

Below we briefly describe how we adapt the above persistence framework for analyzing point cloud data (PCD) representing CD8+ T cells in SARS-CoV-2 infection. Details regarding the approach are provided in Section 2.2.

2.1.3 Computing persistent homology for cytometry datasets:

Our datasets consist of cytometry data for non-naïve CD8+ T cells. Given protein expressions (real values) for d proteins in such a single cell, we can represent it as a d-dimensional point in $\mathbb{R}^d$. Considering a population of single cells, we get a point cloud (PCD) in $\mathbb{R}^d$. Now, we study the shape of this PCD using the persistence framework that we describe above. We compute persistence diagrams for the PCDs generated with protein expressions from different individuals and compare them. It turns out that, for computational purposes, we need a limit on the dimension d for PCD which means we need to choose carefully the proteins that differentiate effectively the subjects of our interest, namely the healthy individuals, COVID-19 patients, and recovered patients. We typically choose 3 (sometimes

2) protein expressions to generate the PCD and call it a PCD in the *P1, P2, P3 space* if it is generated by proteins P1, P2, and P3 respectively.

Flow cytometry data for non-naïve CD8+ T cells in Mathew et al. [46] show generation of CD8+ T cells with larger abundances of the proteins CD38 and HLA-DR (CD38+HLA-DR+ cells) for some COVID-19 patients, forming an 'elbow' in the two dimensional PCD with CD38 and HLA-DR protein expressions (see 2.9 Fig). Moreover, there is an increase in the local density of the points (or single CD8+ T cells) in the 'elbow' region. This suggests that, to study the PCD generated by the protein expressions by persistence framework, we need to choose a filtration that is able to capture such geometric shapes and variations in the local density.

We briefly describe our choice of filtration by considering the example of a point cloud $P \subset \mathbb{R}^2$ shown in Fig 2.2. Mathematical and computational details regarding the filtration are provided in the Section 2.2. We build a filtration according to assigned values to the vertices and edges of a graph connecting the input points. For a vertex p which is a point in the input PCD P, we denote this value $f_v(p)$ (given by Eq 2.1 in Section 2.2). Similarly, we denote the assigned value to an edge e as $f_{\mathrm{e}}(\mathrm{e})$ (given by Eq 2.2 in Section 2.2); see Fig 2.2. The values satisfy the conditions that $f_v(p) < 0$ and $f_{\mathrm{e}}(\mathrm{e}) \geq 0$; implications of this specific choice will become clear in the next paragraph. It is noteworthy to mention that $f_v(p)$ is the negative of distance-to-measure originally defined in [55] and later used in [56] for the PCD case and captures the density distribution of points, whereas $f_{\mathrm{e}}(\mathrm{e})$ captures the inter-point distances between the points in the given point cloud.

The persistence algorithm processes each vertex and edge in the order of their appearance in the filtration. We execute it using a threshold value λ from $-\infty$ to ∞ and generate the persistence diagram accordingly. Intuitively, as λ is increased from $-\infty$ to ∞, vertices p for which $f_v(p) \leq \lambda$ and edges for which $f_{\mathrm{e}}(\mathrm{e}) \leq \lambda$ appear in the filtration for a particular value of λ (see Fig 2.2). Since $f_v(p) < 0$ and $f_{\mathrm{e}}(\mathrm{e}) \geq 0$, all the vertices first appear as λ is increased from $-\infty$ to 0, and then edges start appearing as λ becomes positive. The birth-death events for H_0 and H_1 constituting the persistence diagram (Fig 2.2H) contain information about the density and length scales present in the point-cloud. For example, the points showing birth and death events for the H_0-persistence diagram are more densely organized for the

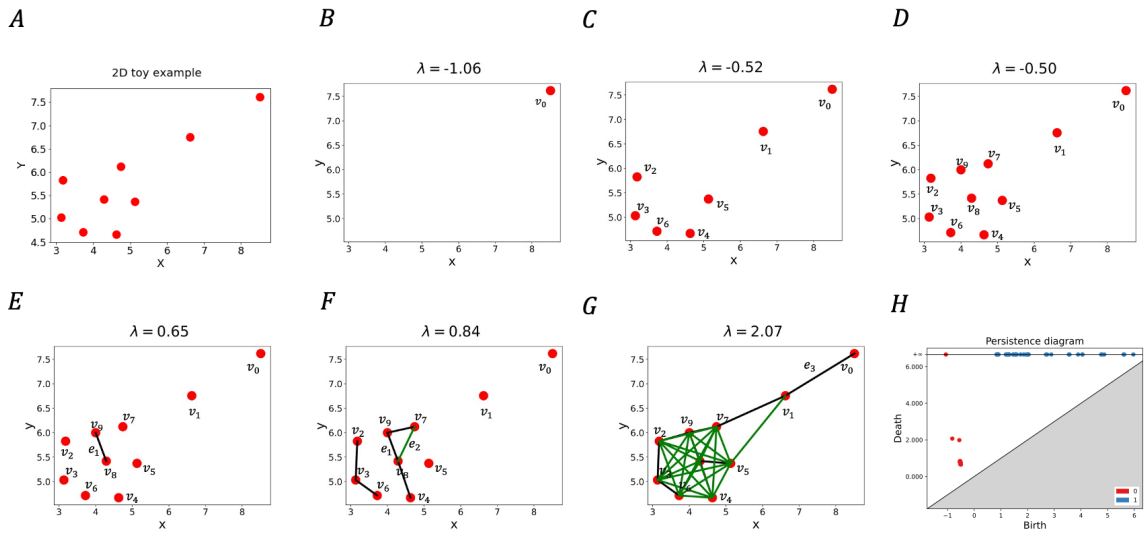

Figure 2.2. Illustration of persistence for a 2D point cloud data (PCD). **(A), (H)** shows a 2D PCD example and its computed persistence diagram. **(B)-(G)** shows important changes in topological feature as λ increases from $-\infty$ to ∞. **(B)** At $\lambda = -1.06$ an isolated point, v_0 appears first. Note that each isolated vertex creates a new component. **(C)** At $\lambda = -0.52$ points in the denser region appears in the filtration, introducing more components. The indices of the vertices denote the order in which they appear in the filtration. **(D)** At $\lambda = -0.50$, all vertices appear in the filtration. Note that, the way we have chosen the filtration function f, vertices appear before the edges since $f_v(v)$ is always negative. **(E)** At $\lambda = 0.65$, the first edge e_1 appears merging two components. By persistence algorithm [51], we pair the edge e_1 with v_9, since v_9 appears later in the filtration. Corresponding to this, we get a persistence pair $(f_v(v_9), f_{\mathrm{e}}(\mathrm{e}_1)) = (-0.50, 0.65)$. **(F)** At $\lambda = 0.84$, the green edge e_2 appears and creates a cycle. Since there is no 2-simplex(triangle) present, the cycle is never destroyed. In the persistence diagram we have this pair as $(f_{\mathrm{e}}(\mathrm{e}_3), \infty) = (0.84, \infty)$. **(G)** At $\lambda = 2.07$, the long edge e_3 appears joining v_0 and v_1, yielding a persistence pair $(-1.06, 2.07)$.

single cell protein expression data from the healthy donor than the SARS-CoV-2 infected patient in the HLA-DR - CD38 plane shown in 2.10 Fig. The denser organization of the birth-death events in the persistence diagram indicates a more homogeneous distribution of CD38 and HLA-DR proteins in the CD8+ T cells in healthy donors compared to that in infected patients. Most of the CD8+ T cells in healthy controls have low amounts of CD38 and HLA-DR abundances and few contain larger values of these proteins, indicating

a greater degree of homogeneity. The birth-death events for H_1 in the persistence diagram (2.10 Fig) in general contain information about the length scales of cyclic structures in the point cloud. It also can capture protrusions like 'elbows' that we have in COVID-19 data. Our filtration allows only birth (and not death) of 1-cycles and therefore, a λ value corresponding to the birth of a 1-cycle captures the length scale of the newly born cycle and hence an 'elbow'. Our analysis of the PCDs in Fig 2.10D and 2.10H indeed shows that λ values for the birth of cycles for the COVID-19 patient is much larger compared to that for the healthy individual indicating the presence of larger length scales in the PCD which is consistent with the presence of an 'elbow' shape in the PCD for the patient.

2.2 Materials and methods

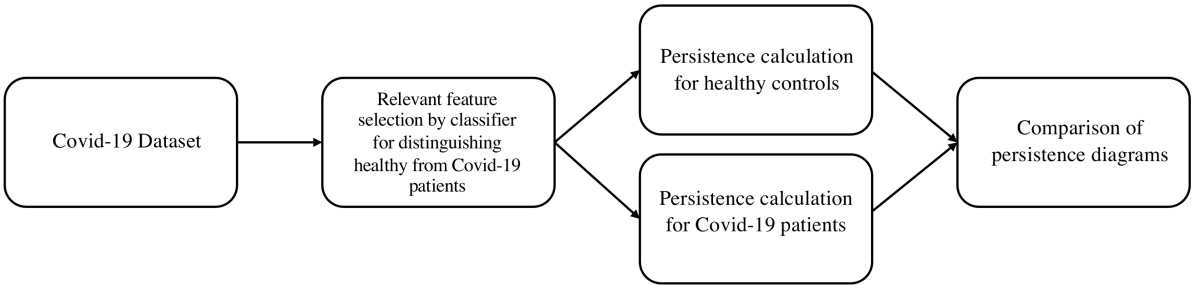

Figure 2.3. Flowchart of computation pipeline. The pipeline includes three main stages, namely, (i) relevant feature selection, (ii) persistence computation, and (iii) comparison of persistence diagrams.

2.2.1 Relevant feature selection by the XGBoost classifier:

Let $D = \left\{c_1, c_2, \ldots, c_m\right\}$ be the collection of m cytometry datasets. Each dataset, c_{i}, can be viewed as a $\mathcal{M}_{n\times p}$ matrix where n is the number of datapoints (cells) and p is the number of proteins with which each c_{i} is generated. We denote the collection of cytometry datasets of healthy individuals as $C_{\mathcal{H}} \subset D$ and similarly the set of individuals infected with SARS-CoV-2 as $C_{\mathcal{P}} \subset D$. We proceed to label the data in the following manner: If $c_{\mathrm{i}} \in C_{\mathcal{H}}$ then we assign the label $+1$ to each of the n datapoints, similarly we assign -1 if $c_{\mathrm{i}} \in C_{\mathcal{P}}$. Essentially, we now

have a binary classification problem where our labeled dataset is $D' = \bigcup D = c_1 \cup c_2 \cup \ldots \cup c_{\mathrm{j}}$, with labels defined as above. We solve this binary classification problem with XGBoost [57], a gradient boosted decision tree based classifier, and as a byproduct we get feature scores that correspond directly to each feature's importance in the classification. The higher the score for a protein, the more important it is for the classifier's decision. After our classifier orders the proteins by their scores, we take first r proteins to construct the point-cloud on which persistence diagrams are computed. We set $r = 3$ for all our analysis reported here. We used data from 56 healthy individuals and 108 COVID-19 patients for our feature selection.

The XGBoost classifier was implemented using the open-source python XGBoost package [57]. The model was then trained and validated with K-fold cross-validation, with $K = 10$. The average accuracy of the classifier was $92.14 \pm 0.04\%$. The protein scores are shown in Fig 2.4.

2.2.2 Random subsampling of the datapoints:

Each PCD can be thought as a set of indexed points. These indices were first shuffled randomly and then $20,000$ indices and hence respective points were sampled uniformly from this shuffled set. The samples drawn from each PCD were further analyzed using persistent homology. We discarded datasets that had less than $20,000$ data points. Among 55 healthy individuals only 1 had less than $20,000$ data points. For the patient data, the number of such datasets was 34.

2.2.3 Details of persistent homology computation:

As mentioned before (Section 2.3), computation of persistence diagrams needs a *filtration*. We set the filtration induced by the function $f = \{f_v, f_{\mathrm{e}}\}$ where $f_v(p)$ computes an 'average'

Euclidean distance between the vertex p and its k neighbors according to Eq 2.1 and $f_{\mathrm{e}}(\mathrm{e})$ computes the length of the edge e according to Eq 2.2.

$$f_v(p) = -\frac{1}{k}\sqrt{\sum_{\mathrm{i}}^{k} \|p - q_{\mathrm{i}}\|^2}\ , p \in P, \text{ and } q_{\mathrm{i}} \in k\text{-Nearest Neighbors of } p.. \tag{2.1}$$

The term $\|p - q_{\mathrm{i}}\|$ in the above equation is the Euclidean distance between the vertices p and q_{i}. The function value $f_{\mathrm{e}}(\mathrm{e})$ for an edge $\mathrm{e} = (p, q)$ is given by the Euclidean distance between p and q. For the experiments, the number of nearest neighbors is fixed to $k = 40$.

$$f_{\mathrm{e}}(\mathrm{e}) = \|p - q\|\ , \forall p, q \in P \text{ and } p \neq q \tag{2.2}$$

We begin by sampling every cytometry PCD c_{i} and take $n(= 20,000)$ samples. We do this to make c_{i} uniform *w.r.t.* number of data points (single CD8+ T cells). We compute a complete weighted graph $G(V, E)$ with vertices in the sampled data. This complete graph G is a key-step that enables us to compute the *persistence diagram*, $Dgm(c_{\mathrm{i}})$ of the dataset c_{i}, *w.r.t.* the filtration function f. We show the algorithm (Algorithm B in 2.2.4 Appendix) that executes this step in detail in the supplementary material. Notice that the graph G is weighted in the sense that each vertex $v \in V$ and edge $\mathrm{e} \in E$ carries a weight of $f_v(v)$ and $f_{\mathrm{e}}(\mathrm{e})$ respectively. Observe that $f : V \cup E \to \mathbb{R}$ constitutes a valid filtration of G.

We compute persistence diagrams for each $c_{\mathrm{i}} \in D$ according to Algorithm C in 2.2.4 Appendix. The next step involves comparing the persistence diagrams. We do this by computing the Wasserstein distance between persistence diagrams and plotting their distributions. We take two persistence diagrams of randomly selected healthy individuals and compute the Wasserstein distance between them with the help of Gudhi [58, 59] and scikit-learn Python library [60]. Similarly, we compute Wasserstein distance between persistence diagrams of a healthy and an infected individual (both are randomly drawn from the collection). We plot the resulting distances. We do this for 108 pairs to obtain two distributions. Note that, results described in Section 2.3 still holds for 200 pairs (Fig 2.11). Intuitively, a large Wasserstein distance between two persistence diagrams implies the datasets on which

Algorithm A DisPers

Input: c: Cytometry dataset, k: Nearest neighbors to consider, n: Number of samples
Output: $Dgm(c')$: Persistence diagram of c' sampled from cytometry data c

begin
  Subsample c
  $c' \leftarrow n$ samples from c
  $G \leftarrow$ GEN-COMPLETE-GRAPH(c', k)
  $Dgm(c') \leftarrow$ COMPUTE-PERSISTENCE(G)
  return $Dgm(c')$
end

they were constructed are structurally very different while a small distance implies they are structurally similar.

2.2.4 Algorithms:

Since a graph is a 1-dimensional simplicial complex, for a graph with m edges and n vertices, we can compute its 0-dim persistence in $\mathcal{O}(m \log n)$ time with Kruskal like minimum spanning tree algorithm [51].

Algorithm B Gen-Complete-Graph

Input: c: Sampled cytometry data, k: Nearest neighbors to consider
Output: $G(V, E)$: Complete Weighted Graph on c

procedure GEN-COMPLETE-GRAPH
  $G(V, E) \leftarrow \phi$ ▷ V is the set of nodes and E is the set of edges of G
  for all $v \in c$ **do**
    Compute $v_1, v_2, \ldots, v_k$, k-nearest neighbors of v
    $w(v) \leftarrow -\frac{1}{k}\sqrt{\sum_{\mathrm{i}} \|v - v_{\mathrm{i}}\|^2}$ ▷ $w(v)$ denotes vertex weight
    $V(G) \leftarrow V(G) \bigcup \Big(v, w(v)\Big)$
  for all $\{u, v\} \in c \times c$ and $u \neq v$ **do**
    $\mathrm{e} \leftarrow \{u, v\}$
    $w(\mathrm{e}) \leftarrow \|u - v\|$ ▷ $w(\mathrm{e})$ denotes weight of the edge
    $E(G) \leftarrow E(G) \bigcup (\mathrm{e}, w(\mathrm{e}))$
  return $G(V, E)$

Algorithm C shows the steps of Persistence computation. Consider a generic step when an edge e $\in G$ is introduced in the minimum spanning forest that joins two forests rooted at nodes v_0 and v_1. For the new tree we will choose the root as node having smaller weight and the edge e pairs with the node having larger weight. Essentially we are choosing to kill the youngest connected component created by, with a little abuse of notation, the vertex $argmax(w(v_0), w(v_1))$. We define persistence of the edge as

$$p(\mathrm{e}) = w(\mathrm{e}) - \max\{w(v_0), w(v_1)\} \tag{2.3}$$

It is important to point out that an edge may or may not necessarily kill a connected component. If it does not kill a connected component it definitely creates a 1-cycle and we pair the edge with a special vertex with $w(v) = \infty$ and define $pers(\mathrm{e}) = \infty$.

Algorithm C Compute Persistence Diagram

Input: $G(V, E)$: Complete weighted graph on sampled cytometry data c
Output: $Dgm(c)$: Persistence Diagram

procedure COMPUTE-PERSISTENCE
  $E' \leftarrow$ Sort E in increasing order of $w(\mathrm{e})$ with e $\in E$
  $P_0 \leftarrow \phi$ ▷ P_0 tracks 0-dim birth and death
  $P_1 \leftarrow \phi$ ▷ P_1 tracks 1-dim birth
  for all e $= (u, v) \in E'$ **do**
    $Root_0 \leftarrow find(u)$
    $Root_1 \leftarrow find(v)$
    if $Root_0 \neq Root_1$ **then**
      $birth \leftarrow max\{w(Root_0), w(Root_1)\}$ ▷ e kills youngest homology class
      $death \leftarrow w(\mathrm{e})$
      $pers(\mathrm{e}) \leftarrow death - birth$
      $merge(Root_0, Root_1)$
      $P_0 \leftarrow P_0 \cup (birth, death)$
    else
      $P_1 \leftarrow P_1 \cup (w(\mathrm{e}), \infty)$ ▷ e is a creator and creates a 1-cycle
  $Dgm(c) \leftarrow \{P_0, P_1\}$
  return $Dgm(c)$

2.2.5 Statistical testing of difference in Wasserstein distance distributions:

2-sided KolmogorovSmirnov (KS) tests were performed on Wasserstein distances for pairs of individuals to determine if they arise from the same or different probability distribution functions [61]. MATLAB's subroutine *kstest2* was used to determine p-values, where the null hypothesis is that the Wasserstein distances from H×H comparisons and the experimental condition (either H×P or H×R) arise from the same non-parametric distribution, and the alternative hypothesis is that they come from different distributions. A p-value of ≤ 0.05 (or ≥ 0.05) indicates the support for the alternate hypothesis (i.e., data occur from different distributions) is statistically significant (or not significant).

2.2.6 Computing quadratic form (QF) distance:

To measure the dissimilarity between a pair of Wasserstein distance distributions, quadratic form (QF) distance was computed as proposed by Bernas et al. in [62] (originally introduced in [63]). The QF distance was calculated using the formula

$$D^2(\mathbf{h},\mathbf{f}) = (\mathbf{h}-\mathbf{f})^T \mathbf{A}^{\mathrm{i}}_{\mathrm{j}} (\mathbf{h}-\mathbf{f}) = \sum_{\mathrm{i}=1}^{n}\sum_{\mathrm{j}=1}^{n} a_{\mathrm{ij}}(h_{\mathrm{i}}-f_{\mathrm{i}})(h_{\mathrm{j}}-f_{\mathrm{j}}) \tag{2.4}$$

where $\mathbf{f}$ and $\mathbf{h}$ are two vectors that list counts corresponding to two histogram bin counts. The quantities $\mathbf{f}$ and $\mathbf{h}$ can be normalised so that $\sum_{\mathrm{i}} f_{\mathrm{i}} = \sum_{\mathrm{i}} h_{\mathrm{i}} = 1$ when indexed by i. In our case, $\mathbf{A}^{\mathrm{i}}_{\mathrm{j}} = [a_{\mathrm{ij}}]$ and defined as $a_{\mathrm{ij}} = 1 - \sqrt{\frac{(\mathrm{i}-\mathrm{j})^2}{d_{max}}}$ with d_{max} being maximum distance between bins.

2.2.7 FlowSOM Analysis:

FlowSOM was performed on the entire non-naïve CD8+ dataset, and separately, the non-naïve B cell dataset. Data was scaled with the transform asinh(x/150) before analysis. See Code Availability for FlowSOM source code. Comparisons of relative cluster abundances between healthy controls and COVID-19 patients were performed with a Wilcoxon rank sum test. Subsequent persistent homology computation was performed on the selected clusters by sampling $20,000$ cells from either the aggregated healthy control data or aggregated COVID-

19 patient data. This sampling was repeated to form “synthetic" individual healthy control or COVID-19 patient data. We cannot sample data from each individual, as was done in the prior computations, because many individuals display too few cells in the selected clusters to reliably sample $20,000$ cells. In the CD8+ T cell analysis, clusters #1 and #3 were chosen for persistent homology calculations because they contain the most cells, and thus are most likely to have cells in each sample and be unaffected by random sampling.

2.2.8 Flow cytometry data for healthy individuals and COVID-19 patients:

The data come from Mathew et al., 2020 [46] and was retrieved from Cytobank. Mathew et al. performed high-dimensional flow cytometry experiments using peripheral blood obtained from 125 patients admitted to the hospital with COVID-19, 36 donors that recovered from documented SARS-CoV-2 infection, and 60 healthy controls. Our analysis focuses on the deposited data available at https://premium.cytobank.org/cytobank/experiments/308357 for non-naïve CD8+ T cells collected at the time of admission (and not any later blood draws, such as at 7 days after admission). Please note that a Cytobank account is currently required for data access. We removed forward- and side-scatter variables and other non-protein measurements, resulting in 25 proteins included in our analysis.

2.3 Results

2.3.1 Application of persistence to healthy and patient data

Our aim is to find out systematic differences in topological features extracted from cytometry data for healthy individuals and COVID-19 patients. Ideally one would like to compute persistence diagrams for all 25 proteins that were measured in single CD8+ T cells, however, this task encounters two major problems. First, as we mentioned before taking the full 25 dimensional PCD introduces the *curse of dimensionality* [64] making it computationally infeasible to produce the persistence diagrams. The second one is more subtle. In order to measure how the density of data differs from a healthy to infected person in a quantitative way, we need to ensure that the number of points in each PCD, to be analyzed by persistent homology, is the same. Cytometry data usually contain different numbers of single cells in

datasets obtained from different donors or replicates. To address the *curse of dimensionality* we use a classifier (XGBoost [57]) that distinguishes single CD8+ T cells in healthy donors from those in COVID-19 patients and we choose the top r (taken to be 3) features (proteins) that are deemed important by the classifier while classifying the data points (cells). This reduces the dimension of the data from 25 to a much smaller value denoted r.

To address the second issue, we perform uniform random sampling on every r-dimensional dataset and take equal number of samples from it. We then use the filtration defined in Eq 2.1 and Eq 2.2 to construct persistence diagrams for each dataset. To quantify the structural differences in the datasets as captured by the corresponding persistence diagrams, we compute the Wasserstein distance [59] between persistence diagrams from randomly selected pairs of either two healthy donors (H×H) or a healthy donor and an infected patient (H×P) and compute distributions of the Wasserstein distances for a large number of (H×H) and (H×P) pairs. The comparison of these distributions via Kolmogorov-Smirnov (KS) tests provides information regarding the systematic differences in shape features in the CD8+ T cell cytometry data across healthy individuals and COVID-19 patients. The computational pipeline is summarized in (Fig 2.3). Below we describe results from the application of our computational pipeline to the CD8+ T cell cytometry data in Mathew et al. [46]

2.3.2 A few protein expressions in CD8+ T cells separate healthy donors from COVID-19 patients:

We use XGBoost [57], a decision tree based classifier, to rank order proteins for their ability to distinguish CD8+ T cell point cloud data between healthy individuals and COVID-19 patients. The average accuracy of the classifier is about 92%. The classifier returns a feature score for each protein that characterizes its importance relative to other proteins in distinguishing cells from healthy individuals and COVID-19 patients. Intuitively, feature score is an indicator of the importance of a particular feature while classifying the data. By ranking the proteins by their feature scores, we can reduce our further analysis to only a subset of the most important proteins. Our analysis (Fig 2.4) shows that the top three most important proteins to the XGBoost classifier are proteins T-bet, Eomes, and Ki-67. T-bet induces gene expressions leading to an increase in cytotoxic functions of CD8+ T cells.

CD8+ T cells with increased cytotoxic functions are known as 'effector' CD8+ T cells and these cells show higher T-bet abundances. Conversely, Eomes induces gene expressions that contribute towards increased life span and re-activation potential of CD8+ T cells to specific antigens [65]. These long-lived T cells are known as 'memory' T cells which show increased expressions of Eomes. Memory T cells provide key protection against re-exposure to the same infection. Ki-67 is a marker for actively proliferating cells [66]. Mathew et al. [46] identified Ki-67 as one marker that is upregulated (increased) in some COVID-19 patients. These three proteins are most likely to distinguish CD8+ T cells in healthy donors from those in patients. Further details regarding the application of the classifier are provided in the Materials and Methods section (Section 2.2).

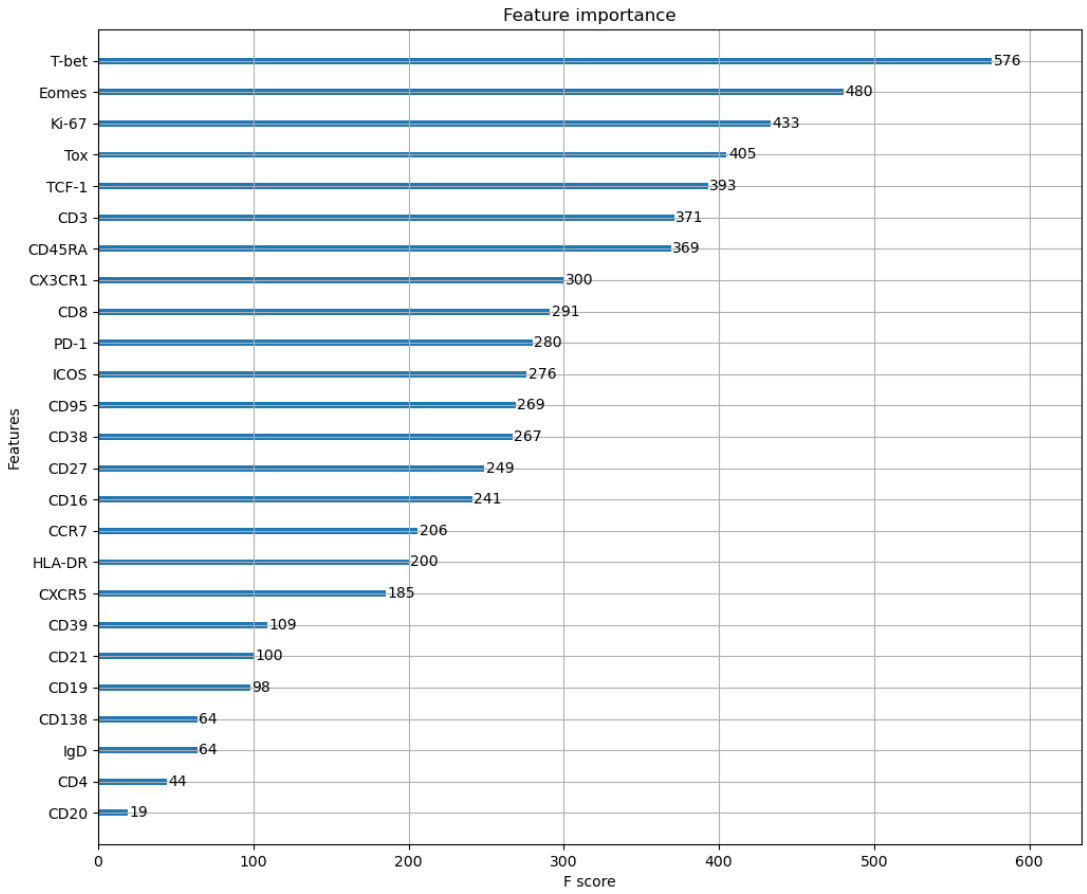

Figure 2.4. Rank ordering of proteins using a decision tree based classifier. Shows rank ordering of proteins by descending values of feature importance generated by the classifier XGBoost.

2.3.3 Persistence diagrams distinguish structural features in CD8+ T cell data occurring in healthy individuals and COVID-19 patients across batch effects and donor-donor variations:

We select the proteins T-bet, Eomes, and Ki-67 as relevant markers and compute the persistence diagrams of the PCD given by them for each individual belonging to groups of healthy donors, COVID-19 patients, and recovered patients. The persistence diagrams vary from individual to individual in each group and between groups which could arise due to batch effects in samples and/or donor-to-donor variations. To determine if there are systematic differences in persistence diagrams for individuals across the three groups (healthy, patient, and recovered), we compute Wasserstein distance between persistence diagrams for 3 categories of pairings: 1) two healthy donors (H×H), 2) one healthy donor and one patient (H×P), and 3) one healthy donor and one recovered individual (H×R). We compute distances for 100 randomly chosen pairs of individuals for each category of pairings. Wasserstein distances of the persistence diagrams for 0-th and 1-st homology groups H_0 and H_1 respectively are higher when comparing H×P pairs than when comparing H×H pairs (Fig 2.5). A 2-sided KS test showed that the Wasserstein distances for H×P and H×H belong to different probability distribution functions ($p \ll 0.01$); see Fig 2.5 and also the description of this test in Section 2.2. This indicates that systematic geometric differences in the flow cytometry PCD with T-bet, Eomes, and Ki-67 between individuals with and without COVID-19 are not attributable to batch effects or donor-to-donor variations alone. Increasing the number of randomly chosen pairs to 200 did not change this conclusion as Figs 2.5 and 2.11 illustrate. The difference between H×H and H×R distributions of distances in the T-bet, Eomes, and Ki-67 space are less prominent (2.12 Fig). We further test if such systematic differences are present for proteins that are at the bottom of the list in Fig 2.4 and find that the distributions of Wasserstein distances for corresponding persistence diagrams overlap between the H×H and H×P pairs (2.13 Fig). This suggests that systematic differences in the geometry of the PCD occur only for specific sets of proteins. Details regarding computation of persistence diagrams and Wasserstein distances are given in Section 2.2.

Next, we select a comparison pair that generates a large Wasserstein distance between H_1-persistence diagrams to further investigate what structural differences exist between the

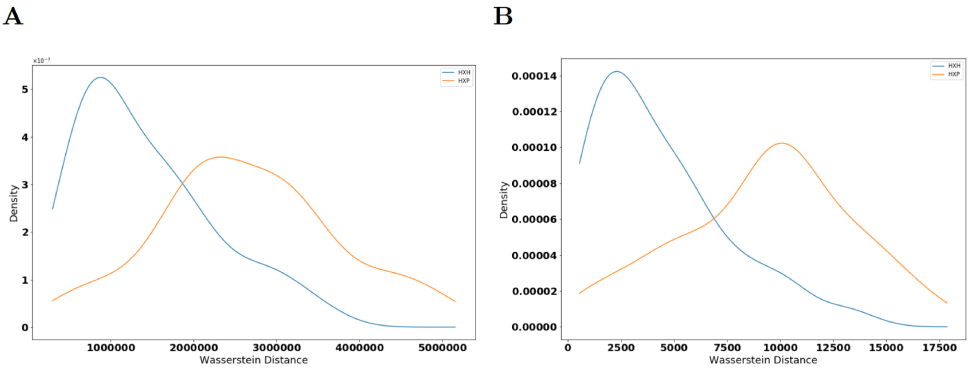

Figure 2.5. Distributions of Wasserstein distances between persistence diagrams. **(A)** Shows distributions of Wasserstein distance between H_0-persistence diagrams for H×H (blue line) and H×P (orange line) pairs ($p = 8.77 \times 10^{-15}$, $QFD = 0.173$). **(B)** Shows distributions of Wasserstein distance between H_1-persistence diagrams for H×H (blue line) and H×P (orange line) for the same pairs in (A) ($p = 3.04 \times 10^{-14}$, $QFD = 0.219$). Persistence diagrams are calculated from point clouds in the T-bet, Eomes, and Ki-67 axes. p-values are calculated from a 2-sided KS test.

datasets. We choose one pair of a healthy control and patient that generated a Wasserstein distance of 4.0×10^6 units in their H_0-persistence diagrams and 1.1×10^4 units in H_1-persistence diagrams. These two individual PCDs and their resulting persistence diagrams are shown in Fig 2.6.

A readily apparent difference between the resulting persistence diagrams is given by the lower birth times in H_1 of the COVID-19 patient compared to the healthy control (Fig 2.6E and 2.6F). This result indicates that the length scale of the data is smaller in the COVID-19 patient, which can be visually confirmed in the scatter plots of the data (Fig 2.6A and 2.6B). Specifically, the single cell abundances of T-bet and Eomes in CD8+ T cells are clustered significantly tighter around the origin for the COVID-19 patients than for the healthy controls. Similar manual inspection of other H×P pairs that generate large Wasserstein distances between their persistence diagrams confirms that this trend is not limited to this pair alone.

