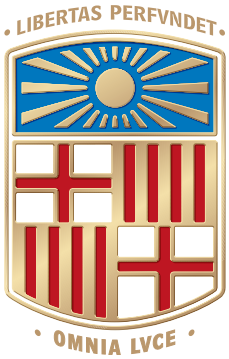
Topology-Enhanced Deep Learning



Rubén Ballester Bautista

Supervised by:

Prof. Sergio Escalera Guerrero
Prof. Carles Casacuberta Vergés
Prof. Bastian Grossenbacher Rieck

Programa de doctorat en Matemàtiques i Informàtica

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

A thesis submitted for the degree of

Doctor of Philosophy

Barcelona 2025

Acknowledgements

Writing this thesis has been a long path, filled with amazing people to whom I'm truly grateful. First of all, I want to express my gratitude to my advisors, Sergio Escalera, Carles Casacuberta, and Bastian Rieck. Sergio, I cannot express enough gratitude for the opportunity you gave me when you decided to support me after leaving my engineering position at Oracle to pursue research. You saw my potential and decided to bet on me when I needed it the most. I'm really happy with what we have achieved together during these 8(!) years. We have conducted substantial research and shared many good moments. Thank you very much, Sergio. Carles, thank you very much for your counseling and your wisdom. I have enjoyed each of our long talks about mathematics, our trips around the world, and the time we have spent together overall. You have not only contributed to my development as a researcher, but as a person. Thank you for motivating me to work hard and to take a close look at the details. Thank you also for doing everything from your side to let me achieve my intellectual objectives. Every seminar you prepared because I was interested in the topic had a deep impact on me. You know, thanks to you, I'm more than motivated to start my new chapter as a researcher in AGI. Thank you very much, Carles. Bastian, although I started working with you recently, you have shown me what a great researcher should look like. Your intelligence, ethics, and deep knowledge about every topic I could think of motivates me to be my best version. I'm really grateful that you gave me the opportunity to work with you on the expressivity paper, and then on the following projects. I think that since I started in AIDOS, I have evolved as a researcher. I cannot conceive topological deep learning without you as a leader in the field, and I'm already missing our interactions. I really hope that this thesis is not our last collaboration: it is hard to leave with such great research topics in AIDOS! Thank you very much, Bastian.

Secondly, I wanted to express my gratitude to my (ex-)*altellers*. Manu, David, German, Artur, Javi, Johnny, Hunor, you mean a world to me. I consider you more than simply colleagues, but friends. I have loved each conversation I have had with you about politics, AI... Each time I share even a few minutes with you, I learn something new. I will always remember our seminars, outdoor walks, and meetings. Extra kudos for Manu and David: you have supported me no matter what, each time I had doubts about