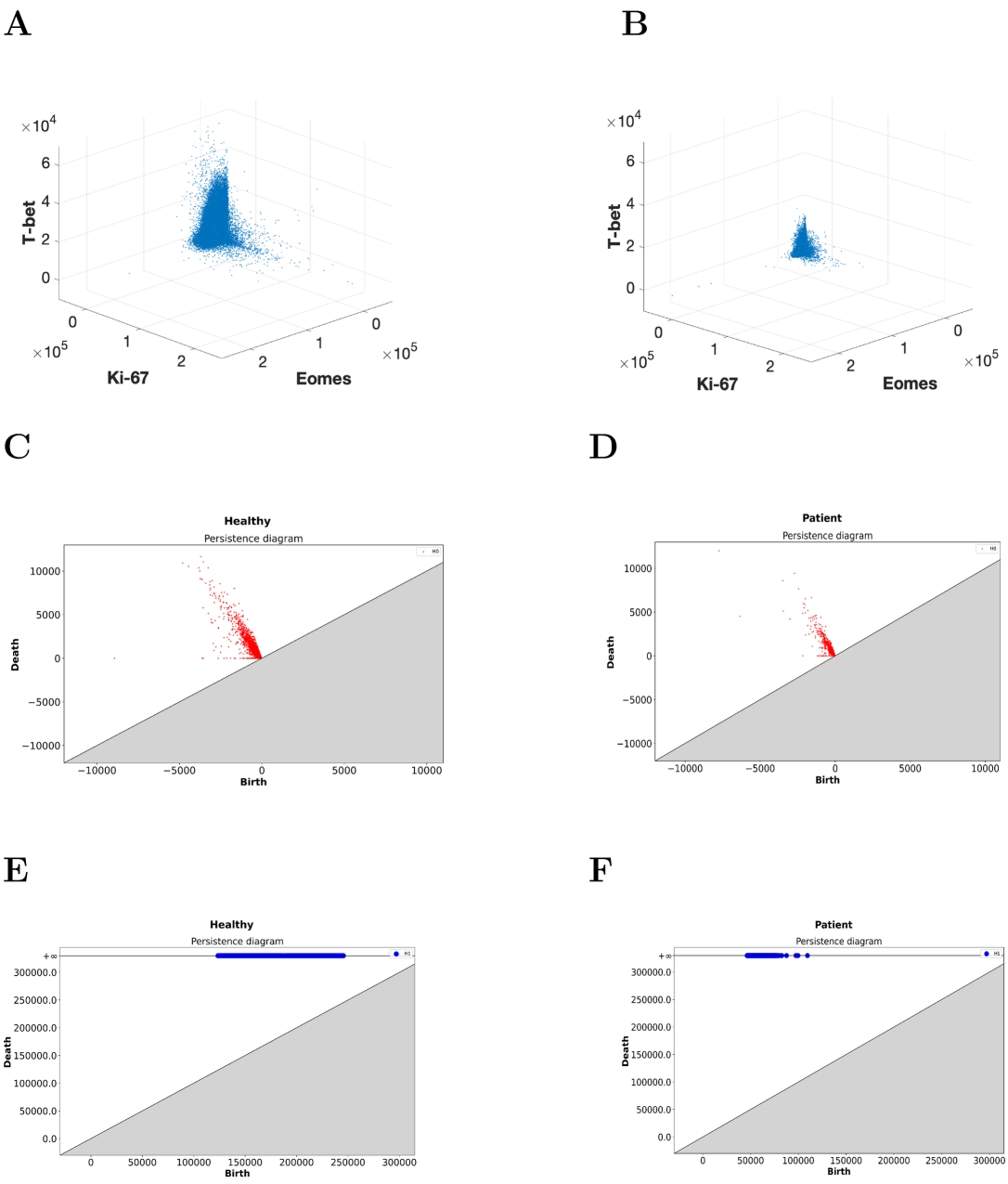

Figure 2.6. Differences in shape features in the 3D point cloud for CD8+ T cells in a H×P pair. CD8+ T cell point cloud for proteins Eomes, Ki-67, and T-bet for **(A)** a healthy control and **(B)** a COVID-19 patient. **(C)** Shows H_0-persistence diagram for the healthy control in (A). **(D)** Shows the H_0-persistence diagram for the COVID-19 patient in (B). **(E)** H_1-persistence diagram for the healthy control in (A). **(F)** H_1-persistence diagram for the COVID-19 patient in (B).

Additionally, the points in the H_0-persistence diagram are spread out more widely for the healthy control than the COVID-19 patient (Fig 2.6C and 2.6D). A wider distribution of births and deaths in the 0-th homology H_0 implies that there are regions of disparate densities. This suggests that the densities in the protein expressions of T-bet and Eomes are more homogeneous in the PCD in the COVID-19 patient than in the healthy control.

The structural change in the PCD for CD8+ T cells in the T-bet/Eomes plane that occurs during COVID-19 infection implies that T-bet and Eomes expression should be downregulated (decreased) in non-naïve CD8+ T cells. This result is consistent with analysis of clusters of CD8+ T cells by Mathew et al. [46] via a software package FlowSOM [67] that shows that clusters high in T-bet and/or Eomes are downregulated in COVID-19 patients.

The relevance of the above proteins in distinguishing healthy controls from patients is further demonstrated by the statistically significant differences (p-values $\ll 10^{-8}$) in the mean T-bet, Eomes, and Ki-67 abundances in the CD8+ T cells between the groups (2.17 Fig). However, the distributions of the mean abundances for the above proteins also showed regions of overlap between healthy and patient populations (2.17 Fig) indicating existence of $H \times P$ pairs with much smaller differences in the mean values between them than the population averaged mean values of these proteins. Our method specifically identifies differences in topological features in the shape of the PCD between a $H \times P$ pair which can be present despite small differences in the mean values of specific proteins (e.g., T-bet). We further investigated this point by analyzing correlations between the Wasserstein distance between the persistence diagrams of $H \times P$ pairs with the difference in the mean protein abundances (Fig 2.18A and 2.18B) which showed moderate correlations (≤ 0.5). However, there are several instances in which the Wasserstein distance captures differences in the shape of the PCD via persistent homology even when the difference in mean protein abundance (e.g., mean T-bet abundance) is small (Fig 2.18C-2.18F). This is due to changes in the shape of the point-cloud that are not easily captured by summary statistics such as the mean. Therefore, our analysis shows that healthy and COVID-19 pairs can be better separated by our persistent homology analysis than by summary statistics measures in such cases. The downregulation of T-bet and Eomes in response to viral infections is not well documented,

as CD8+ T cells commonly differentiate into phenotypes with high T-bet, high Eomes, or both in response to infections [65, 68].

We next explore the application of our approach to other datasets. We apply our method to the single cell cytometry dataset in Mathew et al. [46] for B cells obtained from healthy donors and COVID-19 patients. The B cells are major orchestrators of the humoral component of adaptive immunity against infections. We compute persistence diagrams for the proteins CXCR5, PD-1, and TCF-1, identified by XGBoost as the three most important proteins for classifying healthy donors and patients. A chemokine receptor, CXCR5, is responsible for B cell trafficking and is found to be downregulated in B cells in COVID-19 patients [46]. Both PD-1, a checkpoint inhibitory receptor [69], and TCF-1, a transcription factor important for T cell differentiation and effector functions [70] are increased in B cells in infected individuals (2.19 Fig). Immunosuppressive effects of high PD-1 expression in B cells have been reported earlier [69]. We find that Wasserstein distances of the persistence diagrams for both homology groups (H_0 and H_1) are significantly different (Fig 2.7) between the healthy donors and the patient population B cells. This demonstrates that our approach is able to distinguish healthy individuals from patient populations using PCDs of other immune cells. Furthermore, we find that the margin of separation, quantified by the QF-distance (QFD), in these Wasserstein distances is smaller with B cells than the CD8+ T cells. This implies that the structure of PCDs for CD8+ T cells differ more between healthy controls and patients than that for the B cells. These findings may point to previously uncharacterized impact of PD-1 and TCF-1 on B cell function or phenotype in SARS-CoV-2 infection.

Distributions of mean PD-1 expression on B cells in COVID-19 patients is not largely different than that of healthy controls (2.19B Fig). Therefore, we further analyzed how the difference between mean PD-1 values and the differences in PCDs quantified by the Wasserstein distances are related (Fig 2.20A-2.20D). We found that unlike T-bet for CD8+ T cells, mean PD-1 expression differences in B cells are not tightly correlated with Wasserstein distances in healthy control-patient pairs (Fig 2.20A-2.20B). We visualized the PCD in the space of TCF-1, PD-1, and CXCR5 (Fig 2.20E-2.20F) to gain further insights regarding the differences in the shapes of the PCD in healthy control-patient pairs with similar mean

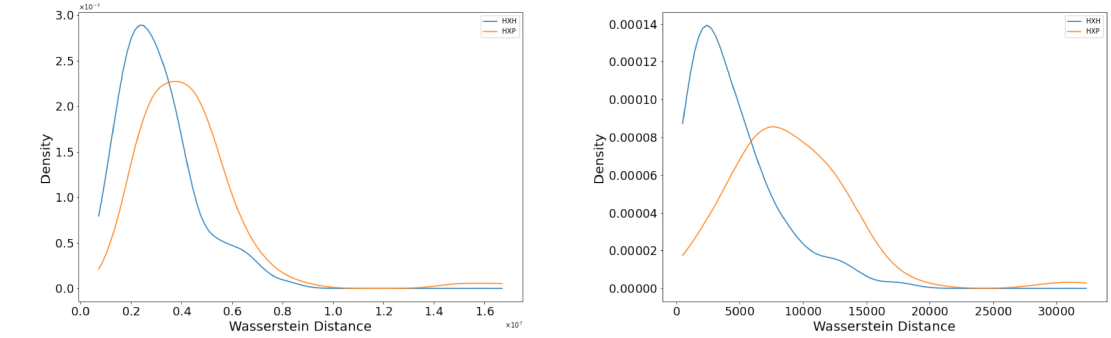

Figure 2.7. Distributions of Wasserstein distances between persistence diagrams for B cells. **(A)** Distributions of Wasserstein distance between H_0-persistence diagrams for H×H (blue line) and H×P (orange line) pairs (p=6.28×10^{-5}, QFD=0.0300). **(B)** Distributions of Wasserstein distance between H_1-persistence diagrams for H×H (blue line) and H×P (orange line) for the same pairs in (A) (p=1.07×10^{-12}, QFD=0.0946). Persistence diagrams are calculated from point clouds in the CXCR5, PD-1, and TCF-1 axes. p-values are calculated from a 2-sided KS test.

PD-1 expressions. The differences in the shape of the PCDs for these pairs can be largely attributed to the higher expressions of TCF-1 in healthy controls (Fig 2.20E-2.20F).

2.3.4 Comparison with Existing Methods:

To determine how our selection of filtration compares with an existing method, we compare how our results might change if we use Rips filtration. In Rips filtration, simplices appear when all of their edges appear in the filtration. The edges appear in non-decreasing order of their lengths. We use the entire dataset as the PCD and generate persistence diagrams subsequently, using Rips filtration [51, 53, 71] rather than the filtration we use in our approach. Note that in the standard Rips filtration, all vertices appear at the same instant whereas in our case the vertices are ordered by Eq 2.1. We then compute Wasserstein distances as done previously. We find that Rips filtration is also able to distinguish persistence diagrams of healthy controls and patients, but the margin of separation is much lower, as

evidenced by a higher p-value and lower QFD than our choice of filtration offers (2.21 Fig). This indicates that our method, which is designed to identify protrusions such as “elbows" in the data, characterizes greater differences in CD8+ T cell protein expression structures than existing methods such as Rips filtration.

We then compare our TDA approach with an existing algorithm FlowSOM [67], widely used for visualizing, clustering, and analyzing PCDs from cytometry experiments. FlowSOM uses a self-organizing map algorithm for generating single cell subsets with unique marker protein expressions. FlowSOM is capable of clustering similar cells together and offers a robust way to determine which cellular subsets are differentially expressed between data sources [72]. We run a FlowSOM analysis and clustering on the CD8+ T cell data and determine that 6 of the 15 clusters are differentially expressed between healthy controls and COVID-19 patients (2.22 Fig).

We select one FlowSOM cluster which is differentially expressed (Cluster #3) and one that is not differentially expressed (Cluster #1) between healthy donors and patients for downstream analysis. Visual inspection of these clusters in the 3-dimensional space of T-bet, Eomes, and Ki-67 shows that PCD structure may be different between healthy controls and patients in Cluster #1 (2.22C Fig). This is because proteins can co-vary in different ways in healthy controls and patients, affecting topological features hidden in the PCD. We then perform our persistent homology analysis to determine if the structure of PCDs for proteins Eomes, Ki-67, and T-bet in subsets of single cells associated with these FlowSOM clusters differ between healthy controls and patients. We find distributions of Wasserstein distances between the persistence diagrams obtained for the above FlowSOM clusters are significantly different (Figs 2.8 and 2.23).

Performing FlowSOM on B cells reveals 12 of the 15 clusters are differentially expressed between healthy controls and COVID-19 patients (2.24A-2.24B Fig). The three clusters (Cluster #1, #2, and #14) that are not differentially expressed are all high in PD-1. Visualization of the PCDs for Cluster #2 and Cluster #4 shows that regions high in TCF-1 2.24C-2.24D Fig) distinguish the PCDs for the pairs. Thus, our method is able to identify topological features hidden in the PCD that separate healthy controls from COVID-19 pa-

tients in FlowSOM clusters, including clusters which are not differentially expressed between these groups.

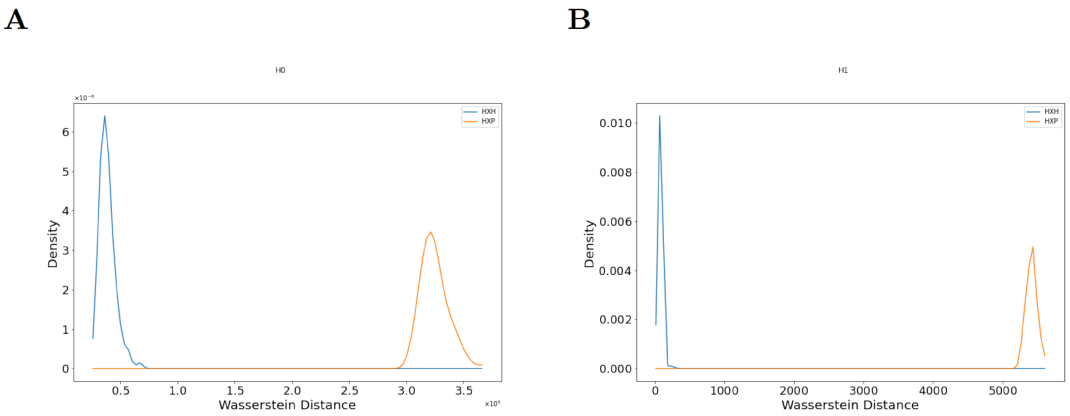

Figure 2.8. Distributions of Wasserstein distances between persistence diagrams from a FlowSOM cluster (Cluster #1) that is not differentially expressed between healthy controls and COVID-19 patients. **(A)** Distributions of Wasserstein distance between H_0-persistence diagrams for H×H (blue line) and H×P (orange line) pairs (p=2.21×10^{-59}, QFD=1.638) computed for the PCD for Eomes, Ki-67, and T-bet for CD8+ T cells. **(B)** Distributions of Wasserstein distance between H_1-persistence diagrams for H×H (blue line) and H×P (orange line) for the same pairs in (A) (p=2.21×10^{-59}, QFD=1.903).

2.4 Discussions and conclusions

We develop a persistent homology based approach to determine topological features hidden in point cloud data representing single cell protein abundances measured in cytometry data. In particular, we characterize the number of connected components or H_0, and the number of holes or H_1 in our persistence calculations, and show that our approach is able to determine systematic shape differences in the cytometry data for CD8+ T cells obtained from healthy individuals and COVID-19 patients. Therefore, the approach is able to successfully determine systematic shape differences that exist in the presence of batch effect noise and donor-donor variations in the cytometry data. Furthermore, our approach does not use data transformations (e.g., arc-sinh transformation) or any ad-hoc subtype gating to determine

these systematic differences, thus we expect persistent homology based approaches will be especially useful in identifying high-dimensional structural trends hidden in cytometry data.

We determine structural changes in T-bet and Eomes abundances in single CD8+ T cells in COVID-19 patients that can be summarized as downregulation. This result is non-intuitive as previous findings show that T-bet and Eomes protein abundances are highest in effector CD8+ T cells, which are induced in response to acute infections, suggesting T-bet and Eomes expressions should be upregulated [65, 68]. The clinical implications of this result are unclear. Mathew et al. [46] describe a immunophenotype in which Eomes+, T-bet+, CD8+ T cells are more abundant in COVID-19 patients who respond poorly to Remdesevir and NSAIDs, have high levels of IL-6, and have fewer eosinophils. Our analysis identifies that this immunophenotype (i.e.,Eomes+, T-bet+, CD8+ T cells) is systematically less prevalent in COVID-19 patients than in healthy controls. The ability of our approach to identify shape features in single immune cell PCD without any 'supervision' (e.g., specific gating) of the cytometry data shows that it can potentially determine more complicated immunologically relevant shape features. Furthermore, our approach inferred finer geometric structures from PCDs in B cells for proteins CRCX5, PD-1, and TCF-1 which helped distinguish COVID-19 patients from healthy individuals. These proteins are associated with cell migration, immunosuppression, and effector functions in lymphocytes and can potentially provide further insights into B cell response in COVID-19.

We compare our approach with an existing algorithm FlowSOM which is widely used for analyzing and visualizing multidimensional cytometry data. Our comparison reveals PCDs for subtypes of CD8+ T cells in FlowSOM clusters that do not differentiate healthy controls and COVID-19 patients contain topological features separating the above groups. Therefore, detecting topological features hidden in the PCDs can provide important biological insights regarding response of the lymphocytes in COVID-19.

Our approach integrates cellular comparisons with dataset comparisons. First, the classifier pools all data and determines which proteins are significant in discriminating whether cells come from healthy controls or COVID-19 patients. In this way, the classifier identifies a way to compare cellular phenotypes across experimental groups. Next, the computation of Wasserstein distances for persistence diagrams compares individuals against each other, inte-

grating cellular phenotypes with donor information (e.g., healthy and COVID-19 patients). Thus, this approach allows us to automatically identify individuals that are associated with distinguishing structural features in the point cloud data.

Currently, the limitations are mostly due to the curse of dimensionality that increases the computational complexity. Since we are computing pairwise distances between datapoints to obtain the persistence diagram (2.2.4 Appendix), computation time increases as the dimension of data increases. Computational cluster resources that we use currently complete all computations in about 20 minutes. This is comparable to other data science applications using large datasets, but this approach can be a barrier to those without access or experience with computational clusters. Additionally, it is unclear how additional dimensions impact the statistical properties of the data and interpretability of the results. To expand into many (i.e. 25) dimensions, computational interpretation and validation tools are necessary.

Available code:

Our current code is available at https://github.com/soham0209/TopoCytometry and will be updated for ease-of-use and performance enhancements.

Acknowledgments:

This work is supported by the NIH awards R01-AI 143740 and R01-AI 146581 to JD, and by the Research Institute at the Nationwide Children's Hospital, and NSF awards CCF 1839252 and 2049010 to TD. The authors would like to thank John Wherry and members of his lab for depositing their data and helping us to understand it.

2.5 Some additional results

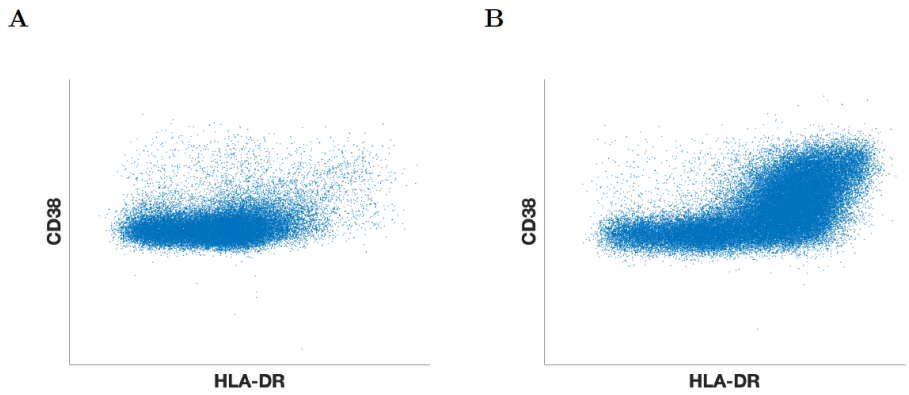

Figure 2.9. Scatter plots demonstrating change in structure of cytometry data. Transformed scatter plot for HLA-DR/CD38 axes for CD8+ T cell PCD in a singular **(A)** healthy donor and **(B)** COVID-19 patient. This plot demonstrates the 'elbow' found by the authors in [46]. The x-axis is asinh(HLA-DR/200) and the y-axis is asinh(CD38/500)

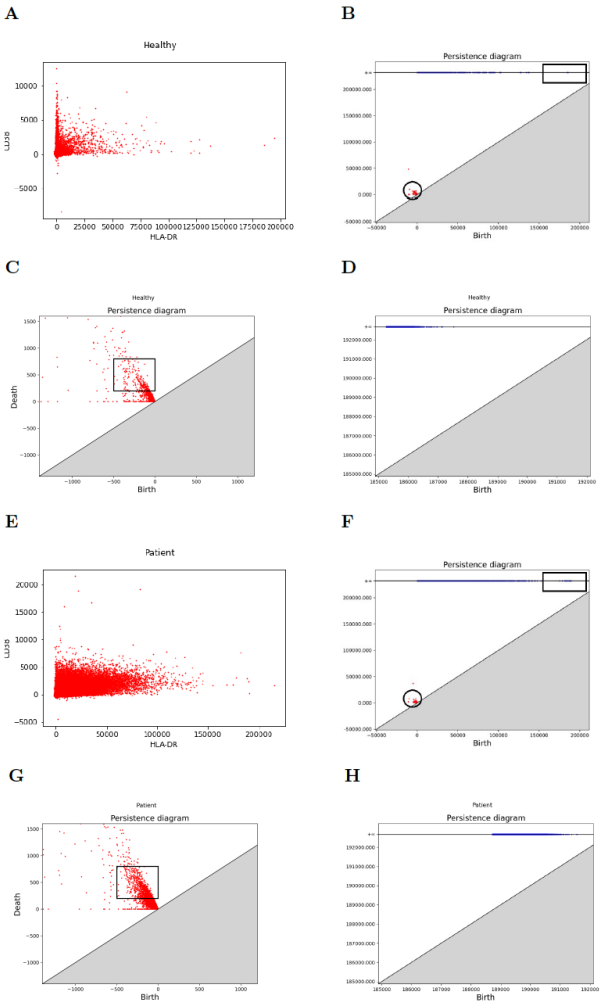

Figure 2.10. Persistence calculations and comparisons for HLA-DR/CD38 axes for CD8+ T cell PCDs shown in Mathew et. al.[46] **(A)** Point cloud for individual healthy control in HLA-DR/CD38 expression levels. **(B)** Complete persistence diagram for the healthy control shown in (A). Boxes indicate zoomed regions for figures (C) and (D); **(C)** Zoomed in region from (B) of H_0 persistence diagram. Box shows area of low density compared to patient persistence diagram; **(D)** Zoomed in region from (B) of H_1 persistence diagram; **(E)-(H)** Same as (A)-(D), but for an individual COVID-19 patient. The box in (G) is more densely populated than the identical box in (C).

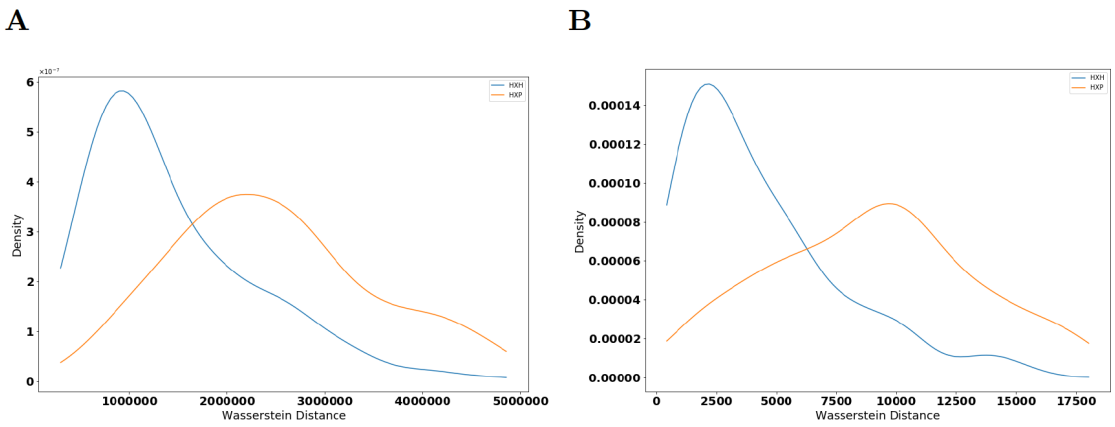

Figure 2.11. Distributions of Wasserstein distances between persistence diagrams calculated from 200 pairs of individuals. Distributions of Wasserstein distances between **(A)** H_0-persistence diagrams ($p = 6.75 \times 10^{-20}$, QFD=0.190) and **(B)** H_1-persistence diagrams ($p = 4.74 \times 10^{-24}$, QFD=0.220) for CD8+ T cells. Distances between pairs of healthy controls (H × H) and pairs of a healthy control and a COVID-19 patient (H × P) are overlaid. Persistence diagrams are calculated from point clouds in the T-bet, Eomes, and Ki-67 axes. This figure plots distributions of **200** randomly selected pairs, while Figure 2.5 plots distributions of 100 randomly selected pairs.

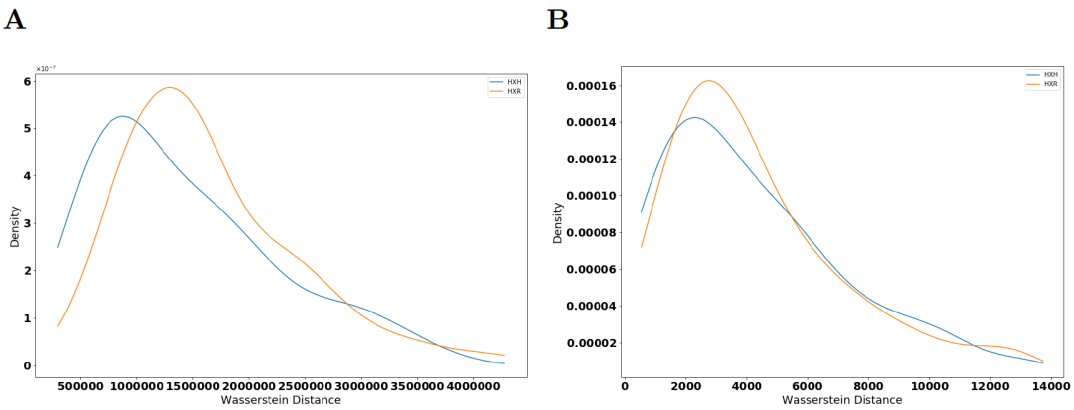

Figure 2.12. Distributions of Wasserstein distances between persistence diagrams for healthy controls and recovered individuals calculated using 3 most important proteins for XGBoost classification of CD8+ T cells from healthy or infected individuals. Distributions of Wasserstein distances between **(A)** H_0-persistence diagrams ($p = 0.131$, QFD=0.005) and **(B)** H_1-persistence diagrams ($p = 0.344$, QFD=0.001) for CD8+ T cells. Distances between pairs of healthy controls (H $\times$ H) and pairs of a healthy control and a individual that recovered from COVID-19 (H $\times$ R) are overlaid. Persistence diagrams are calculated from point clouds in the T-bet, Eomes and Ki-67 axes. p-values are calculated from a 2-sided KS test.

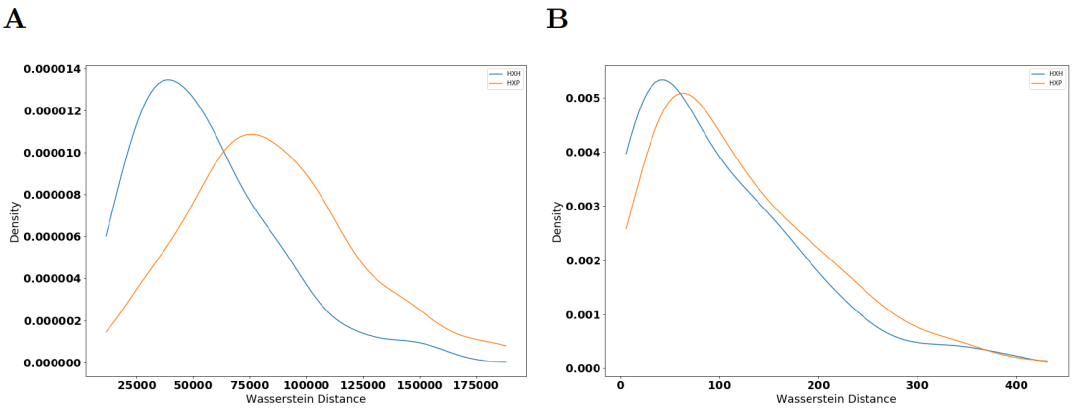

Figure 2.13. Distributions of Wasserstein distances between persistence diagrams calculated using 3 least important proteins for XGBoost classification of CD8+ T cells. Distributions of Wasserstein distances between **(A)** H_0-persistence diagrams ($p = 2.75 \times 10^{-8}$, QFD=0.051) and **(B)** H_1-persistence diagrams ($p = 0.111$, QFD=0.022) for CD8+ T cells. Distances between pairs of healthy controls (H $\times$ H) and pairs of a healthy control and a COVID-19 patient (H $\times$ P) are overlaid. Persistence diagrams are calculated from point clouds in the IgD, CD4 and CD20 axes. p-values are calculated from a 2-sided KS test.

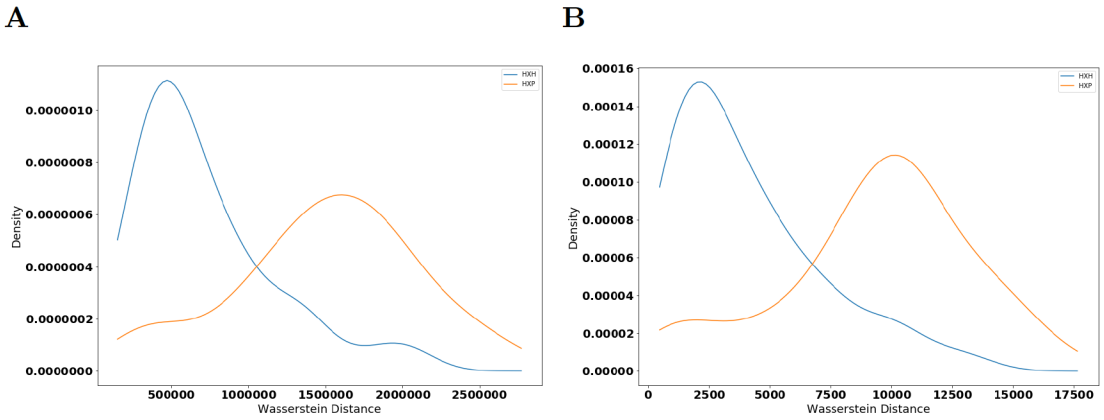

Figure 2.14. Distributions of Wasserstein distances between persistence diagrams calculated using 2 most important proteins for XGBoost classification of CD8+ T cells from healthy or infected individuals. Distributions of Wasserstein distances between **(A)** H_0-persistence diagrams ($p = 6.31 \times 10^{-19}$, QFD=0.261) and **(B)** H_1-persistence diagrams ($p = 6.31 \times 10^{-19}$, QFD=0.276) for CD8+ T cells. Distances between pairs of healthy controls (H $\times$ H) and pairs of a healthy control and a COVID-19 patient (H $\times$ P) are overlaid. Persistence diagrams are calculated from point clouds in the T-bet and Eomes axes. p-values are calculated from a 2-sided KS test.

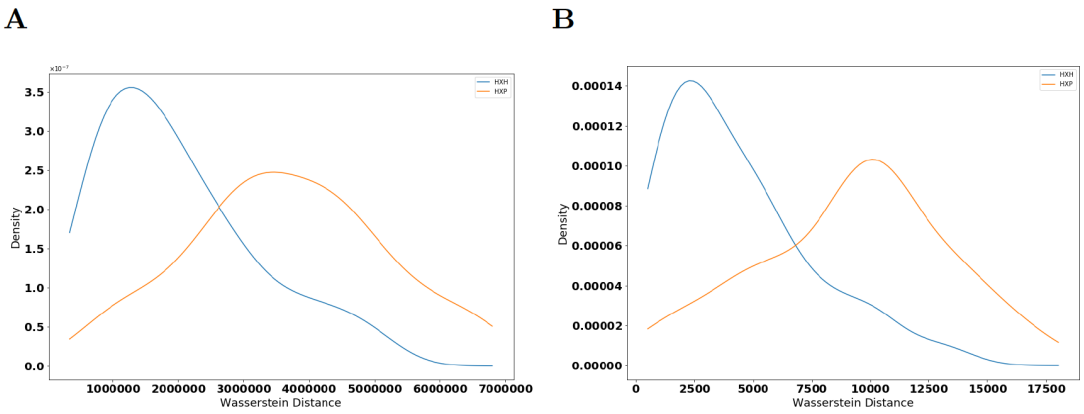

Figure 2.15. Distributions of Wasserstein distances between persistence diagrams calculated using 4 most important proteins for XGBoost classification of CD8+ T cells from healthy or infected individuals. Distributions of Wasserstein distances between **(A)** H_0-persistence diagrams ($p = 3.35 \times 10^{-13}$, QFD=0.267) and **(B)** H_1-persistence diagrams ($p = 3.04 \times 10^{-14}$, QFD=0.265) for CD8+ T cells. Distances between pairs of healthy controls (H $\times$ H) and pairs of a healthy control and a COVID-19 patient (H $\times$ P) are overlaid. Persistence diagrams are calculated from point clouds in the T-bet, Eomes, Tox and TCF-1 axes. p-values are calculated from a 2-sided KS test.

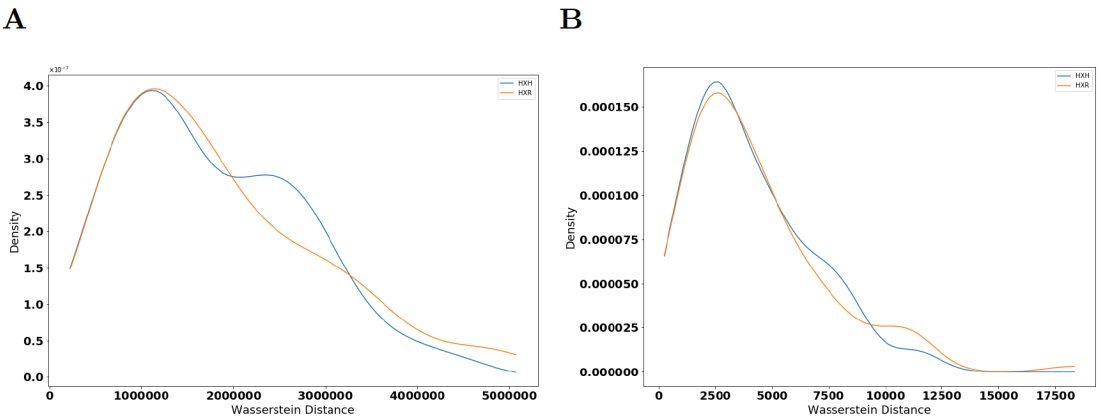

Figure 2.16. Distributions of Wasserstein distances between persistence diagrams calculated using 3 most important proteins for XGBoost classification of recovered CD8+ T cells. Distributions of Wasserstein distances between **(A)** H_0-persistence diagrams ($p = 0.908$, QFD=0.002) and **(B)** H_1-persistence diagrams ($p = 0.994$, QFD=0.001) for CD8+ T cell. Distances between pairs of healthy controls (H × H) and pairs of a healthy control and a individual that recovered from COVID-19 (H × R) are overlaid. Persistence diagrams are calculated from point clouds in the CD45RA, Eomes and TCF-1 axes. These 3 proteins are the best distinguishing features for the XGBoost classifier to distinguish cells from healthy controls from those from recovered individuals. p-values are calculated from a 2-sided KS test.

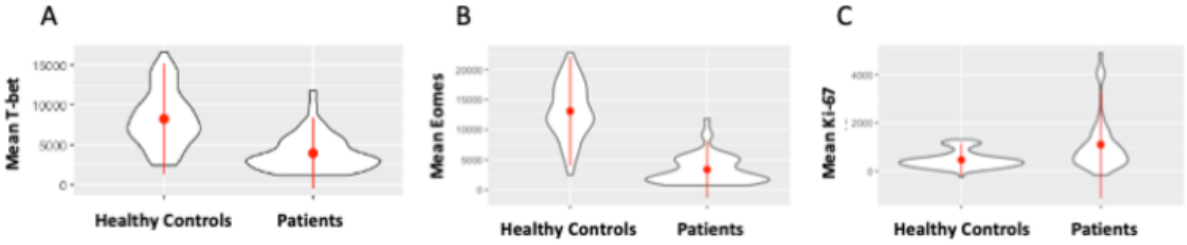

Figure 2.17. Probability distribution function (pdf) of mean abundances of proteins identified by XGBoost to be important to patient classification in CD8+ T cells. PDFs for **(A)** T-bet, **(B)** Eomes, and **(C)** Ki-67 for CD8+ T cell PCDs in each healthy control and COVID-19 patient shown using violin plots. The thickness of the "violin" denotes the value of the pdf. Data distributions are calculated from the mean protein abundances across all non-naïve CD8+ T cells for each individual. Red dots represent the mean of the data, and red lines represent the standard deviation.

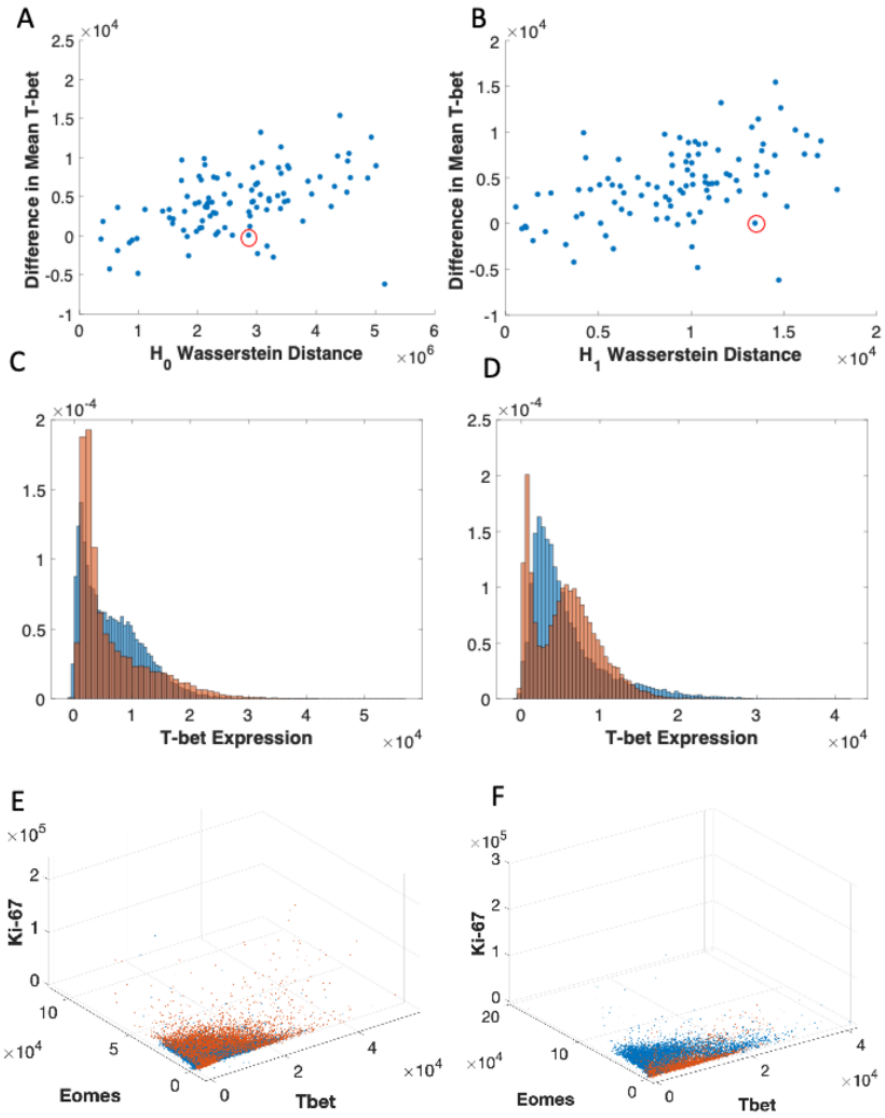

Figure 2.18. Demonstration of persistent homologys ability to capture more information than change in magnitude of single protein measurements in CD8+ T cells. **(A-B)** Scatter plot showing the relationship between Wasserstein distance between H_0 and H_1 persistence diagrams and the difference in mean T-bet abundance for CD8+ T cells for random pairs of healthy controls and COVID-19 patients. Red circles highlight a pair of individuals that generated a large Wasserstein distance despite a small difference in mean T-bet expressions, which are further analyzed in (C-F). **(C-D)** Histograms showing the T-bet expression of non-naïve CD8+ T cells for the healthy control (blue) and COVID-19 patient (red) from the points circled above in (A-B). **(E-F)** Scatter plots showing the T-bet, Eomes, and Ki-67 abundances for the healthy control (blue) and COVID-19 patient (red) from the points circled in (A-B).

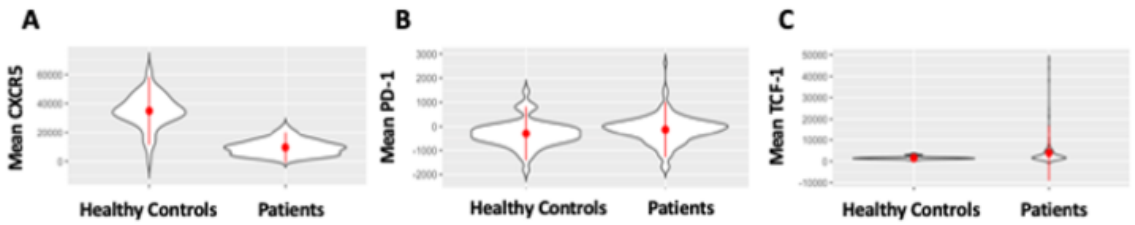

Figure 2.19. Probability distribution function (pdf) of mean abundances of proteins identified by XGBoost to be important to patient classification in B cells. PDFs for **(A)** CXCR5, **(B)** PD-1, and **(C)** TCF-1 in B cells of each healthy control and COVID-19 patient shown using violin plots. The thickness of the "violin" denotes the value of the pdf. Data distributions are calculated from the mean protein abundances across all B cells for each individual. Red dots represent the mean of the data, and red lines represent the standard deviation.

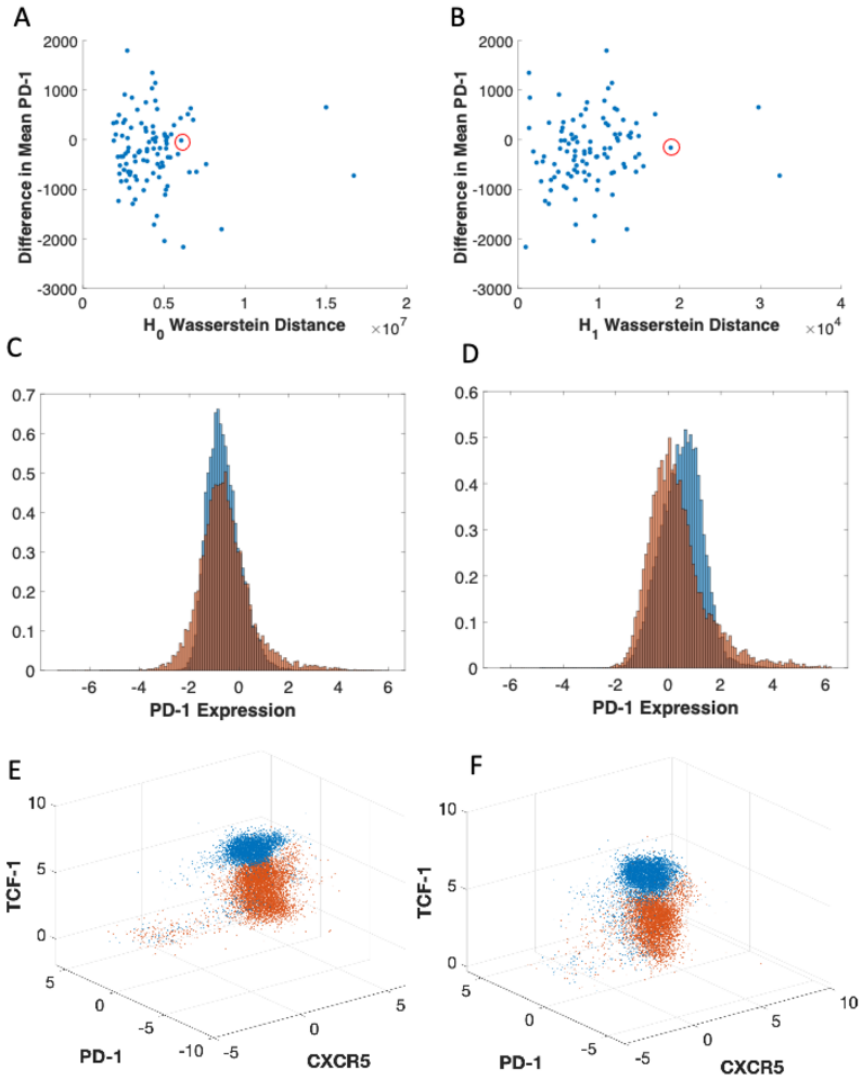

Figure 2.20. Demonstration of persistent homologys ability to capture more information than change in magnitude of single protein measurements in B cells. **(A-B)** Scatter plot showing the relationship between Wasserstein distance between H_0 and H_1 persistence diagrams and the difference in mean PD-1 abundance for B cells for random pairs of healthy controls and COVID-19 patients. Red circles highlight a pair of individuals that generated a large Wasserstein distance despite a small difference in mean PD-1 expressions, which are further analyzed in (C-F). **(C-D)** Histograms showing the PD-1 expression of non-naïve B cells for the healthy control (blue) and COVID-19 patient (red) from the points circled above in (A-B). **(E-F)** Scatter plots showing the CXCR5, PD-1, and TCF-1 abundances for the healthy control (blue) and COVID-19 patient (red) from the points circled in (A-B). Protein expression axes in (C)-(F) are scaled to asinh(x/150), where x is the expression of the given protein.

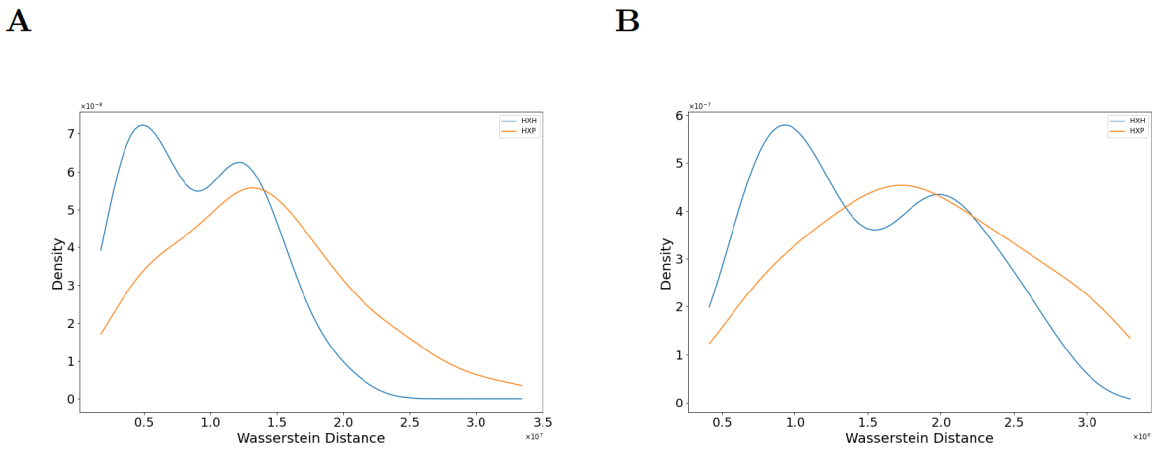

Figure 2.21. Distributions of Wasserstein distances between persistence diagrams calculated using Rips filtration. **(A)** Shows distributions of Wasserstein distance between H_0-persistence diagrams for H×H (blue line) and H×P (orange line) pairs (p=1.20×10^{-4}, QFD=0.0575) for CD8+ T cells. **(B)** Shows distributions of Wasserstein distance between H_1-persistence diagrams for H×H (blue line) and H×P (orange line) for the same pairs in (A) (p=3.73×10^{-3}, QFD=0.0343).

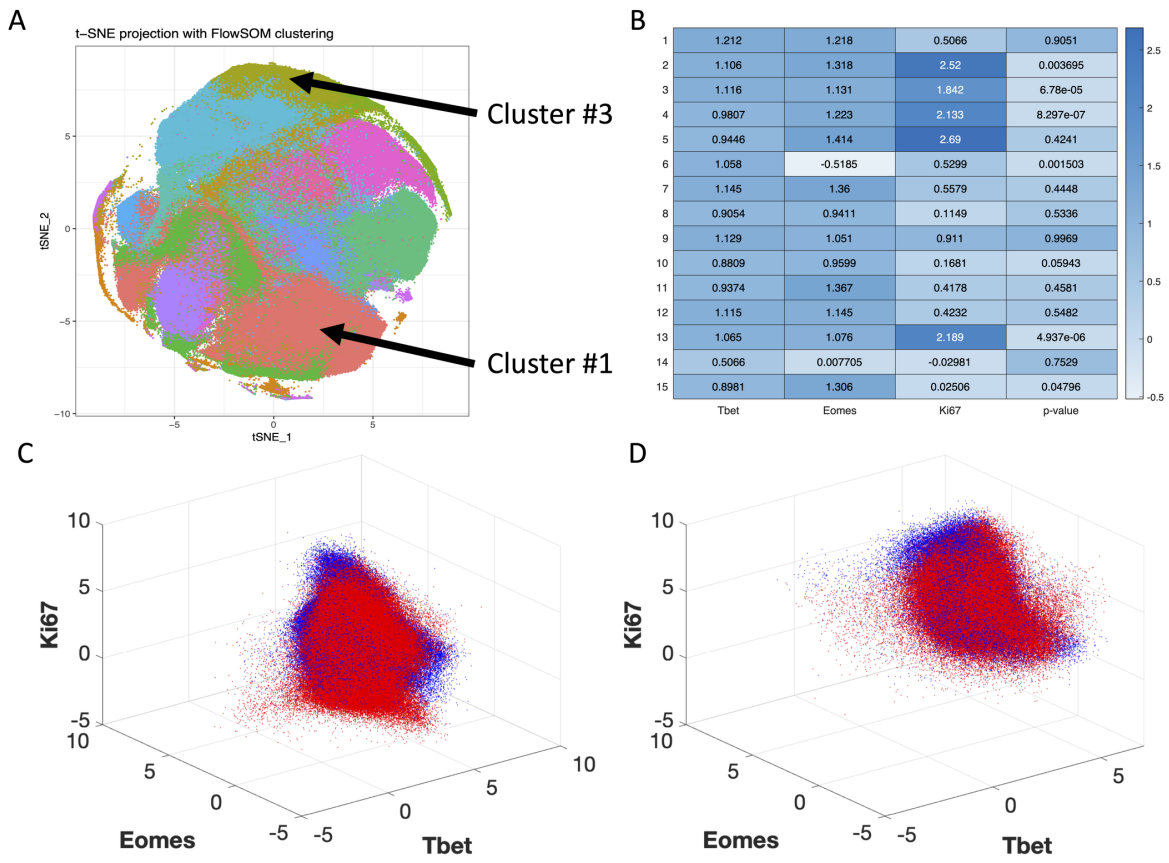

Figure 2.22. Results of FlowSOM analysis for CD8+ T cells. **(A)** t-SNE projection of protein expression data for CD8+ T cell PCD in Mathew et al. Each point represents a cell with 25 protein expressions. Colors represent 15 clusters identified by FlowSOM. Cluster #1 and Cluster #3 are selected for further topological analysis. **(B)** Heatmap showing scaled MFI for T-bet, Eomes, and Ki-67 for each cluster. Each entry in the first three columns is the MFI scaled by the average MFI of the column. The fourth column shows p-values determining differential expression of the cluster between healthy controls and COVID-19 patients. Note that Cluster #1 has $p > 0.05$ and Cluster #3 has $p < 0.05$. **(C-D)** Scatter plots showing the T-bet, Eomes, and Ki-67 abundances for all healthy controls (blue) and COVID-19 patients (red) from the cells in Cluster #1 (C) and Cluster #3 (D). Black circles in (C) indicate regions of single cell protein expressions which contribute to the differences in the PCD structure for the FlowSOM clusters. Axes are scaled to asinh(x/150), where x is the expression of the given protein.

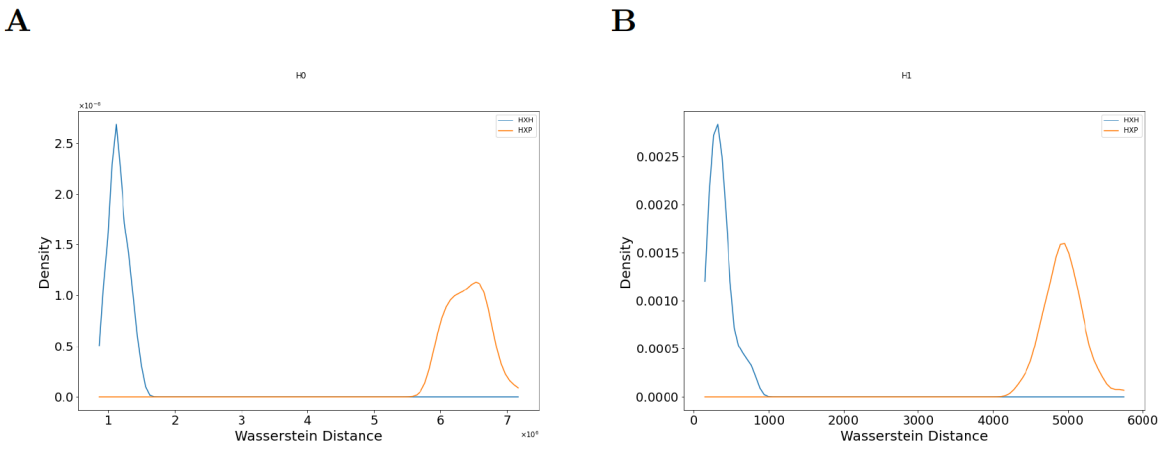

Figure 2.23. Distributions of Wasserstein distances between persistence diagrams from a FlowSOM cluster (Cluster #3) that is differentially expressed between healthy controls and COVID-19 patients. **(A)** Shows distributions of Wasserstein distance between H_0-persistence diagrams for H×H (blue line) and H×P (orange line) pairs (p=2.20×10^{-59}, QFD=1.604). **(B)** Shows distributions of Wasserstein distance between H_1-persistence diagrams for H×H (blue line) and H×P (orange line) for the same pairs in (A) (p=2.21×10^{-59}, QFD=1.573).

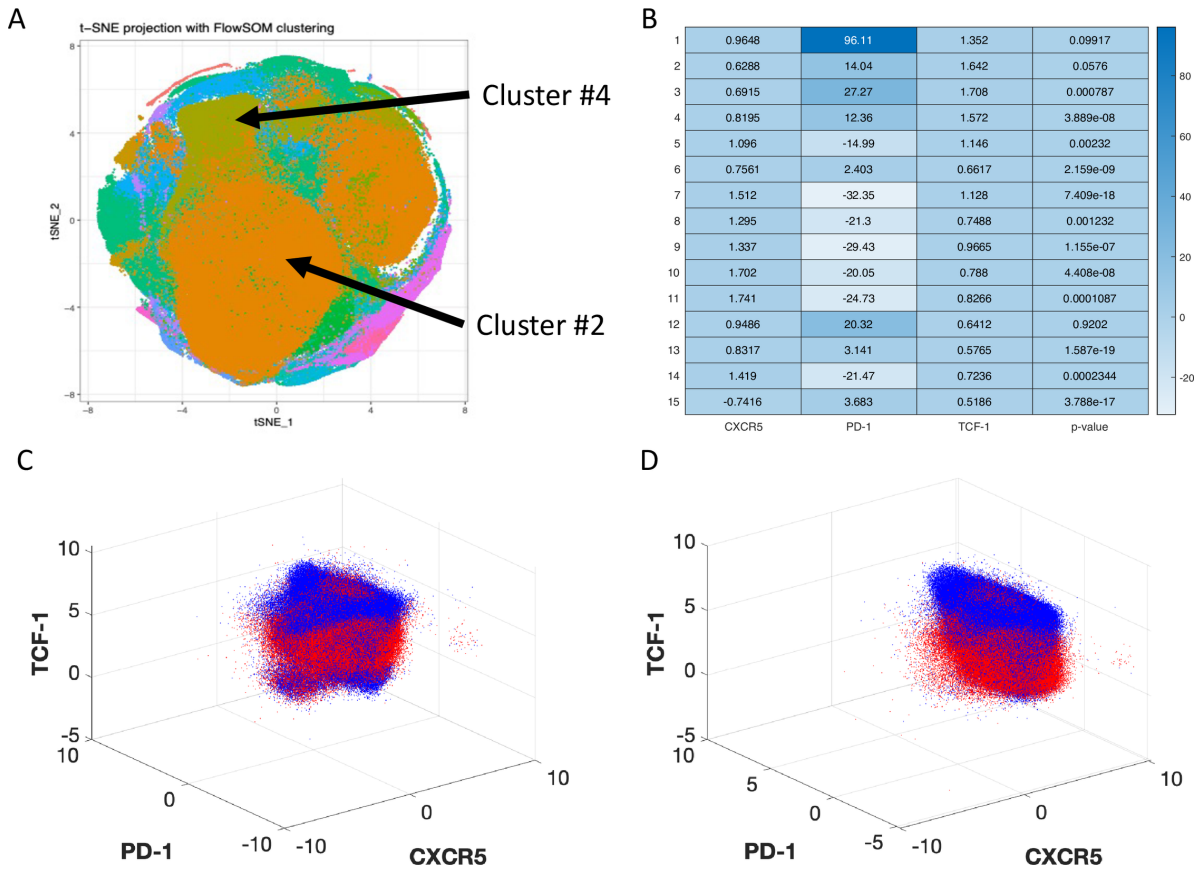

Figure 2.24. Results of FlowSOM analysis for B cells. **(A)** t-SNE projection of protein expression data for B cell PCD in Mathew et al. Each point represents a cell with 25 protein expressions. Colors represent 15 clusters identified by FlowSOM. Cluster #2 and Cluster #4 are selected for further topological analysis. **(B)** Heatmap showing scaled MFI for CXCR5, PD-1, and TCF-1 for each cluster. Each entry in the first three columns is the MFI scaled by the average MFI of the column. The fourth column shows p-values determining differential expression of the cluster between healthy controls and COVID-19 patients. Note that Cluster #2 has $p > 0.05$ and Cluster #4 has $p < 0.05$. **(C-D)** Scatter plots showing the CXCR5, PD-1, and TCF-1 abundances for all healthy controls (blue) and COVID-19 patients (red) from the cells in Cluster #2 (C) and Cluster #4 (D). Axes are scaled to asinh(x/150), where x is the expression of the given protein.

3. LEARNING WITH TDA - DELVING DEEPER

Graph classification is an important task in machine learning. Applications range from classifying social networks to chemical compounds. These applications require global as well as local topological information of a graph to achieve high performance. Message passing graph neural networks (GNNs) are an effective and popular method to achieve this task.

These existing methods crucially lack quantifiable information about the relative prominence of cycles and connected component to make predictions. Extended persistence is an unsupervised technique from topological data analysis that provides this information through a generalization of hierarchical clustering on graphs. It obtains both 1- and 0-dimensional multiscale global homological information.

Existing end-to-end filtration learning methods [73, 74] that use persistent homology do not compute extended persistence because of its high computational cost at scale. A general matrix reduction approach [75] has time complexity of $O((n+m)^\omega)$ for graphs with n nodes and m edges where ω is the exponent for matrix multiplication. We address this by improving upon the work of [76] and introducing a link-cut tree data structure and a parallelism for computation. This allows for $O(\log n)$ update and query operations on a spanning forest with n nodes. In addition to that in this chapter we focus on shape analysis, with an emphasis on non-manifold mesh classification. Shape analysis is a fundamental and challenging field that plays a pivotal role in computer vision, computer graphics, and geometric modeling. Understanding and processing the geometrical and topological properties of objects or shapes are essential for a wide range of applications, such as object recognition, segmentation, registration, and animation. The representation of shapes can vary, ranging from traditional Euclidean models to more complex data structures like meshes and point clouds. However, traditional methods based on graph-based abstractions may not fully capture the rich geometric and topological characteristics of complex shapes.

Recently, topological deep learning (TDL) has emerged as a powerful approach to address the limitations of conventional shape analysis techniques. TDL introduces a new paradigm that leverages higher-order network domains, such as simplicial complexes, cell complexes, and hypergraphs, to model and learn from shape data [77–80]. These higher-order repre-

sentations enable the extraction of more global and topological information, allowing for a more expressive and flexible representation of shapes compared to graph-based methods.

In the context of shape analysis, TDL offers exciting opportunities to advance the state-of-the-art in mesh processing. TDL techniques help to capture complex topological relationships, enabling the discovery of novel shape features that were challenging to obtain with traditional approaches. Most traditional approaches [81–83] also assume that meshes are manifolds and use features that rely on the mesh being a manifold.

In this chapter,

1. We introduce extended persistence and its cycle representatives into the supervised learning framework in an end-to-end differentiable manner, for graph classification.

2. For a graph with m edges and n vertices, we introduce the link-cut tree data structure into the computation of extended persistence, resulting in an $O(m \log n)$ depth and $O(mn)$ work parallel algorithm, achieving more than 60x speedup over the state-of-the-art for extended persistence computation, making extended persistence amenable for machine learning tasks.

3. We first propose a new architecture *Combinatorial Complex Isomorphism Network (CCIN)* which uses higher order networks [84] and demonstrate its use on mesh-related tasks, with a particular focus on "non-manifold" mesh classification. We use vertex, edge and face features of the meshes which do not depend on mesh being a manifold with the CCIN architecture to achieve comparable accuracy with state-of-the-art models.

4. Further, we provide experimental results on non-manifold meshes which show that CCIN along with these features also achieves high accuracy on non-manifold meshes.

3.1 Background on Extended Persistence

Original persistence algorithm, proposed by Edelsbrunner et al. in [85] recognizes the loops in the graph as being persistent forever and the same thing happens for the original connected components in the input graph. Furthermore, it fails to detect the upward

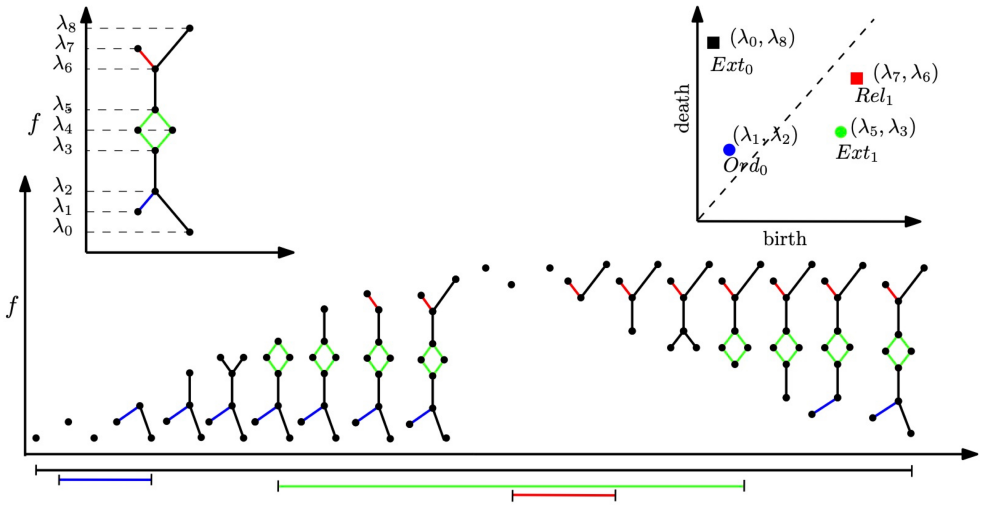

Figure 3.1. Extended persistence diagram w.r.t. a vertex filter function f on a graph. The sequence of sublevel and superlevel graphs shown on the bottom of the figure. The corresponding bars are displayed under the sequence - the black bar represents the connected component of the graph, the blue one represents its upward branch, the red one represents its downward branch and the green one represents its loop. The extended persistence diagram given by the bars is shown on the top-right. Notice that all the bars are finite.

branching (with respect to f) since they do not create connected components when they appear in the filtration. One can use special methods to handle the infinite intervals or can simply ignore them losing topological information in the process [32, 73, 86, 87]. As a remedy, **Extended persistence**($\mathbf{PH_{ext}}$) [75] takes an extended filtration F_{f_G} as input, which is obtained by concatenating *lower star filtration* of the graph G and an *upper star filtration* of the coned space of G induced by a vertex filtration function f_G. Concatenation here simply means concatenating two filtration sequences. More specifically, let α be an additional vertex for the graph G. Define an extended function $f_{G\cup\{\alpha\}}$ whose value is equal to f_G on all vertices except α on which it has a value larger than any other vertices. The cone of a vertex u is given by the edge (α, u) and the cone of an edge (u, v) is given by the triangle (α, u, v). As a result, in extended persistence all 0- and 1-dimensional features die (bars are finite; see [75] for details). Four different persistence pairings or bars result from $\mathbf{PH_{ext}}$. The barcode $\mathcal{B}_0^{low}$ results from the vertex-edge pairs within the lower filtration, the

barcode $\mathcal{B}_0^{up}$ results from the vertex-edge pairs within the upper filtration, the barcode $\mathcal{B}_0^{ext}$ results from the vertex-vertex pairs that represent the persistence of connected components born in the lower filtration and die in the upper filtration, and the barcode $\mathcal{B}_1^{ext}$ results from edge-edge pairs that represent the persistence of cycles that are born in the lower filtration and die in the upper filtration. The barcodes $\mathcal{B}_0^{low}$, $\mathcal{B}_0^{up}$, and $\mathcal{B}_0^{ext}$ represent persistence in the 0th homology H_0. The barcode $\mathcal{B}_1^{ext}$ represents persistence in the 1st homology H_1. In the TDA literature, $\mathcal{B}_0^{low}$, $\mathcal{B}_0^{up}$, $\mathcal{B}_0^{ext}$, and $\mathcal{B}_1^{ext}$ also go by the names of $Ord_0, Rel_1, Ext_0, Ext_1$ respectively.

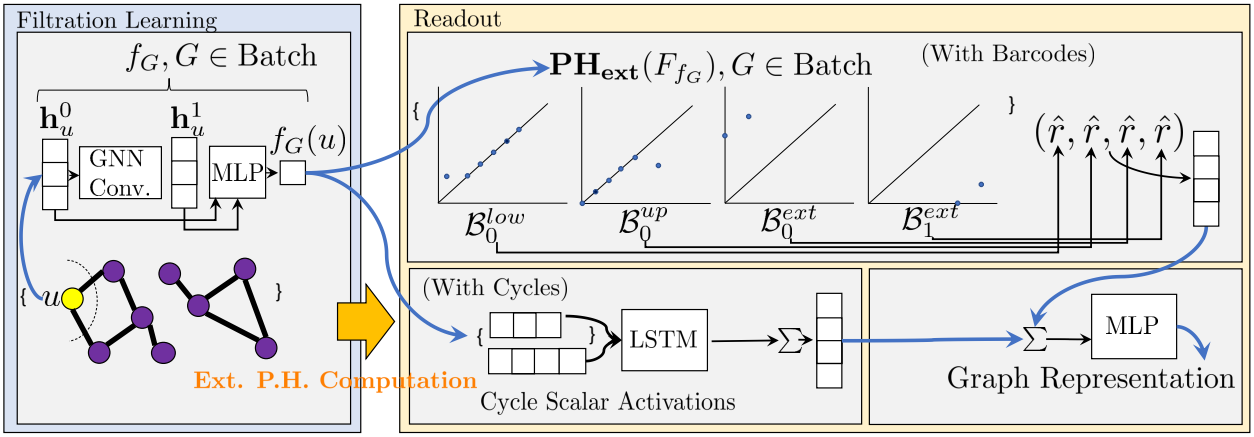

Figure 3.2. The extended persistence architecture (bars+cycles) for graph representation learning. The negative log likelihood (NLL) loss is used for supervised classification. The yellow arrow denotes extended persistence computation, which can compute both barcodes and cycle representatives.

3.2 Extended Persistence as Readout Layer

Our method as illustrated in Figure 3.2 introduces extended persistence as the readout function for graph classification. In our method, an upper and lower filtration, represented by a filtration function, coincides with a set of scalar vertex representations from standard message passing GNNs. This filtration function is thus learnable by *message passing graph neural network* (MPGNN) convolutional layers. Learning filtrations was originally introduced in [14] with standard persistence. In [88], authors show that arbitrary cycle lengths are hard to distinguish by both standard GNN readout functions as well as standard per-

sistence due to the lack of explicitly tracking paths or cycles. Extended persistence, on the other hand, explicitly computes learned displacements on cycles of some cycle basis as determined by the filtration function as well as explicit cycle representatives.

We represent the map from graphs to learnable filtrations by any message passing GNN layer such as GIN, GCN or GraphSAGE followed by a multi layer perceptron (MLP) as a Jumping Knowledge (JK) [89] layer. The JK layer with concatenation is used since we want to preserve the higher frequencies from the earlier layers [90]. Our experiments demonstrate that fewer MPGNN layers perform better than more MPGNN layers. This prevents oversmoothing [91, 92], which is exacerbated by the necessity of scalar representations.

The readout function, the function that consolidates a filtration into a global graph representation, is determined by computing four types of bars for the extended persistence on the concatenation of the lower and upper filtrations followed by compositions with four rational hat functions $\hat{r}$ as used in [14, 73, 74]. To each of the four types of bars in barcode $\mathcal{B}$, we apply the hat function $\hat{r}$ to obtain a k-dimensional vector. The function $\hat{r}$ is defined as:

$$\hat{r}(\mathcal{B}) := \left\{ \sum_{\mathbf{p}\in\mathcal{B}} \frac{1}{1+|\mathbf{p}-\mathbf{c_i}|_1} - \frac{1}{1+\left||r_i|-|\mathbf{p}-\mathbf{c_i}|_1\right|} \right\}_{i=1}^{k} \tag{3.1}$$

where $r_i \in \mathbb{R}$ and $\mathbf{c_i} \in \mathbb{R}^2$ are learnable parameters.

The intent of Equation 3.1 is to have controlled gradients. It is derived from a monotonic function, see [73]. This representation is then passed through MLP layers followed by a softmax to obtain prediction probability vector $\hat{p}_G$ for each graph G. The negative log likelihood loss from standard graph classification is then used on these vectors $\hat{p}_G$.

3.3 Efficient Computation of Extended Persistence

The computation for extended persistence can be reduced to applying a matrix reduction algorithm to a coned matrix as detailed in [51]. In [76], this computation was found to be equivalent to a graph algorithm, which we improve upon (See Algorithm D). We employ the 0-dimensional persistence algorithm, PH_0, using the union find data structure in $O(m \log n)$ time and $O(n)$ memory for the upper and lower filtrations in lines 1 and 2. See Section 3.3.1

for a description of this algorithm. These two lines generate the vertex-edge pairs for $\mathcal{B}_0^{low}$ and $\mathcal{B}_0^{up}$. We then measure the minimum lower filtration value and maximum upper filtration value of each vertex in the union-find data structure found from the PH_0 algorithm as in lines 3 and 4 using the roots of the union-find data structure U_{up} formed by the algorithm. These produce the vertex-vertex pairs in $\mathcal{B}_0^{ext}$.

Algorithm D Efficient Computation of $\mathbf{PH_{ext}}$

Input: $G = (V, E)$, F_{low}: lower filtration function, F_{up}: upper filtration function
Output: $\mathcal{B}_0^{low}, \mathcal{B}_0^{up}, \mathcal{B}_0^{ext}, \mathcal{B}_1^{ext}, \mathcal{C}$: cycle reps.
1: $\mathcal{B}_0^{low}$, $E_{pos}^{low}, E_{neg}^{low}, U_{low} \leftarrow \mathrm{PH}_0(G, F_{low}, lower)$
2: $\mathcal{B}_0^{up}$ $E_{pos}^{up}, E_{neg}^{up}, U_{up} \leftarrow \mathrm{PH}_0(G, F_{up}, upper)$
3: $roots \leftarrow \{\textsc{Get\_Union-Find\_Roots}(U_{up}, v), v \in V\}$
4: $\mathcal{B}_0^{ext} \leftarrow \{\min(roots[v]), \max(roots[v]), v \in V\}$
5: $\mathbf{T} \leftarrow \{\}$ empty link-cut tree; $\mathcal{B}_1^{ext} \leftarrow \{\{\}\}$; $\mathcal{C} \leftarrow \{\}$ empty list of cycle representatives
/* E_{neg}^{up} is sorted by PH_0 in decreasing order of F_{up} values (desc. filtr. values)*/
6: **for** $\mathrm{e} = (u, v) \in E_{neg}^{up}$ **do**
7: $\quad \mathbf{T} \leftarrow \textsc{Link}(\mathbf{T}, \mathrm{e}, \{w\})$ /* $w \notin \mathbf{T}$, $w = u$ or v*/
8: /* E_{pos}^{up} is sorted by PH_0 with respect to F_{up} (descending filtration values) */
9: **for** $\mathrm{e} = (u, v) \in E_{pos}^{up}$ **do**
10: $\quad lca \leftarrow \textsc{Lca}(\mathbf{T}, u, v)$ /*Get the least common ancestor of u and v to form a cycle*/
11: $\quad P_1 \leftarrow \textsc{ListRank}(\textsc{Path}(u, lca))$; $P_2 \leftarrow \textsc{ListRank}(\textsc{Path}(v, lca))$
12: $\quad \mathcal{C} \leftarrow \mathcal{C} \sqcup \{F_{up}(P_1) \sqcup F_{up}(Reverse(P_2))\}$ /*Keep track of the scalar activations on cycle*/
13: $\quad (u', v') \leftarrow \textsc{ArgmaxReduceCycle}(\mathbf{T}, u, v, lca)$ /* Find max edge on cycle using F_{low}*/
14: $\quad \mathbf{T_1}, \mathbf{T_2} \leftarrow \textsc{Cut}(\mathbf{T}, (u', v'))$; $\mathbf{T} \leftarrow \textsc{Link}(\mathbf{T_1}, (u, v), \mathbf{T_2})$
15: $\quad \mathcal{B}_1^{ext} \leftarrow \mathcal{B}_1^{ext} \cup \{(F_{low}(u', v'), F_{up}(u, v))\}$
16: **return** $(\mathcal{B}_0^{low}, \mathcal{B}_0^{up}, \mathcal{B}_0^{ext}, \mathcal{B}_1^{ext}, \mathcal{C})$

For computing edge-edge pairs in $\mathcal{B}_1^{ext}$ with cycle representatives, we implement the algorithm in [76] with a link-cut tree data structure that facilitates deleting and inserting edges in a spanning tree and employ a parallel algorithm to enumerate the edges in a cycle. We collect the max spanning forest T of negative edges, edges that join components, from the upper filtration by repeatedly applying the link operation $n - 1$ times in lines 6-8 in decreasing order of F_{up} values and sort the list of the remaining positive edges, which create cycles in line 9. Then, for each positive edge $\mathrm{e} = (u, v)$, in order of the upper filtration (line 10), we find the least common ancestor (lca) of u and v in the spanning forest T we are

maintaining as in line 11. Next, we apply the parallel primitive [93] of *list ranking* twice, once on the path u to lca and the other on the path v to lca in line 12. List ranking allows a list to populate an array in parallel in logarithmic time. The tensor concatenation of the two arrays is appended to a list of cycle representatives as in line 13. This is so that the cycle maintains order from u to v. We then apply an ArgMaxReduceCycle(T, u, v, lca) which finds the edge having a maximum filtration value in the lower filtration on it over the cycle formed by u, v and lca. We then cut the spanning forest at the edge (u', v'), forming two forests as in line 15. These two forests are then linked together at (u, v) as in line 15. The bar $(F_{low}(u', v'), F_{up}(u, v))$ is now found and added to the multiset $\mathcal{B}_1^{ext}$. The final output of the algorithm is four types of bars and a list of cycle representatives: $((\mathcal{B}_0^{low}, \mathcal{B}_0^{up}, \mathcal{B}_0^{ext}, \mathcal{B}_1^{ext}), \mathcal{C})$.

3.3.1 The PH_0 Algorithm

Algorithm E is the union-find algorithm that computes 0 dimensional persistent homology. It starts with n nodes, 0 edges, and a union-find data structure on n nodes. The edges are sorted in ascending order if a lower filtration function is given. Otherwise, the edges are sorted in descending order. It then proceeds to connect nearest neighbor clusters, or connected components, in a sequential fashion by introducing edges in order one at a time. Two connected components are nearest to each other if they have two nodes closer to each other than any other pair of connected components. This is achieved by iterating through the edges in sorted order and merging the connected components that they connect. When given a lower filtration function, when a connected component merges with another connected component, the connected component with the larger connected component root value has its root filtration function value a birth time. This birth time is paired with the current edge's filtration value and form a birth death pair. The smaller of the two connected component root values is used as birth time when an upper filtration function is given. The two connected components are subsequently merged in a union-find data structure by the Link operation.

Algorithm E PH_0 Algorithm

Input: $G = (V, E)$, F: filtration function, *order*: flag to denote an upper or lower filtration

Output: $\mathcal{B}_0, E_{pos}, E_{neg}, U$: H_0 bars, pos. edges, neg. edges, and union-find data structure

$U \leftarrow V$ /* a union-find data structure populated by n unlinked nodes*/
$\mathcal{B}_0 \leftarrow \{\}$ /*A multiset */
if $order = lower$ **then**
    $\mathrm{SORT}_{incr}(E)$ /*increasing w.r.t. F;*/
else
    $\mathrm{SORT}_{decr}(E)$ /*decreasing w.r.t F;*/
for $\mathrm{e} = (u, v) \in E$ **do**
    $root_u \leftarrow U.\textsc{find}(u)$
    $root_v \leftarrow U.\textsc{find}(v)$
    if $root_u = root_v$ **then**
        $E_{pos} \leftarrow E_{pos} \cup \{\mathrm{e}\}$
    else
        $E_{neg} \leftarrow E_{neg} \cup \{\mathrm{e}\}$
    if $order = lower$ **then**
        $b \leftarrow max(F(root_u), F(root_v)$
    else
        $b \leftarrow min(F(root_u), F(root_v))$
    $d \leftarrow F(\mathrm{e})$
    $\mathcal{B}_0 \leftarrow \mathcal{B}_0 \cup \{\{(b, d)\}\}$
    $U.\textsc{link}(root_u, root_v)$
return $(\mathcal{B}_0, E_{pos}, E_{neg}, U)$

3.4 Graph Classification Experiments

We perform experiments of our method on standard GNN datasets. We also perform timing experiments for our extended persistence algorithm, showing impressive scaling. Finally, we investigate cases where experimentally our method distinguishes graphs that other methods cannot, demonstrating how our method learns to surpass the WL[1] bound.

3.4.1 Experimental Setup

We perform experiments on a 48 core Intel Xeon Gold CPU machine with 1 TB DRAM equipped with a Quadro RTX 6000 NVIDIA GPU with 24 GB of GPU DRAM. For all baseline comparisons, the hyperparameters were set to their repository's standard values. In particular, all training were stopped at 100 epochs using a learning rate of 0.01 with the Adam [94] optimizer. Vertex attributes were used along with vertex degree information as initial vertex labels if offered by the dataset. We perform a fair performance evaluation by performing standard 10-fold cross validation on our datasets. The lowest validation loss is used to determined a test score on a test partition. An average±std. deviation test score over all partitions determines the final evaluation score.

The specific layers of our architecture for the neural network for our filtration function f_G is given by one or two GIN convolutional layers, with the number of layers as determined by an ablation study.

Table 3.1. Average accuracy ± std. dev. of our approach (GEFL) with and without explicit cycle representations, Graph Filtration Learning (GFL), GIN, GCN and TOGL and a readout ablation study on the four TUDatasets: DD, PROTEINS, IMDB-MULTI and MUTAG. Numbers in bold are highest in performance; bold-gray numbers show the second highest. The symbol − denotes that the dataset was not compatible with software at the time.

Model	DD	PROTEINS	IMDB-MULTI	MUTAG
GFL	75.2 ± 3.5	73.0 ± 3.0	46.7 ± 5.0	**87.2 ± 4.6**
Ours+Bars	75.5 ± 2.9	**74.9 ± 4.1**	**50.3 ± 4.7**	**88.3 ± 7.1**
Ours+bars+cycles	**75.9 ± 2.0**	**75.2 ± 4.1**	**51.0 ± 4.6**	86.8 ± 7.1
GIN	72.6± 4.2	66.5 ± 3.8	49.8± 3.0	84.6± 7.9
GCN	72.7± 1.6	59.6± 0.2	50.0 ± 2.0	73.9± 9.3
TOGL	74.7 ± 2.4	66.5 ± 2.5	44.7 ± 6.5	-
Filt.+SUM	75.0 ± 3.2	73.5± 2.8	48.0 ± 2.9	86.7± 8.0
Filt.+MAX	67.6± 3.9	68.6± 4.3	45.5 ± 3.1	70.3± 5.4
Filt.+AVG	69.5± 2.9	67.2± 4.2	46.7 ± 3.8	81.4± 7.9
Filt.+SORT	**76.9± 2.6**	72.6 ± 4.6	49.0± 3.6	85.6± 9.2

3.4.2 Performance on Real World and Synthetic Datasets

We perform experiments with the TUDatasets [95], a standard GNN benchmark. We compare with WL[1] bounded GNNs (GIN, GCN) from the PyTorch Geometric [96, 97] benchmark baseline commonly used in practice. We also compare with existing topology based methods TOGL [74] and GFL [14]. We also perform an ablation study on the readout function, comparing extended persistence as the readout function with the SUM, AVERAGE, MAX, SORT, and SET2SET [98] readout functions. The hyper parameter k is set to the 10th percentile of all datasets when sorting for the top-k nodes activations. We do not compare with [99] since its code is not available online. The performance numbers are listed in Table 3.1. We are able to improve upon other approaches for almost all cases. The real world datasets include DD, MUTAG, PROTEINS and IMDB-MULTI. DD, PROTEINS, and MUTAG are molecular biology datasets, which emphasize cycles, while IMDB-MULTI is a social network, which emphasize cliques and their connections. We use accuracy as our performance score since it is the standard for the TU datasets. In this thesis, we report experimental results that were performed by Soham Mukherjee. The full experimental results are available in [28].

3.5 Topological Deep Learning for Shape Classification

Topological Deep Learning (TDL) [84] is an emerging field that extends the use of graphs, encompassing concepts like simplicial complexes, cell complexes, and hypergraphs, to glean more comprehensive information from data. Central to TDL are deep learning models tailored to data residing on these diverse topological domains. By transcending the confines of binary-relations-only graph abstractions, TDL enables the analysis of higher-order network data, effectively capturing relationships among multiple entities.

Topological Neural Networks (TNNs) are deep learning architectures that extract knowledge from data associated with topological domains such as simplicial/cell complexes and hypergraphs. A TNN, like a GNN, is comprised of stacked layers that transform data into a series of features. In this paper we define a new TNN on combinatorial complexes, a topological domain introduced in [84] that extends hypergraphs and cell complexes.

The concept of Combinatorial Complexes (CCs) draws inspiration from various notions in geometric topology [100–102]. CCs offer a novel perspective that bridges the gap between simplicial/cell complexes and hypergraphs, as discussed in [84]. In essence, CCs provide a unified framework that captures the essential features of these structures while accommodating their variations.

The motivation behind introducing CCs is to establish a more flexible and versatile structure that can handle a wide range of complex relationships in data. While simplicial complexes and cell complexes are well-suited for certain types of data, they may fall short in representing relationships that are not strictly hierarchical or possess missing elements. On the other hand, hypergraphs offer a more general representation, but they might lack the hierarchical aspect of cell complexes. CCs, by encompassing these perspectives, create a holistic framework that can capture both hierarchical and non-hierarchical relationships in data.

We will give a brief introduction to Combinatorial Complex and Tensor Diagrams. We refer the reader to [84] for a detailed description of these concepts.

3.5.1 Combinatorial Complexes

This section reviews combinatorial complexes (CCs) [103], a class of topological domains that generalizes graphs, simplicial complexes, cell complexes, and hypergraphs.

Combinatorial Complexes offer several structural advantages. They serve as a unifying framework for higher-order networks, enabling the study and modeling of complex systems within a common framework. This unification is valuable for understanding the relationships between different types of higher-order networks, leading to insights into the underlying structure and behavior of complex systems. Furthermore, CCs provide flexibility in representing higher-order structures and conducting fine-grained message passing, making them more versatile than other higher-order networks and traditional graphs. This flexibility allows for nuanced analysis of topological properties and facilitates message passing of topological features in deep learning models, enhancing their applicability in various contexts.

Additionally, CCs provide high level of flexibility modeling relations among relations. In cases where constructing meaningful relations is challenging due to limited data support, CCs can accommodate arbitrary set-type relations, removing constraints on permissible relations. This flexibility is particularly advantageous in applications that require the consideration of hierarchical relations among higher-order structures, enabling the simultaneous examination of local and global features.

Definition 3.5.1 (CC)**.** *A* ***combinatorial complex (CC)*** *is a triple* $(\mathcal{V}, \mathcal{X}, rk)$ *consisting of a set* $\mathcal{V}$*, a subset* $\mathcal{X}$ *of* $\mathcal{P}(\mathcal{V}) \setminus \{\emptyset\}$*, and a function* $rk \colon \mathcal{X} \to \mathbb{N}$ *with the following properties:*

1. *For all* $v \in \mathcal{V}$*,* $\{s\} \in \mathcal{X}$*.*
2. *The function* rk *is order-preserving, meaning that if* $x, y \in \mathcal{X}$ *satisfy* $x \subseteq y$*, then* $rk(x) \leq rk(y)$*.*

Elements of $\mathcal{V}$ *are referred to as* ***vertices****, elements of* $\mathcal{X}$ *are known as* ***relations*** *or* ***cells****, and* rk *is denoted as the* ***rank function*** *of the CC.*

In simpler terms, a CC can be thought of as a generalization of both simplicial complexes and hypergraphs. It allows for missing elements within cells, offering greater flexibility than traditional simplicial complexes. Moreover, CCs maintain the hierarchical nature of cell complexes, providing a more structured representation compared to hypergraphs.

3.5.2 Incidence in a Combinatorial Complex (CC)

Definition 3.5.2 (Down/up-incidence neighborhood functions [84])**.** *Let* $(S, \mathcal{X}, rk)$ *be a Combinatorial Complex (CC). Two cells,* x *and* y *in* $\mathcal{X}$*, are called 'incident' if either* $x \subsetneq y$ *or* $y \subsetneq x$*. In particular, the "down-incidence neighborhood function"* $\mathcal{N}_{\searrow}(x)$ *of a cell* $x \in \mathcal{X}$ *is defined as the set:*

$$\mathcal{N}_{\searrow}(x) = \{y \in \mathcal{X} \mid y \subsetneq x\}$$

Similarly, the "up-incidence neighborhood function" $\mathcal{N}_{\nearrow}(x)$ *of cell* x *is defined as the set:*

$$\mathcal{N}_{\nearrow}(x) = \{y \in \mathcal{X} \mid x \subsetneq y\}$$

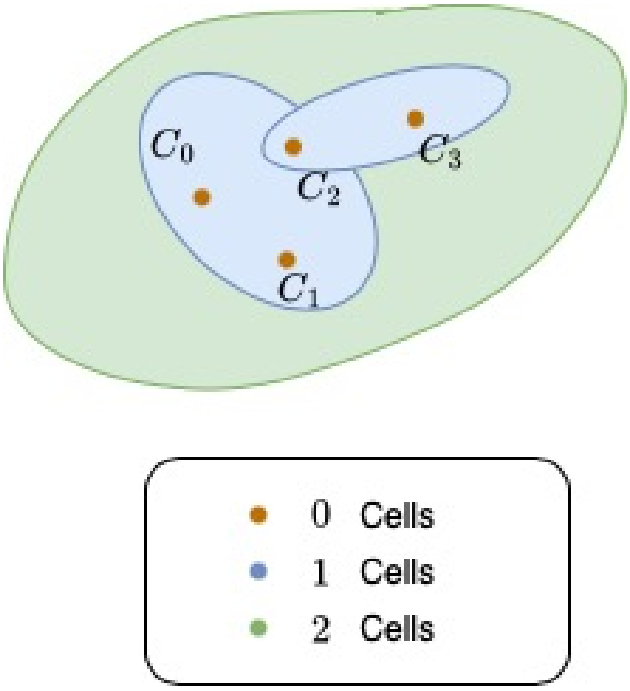

Figure 3.3. A combinatorial complex CC. The 0 cells are denoted by C_{i}. Higher order cells are represented by color schemes.

Definition 3.5.3 (k-down/up incidence neighborhood functions [84])**.** *For any $k \in \mathbb{N}$, the "k-down incidence neighborhood function" $\mathcal{N}_{\searrow,k}(x)$ of a cell $x \in \mathcal{X}$ is defined as the set:*

$$\mathcal{N}_{\searrow,k}(x) = \{y \in \mathcal{X} \mid y \subsetneq x, \text{ and } rk(y) = rk(x) - k\}$$

The "k-up incidence neighborhood function" $\mathcal{N}_{\nearrow,k}(x)$ of cell x is defined as the set:

$$\mathcal{N}_{\nearrow,k}(x) = \{y \in \mathcal{X} \mid y \subsetneq x, \text{ and } rk(y) = rk(x) + k\}$$

Definition 3.5.4 (Incidence matrix [84])**.** *For any $r, k \in \mathbb{Z}_{\geq 0}$ with $0 \leq r < k \leq dim(\mathcal{X})$, the "$(r,k)$-incidence matrix" $B_{r,k}$ between $\mathcal{X}^r$ and $\mathcal{X}^k$ is defined as a binary matrix. The (i,j)-th entry $[B_{r,k}]_{\mathrm{ij}}$ equals one if x^r_{i} is incident to x^k_{j} and zero otherwise.*

3.5.3 Adjacency in a Cell Complex (CC)

Definition 3.5.5 ((Co)adjacency neighborhood functions [84])**.** *Let $(S, \mathcal{X}, rk)$ be a Combinatorial Complex (CC). The "adjacency neighborhood function" $\mathcal{N}_a(x)$ of a cell $x \in \mathcal{X}$ is defined as the set of cells for which there exists another cell $z \in \mathcal{X}$ with rank $rk(z) > rk(x)$ such that both x and the other cell y are subsets of z:*

$$\mathcal{N}_a(x) = \{y \in \mathcal{X} \mid rk(y) = rk(x) \text{ and } \exists z \in \mathcal{X} \text{ with } rk(z) > rk(x) \text{ such that } x, y \subsetneq z\}$$

The "coadjacency neighborhood function" $\mathcal{N}_{co}(x)$ of cell x is defined as the set of cells for which there exists another cell $z \in \mathcal{X}$ with rank $rk(z) < rk(x)$ such that both z is a proper subset of x and z is a proper subset of y:

$$\mathcal{N}_{co}(x) = \{y \in \mathcal{X} \mid rk(y) = rk(x) \text{ and } \exists z \in \mathcal{X} \text{ with } rk(z) < rk(x) \text{ such that } z \subsetneq y \text{ and } z \subsetneq x\}$$

Here, a cell z that satisfies the conditions of either $\mathcal{N}_a(x)$ or $\mathcal{N}_{co}(x)$ is referred to as a "bridge cell."

Definition 3.5.6 (k-(co)adjacency neighborhood functions [84])**.** *For any $k \in \mathbb{N}$, the "k-adjacency neighborhood function" $\mathcal{N}_{a,k}(x)$ of a cell $x \in \mathcal{X}$ is defined as the set of cells for which there exists another cell $z \in \mathcal{X}$ with rank $rk(z) = rk(x) + k$ such that both x and y are subsets of z:*

$$\mathcal{N}_{a,k}(x) = \{y \in \mathcal{X} \mid rk(y) = rk(x) \text{ and } \exists z \in \mathcal{X} \text{ with } rk(z) = rk(x) + k \text{ such that } x, y \subsetneq z\}$$

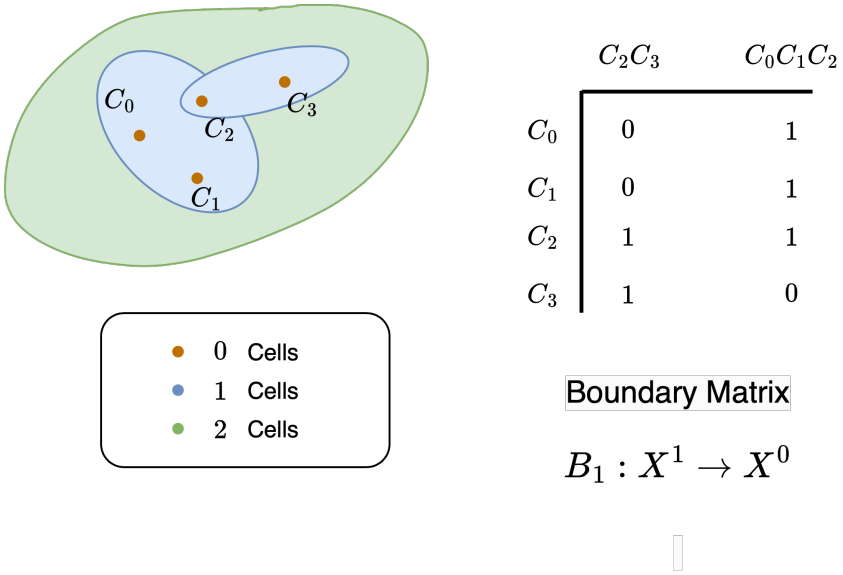

Figure 3.4. Example boundary matrix defined over a CC.

The "k-coadjacency neighborhood function" $\mathcal{N}_{co,k}(x)$ *of cell* x *is defined as the set of cells for which there exists another cell* $z \in \mathcal{X}$ *with rank* $rk(z) = rk(x) - k$ *such that* z *is a proper subset of both* y *and* x*:*

$$\mathcal{N}_{co,k}(x) = \{y \in \mathcal{X} \mid rk(y) = rk(x) \text{ and } \exists z \in \mathcal{X} \text{ with } rk(z) = rk(x) - k \text{ such that } z \subsetneq y \text{ and } z \subsetneq x\}$$

Definition 3.5.7 (Adjacency and Coadjacency matrices [84])**.** *For any* $r \in \mathbb{Z}_{\geq 0}$ *and* $k \in \mathbb{Z}_{>0}$ *with* $0 \leq r < r + k \leq dim(\mathcal{X})$*, the "*$(r, k)$*-adjacency matrix"* $A_{r,k}$ *among the cells of* $\mathcal{X}^r$ *with respect to the cells of* $\mathcal{X}^k$ *is defined as a binary matrix. The* (i, j)*-th entry* $[A_{r,k}]_{\mathrm{ij}}$ *equals one if* x^r_{i} *is* k*-adjacent to* x^r_{j} *and zero otherwise.*

For any $r \in \mathbb{Z}_{\geq 0}$ *and* $k \in \mathbb{N}$ *with* $0 \leq r - k < r \leq dim(\mathcal{X})$*, the "*$(r, k)$*-coadjacency matrix"* $coA_{r,k}$ *among the cells of* $\mathcal{X}^r$ *with respect to the cells of* $\mathcal{X}^k$ *is defined as a binary matrix. The* (i, j)*-th entry* $[coA_{r,k}]_{\mathrm{ij}}$ *has a value of* 1 *if* x^r_{i} *is* k*-coadjacent to* x^r_{j} *and* 0 *otherwise.*

3.5.4 Tensor Diagrams

In contrast to traditional graphs, which primarily deal with signals associated with nodes and edges, Topological Neural Networks (TNNs) encompass a more diverse set of signals, including higher-order relationships. Consequently, when constructing a TNN that incorporates these higher-order signals, the process involves assembling a multitude of interconnected sub-networks. This complexity arises from the need to account for a significant number of cochains within the TNN structure.

To address this complexity and aid in the comprehension of TNN architectures, we introduce the concept of tensor diagrams. Tensor diagrams serve as a visual and diagramatic notation for representing TNNs. These diagrams provide a graphical abstraction that illustrates the flow of various signals on the complex. Specifically, a tensor diagram portrays a TNN operating on a topological domain through the use of a directed graph. This directed graph illustrates how the different elements within the TNN, such as nodes, edges and faces, interact with each other to process the signals. In a tensor diagram, the signals follow a directional flow, moving from source nodes to target nodes. This flow of signals mirrors the propagation of information within the TNN.

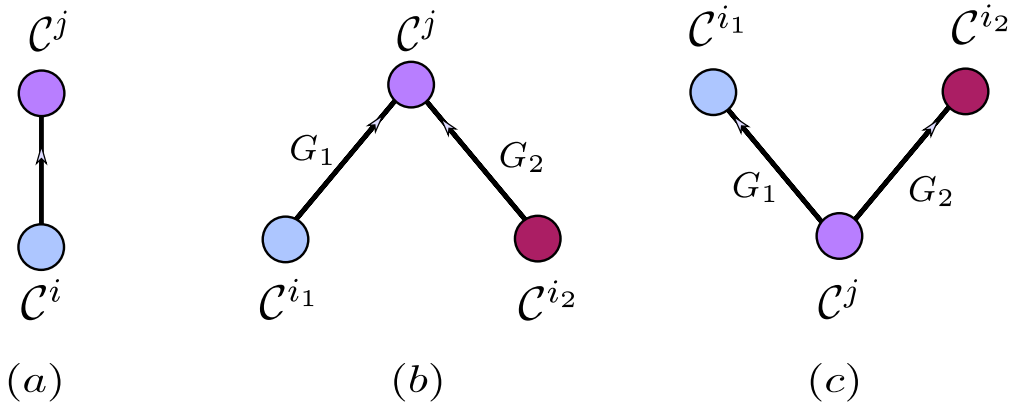

Figure 3.5. Elementary tensor diagrams. (a) Tensor diagram of the push-forward operator (b) Tensor diagram of the merge operator (c) Tensor diagram of the split operator.

3.6 Combinatorial Complex Isomorphism Networks (CCINs)

Combinatorial Complex Isomorphism Networks (CCINs) are constructed using a framework based on three fundamental tensor diagrams: merge, split, and push-forward. These tensor diagrams are the building blocks that enable the creation of CCINs for various applications.

To comprehend Combinatorial Complex Isomorphism Networks (CCINs), let's delve into three fundamental tensor operations: push-forward, merge nodes, and split nodes. These operations serve as the foundational elements for constructing CCINs using tensor diagrams.

3.6.1 Push-Forward Operator

The core of CCINs lies in the push-forward operator, often represented by a map, denoted as $G : \mathcal{C}^{\mathrm{i}} \to \mathcal{C}^{\mathrm{j}}$. This operator is pivotal for transmitting information from one layer of the network to the next. In simpler terms, it maps data from a lower-dimensional space to a higher-dimensional space.

In tensor diagrams, the push-forward operation is depicted as an arrow connecting two tensor spaces, illustrating the flow of information from input to output.

3.6.2 Merge Node

Consider situations where information originates from different sources or maps. The merge node offers a means to amalgamate these sources into a unified output. For instance, if we have two maps, $G_1 : \mathcal{C}^{\mathrm{i}_1} \to \mathcal{C}^{\mathrm{j}}$ and $G_2 : \mathcal{C}^{\mathrm{i}_2} \to \mathcal{C}^{\mathrm{j}}$, we can merge their outputs into a single output space.

In tensor diagrams, the merge node is represented by multiple arrows converging into a single output space, symbolizing the fusion of information.

3.6.3 Split Node

Conversely, the split node enables the division of a single source of information into multiple paths. Suppose we have a map $G_1 : \mathcal{C}^{\mathrm{j}} \to \mathcal{C}^{\mathrm{i}_1}$ and another map $G_2 : \mathcal{C}^{\mathrm{j}} \to \mathcal{C}^{\mathrm{i}_2}$; a split node will separate the input into two distinct streams based on these maps.

In tensor diagrams, a split node is depicted as a single input branching into multiple paths, illustrating how data diverges.

3.6.4 Elementary Tensor Operations

Collectively, these three operationspush-forward, merge nodes, and split nodesare referred to as elementary tensor operations. They serve as the fundamental tools for constructing complex CCINs. Think of them as the building blocks of CCIN construction. By arranging and connecting these operations, we can create intricate neural network architectures.

3.6.5 Flexibility in CCIN Design

What's intriguing is that with just these three elementary tensor operations, we can define a wide range of topological neural networks. For example, we can define convolutional and attention-based CCINs by specifying the push-forward and merge operators. Beyond these standard architectures, these operations offer limitless possibilities for creating novel topological neural networks.

3.6.6 CCIN Architecture

In Figure 3.6 we demonstrate the design of CCIN schematically. Cell features of rank $0, 1$ and 2 are passed through an attention layer first. Then we use *merge node* to merge messages coming from 0 to 1 cells and 1 to 2 cells respectively. The merged signals are then again merged through neighborhood matrices (B_2^T and $A_{\downarrow,2}$). The final classification is obtained after a *mean* pooling layer and an MLP. Note that the message passing layer is a

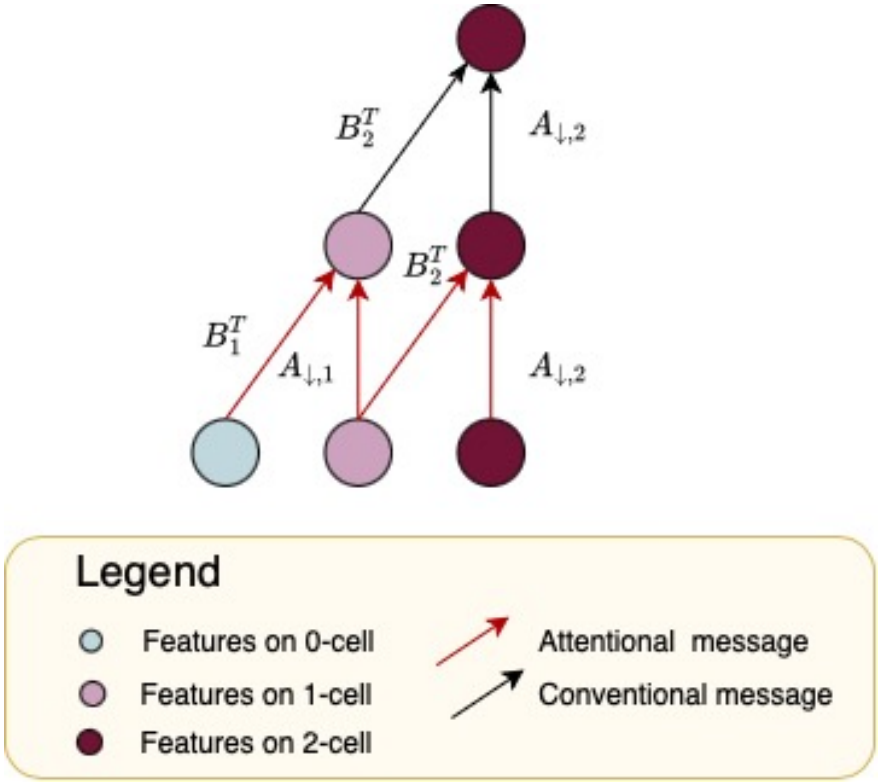

Figure 3.6. Tensor diagram of CCIN network. In the figure B_r denotes incidence matrix, B_r^T denotes transpose of the incidence matrix, $L_{\downarrow,r} = B_r B_r^T$ denotes rth down Laplacian matrix and $A_{\downarrow,r} = D_{\downarrow,r} - L_{\downarrow,r}$.

generalization of *graph isomorphism network* (GIN). Specifically, if a message $m_{y\to x}^{(r'\to r)}$ travels from r' cell x to r cell y through a neighborhood k of x denoted as $\mathcal{N}_k(x)$, then

$$m_{y\to x}^{(r'\to r)} = M_{\mathcal{N}_k(x)}\Big(h_x^{t,(r)}, h_y^{t,(r')}, \Theta^t\Big) \tag{3.2}$$

In Equation 3.6.6, $h_x^{t,(r)}, h_y^{t,(r')}$ are features and Θ^t are learnable parameters. $M_{\mathcal{N}_k}$ are commonly used non-linearity composed with MLP. Then CCIN updates its node represntations as

$$h_x^{t,(r)} = MLP^{(t)}\Big((1+\varepsilon^{(t)})\cdot h_x^{t-1,(r)} + AGG_{\mathcal{N}_k\in\mathcal{N}} m_x^{(r'\to r)}\Big) \tag{3.3}$$

In Equation 3.6.6, ε^t is a trainable parameter at layer t and AGG is commonly used permutation invariant aggregate operation such as sum, max, mean etc.

3.7 Experiments on Noisy and Non-manifold Dataset

We use the flexibility offered by CCINs on noisy meshes which may or may not be manifolds. The existing approaches [81, 82, 104] assume that meshes are manifolds and derive mesh features based on this assumption. However, we use features that do not depend on the 'manifoldness' of the mesh with CCIN architecture and obtain comparable results to state-of-the-art models. To the best of our knowledge, this is one of the first works to deal with non-manifold meshes. Further, we also present results on how well our model performs in a transfer learning framework where the model is trained on manifold mesh data and is evaluated on noisy test data which violates the manifold condition. We observe that the model generalizes well.

We provide an overview of the datasets used followed by mesh classification results.

3.7.1 Datasets

For this chapter we use SHREC11 originally made available by [82].

- **SHREC11 Classification Dataset.** SHREC 2011 [105] is a dataset containing 600 nonrigid deformed shapes from 30 categories, with 480 shapes in the training set and 120 in the test set. We use this dataset for mesh classification.

3.7.2 Results on Mesh Classification

Table 3.2. Mesh Classification Accuracies on SHREC-11 dataset. CCIN denotes that the model used vertex, edge and face features.

Model	Accuracy
HodgeNet	99.10
PD-MeshNet	99.70
MeshCNN	98.60
CCIN (Ours)	92.70

In the previous section we introduced CCIN, a new architecture and message passing scheme for combinatorial complexes. In this section, we demonstrate the use of this architecture on 'non-manifold' and noisy mesh classification.

We used SHREC-11 dataset which is a 30-class mesh dataset. On the vertices of these meshes we consider the first 15 eigen values of the graph laplacian, vertex positions and vertex normals as vertex features, edge length as edge feature and face angles, face normal and face area as face features. The dataset is very small and has 16 training samples per class. We augment the train data with 30 random rotations of each mesh. We report the accuracy in Table 3.2.

We note that the vertex, edge and face features we used do not depend on the fact that the mesh is a manifold, e.g., each edge need not be shared only by two faces. We see that the model achieves comparable performance to other state-of-the-art methods with these features and the CCIN architecture. We perform a series of experiments to test our model on non-manifold meshes and noisy meshes. Since this is one of the first works to deal with non-manifold meshes, we provide a comparison of the performance of our model in different noise ratio and non-manifold ratio settings. We report the results in 3.3

Table 3.3. Mesh Classification Accuracies on noisy and non-manifold SHREC-11 dataset.

Noise ratio	**Non-manifold ratio**	**CCIN Accuracy**
0.01	0.01	84.17
0.01	0	85.83
0	0.001	92.50
0	0.005	84.17
0	0.01	84.17
0	0.05	80.00

In order to incorporate noise in the meshes, we select a percentage (denoted as noise ratio) of vertices and perturb their positions by adding uniform noise. For making a mesh non-manifold, we select a percentage of faces and at these faces, we create a new vertex and a new face.

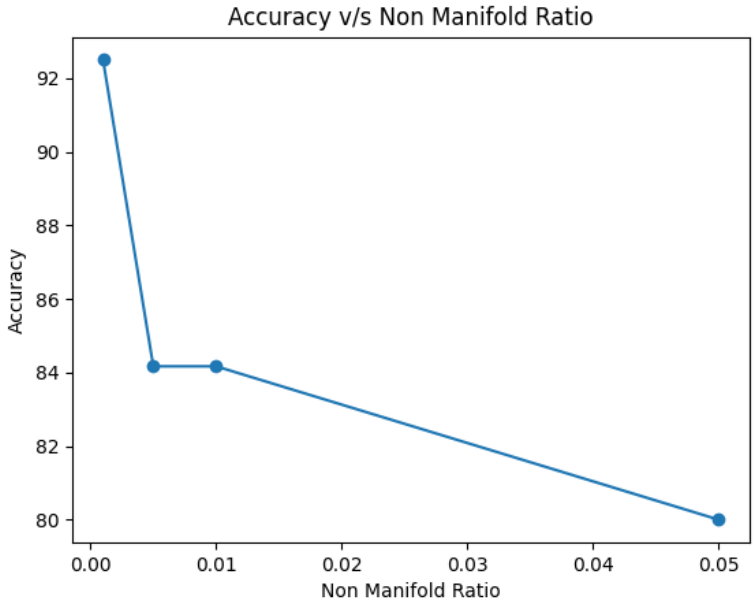

Figure 3.7. Plot of accuracy vs non-manifold ratio.

In Figure 3.7, we reported the test accuracy on noisy and non-manifold meshes when the model is trained on noisy and non-manifold meshes. We perform another series of experiments to test how well our model generalizes. We train the model on non-noisy and manifold (clean) data and test it on noisy and non-manifold data to test if the learning can be transferred. We report the results in Table 3.4

Table 3.4. Mesh Classification Accuracies on noisy and non-manifold SHREC-11 dataset. CCIN-V refers to the model which uses only vertex features. CCIN-Mani denotes that the model is trained on clean (manifold) data and inference is done on noisy and non-manifold data.

Model	Noise ratio	Non-manifold ratio	Accuracy
CCIN-Mani	0.01	0.01	74.17
CCIN-Mani	0.01	0	90.83
CCIN-Mani	0	0.01	75.83

3.8 Concluding remarks

In this chapter we introduced extended persistence into the supervised learning framework, bringing in crucial global connected component and cycle measurement information into the graph representations. We address a fundamental limitation of MPGNNs, which is their inability to measure cycles lengths. Our method hinges on an efficient algorithm for computing extended persistence. This is a parallel differentiable algorithm with an $O(m \log n)$ depth $O(mn)$ work complexity and scales impressively over the state-of-the-art. The speed with which we can compute extended persistence makes it feasible for machine learning. Furthermore, we have introduced a new architecture CCIN in the realm of TDL. By performing a series of experiments we have shown its effectiveness on mesh-classification. Although we have provided empirical results on non-manifold and noisy mesh processsing, we would like to point out that our framework is general. We hope that this contribution in the field of TDL will encourage further discussions to find applications in diverse fields, such as protein-protein interaction networks or drug-protein interaction analyzing social networks, communication patterns, or collaboration networks.

4. EXPLORING GRIL - EXPERIMENTATION AND IMPLEMENTATION OF A 2-PARAMETER PERSISTENCE BASED VECTORIZATION

Machine learning models such as Graph Neural Networks (GNNs) [106–109] are well-known successful tools from the geometric deep learning community. Some recent research has indicated that the representation power of such models can be augmented by infusing topological information [12, 15, 110, 111]. One way to do that is by applying *persistent homology*, which is a powerful tool for characterizing the shape of data, rooted in the theory of algebraic topology. It has spawned the flourishing area of Topological Data Analysis. The classical persistent homology, also known as, 1-parameter *persistence module*, has attracted plenty of attention from both theory and applications [51, 112–114].

In essence, a 1-parameter persistence homology captures the evolution of some topological information within a topological space $\mathcal{X}$ along an ascending filtration determined by a scalar function $\mathcal{X} \to \mathbb{R}$. It can be losslessly summarized by a complete discrete invariant such as a *persistence diagram*, *rank invariant* or *barcode*. In recent years, many works have successfully integrated persistence homology with machine learning models [12, 14, 15, 23–35].

To further enhance the capacity of persistent homology, it is natural to consider a more general multivariate filtration function $\mathcal{X} \to \mathbb{R}^d$ for $d \geq 2$ in place of a real valued function, and represent its topological information by multiparameter persistence modules. However, the structure of multiparameter persistence modules is much more complicated than 1-parameter persistence modules. In 1-parameter case, the modules are completely characterized by what is called *barcode* or *persistence diagram* [36, 37]. Unfortunately, there is no such discrete complete invariant which can summarize multiparameter persistence modules completely [38]. Given this limitation, building a useful vector representation from multiparameter persistence modules while capturing as much topological information as possible for machine learning models becomes an important but challenging problem.

To address this challenge, different kinds of vector representations have been proposed for 2-parameter persistence modules [23, 24, 35].

All these works are essentially based on the invariant called fibered (sliced) barcodes [115].

However, such representations capture as much topological information as determined by the well-known incomplete summary called rank invariant [38] which is equivalent to fibered barcodes.

In this paper, we propose a new vector representation to extend its expressive power in terms of capturing topological information from a 2-parameter persistence module:

- We introduce *Generalized Rank Invariant Landscape* (Gril), a new vector representation encoding richer information beyond fibered barcodes for 2-parameter persistence modules, based on the idea of *generalized rank invariant* [39] and its computation by zigzag persistence [40]. The construction of Gril can be viewed as a generalization of persistence landscape [16, 24], hence has more discriminating power.

- We show that this vector representation Gril is 1-Lipschitz stable and differentiable with respect to the filtration function f, which allows one to build a topological representation as a machine learning model.

- We propose an efficient algorithm to compute (Gril), demonstrate its use on synthetic and benchmark graph datasets, and compare the results with previous vector representations of 1-parameter and 2-parameter persistence modules. Specifically, we present results indicating that GNNs may improve when augmented with Gril features for graph classification task.

4.1 Background

In this section, we start with an overview of single and multiparameter persistence modules followed by formal definitions of basic concepts. Then we provide a high-level idea of how to construct our vector representation Gril. For a more comprehensive introduction to persistence modules, we refer the interested readers to [51, 112–114].

The standard pipeline of 1-parameter persistence module is as follows: Given a domain of interest $\mathcal{X}$ (e.g. a topological space, point cloud data, a graph, or a simplicial

complex) with a scalar function $f : \mathcal{X} \to \mathbb{R}$, one filters the domain $\mathcal{X}$ by the sublevel sets $\mathcal{X}_\alpha \triangleq \{x \in \mathcal{X} \mid f(x) \leq \alpha\}$ along with a continuously increasing threshold $\alpha \in \mathbb{R}$. The collection $\{\mathcal{X}_\alpha\}$, which is called a *filtration*, forms an increasing sequence of subspaces $\emptyset = \mathcal{X}_{-\infty} \subseteq \mathcal{X}_{\alpha_1} \subseteq \cdots \subseteq \mathcal{X}_{+\infty} = \mathcal{X}$. Along with the filtration, topological features appear, persist, and disappear over a collection of intervals. We consider pth homology groups $H_p(-)$ over a field, say $\mathbb{Z}_2$, of the subspaces in this filtration, which results into a sequence of vector spaces. These vector spaces are connected by inclusion-induced linear maps forming an algebraic structure $0 = H_p(\mathcal{X}_{-\infty}) \to H_p(\mathcal{X}_{\alpha_1}) \to \cdots \to H_p(\mathcal{X}_{+\infty})$. (see [116]). This algebraic structure, known as 1-parameter persistence module induced by f and denoted as M^f, can be uniquely decomposed into a collection of atomic modules called interval modules, which completely characterizes the topological features in regard to the three behaviors–appearance, persistence, and disappearance of all p-dimensional cycles. This unique decomposition of a 1-parameter persistence module is commonly summarized as a complete discrete invariant, *persistence diagram* [85] or *barcode* [117]. Figure 4.1 (left) shows a filtration of a simplicial complex that induces a 1-parameter persistence module and its decomposition into bars.

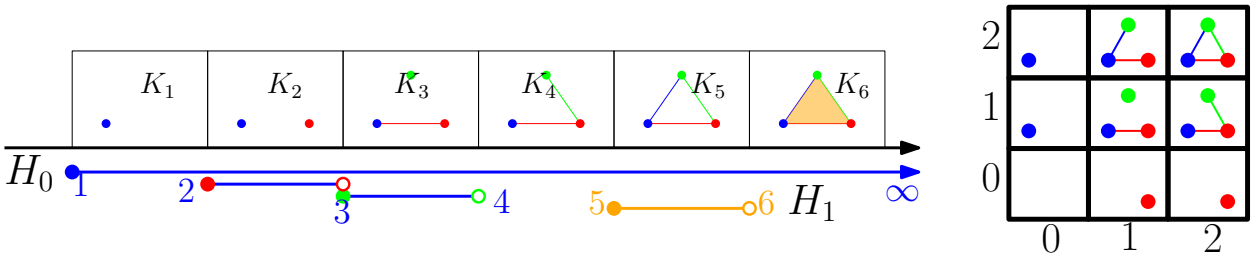

Figure 4.1. (left) 1-parameter filtration and bars; (right) a 2-parameter filtration inducing a 2-parameter persistence module whose decomposition is not shown.

Some problems in practice may demand tracking the topological information in a filtration that is not necessarily linear. For example, in [118], 2-parameter persistence modules are shown to be better for classifying hepatic lesions compared to 1-parameter persistence modules. In [27, 119], a virtual screening system based on 2-parameter persistence modules are shown to be effective for searching new candidate drugs. In such applications, instead of studying a sequential filtration filtered by a scalar function, one may study a grid-filtration

induced by a $\mathbb{R}^2$-valued bi-filtration function $f : \mathcal{X} \to \mathbb{R}^2$ with $\mathbb{R}^2$ equipped with partial order $\boldsymbol{u} \leq \boldsymbol{v} : u_1 \leq v_1, u_2 \leq v_2$; see Figure 4.1(right) for an example of 2-parameter filtration. Following a similar pipeline as the 1-parameter persistence module, one will get a collection of vector spaces $\{M^f_{\boldsymbol{u}}\}_{\boldsymbol{u}\in\mathbb{R}^2}$ indexed by vectors $\boldsymbol{u} = (u_1, u_2) \in \mathbb{R}^2$ and linear maps $\{M^f_{\boldsymbol{u}\to\boldsymbol{v}} : M^f_{\boldsymbol{u}} \to M^f_{\boldsymbol{v}} \mid \boldsymbol{u} \leq \boldsymbol{v} \in \mathbb{R}^2\}$ for all comparable $\boldsymbol{u} \leq \boldsymbol{v}$. The entire structure M^f, in analogy to the 1-parameter case, is called a 2-parameter persistence module induced from f.

Unlike 1-parameter case, there is no *complete* discrete invariant like persistence diagrams or barcodes that can losslessly summarize the whole structure of 2-parameter persistence modules [38]. A good non-complete invariant for 2-parameter persistence modules should characterize many non-isomorphic topological features, ideally as many as possible. At the same time, it should be stable with respect to small perturbations of filtration functions, which guarantees its important properties of continuity and differentiability for machine learning models. Therefore, building a good summary in general for 2-parameter persistence modules which is also applicable to machine learning models is an important and challenging problem.

We now formally define some of the concepts discussed above.

Definition 4.1.1 (Simplicial Complex)**.** *An abstract simplicial complex is a pair* (V, Σ) *where* V *is a finite set and* Σ *is a collection of non-empty subsets of* V *such that if* $\sigma \in \Sigma$ *and if* $\tau \subseteq \sigma$ *then* $\tau \in \Sigma$*. A topological space* $|(V, \Sigma)|$ *can be associated with the simplicial complex which can be defined using a bijection* $t\colon V \to \{1, 2, \ldots, \|V\|\}$ *as the subspace of* $\mathbb{R}^{\|V\|}$ *formed by the union* $\bigcup_{\sigma\in\Sigma} h(\sigma)$*, where* $h(\sigma)$ *denotes the convex hull of the set* $\{\mathrm{e}_{t(s)}\}_{s\in\sigma}$*, where* e_{i} *denotes the standard basis vector in* $\mathbb{R}^{\|V\|}$*.*

Definition 4.1.2 (Simplicial Filtration)**.** *A d-parameter* simplicial filtration *over* $\mathbb{R}^d$ *for some* $d \in \mathbb{Z}_+$ *is a collection of simplicial complexes* $\{X_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{R}^d}$ *with inclusion maps* $X_{\mathbf{u}} \hookrightarrow X_{\mathbf{v}}$ *for* $\mathbf{u} \leq \mathbf{v}$*, that is,* $u_1 \leq u_2$ *and* $v_1 \leq v_2$ *where* $\mathbf{u} = (u_1, u_2)$ *and* $\mathbf{v} = (v_1, v_2)$*. When* $d = 2$*, it is also called bi-filtration,*

Definition 4.1.3 (Persistence Module)**.** *A d-parameter Persistence Module is a collection of vector spaces* $\{X_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{R}^d}$ *indexed by* $\mathbb{R}^d$*, together with a collection of linear maps* $\{M_{\boldsymbol{u}\to\boldsymbol{v}} : M_{\mathbf{u}} \to M_{\mathbf{v}} \mid \boldsymbol{u} \leq \boldsymbol{v} \in \mathbb{R}^d\}$ *such that* $M_{\boldsymbol{v}\to\boldsymbol{w}} = M_{\boldsymbol{v}\to\boldsymbol{w}} \circ M_{\boldsymbol{u}\to\boldsymbol{v}}, \forall \boldsymbol{u} \leq \boldsymbol{v} \leq \boldsymbol{w}$*.*

Remark 4.1.1. *In this paper, we study* 1 *and* 2*-parameter persistence modules for* $d = 1, 2$*. Each* $M_{\boldsymbol{u}}$ *is the homology vector space of* $X_{\boldsymbol{u}}$ *in a simplicial (bi-)filtration. And each* $M_{\boldsymbol{v} \to \boldsymbol{w}}$ *is the induced linear map from the inclusion* $X_{\boldsymbol{u}} \hookrightarrow X_{\boldsymbol{v}}$*.*

We now define the notion of an interval in $\mathbb{R}^2$. In the definition, we shall make use of the standard partial order on $\mathbb{R}^2$, i.e., $\boldsymbol{u} \leq \boldsymbol{v}$ if $u_1 \leq v_1$ and $u_2 \leq v_2$ for $\boldsymbol{u} = (u_1, u_2)$ and $\boldsymbol{v} = (v_1, v_2)$.

Definition 4.1.4 (Interval)**.** *A connected subset* $\emptyset \neq I \subseteq \mathbb{R}^2$ *is an* interval *if* $\forall \boldsymbol{u} \leq \boldsymbol{v} \leq \boldsymbol{w}, [\boldsymbol{u} \in I, \boldsymbol{w} \in I] \implies [\boldsymbol{v} \in I]$*.*

We also give the definition of a zigzag filtration and the zigzag persistence module induced by it as follows:

Definition 4.1.5 (Zigzag filtration)**.** *A* zigzag filtration *is a sequence of simplicial complexes where both insertions and deletions of simplices are allowed, the possibility of which we indicate with double arrows:*

$$X_0 \leftrightarrow X_1 \leftrightarrow \cdots \leftrightarrow X_n = \mathcal{X}.$$

Applying homology functor on such a filtration we get a zigzag persistence module that is a sequence of vector spaces connected either by forward or backward linear maps:

$$H_*(X_0) \leftrightarrow H_*(X_1) \leftrightarrow \cdots \leftrightarrow H_*(X_n).$$

We end this section by providing an overview of some high-level ideas for constructing our vector representation Gril.

Overview:

Our approach computes a *landscape function* over the 2-parameter domain and then vectorizes it. At this high level, this is similar to the approach in [24]. However, the

landscape function we construct is much more general and thus potentially has the power of capturing more topological information. In particular, we use the concept of *generalized rank invariant* introduced in [39], which indeed generalizes the traditional rank invariant used in [24]. As opposed to simple rank invariant which is defined over rectangles, generalized ranks are defined over their generalizations called *intervals*. We define it more formally in section 4.2 below.

One difficulty facing the use of the generalized ranks in TDA was that its efficient computation was not known. Recently, in [40], the authors showed that generalized ranks for intervals in 2-parameter persistence modules can be obtained by considering a persistence module supported on a linear poset induced by the boundary of the interval in question. However, this linear poset is not totally ordered as in 1-parameter persistence, and thus gives rise to what is called *zigzag persistence* [120] where the inclusions can both be in forward and backward directions unlike traditional 1-parameter persistence where they are only in forward directions; With this result, computing generalized ranks efficiently boils down to computing zigzag persistence efficiently. For this purpose, we use a recently discovered fast zigzag algorithm and its efficient implementation [121][1].

Our method samples a subset of grid points from the 2-parameter grid spanned by a given bi-filtration function, and computes the landscape function values (Definition 4.2.1) at those points based on generalized ranks. For this, the algorithm considers an expanding sequence of intervals which we call *worms* centered at each point $\boldsymbol{p}$ and computes generalized rank over them to determine the 'width' of the maximal worm sustaining a chosen rank. This maximization is achieved by a binary search over the sequence of worms centering $\boldsymbol{p}$; section 4.3 describes this procedure. The widths, thus computed for each sample point, constitute the landscape function values which become the basis for our vector representation.

4.2 Generalized Rank Invariant Landscape

In this section, we introduce Generalized Rank Invariant Landscape, abbreviated as Gril, a stable and differentiable vector representation of 2-parameter persistence modules.

[1] ↑https://github.com/taohou01/fzz

Let $M = M^f$ be a 2-parameter persistence module induced by a filtration function f. The restriction of M to an interval I, denoted as $M|_I$, is the collection of vector spaces $\{M_{\boldsymbol{u}} \mid \boldsymbol{u} \in I\}$ along with linear maps $\{M_{\boldsymbol{u}\to\boldsymbol{v}} \mid \boldsymbol{u} \le \boldsymbol{v} \in I)\}$.

One can define the generalized rank of $M|_I$ as the rank of the canonical linear map from limit $\varprojlim M|_I$ to colimit $\varinjlim M|_I$ of $M|_I$:

$$\mathsf{rk}^M(I) \triangleq \operatorname{rank}\left[\varprojlim M|_I \to \varinjlim M|_I\right]$$

A formal explanation of limit and colimit is beyond the scope of this article; we refer readers to [122] for their definitions and also the construction of the canonical limit-to-colimit map in category theory. Intuitively, $\mathsf{rk}^M(I)$ captures the number of *independent* topological features encoded in M with the support over the entire interval I. Specially, when $I = [\boldsymbol{u}, \boldsymbol{v}] \triangleq \{\boldsymbol{w} \in \mathbb{R}^2 \mid \boldsymbol{u} \le \boldsymbol{w} \le \boldsymbol{v}\}$ is a rectangle, $\varprojlim M|_I = M_{\boldsymbol{u}}$ and $\varinjlim M|_I = M_{\boldsymbol{v}}$. Then $\mathsf{rk}^M(I)$ equals the traditional rank of the linear map $M_{\boldsymbol{u}\to\boldsymbol{v}}$.

Remark 4.2.1. *An interesting property of the generalized rank invariant is that its value over a larger interval is less than or equal to its value over any interval contained inside the larger interval. Formally,* $I \subseteq J \implies \mathsf{rk}^M(I) \ge \mathsf{rk}^M(J)$. *We implicitly use this* monotone *property in the definition of* Gril.

The basic idea of Gril is to consider a collection of generalized ranks $\{\mathsf{rk}^M(I)\}_{I\in\mathcal{W}}$ over some covering set $\mathcal{W}$ on $\mathbb{R}^2$, which is called a *generalized rank invariant* of M over $\mathcal{W}$.

Let $\boxed{\boldsymbol{p}}_\delta \triangleq \{\boldsymbol{w} : \|\boldsymbol{p} - \boldsymbol{w}\|_\infty \le \delta\}$ be the δ-square centered at $\boldsymbol{p}$ with side 2δ.

For given $\boldsymbol{p} \in \mathbb{R}^2, \ell \ge 1, \delta > 0$, we define an *$\ell$-worm* $\boxed{\boldsymbol{p}}_\delta^\ell$ to be the union over all δ-squares $\boxed{\boldsymbol{q}}_\delta$ centered at some point $\boldsymbol{q}$ on the off-diagonal line segment $\boldsymbol{p} + \alpha \cdot (1, -1)$ with $|\alpha| \le (\ell - 1)\delta$. See Figure 4.3 for an illustration.

Formally,

$$\boxed{\boldsymbol{p}}_\delta^\ell \triangleq \bigcup_{\substack{\boldsymbol{q}=\boldsymbol{p}+(\alpha,-\alpha)\\ |\alpha|\le(l-1)\delta}} \boxed{\boldsymbol{q}}_\delta$$

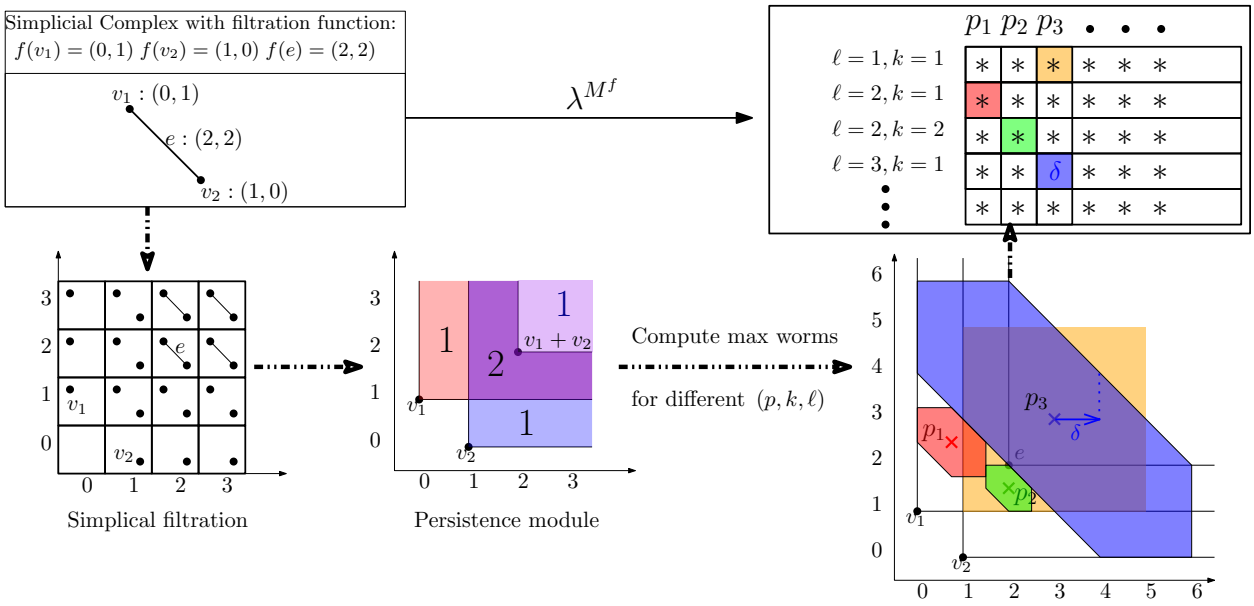

Figure 4.2. The construction starts from a simplicial complex with a bi-filtration function as shown on the top left. The simplicial complex consists of two vertices connected by one edge. Based on the bi-filtration, a simplicial bi-filtration can be defined as shown on the bottom left. On the mid bottom, a 2-parameter persistence module is induced from the above simplicial filtration. If we check the dimensions of the vector spaces on all points of the plane, there are 1-dimensional vector spaces on red, blue and light purple regions. On the L-shaped dark purple region, the vector spaces have dimension 2. For this 2-parameter persistence module, we calculate $\lambda^{M^f}(\boldsymbol{p}, k, \ell)$ for all tuples $(\boldsymbol{p}, k, \ell) \in \mathcal{P} \times K \times L$ to get our GRIL vector representation. By Definition 4.2.1 the value $\lambda^{M^f}(\boldsymbol{p}, k, \ell)$ corresponds to the width of the maximal ℓ-worm on which the generalized rank is at least k. On the bottom right, the interval in red is the maximal 2-worm for $\lambda^{M^f}(\boldsymbol{p}_1, k = 1, \ell = 2)$. The green interval is the maximal 2-worm for $\lambda^{M^f}(\boldsymbol{p}_2, k = 2, \ell = 2)$. The yellow square is the maximal 1-worm for $\lambda^{M^f}(\boldsymbol{p}_3, k = 1, \ell = 1)$, and the blue interval is the maximal 3-worm for $\lambda^{M^f}(\boldsymbol{p}_3, k = 1, \ell = 3)$. Finally, on the top right, we have our GRIL vector representation λ^{M^f} which is a collection of vectors. Each vector corresponding to a different ℓ and k consists of values as the width of maximal worms at each center point $\boldsymbol{p}$. As an example, the blue one on the last vector at position p_3 has value δ which is the width of the blue worm.

We call $\boldsymbol{p}$ the *center point* and δ the *width* of the ℓ-worm $\boxed{\boldsymbol{p}}_\delta^\ell$. As a special case, when $\ell = 1$, $\boxed{\boldsymbol{p}}_\delta^1 = \boxed{\boldsymbol{p}}_\delta$ is just the δ-square with side 2δ.

We choose $\mathcal{W}$ to be a set of *Worms* defined as follows:

$$\mathcal{W} \triangleq \left\{ W = \boxed{\boldsymbol{p}}_{\delta}^{\ell} \mid \delta > 0, \ell \geq 1, \boldsymbol{p} \in \mathbb{R}^2 \right\}$$

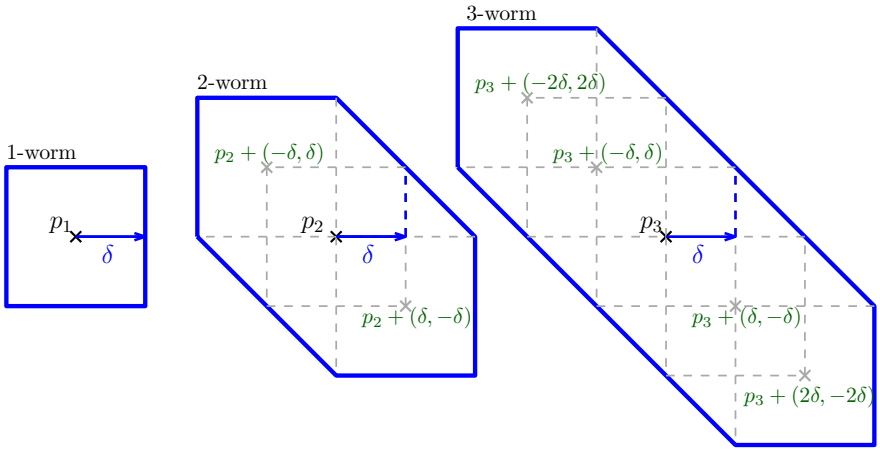

Figure 4.3. Examples of three ℓ-worms with $\ell = 1, 2, 3$.

Now we are ready to define the main construction in this paper which uses the monotone property of generalized rank mentioned in Remark 4.2.1.

Definition 4.2.1 (Generalized Rank Invariant Landscape (GRIL))**.** *For a persistence module M, the* Generalized Rank Invariant Landscape (GRIL) *of M is a function $\lambda^M : \mathbb{R}^2 \times \mathbb{N}_+ \times \mathbb{N}_+ \to \mathbb{R}$ defined as*

$$\lambda^M(\boldsymbol{p}, k, \ell) \triangleq \sup_{\delta \geq 0} \{ \mathsf{rk}^M(\boxed{\boldsymbol{p}}_{\delta}^{\ell}) \geq k \}. \tag{4.1}$$

We can see from the definition that given a persistence module M, a point $\boldsymbol{p}$, a rank k and ℓ, the value of GRIL ($\lambda^M(\boldsymbol{p}, k, \ell)$) is, in essence, the width δ of the "maximal" ℓ-worm $W = \boxed{\boldsymbol{p}}_{\delta}^{\ell}$ centered at $\boldsymbol{p}$ such that the value of the generalized rank over W is greater than or equal to k. See Figure 4.2 bottom right for some examples of maximal worms.

It turns out that, GRIL as an invariant is equivalent to the generalized rank invariant over $\mathcal{W}$.

Proposition 4.2.1. *GRIL is equivalent to the generalized rank invariant over $\mathcal{W}$. Here the equivalence means bijective reconstruction from each other.*

Proof. Constructing GRIL from generalized rank invariant over $\mathcal{W}$ is immediate from the definition of GRIL.

On the other direction, for any $\boldsymbol{p}, \delta, \ell$, the generalized rank $\mathsf{rk}^M(\boxed{\boldsymbol{p}}_\delta^\ell)$ can be reconstructed by GRIL as follows:

$$\mathsf{rk}^M(\boxed{\boldsymbol{p}}_\delta^\ell) = \arg\max_k \{\lambda(\boldsymbol{p}, k, \ell) \geq \delta\} \tag{4.2}$$

It is not hard to check that, this construction, combined with the construction of persistence landscape, gives a bijective mapping between generalized rank invariants over $\mathcal{W}$ and GRILs. □

See Figure 4.2 for an illustration of the overall pipeline of our construction of λ^M starting from a filtration function on a simplicial complex. Figure 4.4 shows the discriminating power of GRIL where we see that GRIL can differentiate between shapes that are topologically non-equivalent.

4.3 Algorithm

We present our algorithm to compute GRIL in this section.

In practice, we choose center points $\boldsymbol{p}$ from some finite subset $\mathcal{P} \subset \mathbb{R}^2$, e.g. a finite uniform grid in $\mathbb{R}^2$, and consider $k \leq K, \ell \leq L$ for some fixed $K, L \in \mathbb{N}_+$. Then, GRIL $\{\lambda^M(\boldsymbol{p}, k, \ell)\}$ can be viewed as a vector of dimension $|\mathcal{P}| \times K \times L$.

The high-level idea of the algorithm is as follows: Given a bi-filtration function $f : \mathcal{X} \to \mathbb{R}^2$, for each triple $(\boldsymbol{p}, k, \ell) \in \mathcal{P} \times K \times L$, we compute $\lambda^{M^f}(\boldsymbol{p}, k, \ell) = \sup_{\delta \geq 0}\{\mathsf{rk}^{M^f}(\boxed{\boldsymbol{p}}_\delta^\ell) \geq k\}$. In essence, we need to compute the maximum width over worms on which the generalized rank is at least k. In order to find the value of this width, we use binary search. We compute generalized rank $\mathsf{rk}^{M^f}\left(\boxed{\boldsymbol{p}}_\delta^\ell\right)$ by applying the algorithm proposed in [40], which uses zigzag persistence on a boundary path. This zigzag persistence is computed efficiently by a recent algorithm proposed in [121]. We denote the sub-routine to compute generalized rank over a worm by COMPUTERANK in algorithm F mentioned below. COMPUTERANK(f, I) takes as input a bi-filtration function f and an interval I, and outputs generalized rank over that interval. In order to use the algorithm proposed in [40], the worms need to have their

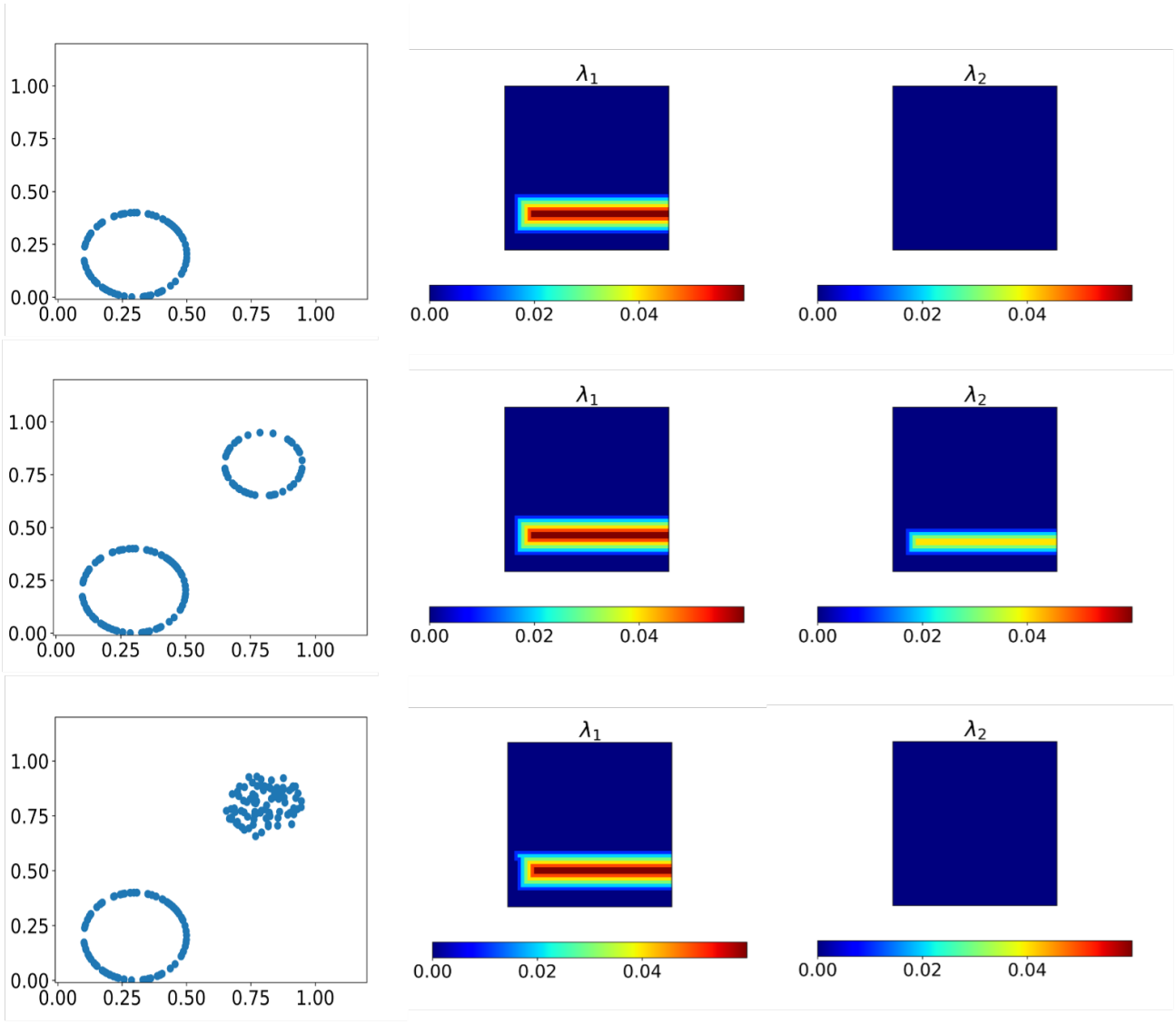

Figure 4.4. GRIL as a topological discriminator: each row shows a point cloud P, GRIL value heatmap for ranks $k = 1$ and $k = 2$ in homology of degree 1 named as λ_1 and λ_2 respectively. First Betti number (β_1) of a circle is 1 which is reflected in λ_1 being non-zero. β_1 for two circles is 2 which is reflected in both λ_1 and λ_2 being non-zero. Similarly, β_1 of a circle and disk together is 1 which is reflected in λ_1 being non-zero but λ_2 being zero for this point cloud.

boundaries aligned with a grid structure defined on the range of f. Thus, we normalize f to be in the range $[0, 1] \times [0, 1]$, define a grid structure on $[0, 1] \times [0, 1]$ and discretize the worms. Let $\text{Grid} = \{\left(\frac{m}{M}, \frac{n}{M}\right) \mid m, n \in \{0, 1, \ldots, M\}\}$ for some $M \in \mathbb{Z}_+$. We denote the grid

resolution as $\rho \triangleq 1/M$. We take the set of center points $\mathcal{P} \subseteq \text{Grid}$ as a uniform subgrid of Grid. We consider the discrete worms for $\boldsymbol{p} \in \mathcal{P}, \delta = d \cdot \rho, d \in \mathbb{Z}_{\geq 0}$ as follows:

$$\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{\delta} \triangleq \bigcup_{\substack{\boldsymbol{q}=\boldsymbol{p}+(\alpha,-\alpha) \\ |\alpha| \leq (l-1)\delta \\ \boldsymbol{q} \in \text{Grid}}} \boxed{\boldsymbol{q}}_{\delta}. \tag{4.3}$$

Essentially, a discrete ℓ-worm $\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{\delta}$ centered at $\boldsymbol{p}$ with width δ is the union of $2\ell - 1$ squares with width δ centered at $\boldsymbol{p} \pm (c\delta, -c\delta)$ for $c \in \{0, 1, \ldots, \ell - 1\}$ along with the intermediate staircases between two consecutive squares of step-size equal to *grid resolution* (ρ). Figure 4.5 (middle) shows the discretization of a 2-worm. This construction is sensitive to the grid resolution.

Now all such discrete worms $\widehat{\boxed{\boldsymbol{p}}}$ are intervals whose boundaries are aligned with the Grid. We apply the procedure ComputeRank(f, I) to compute $\mathsf{rk}^{M^f}(I)$ for $I = \widehat{\boxed{\boldsymbol{p}}}^{\ell}_{\delta}$. Denote

$$\hat{\lambda}^{M^f}(\boldsymbol{p}, k, \ell) = \sup_{\delta \geq 0} \{\mathsf{rk}^{M^f}(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{\delta}) \geq k\}. \tag{4.4}$$

Remark 4.3.1. *One can observe that*

$$\lambda^{M^f}(\boldsymbol{p}, k, \ell) \leq \hat{\lambda}^{M^f}(\boldsymbol{p}, k, \ell) \leq \lambda^{M^f}(\boldsymbol{p}, k, \ell) + \rho$$

Therefore, we compute $\hat{\lambda}$ as an approximation of λ in practice.

The pseudo-code is given in Algorithm F. The algorithm is described in detail in Appendix 4.4.

Time complexity. Assuming a grid with t nodes and a bi-filtration of a complex with n simplices on it, one can observe that each probe in the binary search takes $O(n^{\omega})$ time where $\omega < 2.37286$ is the matrix multiplication exponent [123]. This is because each probe generates a zigzag filtration of length $O(n)$ with $O(n)$ simplices. Therefore, the binary search takes $O(n^{\omega} \log t)$ time giving a total time complexity of $O(tn^{\omega} \log t))$ that accounts for $O(t)$ worms.

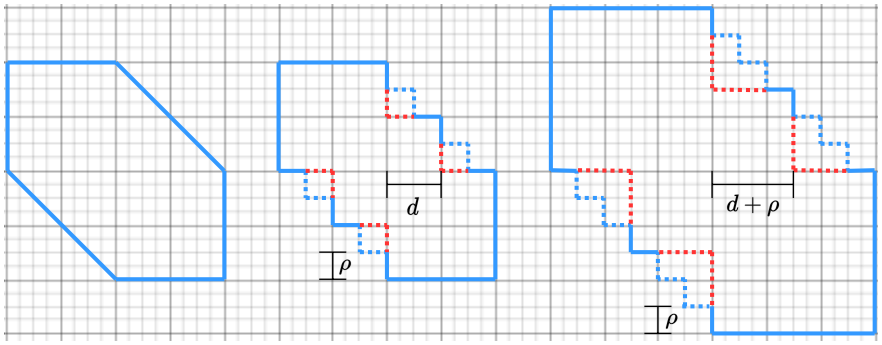

Figure 4.5. A 2-worm, discretized 2-worm and expanded discretized 2-worm; ρ denotes grid resolution. The blue dotted lines show the intermediate staircase with step-size ρ. The red dotted lines form parts of the squares with size d which are replaced by the blue dotted lines in the worm. The last figure shows the expanded 2-worm with red and blue dotted lines. The expanded 2-worm has width $d + \rho$ which is the one step expansion of the worm with width d.

Algorithm F ComputeGril

Input: f : Bi-filtration function, $\ell \geq 0, k \geq 1, \boldsymbol{p} \in \mathcal{P} \subseteq$ Grid, ρ: grid resolution
Output: $\hat{\lambda}(\boldsymbol{p}, k, l)$: Gril value at a point $\boldsymbol{p}$ for fixed k and ℓ
Initialize: $d_{min} \leftarrow \rho, d_{max} \leftarrow 1, \lambda \leftarrow 0$
while $d_{min} \leq d_{max}$ **do**
  $d \leftarrow (d_{min} + d_{max})/2.$
  $I \leftarrow \boxed{\widehat{\boldsymbol{p}}^{\ell}}_{d}$
  $r \leftarrow$ ComputeRank(f, I)
  if $r \geq k$ **then**
    $\lambda \leftarrow d$
    $d_{min} \leftarrow d + \rho$
  else
    $d_{max} \leftarrow d - \rho$
return λ

Speeding up the implementation. In implementation, we use some observations that help run ComputeGril more efficiently in practice. When computing Gril for $k = 1, 2, \ldots, n$, we use the monotone property described in Remark 4.2.1 to reduce the scope of the binary search for successive values of k. For example, the value of Gril for k is always greater than or equal to the value of Gril for $k + 1$. Thus, we can reduce the scope of the binary search while computing for $k + 1$ by setting the maximum in the binary search to

be the value of GRIL at k. Further, we store the values of rank for a given width d while computing the value of GRIL for a k. This information can be reused in later computations. For example, we store the values of generalized ranks of worms for different values of d at a center point $\boldsymbol{p}$ during the binary search for, say $k = k_0$. We use this information for successive binary searches for all $k > k_0$ and save on the zigzag persistence computation for those values of d. While computing zigzag persistent, along with the barcode for 0th homology group, the barcode for 1st homology group is also computed. We store this information and reuse it while computing GRIL values for 1st homology group. These observations reduce the total number of zigzag persistence computations to a significant extent resulting in reducing the total computational time.

4.4 Implementation details

In this section we describe the algorithm in detail. In practice, we are usually presented with a piecewise linear (PL) approximation $\hat{f}$ of a $\mathbb{R}^2$-valued function f on a discretized domain such as a finite simplicial complex. The PL-approximation $\hat{f}$ itself is $\mathbb{R}^2$-valued. Discretizing the parameter space $\mathbb{R}^2$ by a grid, we consider a *lower star* bi-filtration of the simplicial complex. Analogous to the 1-parameter case, a lower star bi-filtration is obtained by assigning every simplex the maximum of the values over all of its vertices in each of the two co-ordinates. With appropriate scaling, these (finite) values can be mapped to a subset of points in a uniform finite grid over $[0, 1] \times [0, 1]$. Observe that because of the maximization of values over all vertices, we have the property that two simplices $\sigma \subseteq \tau$ have values $\hat{f}(\sigma) \in \mathbb{R}^2$ and $\hat{f}(\tau) \in \mathbb{R}^2$ where $\hat{f}(\sigma) \leq \hat{f}(\tau)$. A partial order of the simplices according to these values provide a bi-filtration over the grid $[0, 1] \times [0, 1]$.

Computing generalized ranks. We need to compute the generalized rank $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_d)$ for every worm $\widehat{\boxed{\boldsymbol{p}}}^{\ell}_d$ to decide whether to increase its width or not. We use a result of [40] to compute $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_d)$. It says that $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_d)$ can be computed by considering a zigzag module and computing the number of full bars (bars that begin at the start of the zigzag filtration and persist until the end of the filtration) in its decomposition. This zigzag module

decomposition can be obtained by restricting the bi-filtration on the boundary of $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d})$ and using any of the zigzag persistence algorithms on the resulting zigzag filtration.

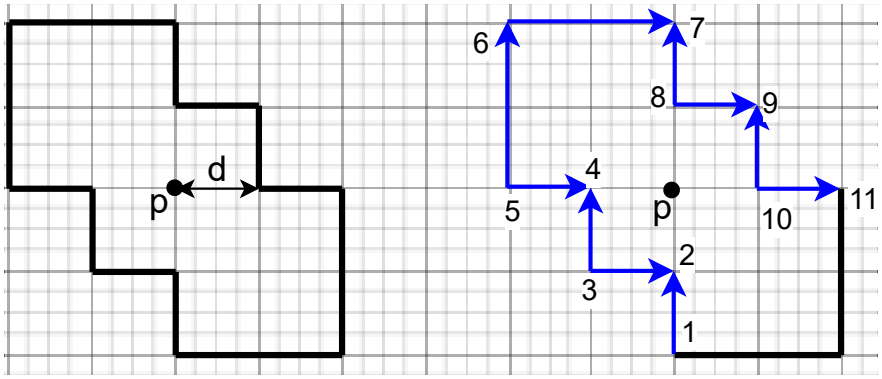

Figure 4.6. (Left) The figure shows the 2-worm centered at p with width d. (Right) The highlighted part denotes the boundary cap of the worm. The arrows in the figure denote the direction of arrows in the zigzag filtration.

Computing zigzag module decomposition. For a given d we do not construct $\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d}$ explicitly. We store the coordinates where each horizontal and vertical line segment of the ℓ-worm intersect. Let us denote these points by a_{i}. We consider the arrow between a_{i} to $a_{\mathrm{i}+1}$, where $\mathrm{i} \in [1, 4\ell+2]$ and denote it by $\overline{a_{\mathrm{i}}a_{\mathrm{i}+1}}$. We find the simplices that project onto these segments. We mark the simplices as 'inserted' or 'deleted' depending upon the direction of the arrow (See Figure 4.7). Notice that since the coordinates of each a_{i} is known we can

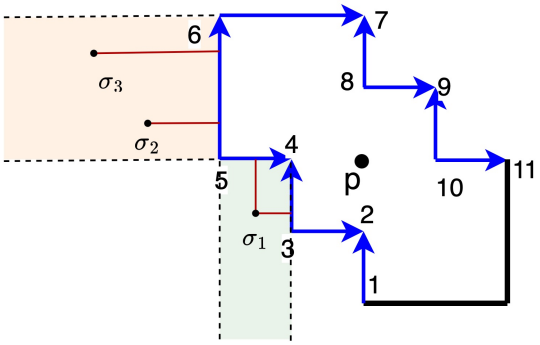

Figure 4.7. σ_1 gets projected to the segment $\overline{a_3a_4}$ and $\overline{a_4a_5}$, whereas σ_2 and σ_3 gets projected to the segment $\overline{a_5a_6}$. During iteration σ_1 gets inserted when $\overline{a_3a_4}$ segment is considered and gets deleted when $\overline{a_4a_5}$ is considered.

efficiently find the projected simplices with vectorized operations available in PyTorch [97]

or NumPy [124]. The resulting zigzag filtration can be decomposed using any of the zigzag peersistence algorithm. We use the recently published efficient algorithm and its associated software [121] for computing zigzag persistence.

Computing the value of Gril using binary search. For a worm $\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d}$ and a given $k \geq 1$, we apply binary search to compute the value of Gril.

Let us denote the grid resolution by ρ. We do the binary search for d in the range $[d_{\min}, d_{\max}]$ where $d_{\min} = \rho$ and $d_{\max} = 1$. In each iteration, we compute $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d})$ for $d = (d_{\min} + d_{\max})/2$ and check if $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d}) \geq k$. We increase the width of the worm by updating

$d_{\min}$ to be $d + \rho$ if $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d}) \geq k$. Otherwise, we decrease the width of the worm by updating $d_{\max}$ to be $d - \rho$. The binary search stops and returns d when $d_{\max} < d_{\min}$. This ensures that we have searched through all possible values of d for which $\mathsf{rk}^M(\widehat{\boxed{\boldsymbol{p}}}^{\ell}_{d}) \geq k$ and returned the maximum of these values.

Since the above steps can be computed independently for each centre point $\boldsymbol{p}$, we take advantage of thread-level parallelism. We use OpenMP [125] to parallelize the computation. Refer to Figure 4.6 for an illustration of the zigzag filtration along the boundary cap of a 2-worm.

4.5 Experiments

Our method Gril exploits generalized rank invariant whereas existing methods exploit rank invariant which is equivalent to fibered barcode. Although both invariants are known to be incomplete for multiparameter persistence as any other discrete invariant, the generalized rank invariant is more informative in theory. Our experiments support this theoretical hypothesis in practice to some extent as we obtain better accuracy for 13 out of 20 cases in Table 4.1 in comparison to existing methods applying some form of fibered barcodes. We perform experiments on synthetic graph benchmark datasets. On these datasets, we define a bi-filtration and compute Gril values $\lambda(\boldsymbol{p}, k, \ell)$ for $\ell = 2$ and for each $k \in \{1, 2, \ldots, 5\}$ where $\boldsymbol{p}$ is chosen over a uniform subgrid. Some datasets require a finer resolution for capturing

meaningful information while for others, finer resolutions capture redundant information and a coarser resolution performs better. Therefore, we sample subgrids with different step-sizes from the discretized grid described in section 4.3 and vary $\boldsymbol{p}$ over these subgrids.We first describe an experiment on a synthetic data set and follow it with experiments on benchmark graph data sets.

We perform a series of experiments on graph classification to test the proposed model. We use standard datasets such as PROTEINS, DHFR, COX2, IMDB-BINARY and MUTAG [95]. A quantitative summary of these datasets is given in Section 4.6.

4.5.1 Classifying Gril representations directly

We compare the performance of GRIL with other models such as multiparameter persistence landscapes (MP-L) [24], multiparameter persistence images (MP-I) [35], multiparameter persistence kernel (MP-K) [23].

In [35], the authors use the heat kernel signature (HKS) and Ricci curvature to form a bi-filtration on the graph datasets. We also use the same bi-filtration and report the result in Table 4.1. We use XGBoost classifier [126] as done in [35] for a fair comparison. We also report the results of GRIL with different classifiers in Table 4.8. The reported accuracies are averaged over 5 train/test splits of the datasets obtained with 5 stratified folds. The full details of the experiments are given in Section 4.6.

Table 4.1. Test accuracy of different models on graph datasets. The values of the MP-I, MP-K, MP-L and P columns are as reported in [35]. P denotes 1-parameter persistence as reported in [35].

Dataset	MP-I	MP-K	MP-L	P	Gril
PROTEINS	67.3 ± 3.5	67.5 ± 3.1	65.8 ± 3.3	65.4 ± 2.7	70.9 ± 3.1
DHFR	80.2 ± 2.3	81.7 ± 1.9	79.5 ± 2.3	70.9 ± 3.1	77.6 ± 2.5
COX2	77.9 ± 2.7	79.9 ± 1.8	79.0 ± 3.3	76.0 ± 4.1	79.8 ± 2.9
MUTAG	85.6 ± 7.3	86.2 ± 2.6	85.7 ± 2.5	79.2 ± 7.7	87.8 ± 4.2
IMDB-BINARY	71.1 ± 2.1	68.2 ± 1.2	71.2 ± 2.0	54.0 ± 1.9	65.2 ± 2.6

From Table 4.1, we can see that the performance of GRIL on IMDB-BINARY is slightly lower than the other methods. This is because the graphs in IMDB-BINARY do not contain

many cycles and hence, there is not enough information to capture in H_1 (See Appendix 4.7 for a visual interpretation). However, when there is information available, GRIL captures it better than the existing methods as can be seen from the accuracy values on other datasets.

4.5.2 Augmenting GNNs with Gril features

Experimental Setup:

In another set of experiments, we augment standard GNNs with GRIL features and compare the performance of the model with the existing ones. We use 3 layers of message-passing with hidden dimensionality of 64. The latent node representations are passed through a pooling layer and a two layer MLP to obtain the final classification. We use sum pooling to maintain uniformity among experiments and we do not claim that this is the optimal choice in any sense. For the GNN+GRIL architectures, we concatenate H_0 and H_1 and pass it through a 1-layer MLP. We concatenate the transformed GRIL values with the graph-level representations obtained from the pooling layer before passing through the final MLP classifier.

Table 4.2. Performance comparison of baseline GNNs and GRIL augmented GNNs on graph benchmark datasets.

Model	PROTEINS	DHFR	COX2	MUTAG	IMDB-BINARY
GCN	71.15 ± 2.31	78.70 ± 2.35	78.80 ± 2.13	88.26 ± 3.70	73.1 ± 2.20
GCN + GRIL	74.21 ± 2.08	75.66 ± 3.08	80.30 ± 1.57	88.80 ± 3.60	72.6 ± 1.46
GAT	67.66 ± 3.92	77.78 ± 4.50	79.45 ± 3.68	86.69 ± 6.36	74.90 ± 2.98
GAT + GRIL	71.60 ± 3.92	79.64 ± 6.29	80.52 ± 3.30	84.03 ± 7.85	71.60 ± 3.04
GIN	69.09 ± 3.77	79.77 ± 6.72	78.80 ± 4.88	83.97 ± 6.04	73.7 ± 3.34
GIN + GRIL	71.87 ± 3.22	78.46 ± 5.80	79.22 ± 4.89	89.32 ± 4.81	74.2 ± 2.82

Training and evaluation:

The models are trained for 100 epochs with Adam [94] as the optimizer. The initial learning rate was set to be 10^{-2} halving every 20 epochs. No hyperparameter tuning and early stopping was done. Though restrictive for practical scenarios, we follow earlier works

Table 4.3. Performance comparison of baseline GNNs and GRIL augmented GNNs on social network datasets without node attributes.

Model	IMDB-BINARY	IMDB-MULTI	REDDIT-BINARY	REDDIT-MULTI-5K
	initial_node_features: deg(v)		initial_node_features: uninformative	
GIN	73.70 ± 3.34	49.60 ± 3.02	90.30 ± 1.30	53.77 ± 1.85
GIN + GRIL	74.20 ± 2.82	50.33 ± 2.58	87.35 ± 2.77	53.85 ± 2.60

(see [127, 128] for more details). We report cross-validation accuracy averaged over 10 folds of the model obtained in the final training epoch.

Results:

We can see from Table 4.2 that GRIL captures topological information that the GNN architectures are unable to capture and hence we see a clear increase in performance. However this is not the case for social network datasets. For the experiments reported in table 4.3 the node features are set as *uninformative* following the settings of [109]. For the IMDB-*, REDDIT-MULTI-5K datasets, the augmented GRIL features improve the baseline GIN accuracy. For the REDDIT-BINARY dataset, since the graphs are highly sparse GRIL features computed with HKS-RC bifiltration fails to capture important features and as a consequence, the performance decreases.

4.6 Additional experiments

We performed a series of experiments on graph classification using GRIL. We used standard datasets with node features such as PROTEINS, DHFR, COX2, MUTAG and IMDB-BINARY [95]. Description of the graph classification tasks is given in Table 4.4.

The Heat Kernel Signature-Ricci Curvature bi-filtration, as done in [35], values are normalized so that they lie between 0 and 1. For the experiments reported in Section 4.5, we fix the grid resolution $\rho = 0.01$. Thus, the square $[0, 1] \times [0, 1]$ has 100×100 many grid points. We sample a uniform subgrid of center points, $\boldsymbol{p}$, out of these grid points. We fix $l = 2$ for our experiments. We compute $\lambda(\boldsymbol{p}, k, \ell)$ where $\boldsymbol{p}$ varies over the sampled center

Table 4.4. Description of Graph Datasets.

Dataset	Num Graphs	Num Classes	Avg. No. Nodes	Avg. No. Edges
PROTEINS	1113	2	39.06	72.82
COX2	467	2	41.22	43.45
DHFR	756	2	42.43	44.54
MUTAG	188	2	17.93	19.79
IMDB-BINARY	1000	2	19.77	96.53

points and k varies from 1 to 5. Each such computation is done for dimension 0 homology (H_0) and dimension 1 homology (H_1). We use XGBoost [126] classifier for these experiments.

4.6.1 Ablation Studies:

We have performed experiments with different subgrid sizes and the results are reported in Table 4.5. The reported accuracies are averaged over 5 train/test splits of the datasets obtained with 5 stratified folds. We can see from the table that for different datasets, different subgrid sizes give the best results. This can be attributed to the fact that for some datasets, topological information needs to be captured at a finer level while for other datasets, capturing such finer details can be redundant.

Table 4.5. Test accuracies of GRIL on subgrids of different sizes.

Grid Size	50×50	25×25	10×10	5×5
PROTEINS	70.8 ± 2.7	70.2 ± 1.8	69.8 ± 2.4	68.5 ± 2.6
DHFR	77.6 ± 2.5	77.2 ± 3.4	77.5 ± 3.5	77.5 ± 3.5
COX2	79.8 ± 3.0	78.9 ± 2.4	79.8 ± 2.9	78.9 ± 3.5
MUTAG	87.3 ± 3.8	87.8 ± 4.2	87.8 ± 4.5	86.8 ± 3.3
IMDB-BINARY	62.2 ± 4.3	65.2 ± 2.6	62.2 ± 2.3	63.5 ± 3.2

We report the computation times of GRIL for these datasets in Table 4.6. The values denote the total computation time for all the center points on a 50×50 subgrid for a 2-worm. The computations were done on a Intel(R) Xeon(R) Gold 6248R CPU machine and the computation was carried out on 32 cores.

Table 4.6. Computation times for GRIL for each dataset with a 2-worm and 50×50 subgrid.

Dataset	Computation time
PROTEINS	6 hr 13 min 38 s
DHFR	4 hr 15 min 54 s
COX2	2 hr 44 min 23 s
MUTAG	0 hr 56 min 48 s
IMDB-BINARY	4 hr 03 min 35 s

In Table 4.7, we show the performance of GRIL with different grid resolutions (ρ) and ℓ-worms. For these experiments, we used a 50×50 subgrid for the center points. The reported accuracies are averaged over 5 train/test splits of the datasets obtained with 5 stratified folds. We test it on MUTAG and COX2 and we can see that for $\rho = 0.01$, we get the highest accuracy of the model on both the datasets. We can see from the table that there is an improvement in accuracy from $\ell = 1$ to $\ell = 2$. However, there is no significant improvement from $\ell = 2$ to $\ell = 3$.

Table 4.7. Test accuracy for different grid resolutions and for ℓ-worms with different values of ℓ.

Dataset	$\rho = \mathbf{0.02}$	$\rho = \mathbf{0.01}$	$\rho = \mathbf{0.005}$	$\ell = \mathbf{1}$	$\ell = \mathbf{2}$	$\ell = \mathbf{3}$
MUTAG	86.3 ± 4.2	87.8 ± 4.5	85.2 ± 3.9	85.7 ± 4.2	87.8 ± 4.5	87.8 ± 3.9
COX2	78.2 ± 1.7	79.8 ± 2.9	77.8 ± 1.4	79.3 ± 2.9	79.8 ± 2.9	78.9 ± 3.5

Table 4.8. Test accuracies of GRIL using different classifiers.

Dataset	SVM	LR	XGBoost	3-MLP
PROTEINS	73.3 ± 1.5	72.7 ± 2.6	70.9 ± 3.1	71.3 ± 2.1
DHFR	61.7 ± 0.4	77.8 ± 1.9	77.6 ± 2.5	72.3 ± 4.3
COX2	77.2 ± 0.8	78.5 ± 2.5	79.8 ± 2.9	77.0 ± 1.2
MUTAG	80.0 ± 3.9	86.3 ± 3.8	87.8 ± 4.2	76.8 ± 9.1
IMDB-BINARY	65.1 ± 3.6	63.2 ± 2.1	65.2 ± 2.6	61.2 ± 6.6

In Table 4.8, we report the performance of Gril on graph benchmark datasets with different classifiers such as support vector machine (SVM) [129, 130], logistic regression (LR) [131], Multilayer Perceptron (3-MLP) implemented using *scikit-learn* [60] library. The reported accuracies are averaged over 5 train/test splits of the datasets obtained with 5 stratified folds.

4.7 Visualization of Gril for graph datasets

The plot for first 5 Gril values are shown in Figure 4.8. The figure contains landscape values for 5 random graph samples of each dataset. In Figure 4.9, we plot the first two eigen vectors given by principal component analysis (PCA) of the computed Gril values for each dataset. Plots for H_0 and H_1 are shown separately.

4.8 Concluding remarks

In this work, we propose Gril, a 2-parameter persistence vectorization based on generalized rank invariant. Furthermore, we present an algorithm for computing Gril which is a synergistic confluence of the recent developments in computing generalized rank invariant of a 2-parameter module and an efficient algorithm for computing zigzag persistence. As a topological feature extractor, Gril performs better than Graph Convolutional Networks (GCNs) and Graph Isomorphism Networks (GINs) on our synthetic dataset. It also performs better than the existing multiparameter persistence methods on some graph benchmark datasets while achieves comparable performance on others.

We believe that the additional topological information that a 2-parameter persistence module encodes, as compared to a 1-parameter persistence module, can be leveraged to learn better representations. Further directions of research include using Gril with GNNs for filtration learning to learn more powerful representations. We expect that this work motivates further research in this direction.

4.9 Acknowledgement

This research is partially supported by NSF grant CCF 2049010.

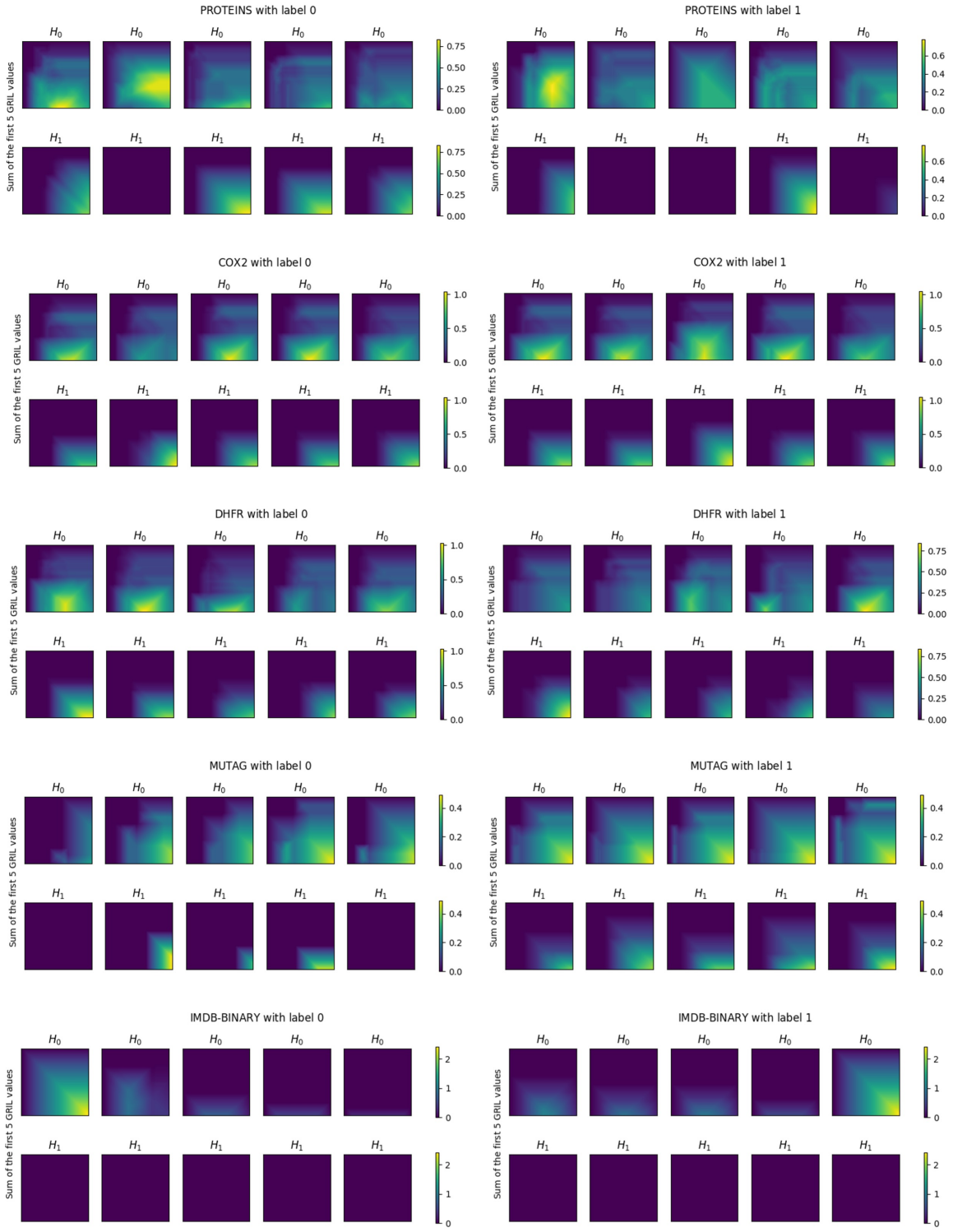

Figure 4.8. GRIL of 5 random graph samples of each dataset. GRIL values of H_0 and H_1 are shown separately columnwise.

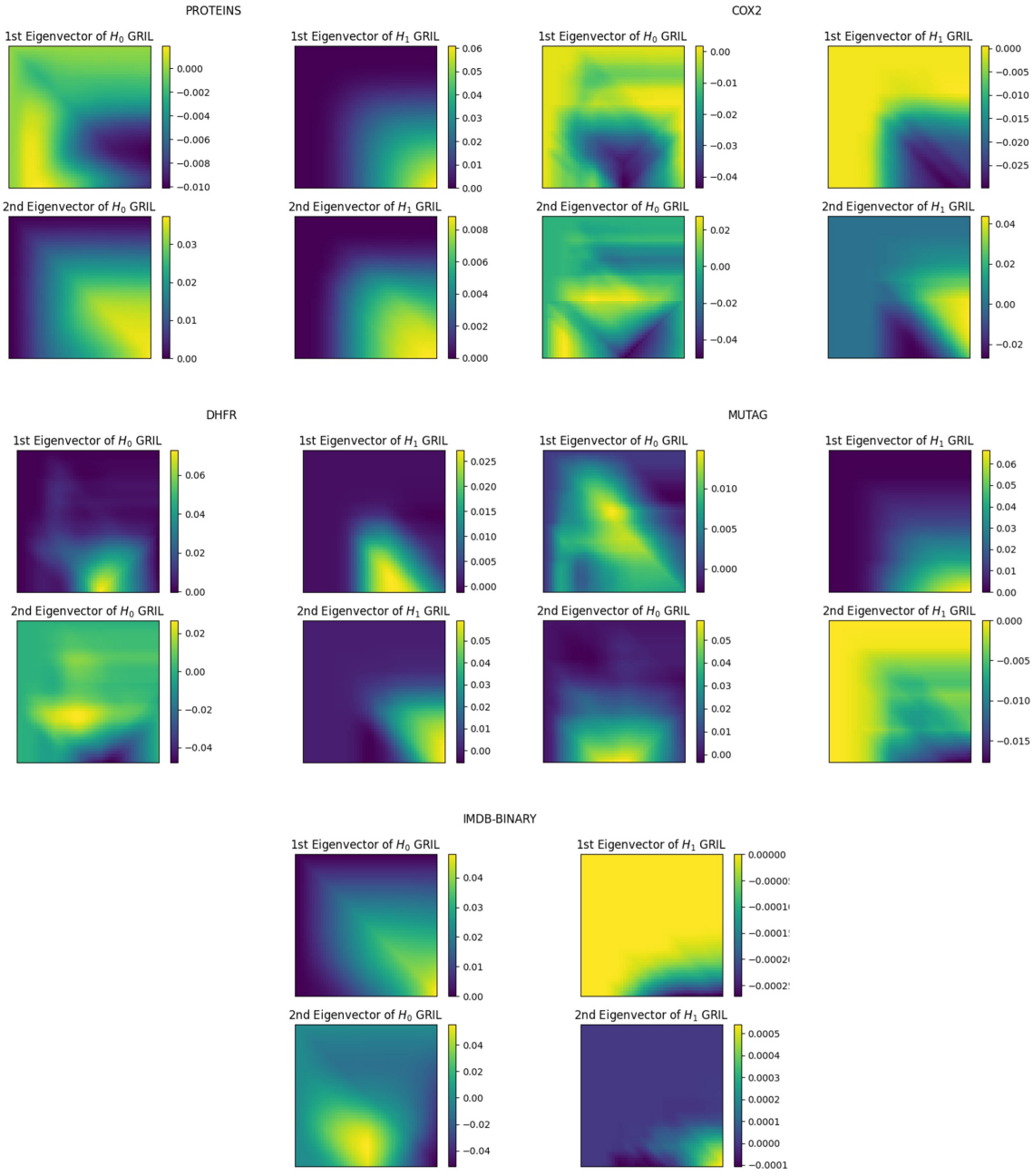

Figure 4.9. Plot of the first two eigen vectors given by PCA on the entire dataset for H_0 and H_1 respectively.

5. GRIL-D: A DIFFERENTIABLE 2-PARAMETER PERSISTENCE LAYER FOR DRUG DISCOVERY

In chapter 4 we discussed Generalized Rank Invariant Landscape (Gril), a new vector representation, and its effectiveness in capturing encoding richer information beyond fibered barcodes for 2-parameter persistence modules. We showed this empirically by augmenting the features extracted by Gril in a machine learning pipeline, specifically augmenting the features extracted by Gril as an input to Graph Neural Networks (GNNs). In this chapter, we propose a molecular fingerprint based on 2 parameter persistence, specifically Gril, to predict the bio-activity of a synthesized drug. Contrary to previous multiparameter approaches (such as ToDD [27], PHoS [119]) where the bi-filtration function is fixed, our method is differentiable, meaning our multiparameter representations are learnt in a data-driven way. We leverage this by proposing an *end-to-end* pipeline in conjunction with GNNs. We first use graph isomorphism network (GIN) to obtain a bi-filtration function f and pass it through the proposed Gril-D to obtain fingerprints of compounds as 2D matrices. Then, we use an MLP layer to successfully predict the compounds that show bio-activity. We also demonstrate the versatility of Gril fingerprints from a pre-trained neural network and classifying with traditional ML algorithms such as XGBoost [126], logistic regression [131], SVM [130]. The key contributions of this chapter are:

- A *differentiable* 2-parameter persistence based molecular fingerprint.
- To the best of our knowledge, we introduced the first approach to actively integrate 2-parameter persistence within the realm of GNNs, where the bi-filtration function, f is learnt.
- We perform extensive experiments in bio-activity prediction, showing that our method outperforms the static multiparameter fingerprints, demonstrating the need for an *end-to-end* pipeline.

5.1 Related Works

5.1.1 Virtual Screening

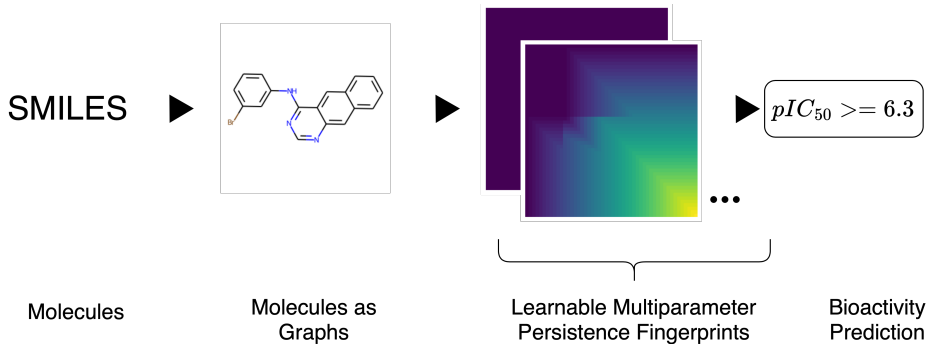

Figure 5.1. General framework of our approach.

A crucial step of drug discovery involves identifying biologically active compounds that can potentially be developed into drug candidates. To achieve this, researchers often turn to computational methods for compound prioritization with desired properties. One widely used approach for this purpose is virtual screening (VS), which can be categorized into two major types: structure-based virtual screening (SBVS) and ligand-based virtual screening (LBVS) [132].

SBVS relies on the 3D structural information of both the compound (ligand) and the target protein as a complex. This method requires a deep understanding of the target protein's 3D structure to explore how a compound can fit into its binding pocket effectively. However, this level of detail makes SBVS computationally expensive [133–135].

In contrast, LBVS methods take a different approach. They compare structural similarities of a library of compounds with a known active ligand, assuming that similar compounds are likely to exhibit similar biological activity. LBVS, unlike SBVS, solely uses ligand information and aims to create effective fingerprints of the compounds using machine learning (ML) tools to find similarities. As a result, LBVS can be more efficient, especially when working with larger chemical datasets and when the structure of the target receptor is not well understood [136].

Over the past three decades, various LBVS methods have emerged, employing different approaches. These methods can be categorized into three classes based on the type of fingerprints they generate: **(i) 1D-Methods:** Examples of 1D-methods include SMILES [137] and SMARTS [138], which produce 1D-fingerprints. These methods compress compound information into a vector representation. **(ii) 2D-Methods:** The realm of 2D methods comprises of RASCAL [139], MOLPRINT2D [140], ECFP [141], MACSS [142], and Morgan [143]. These methods use 2D-structure fingerprints and graph matching to assess structural similarities among compounds. **(iii) 3D-Methods:** On the 3D front, methods like ROCS, USR, and PatchSurfer take into account the 3D structure of compounds and their conformations, which includes the 3D position of the compound [144–149].

5.1.2 Multiparameter Persistence in Virtual Screening

Topology has found its way into the realm of medicinal chemistry, and we are not the pioneers in this endeavor. Early on, persistent homology, a topological technique, was applied to the field of protein docking [150]. This approach continues to be actively explored [151–153], though the specific methods employed in those studies differ from our approach. More recently, there have been efforts in structure-based screening that combine single-parameter persistent homology with machine learning [12, 14, 17, 28, 31, 154, 155]. In contrast to these approaches, our method achieves state-of-the-art results in ligand-based screening without the necessity of training a machine learning model. We accomplish this by employing multiparameter persistence, which allows us to not only capture essential molecular shape properties but also incorporate non-shape information, such as electrostatics, in a coherent and effective manner.

In the ever-evolving landscape of medicinal chemistry, deep learning has gained attention [156], and there have even been attempts to combine single-parameter persistent homology with deep learning in a docking-based approach [26, 154].

However, the structure of multiparameter persistence modules is much more complicated than 1-parameter persistence modules. In the 1-parameter case, the modules are completely characterized by what is called *barcode* or *persistence diagram* [36, 37]. Unfor-

tunately, there is no such discrete complete invariant that can summarize multiparameter persistence modules completely [38]. Given this limitation, leveraging multiparameter persistence in virtual screening becomes an important but challenging problem. To address this challenge, different kinds of vector representations have been proposed for 2-parameter persistence modules [23, 24, 35]. However, multiparameter persistence in the domain of drug-discovery remains largely unexplored barring PHoS [119] and ToDD [27]. In a different domain, multi-parameter persistent homology has also been employed for classifying hepatic lesions in computed tomography images [157]. This demonstrates the versatility of topological methods in diverse scientific applications.

5.2 Multiparameter Persistence Based Fingerprinting

5.2.1 Overall Framework

While our approach is versatile and can be employed with different types of data, our primary emphasis in this context is on graphs, with a specific focus on utilizing them for the virtual screening of compounds. Our approach produces fingerprints of compounds based on multiparameter persistence, to be specific, the *differentiable* variant of GRIL which we call GRIL-D. The overall pipeline is explained in the following steps:

Step 1. Bi-filtrations: In all of the previous approaches that involve multiparameter persistence for fingerprinting, the input bi-filtration is fixed *a-priori*. For example, in [119] authors employ Vietoris-Rips bi-filtration, and in [27] authors use an ensemble of atomic mass, partial charge, bond type, electron affinity, ionization energy to perform graph filtration. Owing to the theoretical development (Explained in Section 5.3) we employ a learnable bi-filtration that is suitable for downstream tasks. Since we limit ourselves to graphs in this paper, graph neural networks (GNNs) are a natural choice to obtain bi-filtrations. Careful readers will note that through GNNs we obtain node embeddings and if we restrict the embeddings to $\mathbb{R}^2$ we obtain a vertex filter function, i.e. $f_v \colon \mathbb{V} \to \mathbb{R}^2$. We can interpolate f_v in a piecewise-constant fashion to obtain a valid bi-filtration function f.

Step 2. *Differentiable* GRIL fingerprints: Our method samples a subset of grid points from the 2-parameter grid spanned by the bi-filtration function coming from GNN

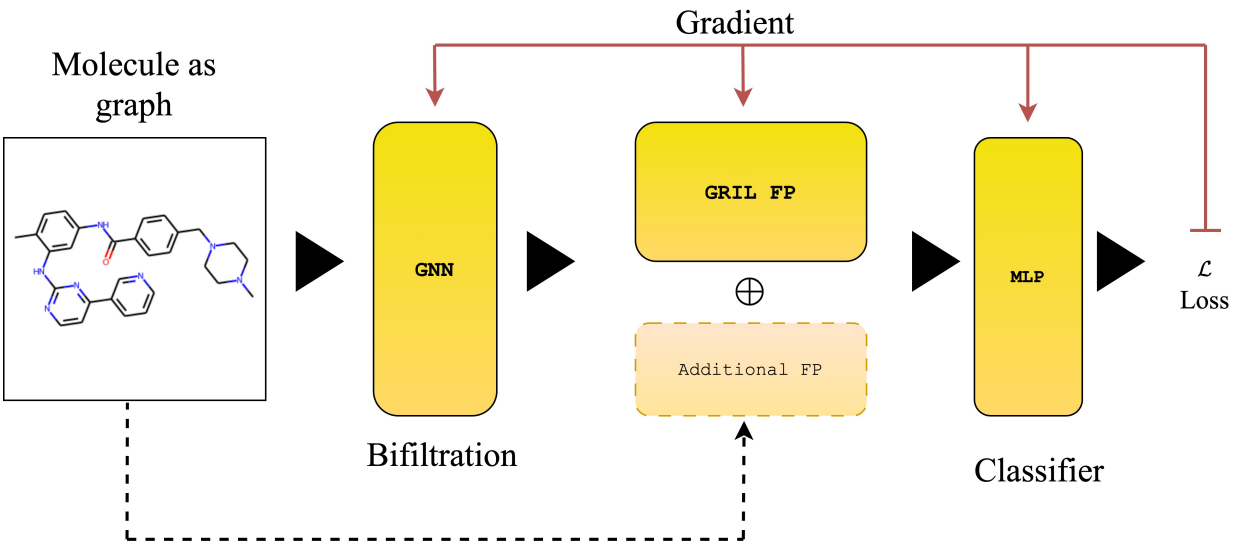

Figure 5.2. Architecture choice for bio-activity prediction. Notice that the bi-filtration function f is learnt compared to the standard multiparameter pipeline.

and computes the landscape function values (Definition 5.2.2) at those points based on generalized ranks. For this, the algorithm considers an expanding sequence of intervals which we call *worms* centered at each point $\boldsymbol{p}$, and computes generalized rank over them to determine the 'width' of the maximal worm sustaining a chosen rank. This maximization is achieved by a binary search over the sequence of worms centering $\boldsymbol{p}$. The widths, thus computed for each sample point, constitute the landscape function values which become the basis for our fingerprints. Note that the previous multiparameter signatures that have been proposed to study compounds till now are variants of *fibered* or *sliced* barcodes based on rank invariants, whereas the signatures used by us are based on generalized rank invariants, which in theory is a stronger invariant and captures more information.

Step 3. Biologically active compound classification: The fingerprints are then passed through a 3-layer MLP for final classification. Please see Section 5.5 for more details.

5.2.2 Background and Definitions

In this subsection, we recall some definitions to prove the differentiability of the proposed *differentiable* GRIL layer.

Definition 5.2.1 ((discrete) ℓ-worm, [42]). *Let* $\boxed{\mathbf{p}}_\delta := \{\mathbf{w} : |\mathbf{p} - \mathbf{w}|_\infty \leq \delta\}$ *be the* δ*-square centered at* $\mathbf{p}$ *with side* 2δ*. Given* $\mathbf{p} \in \mathbb{R}^2, \ell \geq 1, \delta > 0$*, we define an* ℓ-worm $\boxed{\mathbf{p}}_\delta^\ell$ *to be the union over all* δ*-squares* $\boxed{\mathbf{q}}_\delta$ *centered at some point* $\mathbf{q}$ *on the off-diagonal line segment* $\mathbf{p} + \alpha \cdot (1, -1)$ *with* $|\alpha| = \mathrm{j} \cdot \delta$ *where* $\mathrm{j} \in \{1, \ldots, \ell - 1\}$.

Definition 5.2.2 (GRIL, [42]). *For a 2-parameter persistence module* M*, Generalized Rank Invariant Landscape (GRIL) is defined as a function* $\lambda^M \colon \mathbb{R}^2 \times \mathbb{N}_+ \times \mathbb{N}_+ \to \mathbb{R}$ *given by*

$$\lambda^M(\mathbf{p}, k, \ell) := \sup_{\delta \geq 0} \left\{ rk^M \left(\boxed{\mathbf{p}}_\delta^\ell \right) \geq k \right\}$$

where rk^M *is the generalized rank invariant as defined in [39].*

Proposition 5.2.1 ([42]). *GRIL is stable with respect to the input bi-filtration functions, i.e., given* $f, g : \mathbf{X} \to \mathbb{R}^2$*, then*

$$|\lambda^{M_f}(\boldsymbol{p}, k, \ell) - \lambda^{M_g}(\boldsymbol{p}, k, \ell)| \leq \|f - g\|_\infty$$

for all $\boldsymbol{p}, k, \ell$.

5.3 Differentiability

Consider a simplicial complex K with n simplices. Consider a bi-filtration function f on K. Then, f can be viewed as a vector $x_f \in \mathbb{R}^{2n}$. Let $\Lambda_{k,\ell}^{M_f}$ denote the GRIL vector in $\mathbb{R}^c$ where c is the number of center points $\{p_\mathrm{j}\}_{\mathrm{j}=1}^c$ and fixed k, ℓ for the persistence module induced by f. Here we state the main result (see Proposition 5.3.2)) that will permit us to use GRIL in a differentiable framework. For the actual proof, we refer the readers to [158].

Proposition 5.3.1 ([42]). *$\mathcal{G}$ is Lipschitz continuous.*

Corollary 5.3.1 ([42]). *$\mathcal{G}$ is differentiable almost everywhere.*

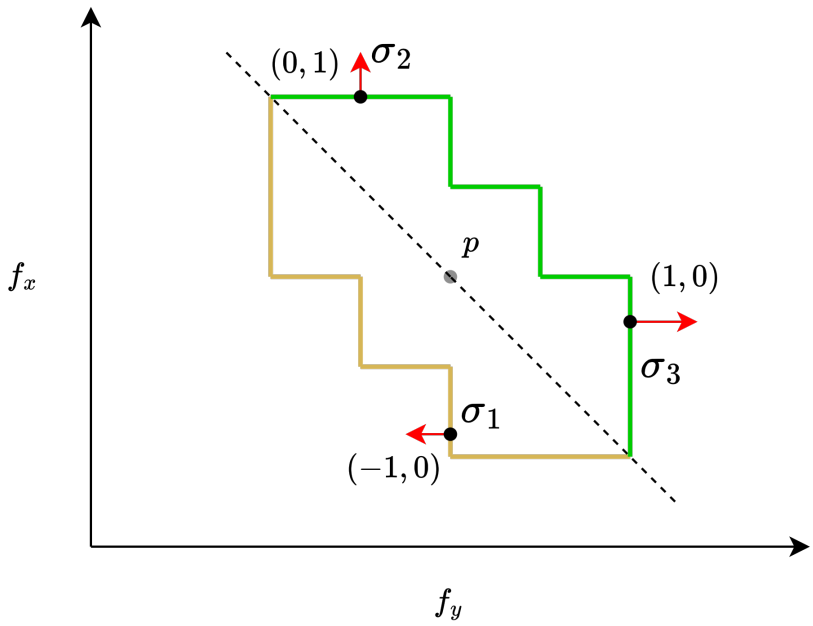

Figure 5.3. Gradient assignment for the ℓ-worm centred at point p. The yellow and green lines in the figure are *lower* and *upper* boundaries of ℓ-worm. σ_1 is a lower x constraining simplex and thus assigned a gradient of $(-1, 0)$, σ_2 is an upper y constraining simplex with an assignment of $(0, 1)$ and σ_3 is an upper x constraining simplex with gradient assigned as $(1, 0)$.

Proposition 5.3.2. *Given a bi-filtration function f which satisfies the generic condition, $k \in \mathbb{N}$, $\ell \in \mathbb{N}$. Let $\{p_j\}_{j=1}^{c}$ denote the center points of the worms and let $\{d_j\}_{j=1}^{c}$ denote the corresponding* GRIL *values. If there exists a unique constraining simplex σ_{i_j} for each worm $\boxed{\boldsymbol{p}}_{d_j}^{\ell}$ then $\mathcal{G}$ is differentiable and the derivative is given by*

$$\begin{pmatrix} \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_1)}{\partial \sigma_1^x} & \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_1)}{\partial \sigma_1^y} & \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_1)}{\partial \sigma_2^x} & \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_1)}{\partial \sigma_2^y} & \cdots & \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_1)}{\partial \sigma_n^y} \\ \vdots & & & & & \\ \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_c)}{\partial \sigma_1^x} & & \cdots & & & \frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_c)}{\partial \sigma_n^y} \end{pmatrix}_{c \times 2n}$$

where,

$$\frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_{\mathrm{j}})}{\partial \sigma_{\mathrm{i}}^{x}} = \begin{cases} -1, & \textit{if } \mathrm{i} = \mathrm{i}_{\mathrm{j}} \textit{ and } \sigma_{\mathrm{i}_{\mathrm{j}}} \textit{ is lower } x\textit{-constraining} \\ +1, & \textit{if } \mathrm{i} = \mathrm{i}_{\mathrm{j}} \textit{ and } \sigma_{\mathrm{i}_{\mathrm{j}}} \textit{ is upper } x\textit{-constraining} \\ 0, & \textit{otherwise} \end{cases}$$

$$\frac{\partial \lambda_{k,\ell}(\boldsymbol{p}_{\mathrm{j}})}{\partial \sigma_{\mathrm{i}}^{y}} = \begin{cases} -1, & \textit{if } \mathrm{i} = \mathrm{i}_{\mathrm{j}} \textit{ and } \sigma_{\mathrm{i}_{\mathrm{j}}} \textit{ is lower } y\textit{-constraining} \\ +1, & \textit{if } \mathrm{i} = \mathrm{i}_{\mathrm{j}} \textit{ and } \sigma_{\mathrm{i}_{\mathrm{j}}} \textit{ is upper } y\textit{-constraining} \\ 0, & \textit{otherwise} \end{cases}$$

5.4 Datasets

The data is extracted from the ChEMBL database [159, 160]. Each of the datasets contains SMILES encoding of a novel drug (compound) and activity pairs for a target of interest. For example, the EGFR dataset contains all the molecules that have been tested against epidermal growth factor receptor (EGFR) kinase and their measured bio-activity. The bio-activity is measured by *half maximal inhibitory concentration* (IC_{50}), which measures qualitatively indicates how much a drug is needed, *in vitro*, to inhibit a particular process by 50%. To facilitate the comparison of IC_{50} values it is common practice to convert it to $pIC_{50} = -\log_{10}(IC_{50})$, expressed in molar units. The threshold for activity cutoff $pIC_{50} = 6.3$ is used throughout the paper.

EGFR:

This dataset contains 4635 molecules reacted against epidermal growth factor receptor (EGFR) kinase. The dataframe contains ChEMBL-ID, SMILES encoding of the corresponding compound, and measured affinity (pIC_{50}) value.

ERRB2:

This dataset contains 1822 molecules reacted against Receptor protein-tyrosine kinase erbB-2 (ERRB2). The dataframe contains ChEMBL-ID, SMILES encoding of the corresponding compound, and measured affinity (pIC_{50}) value.

The datasets are publicly available on the ChEMBL website and can be downloaded following the tutorial mentioned in [161]. More details of these datasets are in Table 5.1. The SMILES encoding of the molecule is then converted to PyG [96] graph objects by Molfeat [162]. During the conversion of molecules to graphs, Molfeat computes additional node features that are fed to the GNN to get the input bi-filtration function. The computed 82 dimensional node features are (i) atom-one-hot, (ii) atom-degree-one-hot, (iii) atom-implicit-valence-one-hot, (iv) atom-hybridization-one-hot, (v) atom-is-aromatic, (vi) atom-formal-charge, (vii) atom-num-radical-electrons, (viii) atom-is-in-ring, (ix) atom-total-num-H-one-hot, (x) atom-chiral-tag-one-hot and (xi) atom-is-chiral-center. This is the default setting of Molfeat and we do not claim, in any way, that these are the optimal node features that are to be used.

Table 5.1. Details of the ChEMBL datasets. Note that compounds with $pIC_{50} >= 6.3$ are considered active molecules.

Dataset	Num Graphs	Active	Inactive	Avg. Num Nodes	Avg. Num Edges
EGFR	4635	2631	2004	28.97	31.73
ERRB2	1818	1140	678	33.33	36.70
CHEMBL1163125	2719	1507	1212	30.77	34.23
CHEMBL203	6816	4234	2582	31.88	34.98
CHEMBL2148	3200	2380	820	29.89	33.28
CHEMBL279	7461	5104	2357	31.82	35.11
CHEMBL2815	3143	2484	659	33.72	37.30
CHEMBL4005	4790	3195	1595	31.79	35.26
CHEMBL4282	3004	2112	892	33.81	37.69
CHEMBL4722	2565	1750	815	31.93	35.29

5.5 Experiments

5.5.1 Setup

Since we want to test the discriminating power of the new topological fingerprints, we restrict ourselves to the message passing layer to 1. Particularly we use graph isomorphism network (GIN) with 1 hidden layer of dimension 64. For a graph G, the GIN layer gives us a map from 82 dimensional node features to $\mathbb{R}^2$. We construct a valid bi-filtration function, f, by extending those embeddings in a piecewise constant manner. To be specific, let f_v denote the map from node features to $\mathbb{R}^2$, i.e., $f_v \colon \mathbb{R}^{82} \to \mathbb{R}^2$. Then for each edge (u, v) we get $f(u, v) = \{\max(f_{u_1}, f_{v_1}), \max(f_{u_2}, f_{v_2}) \}$, thereby giving us a valid filtration function. We get the GRIL values w.r.t f and feed it to a 3-layer MLP for classification. For each of the datasets, the train-val-test split is $70 - 10 - 20$ % with the accuracies and ROC-AUC reported averaged over 5 fold cross-validation. Each fold consists of 50 epochs and utmost care was taken to reinitialize the model over the folds. The initial learning rate is set to be $1e^{-2}$ halved every 10 epoch. The optimizer used was Adam [94].

5.5.2 Results

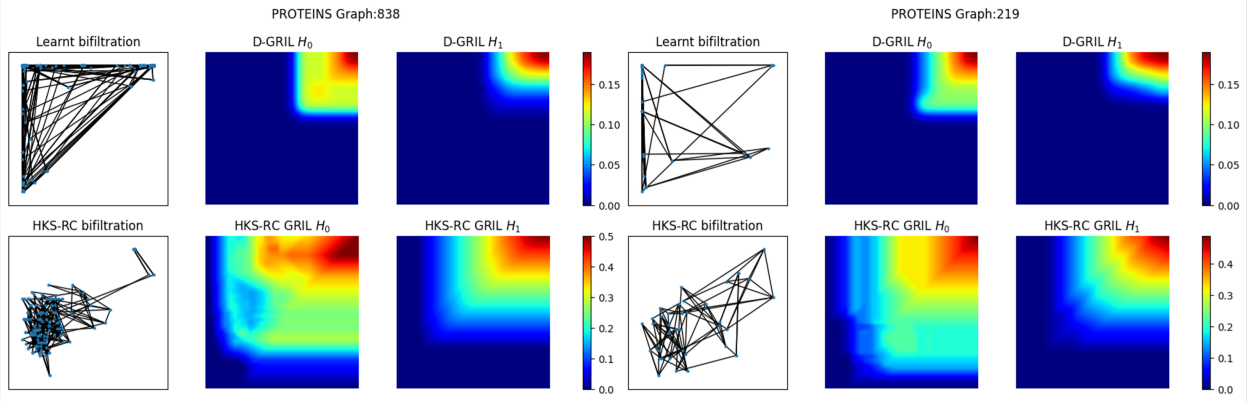

Figure 5.4. The figure compares the learnt bifiltration function with the Heat-Kernel Signature-Ricci Curvature (HKS-RC) bifiltration on two random graph instances (838 and 219) of PROTEINS dataset. In the first column, the bifiltration function on the vertices of these graphs is plotted.

We perform a series of experiments on the datasets mentioned in Section 5.4 and report the results in Table 5.2 and Table 5.3. In the first set of experiments, we compare GrilD with (i) a standard GNN model, GIN, with sum-pooling as a representative, (ii) with GIN and Gril as a readout layer. Note that for the results on GIN-Gril, Gril is used as a passive readout layer, i.e., we obtained a bifiltration function from a pre-trained GIN (pre-trained for graph classification on the same dataset) and computed Gril on it. We use the same classifier (3-layer MLP) in all the cases. From Table 5.2, it is clear that adding topological information in an end-to-end learning framework appears to be beneficial for bio-activity prediction on these datasets.

We plot the learnt bifiltration function and compare it with the Heat-Kernel Signature-Ricci Curvature (HKS-RC) bifiltration function of two random graph instances from Proteins dataset. We also plot the GrilD vectors for H_0 and H_1 and compare them with the Gril vectors computed on the HKS-RC bifiltration function on these graphs. The figures are shown in Figure 5.4. It is clear from the figure that the model is learning a different bifiltration function than the HKS-RC bifiltration and consequently, the Gril vectors in these cases also look very different.

Table 5.2. Test ROC-AUC on ChEMBL datasets. GrilD performs better than GIN with *sum* pooling and Gril with bifiltration obtained from pre-trained GIN.

Dataset	GIN	GIN-GRIL	GrilD
EGFR	55.60 ± 8.61	58.39 ± 2.51	62.83 ± 1.92
ERRB2	57.06 ± 8.58	61.28 ± 3.66	62.26 ± 4.43
CHEMBL1163125	58.48 ± 4.37	54.13 ± 1.09	68.67 ± 3.50
CHEMBL2148	52.88 ± 0.91	50.24 ± 0.18	53.63 ± 3.52
CHEMBL4005	55.90 ± 4.63	51.85 ± 3.55	57.20 ± 4.48

We perform a set of additional experiments where we classify Gril fingerprints obtained from pre-trained neural network (See Table 5.4 and 5.5). We classify the obtained Gril fingerprints with traditional linear classifiers, specifically with logistic regression (LR) and support vector machines (SVM).

Table 5.3. Test ROC-AUC scores on ChEMBL datasets. Augmenting GrilD with ECFP, Morgan2, and Morgan3 fingerprints increases the classification performance for most of the datasets.

Dataset	ECFP	ECFP+GrilD	Morgan2	Morgan2+GrilD	Morgan3	Morgan3+GrilD
EGFR	83.27 ± 1.10	83.40 ± 2.06	83.39 ± 1.23	83.39 ± 1.86	82.39 ± 1.35	82.88 ± 1.71
ERRB2	83.53 ± 1.31	84.29 ± 1.10	82.66 ± 1.45	84.29 ± 1.10	83.19 ± 1.26	83.26 ± 2.42
CHEMBL1163125	83.94 ± 1.23	84.74 ± 1.08	83.89 ± 1.40	84.77 ± 0.87	83.55 ± 1.20	83.57 ± 1.52
CHEMBL203	81.74 ± 0.90	82.17 ± 1.37	81.31 ± 1.10	82.17 ± 1.37	80.85 ± 1.32	81.03 ± 0.93
CHEMBL2148	73.96 ± 2.59	73.93 ± 3.49	75.18 ± 2.06	73.81 ± 3.64	72.79 ± 1.97	72.98 ± 2.18
CHEMBL279	76.72 ± 1.34	77.74 ± 1.05	76.90 ± 1.34	77.85 ± 1.03	76.76 ± 0.50	77.58 ± 0.78
CHEMBL2815	73.69 ± 1.53	74.64 ± 1.69	73.13 ± 1.62	74.58 ± 1.73	73.42 ± 0.56	74.51 ± 1.01
CHEMBL4005	80.45 ± 1.30	80.45 ± 1.40	79.88 ± 1.52	80.49 ± 1.31	79.95 ± 1.30	81.11 ± 0.86
CHEMBL4282	79.08 ± 2.75	78.39 ± 1.70	78.78 ± 2.76	78.44 ± 1.71	77.61 ± 2.37	78.32 ± 1.88
CHEMBL4722	78.05 ± 1.64	77.65 ± 1.87	77.96 ± 1.07	77.54 ± 2.03	78.76 ± 1.37	78.82 ± 1.57

Table 5.4. Experiments with Gril fingerprints obtained from pre-trained neural network on the EGFR dataset.

Classifier	Morgan2	Morgan2+GrilD	Morgan3	Morgan3+GrilD	ECFP	ECFP+GrilD
LR	83.56	83.56	83.88	83.77	83.56	83.56
SVM	85.44	85.44	85.39	85.39	85.44	85.44

Table 5.5. Experiments with Gril fingerprints obtained from pre-trained neural network on the ERRB2 dataset.

Classifier	Morgan2	Morgan2+GrilD	Morgan3	Morgan3+GrilD	ECFP	ECFP+GrilD
LR	87.26	87.26	87.67	87.67	87.26	87.26
SVM	87.67	87.53	88.08	88.08	87.67	87.53

Table 5.6. Accuracy of GrilD on benchmark graph datasets.

Dataset	GrilD	Gril	MP-I	MP-L	MP-K	P
MUTAG	85.59 ± 6.71	83.49 ± 3.64	74.99 ± 2.79	82.42 ± 3.72	79.27 ± 2.45	66.50 ± 0.87
PROTEINS	67.74 ± 2.27	66.31 ± 2.34	70.80 ± 3.09	70.80 ± 1.31	61.70 ± 2.98	59.57 ± 0.08
DHFR	67.06 ± 4.37	61.64 ± 1.66	60.98 ± 0.10	61.11 ± 0.25	60.98 ± 0.10	60.98 ± 0.10
COX2	78.59 ± 1.09	78.16 ± 0.41	78.16 ± 0.41	78.16 ± 0.41	78.16 ± 0.41	78.16 ± 0.41
IMDB-BINARY	58.70 ± 4.92	50.00 ± 0.00	56.60 ± 2.94	50.00 ± 0.00	50.30 ± 1.12	50.00 ± 0.00

GrilD can be used in a more general setting, for filtration learning on graph datasets. We perform a set of experiments with benchmark graph datasets such as Mutag, Proteins, Dhfr, Cox2 and compare them with existing multiparameter persistence methods.

For a valid comparison, we used a 3-layer MLP as the classifier for all the multiparameter signatures. We can see from Table 5.6 that learning the bifiltration function seems to perform better than multiparameter persistence methods on popular choices of bifiltration functions on most datasets. In fact, we can see from the table that GRILD performs better than GRIL, supporting our argument for an end-to-end learning framework.

5.6 Concluding remarks

In this chapter, we presented a topological molecular fingerprint derived from multiparameter persistence. To our knowledge, our approach represents the initial endeavor to actively incorporate multiparameter persistent homology into the domain of bio-activity prediction, introducing a unique category of molecular fingerprinting. Across all experiments, our framework consistently outperforms the conventional *static* multiparameter-based fingerprints. This highlights the necessity of employing an *end-to-end* differentiable pipeline for such scenarios.

6. CONCLUSIONS

In this dissertation, we have expanded upon the synergy of computational topology and machine learning. In Chapter 2, TDA helped to analyze cytometry data that is difficult to interpret manually. We found out that a few protein expressions in CD8+ T cells separate healthy donors from COVID-19 patients. Furthermore, persistence diagrams help to distinguish structural features in CD8+ T cell data occurring in healthy individuals and COVID-19 patients. Next, in Chapter 3 we introduced Extended persistence in a suprevised graph classification setting. Furthermore, we showed how we can leverage higher dimensional simplices to enhance message passing. Defining higher order networks allowed us to process noisy and non-manifold meshes that were not possible by traditional methods such as [82, 104]. Moreover, to ensure the applicability of our research in the field of machine learning, we introduced a vector representation, denoted as Gril, designed specifically for 2-parameter persistence modules. Gril surpasses previous constraints by capturing more intricate information, offering a Lipschitz stable and differentiable representation concerning the filtration function, f. Additionally, we developed an efficient algorithm for computing Gril, demonstrating its practical application on synthetic and benchmark graph datasets. Initial findings suggest that Graph Neural Networks (GNNs) enhanced with Gril features exhibit improved performance in graph classification tasks. Lastly, in Chapter 5 we a introduce topological molecular fingerprint derived from 2-parameter persistence. To our knowledge, our approach represents the initial endeavor to actively incorporate multiparameter persistent homology into the domain of bio-activity prediction, introducing a unique category of molecular fingerprinting. We hope that this dissertation encourages future research work in the following directions:

- We determine structural changes in T-bet and Eomes abundances in single CD8+ T cells in COVID-19 patients that can be summarized as downregulation. This result is non-intuitive as previous findings show that T-bet and Eomes protein abundances are highest in effector CD8+ T cells, which are induced in response to acute infections, suggesting T-bet and Eomes expressions should be upregulated [65, 68]. The clinical implications of this result are unclear.

- Applications of TDL in the field of drug discovery, and protein-protein interactions remain uncharted. We believe TDL can provide an alternative approach to decipher protein fingerprints described in [148, 149].

- Further experimentation with Gril, specifically extending its application to other complex datasets and across different problem domains, may open up new avenues for applying multiparameter persistence in machine learning.

- Studying Gril fingerprints obtained from a pre-trained neural network may provide new avenues for the interpretability of deep learning.

In summary, this dissertation not only pushes the boundaries of single and multiparameter persistence modules' applications but also sets the stage for compelling future research opportunities in the intersection of topological data analysis and machine learning.

REFERENCES

[1] M. Nicolau, A. J. Levine, and G. Carlsson, "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival," *Proceedings of the National Academy of Sciences*, vol. 108, no. 17, pp. 7265–7270, 2011.

[2] Y. Li, D. Wang, G. A. Ascoli, P. Mitra, and Y. Wang, "Metrics for comparing neuronal tree shapes based on persistent homology," *PloS one*, vol. 12, no. 8, e0182184, 2017.

[3] S. Mukherjee, "Denoising with discrete morse theory," *The Visual Computer*, vol. 37, no. 9-11, pp. 2883–2894, 2021.

[4] T. K. Dey, S. Mandal, and S. Mukherjee, "Gene expression data classification using topology and machine learning models," *BMC bioinformatics*, vol. 22, no. 10, pp. 1–22, 2021.

[5] J. Zhang, C. Cai, G. Kim, Y. Wang, and W. Chen, "Composition design of high-entropy alloys with deep sets learning," *npj Computational Materials*, vol. 8, no. 1, p. 89, 2022.

[6] T. K. Dey, M. Lipiski, M. Mrozek, and R. Slechta, "Computing connection matrices via persistence-like reductions," *arXiv preprint arXiv:2303.02549*, 2023.

[7] A. V. Patel, T. Hou, J. D. B. Rodriguez, T. K. Dey, and D. P. Birnie, "Topological filtering for 3d microstructure segmentation," *Computational Materials Science*, vol. 202, p. 110 920, 2022, ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2021.110920. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927025621006224.

[8] J. Yang, X. Hu, C. Chen, and C. Tsai, "3d topology-preserving segmentation with compound multi-slice representation," in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, Apr. 2021, pp. 1297–1301. DOI: 10.1109/ISBI48211.2021.9433941.

[9] R. Ghrist, "Barcodes: The persistent topology of data," *Bulletin of the American Mathematical Society*, vol. 45, no. 1, pp. 61–75, 2008.

[10] D. Horak, S. Maleti, and M. Rajkovi, "Persistent homology of complex networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2009, no. 03, P03034, 2009.

[11] H. Adams *et al.*, "Persistence images: A stable vector representation of persistent homology," *Journal of Machine Learning Research*, vol. 18, 2017.

[12] M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda, "Perslay: A neural network layer for persistence diagrams and new graph topological signatures," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, S. Chiappa and R. Calandra, Eds., ser. Proceedings of Machine Learning Research, vol. 108, PMLR, Aug. 2020, pp. 2786–2796. [Online]. Available: https://proceedings.mlr.press/v108/carriere20a.html.

[13] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt, "A stable multi-scale kernel for topological machine learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4741–4748.

[14] C. D. Hofer, F. Graf, B. Rieck, M. Niethammer, and R. Kwitt, "Graph filtration learning," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 4314–4323. [Online]. Available: http://proceedings.mlr.press/v119/hofer20b.html.

[15] M. Horn, E. D. Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. M. Borgwardt, "Topological graph neural networks," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=oxxUMeFwEHd.

[16] P. Bubenik, "Statistical topological data analysis using persistence landscapes," *J. Mach. Learn. Res.*, vol. 16, pp. 77–102, 2015. DOI: 10.5555/2789272.2789275. [Online]. Available: https://dl.acm.org/doi/10.5555/2789272.2789275.

[17] Z. Yan, T. Ma, L. Gao, Z. Tang, Y. Wang, and C. Chen, "Neural approximation of extended persistent homology on graphs," in *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=H9xJHbJ6e5.

[18] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," in *Proceedings of the Twenty-First Annual Symposium on Computational Geometry*, ser. SCG '05, Pisa, Italy: Association for Computing Machinery, 2005, pp. 263–271, ISBN: 1581139918. DOI: 10.1145/1064092.1064133. [Online]. Available: https://doi.org/10.1145/1064092.1064133.

[19] M. R. McGuirl, A. Volkening, and B. Sandstede, "Topological data analysis of zebrafish patterns," *Proceedings of the National Academy of Sciences*, vol. 117, no. 10, pp. 5113–5124, 2020, ISSN: 0027-8424. DOI: 10.1073/pnas.1917763117. eprint: https://www.pnas.org/content/117/10/5113.full.pdf. [Online]. Available: https://www.pnas.org/content/117/10/5113.

[20] A. O. Komendantov, S. Venkadesh, C. L. Rees, D. W. Wheeler, D. J. Hamilton, and G. A. Ascoli, "Quantitative firing pattern phenotyping of hippocampal neuron types," *Scientific Reports*, vol. 9, no. 1, pp. 1–17, Nov. 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-52611-w.

[21] T. Lakshmikanth *et al.*, "Mass cytometry and topological data analysis reveal immune parameters associated with complications after allogeneic stem cell transplantation," *Cell reports*, vol. 20, no. 9, pp. 2238–2250, 2017.

[22] A. H. Rizvi *et al.*, "Single-cell topological rna-seq analysis reveals insights into cellular differentiation and development," *Nature Biotechnology*, vol. 35, no. 6, pp. 551–560, Jun. 2017, ISSN: 1546-1696. DOI: 10.1038/nbt.3854.

[23] R. Corbet, U. Fugacci, M. Kerber, C. Landi, and B. Wang, "A kernel for multi-parameter persistent homology," *Computers & Graphics: X*, vol. 2, p. 100 005, 2019, ISSN: 2590-1486. DOI: https://doi.org/10.1016/j.cagx.2019.100005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590148619300056.

[24] O. Vipond, "Multiparameter persistence landscapes," *Journal of Machine Learning Research*, vol. 21, no. 61, pp. 1–38, 2020. [Online]. Available: http://jmlr.org/papers/v21/19-054.html.

[25] G. Bouritsas, F. Frasca, S. P. Zafeiriou, and M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2022. DOI: 10.1109/TPAMI.2022.3154319. [Online]. Available: https://doi.org/10.1109/TPAMI.2022.3154319.

[26] Z. Cang and G.-W. Wei, "Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions," *PLOS Computational Biology*, vol. 13, no. 7, pp. 1–27, Jul. 2017. DOI: 10.1371/journal.pcbi.1005690. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1005690.

[27] A. Demir, B. Coskunuzer, Y. Gel, I. Segovia-Dominguez, Y. Chen, and B. Kiziltan, "ToDD: Topological compound fingerprinting in computer-aided drug discovery," in *Advances in Neural Information Processing Systems*, 2022. [Online]. Available: https://openreview.net/forum?id=8hs7qlWcnGs.

[28] S. Zhang, S. Mukherjee, and T. K. Dey, "GEFL: extended filtration learning for graph classification," in *Proceedings of the First Learning on Graphs Conference*, ser. Proceedings of Machine Learning Research, vol. 198, PMLR, Dec. 2022, 16:1–16:26. [Online]. Available: https://proceedings.mlr.press/v198/zhang22b.html.

[29] X. Liu, H. Feng, J. Wu, and K. Xia, "Dowker complex based machine learning (dcml) models for protein-ligand binding affinity prediction," *PLOS Computational Biology*, vol. 18, no. 4, pp. 1–17, Apr. 2022. DOI: 10.1371/journal.pcbi.1009943. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1009943.

[30] C. Chen, X. Ni, Q. Bai, and Y. Wang, "A topological regularizer for classifiers via persistent homology," in *The 22nd International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 2573–2582.

[31] K. Kim, J. Kim, M. Zaheer, J. Kim, F. Chazal, and L. Wasserman, "PLLay: Efficient topological layer based on persistent landscapes," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 15 965–15 977. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/b803a9254688e259cde2ec0361c8abe4-Paper.pdf.

[32] R. B. Gabrielsson, B. J. Nelson, A. Dwaraknath, and P. Skraba, "A topology layer for machine learning," in *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, ser. Proceedings of Machine Learning Research, vol. 108, PMLR, 2020, pp. 1553–1563. [Online]. Available: http://proceedings.mlr.press/v108/gabrielsson20a.html.

[33] Q. Zhao, Z. Ye, C. Chen, and Y. Wang, "Persistence enhanced graph neural network," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 108, PMLR, Aug. 2020, pp. 2896–2906. [Online]. Available: https://proceedings.mlr.press/v108/zhao20d.html.

[34] N. Swenson, A. S. Krishnapriyan, A. Buluc, D. Morozov, and K. Yelick, "PersGNN: Applying topological data analysis and geometric deep learning to structure-based protein function prediction," *arXiv preprint arXiv:2010.16027*, 2020.

[35] M. Carrière and A. Blumberg, "Multiparameter persistence image for topological machine learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 22 432–22 444. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/fdff71fcab656abfbefaabecab1a7f6d-Paper.pdf.

[36] F. Chazal, D. Cohen-Steiner, M. Glisse, L. Guibas, and S. Y. Oudot, "Proximity of persistence modules and their diagrams," in *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, ser. SCG '09, 2009, pp. 237–246.

[37] M. Lesnick, "The theory of the interleaving distance on multidimensional persistence modules," *Found. Comput. Math.*, vol. 15, no. 3, pp. 613–650, Jun. 2015, ISSN: 1615-3375. DOI: 10.1007/s10208-015-9255-y. [Online]. Available: https://doi.org/10.1007/s10208-015-9255-y.

[38] G. Carlsson and A. Zomorodian, "The theory of multidimensional persistence," *Discrete & Computational Geometry*, vol. 42, no. 1, pp. 71–93, Jul. 2009, ISSN: 1432-0444. DOI: 10.1007/s00454-009-9176-0. [Online]. Available: https://doi.org/10.1007/s00454-009-9176-0.

[39] W. Kim and F. Mémoli, "Generalized persistence diagrams for persistence modules over posets," *Journal of Applied and Computational Topology*, vol. 5, no. 4, pp. 533–581, Dec. 2021, ISSN: 2367-1734. DOI: 10.1007/s41468-021-00075-1. [Online]. Available: https://doi.org/10.1007/s41468-021-00075-1.

[40] T. K. Dey, W. Kim, and F. Mémoli, "Computing generalized rank invariant for 2-parameter persistence modules via zigzag persistence and its applications," in *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, X. Goaoc and M. Kerber, Eds., ser. LIPIcs, vol. 224, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 34:1–34:17. DOI: 10.4230/LIPIcs.SoCG.2022.34. [Online]. Available: https://doi.org/10.4230/LIPIcs.SoCG.2022.34.

[41] D. Loiseaux, L. Scoccola, M. Carrière, M. B. Botnan, and S. Oudot, "Stable vectorization of multiparameter persistent homology using signed barcodes as measures," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[42] C. Xin, S. Mukherjee, S. N. Samaga, and T. K. Dey, "GRIL: a 2-parameter persistence based vectorization for machine learning," in *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, ser. Proceedings of Machine Learning Research, vol. 221, PMLR, Jul. 2023, pp. 313–333. [Online]. Available: https://proceedings.mlr.press/v221/xin23a.html.

[43] S. Mukherjee, D. Wethington, T. K. Dey, and J. Das, "Determining clinically relevant features in cytometry data using persistent homology," *PLOS Computational Biology*, vol. 18, no. 3, pp. 1–22, Mar. 2022. DOI: 10.1371/journal.pcbi.1009931. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1009931.

[44] M. H. Spitzer and G. P. Nolan, "Mass cytometry: Single cells, many features," *Cell*, vol. 165, no. 4, pp. 780–791, 2016, ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2016.04.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009286741630410X.

[45] Y. Simoni, M. H. Y. Chng, S. Li, M. Fehlings, and E. W. Newell, "Mass cytometry: A powerful tool for dissecting the immune landscape," *Current Opinion in Immunology*, vol. 51, pp. 187–196, 2018, ISSN: 0952-7915. DOI: https://doi.org/10.1016/j.coi.2018.03.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952791518300293.

[46] D. Mathew *et al.*, "Deep immune profiling of covid-19 patients reveals distinct immunotypes with therapeutic implications," *Science*, vol. 369, no. 6508, Z. Alam *et al.*, Eds., 2020, ISSN: 0036-8075. DOI: 10.1126/science.abc8511. eprint: https://science.sciencemag.org/content/369/6508/eabc8511.full.pdf. [Online]. Available: https://science.sciencemag.org/content/369/6508/eabc8511.

[47] D. M. Strauss-Albee *et al.*, "Human nk cell repertoire diversity reflects immune experience and correlates with viral susceptibility," *Science Translational Medicine*, vol. 7, no. 297, 297ra115–297ra115, 2015, ISSN: 1946-6234. DOI: 10.1126/scitranslmed.aac5722. eprint: https://stm.sciencemag.org/content/7/297/297ra115.full.pdf. [Online]. Available: https://stm.sciencemag.org/content/7/297/297ra115.

[48] J. A. Wargo, S. M. Reddy, A. Reuben, and P. Sharma, "Monitoring immune responses in the tumor microenvironment," *Current Opinion in Immunology*, vol. 41, pp. 23–31, 2016, ISSN: 0952-7915. DOI: https://doi.org/10.1016/j.coi.2016.05.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952791516300462.

[49] A. Azad, B. Rajwa, and A. Pothen, "Immunophenotype discovery, hierarchical organization, and template-based classification of flow cytometry samples," *Frontiers in Oncology*, vol. 6, p. 188, 2016, ISSN: 2234-943X. DOI: 10.3389/fonc.2016.00188. [Online]. Available: https://www.frontiersin.org/article/10.3389/fonc.2016.00188.

[50] E. del Barrio, H. Inouzhe, J.-M. Loubes, C. Matrán, and A. Mayo-Íscar, "optimalFlow: Optimal transport approach to flow cytometry gating and population matching," en, *BMC Bioinformatics*, vol. 21, no. 1, p. 479, Dec. 2020, ISSN: 1471-2105. DOI: 10.1186/s12859-020-03795-w. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-03795-w.

[51] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction* (Applied Mathematics). American Mathematical Society, 2010, ISBN: 9780821849255.

[52] A. Zomorodian, "Topological data analysis," *Advances in applied and computational topology*, vol. 70, pp. 1–39, 2012.

[53] M. Buchet, F. Chazal, S. Y. Oudot, and D. R. Sheehy, "Efficient and robust persistent homology for measures," *Computational Geometry*, vol. 58, pp. 70–96, 2016, ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2016.07.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772116300633.

[54] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas, "Persistence barcodes for shapes," *International Journal of Shape Modeling*, vol. 11, no. 02, pp. 149–187, 2005.

[55] F. Chazal, D. Cohen-Steiner, and Q. Mérigot, "Geometric inference for probability measures," *Foundations of Computational Mathematics*, vol. 11, no. 6, pp. 733–751, 2011.

[56] M. Buchet, T. K. Dey, J. Wang, and Y. Wang, "Declutter and resample: Towards parameter free denoising," in *33rd International Symposium on Computational Geometry, SoCG 2017*, Schloss Dagstuhl, Leibniz-Zentrum fü Informatik GmbH, 2017, pp. 231–2316.

[57] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794, ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. [Online]. Available: https://doi.org/10.1145/2939672.2939785.

[58] The GUDHI Project, *GUDHI User and Reference Manual*, 3.4.1. GUDHI Editorial Board, 2021. [Online]. Available: https://gudhi.inria.fr/doc/3.4.1/.

[59] M. Kerber, D. Morozov, and A. Nigmetov, "Geometry helps to compare persistence diagrams," *ACM J. Exp. Algorithmics*, vol. 22, Sep. 2017, ISSN: 1084-6654. DOI: 10.1145/3064175. [Online]. Available: https://doi.org/10.1145/3064175.

[60] L. Buitinck *et al.*, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[61] D. Panchenko, *Statistics for applications: 18.650*, 2006. [Online]. Available: https://ocw.mit.edu.

[62] T. Bernas, E. K. Asem, J. P. Robinson, and B. Rajwa, "Quadratic form: A robust metric for quantitative comparison of flow cytometric histograms," *Cytometry Part A: the journal of the International Society for Analytical Cytology*, vol. 73, no. 8, pp. 715–726, 2008.

[63] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient color histogram indexing for quadratic form distance functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 729–736, 1995. DOI: 10.1109/34.391417.

[64] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966, ISSN: 0036-8075. DOI: 10.1126/science.153.3731.34. eprint: https://science.sciencemag.org/content/153/3731/34.full.pdf. [Online]. Available: https://science.sciencemag.org/content/153/3731/34.

[65] N. Takemoto, A. M. Intlekofer, J. T. Northrup, E. J. Wherry, and S. L. Reiner, "Cutting edge: Il-12 inversely regulates t-bet and eomesodermin expression during pathogen-induced cd8+ t cell differentiation," *The Journal of Immunology*, vol. 177, no. 11, pp. 7515–7519, 2006.

[66] T. Scholzen and J. Gerdes, "The ki-67 protein: From the known and the unknown," *Journal of Cellular Physiology*, vol. 182, no. 3, pp. 311–322, 2000. DOI: https://doi.org/10.1002/(SICI)1097-4652(200003)182:3<311::AID-JCP1>3.0.CO;2-9. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-4652%28200003%29182%3A3%3C311%3A%3AAID-JCP1%3E3.0.CO%3B2-9. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4652%28200003%29182%3A3%3C311%3A%3AAID-JCP1%3E3.0.CO%3B2-9.

[67] S. Van Gassen *et al.*, "Flowsom: Using self-organizing maps for visualization and interpretation of cytometry data," *Cytometry Part A*, vol. 87, no. 7, pp. 636–645, 2015.

[68] J. J. Knox, G. L. Cosma, M. R. Betts, and L. M. McLane, "Characterization of t-bet and eomes in peripheral human immune cells," *Frontiers in Immunology*, vol. 5, p. 217, 2014, ISSN: 1664-3224. DOI: 10.3389/fimmu.2014.00217. [Online]. Available: https://www.frontiersin.org/article/10.3389/fimmu.2014.00217.

[69] M.-L. Thibult *et al.*, "Pd-1 is a novel regulator of human b-cell activation," *International immunology*, vol. 25, no. 2, pp. 129–137, 2013.

[70] Y. Wang *et al.*, "The transcription factor tcf1 preserves the effector function of exhausted cd8 t cells during chronic viral infection," *Frontiers in Immunology*, vol. 10, p. 169, 2019, ISSN: 1664-3224. DOI: 10.3389/fimmu.2019.00169. [Online]. Available: https://www.frontiersin.org/article/10.3389/fimmu.2019.00169.

[71] T. Dey and Y. Wang, *Computational Topology for Data Analysis*. Cambridge University Press, 2022, ISBN: 9781009098168. [Online]. Available: https://books.google.com/books?id=PWtYEAAAQBAJ.

[72] K. Quintelier, A. Couckuyt, A. Emmaneel, J. Aerts, Y. Saeys, and S. Van Gassen, "Analyzing high-dimensional cytometry data using flowsom," *Nature Protocols*, pp. 1–27, 2021.

[73] C. D. Hofer, R. Kwitt, and M. Niethammer, "Learning representations of persistence barcodes.," *J. Mach. Learn. Res.*, vol. 20, no. 126, pp. 1–45, 2019.

[74] M. Horn, E. De Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt, "Topological graph neural networks," *arXiv preprint arXiv:2102.07835*, 2021.

[75] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Extending persistence using poincaré and lefschetz duality," *Foundations of Computational Mathematics*, vol. 9, no. 1, pp. 79–103, 2009.

[76] Z. Yan, T. Ma, L. Gao, Z. Tang, and C. Chen, "Link prediction with persistent homology: An interactive view," in *International Conference on Machine Learning*, PMLR, 2021, pp. 11 659–11 669.

[77] C. Bick, E. Gross, H. A. Harrington, and M. T. Schaub, "What are higher-order networks?" *arXiv:2104.11329*, 2021.

[78] F. Battiston *et al.*, "Networks beyond pairwise interactions: Structure and dynamics," *Physics Reports*, vol. 874, pp. 1–92, 2020.

[79] A. R. Benson, D. F. Gleich, and D. J. Higham, "Higher-order network analysis takes off, fueled by classical ideas and new data," *arXiv preprint arXiv:2103.05031*, 2021.

[80] L. Torres, A. S. Blevins, D. Bassett, and T. Eliassi-Rad, "The why, how, and when of representations for complex systems," *SIAM Review*, vol. 63, no. 3, pp. 435–485, 2021.

[81] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "Meshnet: Mesh neural network for 3d shape representation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8279–8286.

[82] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: A network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[83] F. Milano, A. Loquercio, A. Rosinol, D. Scaramuzza, and L. Carlone, "Primal-dual mesh convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 952–963, 2020.

[84] M. Hajij *et al.*, "Topological deep learning: Going beyond graph data," *arXiv preprint arXiv:1906.09068 (v3)*, 2023.

[85] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 454–463. DOI: 10.1109/SFCS.2000.892133.

[86] C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl, "Deep learning with topological signatures," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 1633–1643, ISBN: 9781510860964.

[87] M. Carrière, M. Cuturi, and S. Oudot, "Sliced wasserstein kernel for persistence diagrams," *arXiv preprint arXiv:1706.03358*, 2017.

[88] C. Vignac, A. Loukas, and P. Frossard, "Building powerful and equivariant graph neural networks with structural message-passing," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 143–14 155, 2020.

[89] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International conference on machine learning*, PMLR, 2018, pp. 5453–5462.

[90] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7793–7804, 2020.

[91] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Tackling over-smoothing for general graph convolutional networks," *arXiv preprint arXiv:2008.09864*, 2020.

[92] K. Oono and T. Suzuki, "Optimization and generalization analysis of transduction through gradient boosting and application to multi-scale graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 917–18 930, 2020.

[93] G. E. Blelloch and B. M. Maggs, "Parallel algorithms," in *Algorithms and theory of computation handbook: special topics and techniques*, 2010, pp. 25–25.

[94] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[95] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "TU-Dataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. arXiv: 2007.08663. [Online]. Available: www.graphlearning.io.

[96] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[97] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[98] H. Hu and X. He, "Sets2sets: Learning from sequential sets with neural networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1491–1499.

[99] H. Gao, Y. Liu, and S. Ji, "Topology-aware graph pooling networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4512–4518, 2021.

[100] M. Aschbacher, "Combinatorial cell complexes," in *Progress in Algebraic Combinatorics*, Mathematical Society of Japan, 1996, pp. 1–80.

[101] T. Basak, "Combinatorial cell complexes and poincaré duality," *Geometriae Dedicata*, vol. 147, no. 1, pp. 357–387, 2010.

[102] M. Savoy, "Combinatorial cell complexes: Duality, reconstruction and causal cobordisms," *arXiv preprint arXiv:2201.12846*, 2022.

[103] M. Hajij, G. Zamzmi, T. Papamarkou, N. Miolane, A. Guzmán-Sáenz, and K. N. Ramamurthy, "Higher-order attention networks," *arXiv preprint arXiv:2206.00606 (v1)*, 2022.

[104] T. M. Roddenberry and S. Segarra, "Hodgenet: Graph neural networks for edge data," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2019, pp. 220–224.

[105] Z. Lian *et al.*, "Shape retrieval on non-rigid 3D watertight meshes," in *Eurographics workshop on 3d object retrieval (3DOR)*, Citeseer, 2011.

[106] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, 2005, 729–734 vol. 2. DOI: 10.1109/IJCNN.2005.1555942. [Online]. Available: https://doi.org/10.1109/IJCNN.2005.1555942.

[107] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: 10.1109/TNN.2008.2005605. [Online]. Available: https://doi.org/10.1109/TNN.2008.2005605.

[108] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl.

[109] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km.

[110] C. D. Hofer, R. Kwitt, M. Niethammer, and A. Uhl, "Deep learning with topological signatures," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 1634–1644. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/883e881bb4d22a7add958f2d6b052c9f-Abstract.html.

[111] N. Dehmamy, A.-L. Barabási, and R. Yu, "Understanding the representation power of graph neural networks in learning graph topology," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[112] S. Y. Oudot, *Persistence Theory: From Quiver Representations to Data Analysis* (Mathematical Surveys and Monographs 209). American Mathematical Society, 2015, p. 218. [Online]. Available: https://hal.inria.fr/hal-01247501.

[113] G. Carlsson and M. Vejdemo-Johansson, *Topological Data Analysis with Applications*. Cambridge University Press, 2021. DOI: 10.1017/9781108975704.

[114] T. K. Dey and Y. Wang, *Computational Topology for Data Analysis*. Cambridge University Press, 2022. DOI: 10.1017/9781009099950.

[115] M. Lesnick and M. Wright, "Interactive visualization of 2-d persistence modules," *CoRR*, vol. abs/1512.00180, 2015. arXiv: 1512.00180. [Online]. Available: http://arxiv.org/abs/1512.00180.

[116] A. Hatcher, *Algebraic topology*. Cambridge: Cambridge Univ. Press, 2000. [Online]. Available: https://cds.cern.ch/record/478079.

[117] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, Feb. 2005, ISSN: 1432-0444. DOI: 10.1007/s00454-004-1146-y. [Online]. Available: https://doi.org/10.1007/s00454-004-1146-y.

[118] A. Adcock, D. Rubin, and G. Carlsson, "Classification of hepatic lesions using the matching metric," *Comput. Vis. Image Underst.*, vol. 121, pp. 36–42, Apr. 2014, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2013.10.014. [Online]. Available: https://doi.org/10.1016/j.cviu.2013.10.014.

[119] B. Keller, M. Lesnick, and T. L. Willke, "Phos: Persistent homology for virtual screening," *ChemRxiv*, 2018. DOI: 10.26434/chemrxiv.6969260.v1.

[120] G. Carlsson and V. De Silva, "Zigzag persistence," *Foundations of computational mathematics*, vol. 10, no. 4, pp. 367–405, 2010. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s10208-010-9066-0.pdf.

[121] T. K. Dey and T. Hou, "Fast computation of zigzag persistence," in *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, ser. LIPIcs, vol. 244, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 43:1–43:15. DOI: 10.4230/LIPIcs.ESA.2022.43. [Online]. Available: https://doi.org/10.4230/LIPIcs.ESA.2022.43.

[122] S. Mac Lane, *Categories for the working mathematician*. Springer Science & Business Media, 2013, vol. 5.

[123] J. Alman and V. V. Williams, "A refined laser method and faster matrix multiplication," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021, pp. 522–539. DOI: 10.1137/1.9781611976465.32. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976465.32. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611976465.32.

[124] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[125] L. Dagum and R. Menon, "OpenMP: An industry standard api for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998.

[126] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794, ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. [Online]. Available: https://doi.org/10.1145/2939672.2939785.

[127] C. Morris *et al.*, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4602–4609.

[128] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[129] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018. [Online]. Available: https://doi.org/10.1007/BF00994018.

[130] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1–27:27, 3 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[131] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: a library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008, ISSN: 1532-4435.

[132] C. N. Cavasotto and A. J. W Orry, "Ligand docking and structure-based virtual screening in drug discovery," *Current topics in medicinal chemistry*, vol. 7, no. 10, pp. 1006–1014, 2007.

[133] N. Brooijmans and I. D. Kuntz, "Molecular recognition and docking algorithms," *Annual review of biophysics and biomolecular structure*, vol. 32, no. 1, pp. 335–373, 2003.

[134] T. B. Kimber, Y. Chen, and A. Volkamer, "Deep learning in virtual screening: Recent applications and developments," *International journal of molecular sciences*, vol. 22, no. 9, p. 4435, 2021.

[135] V. B. Sulimov, D. C. Kutov, and A. V. Sulimov, "Advances in docking," *Current medicinal chemistry*, vol. 26, no. 42, pp. 7555–7580, 2019.

[136] C. Lemmen and T. Lengauer, "Computational methods for the structural alignment of molecules," *Journal of Computer-Aided Molecular Design*, vol. 14, pp. 215–232, 2000.

[137] D. Weininger, "SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[138] I. Daylight Chemical Information Systems, *Smarts-a language for describing molecular patterns*, 2007.

[139] J. W. Raymond, E. J. Gardiner, and P. Willett, "Rascal: Calculation of graph similarity using maximum common edge subgraphs," *The Computer Journal*, vol. 45, no. 6, pp. 631–644, 2002.

[140] A. Bender, H. Y. Mussa, R. C. Glen, and S. Reiling, "Similarity searching of chemical databases using atom environment descriptors (MOLPRINT 2D): Evaluation of performance," *Journal of chemical information and computer sciences*, vol. 44, no. 5, pp. 1708–1718, 2004.

[141] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[142] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, "Reoptimization of mdl keys for use in drug discovery," *Journal of chemical information and computer sciences*, vol. 42, no. 6, pp. 1273–1280, 2002.

[143] H. L. Morgan, "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service.," *Journal of chemical documentation*, vol. 5, no. 2, pp. 107–113, 1965.

[144] P. C. Hawkins, A. G. Skillman, and A. Nicholls, "Comparison of shape-matching and docking as virtual screening tools," *Journal of medicinal chemistry*, vol. 50, no. 1, pp. 74–82, 2007.

[145] P. J. Ballester and W. G. Richards, "Ultrafast shape recognition to search compound databases for similar molecular shapes," *Journal of computational chemistry*, vol. 28, no. 10, pp. 1711–1723, 2007.

[146] B. Hu, X. Zhu, L. Monroe, M. G. Bures, and D. Kihara, "Pl-patchsurfer: A novel molecular local surface-based method for exploring protein-ligand interactions," *International journal of molecular sciences*, vol. 15, no. 9, pp. 15 122–15 145, 2014.

[147] W.-H. Shin, X. Zhu, M. G. Bures, and D. Kihara, "Three-dimensional compound comparison methods and their application in drug discovery," *Molecules*, vol. 20, no. 7, pp. 12 841–12 862, 2015.

[148] P. Gainza *et al.*, "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning," *Nature Methods*, vol. 17, no. 2, pp. 184–192, 2020.

[149] F. Sverrisson, J. Feydy, B. E. Correia, and M. M. Bronstein, "Fast end-to-end learning on protein surfaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 272–15 281.

[150] P. K. Agarwal, H. Edelsbrunner, J. Harer, and Y. Wang, "Extreme elevation on a 2-manifold," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 357–365.

[151] K. Xia and G.-W. Wei, "Persistent homology analysis of protein structure, flexibility, and folding," *International journal for numerical methods in biomedical engineering*, vol. 30, no. 8, pp. 814–844, 2014.

[152] K. Xia and G.-W. Wei, "Multidimensional persistence in biomolecular data," *Journal of computational chemistry*, vol. 36, no. 20, pp. 1502–1520, 2015.

[153] V. Kovacev-Nikolic, P. Bubenik, D. Nikoli, and G. Heo, *Statistical Applications in Genetics and Molecular Biology*, vol. 15, no. 1, pp. 19–38, 2016. DOI: doi:10.1515/sagmb-2015-0057. [Online]. Available: https://doi.org/10.1515/sagmb-2015-0057.

[154] Z. Cang, L. Mu, and G.-W. Wei, "Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening," *PLoS computational biology*, vol. 14, no. 1, e1005929, 2018.

[155] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, "Topological exploration of unknown and partially known environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3851–3858.

[156] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *Journal of computer-aided molecular design*, vol. 30, pp. 595–608, 2016.

[157] A. Adcock, D. Rubin, and G. Carlsson, "Classification of hepatic lesions using the matching metric," *Computer vision and image understanding*, vol. 121, pp. 36–42, 2014.

[158] S. Samaga, S. Mukherjee, C. Xin, S. Oudot, and T. K. Dey, "End-to-end pipeline for topological fingerprinting by 2-parameter persistence," in *to appear*, 2024.

[159] A. Gaulton *et al.*, "Chembl: A large-scale bioactivity database for drug discovery," *Nucleic acids research*, vol. 40, no. D1, pp. D1100–D1107, 2012.

[160] M. Davies *et al.*, "Chembl web services: Streamlining access to drug discovery data and utilities," *Nucleic acids research*, vol. 43, no. W1, W612–W620, 2015.

[161] D. Sydow *et al.*, "TeachOpenCADD 2022: open source and fair python pipelines to assist in structural bioinformatics and cheminformatics research," *Nucleic Acids Research*, vol. 50, no. W1, W753–W760, 2022. DOI: 10.1093/nar/gkac267.

[162] E. Noutahi *et al.*, *Datamol-io/molfeat: 0.9.4*, version 0.9.4, Sep. 2023. DOI: 10.5281/zenodo.8373019. [Online]. Available: https://doi.org/10.5281/zenodo.8373019.

VITA

Soham Mukherjee is a PhD student in the Department of Computer Science at Purdue University, having joined in August 2020. Prior to Purdue, he began his PhD journey in the Department of Computer Science and Engineering at the Ohio State University in 2017. Soham holds a Master of Science degree in Computer Science from Ohio State University and a Bachelor's degree in Electronics and Tele-communication Engineering from Jadavpur University, India.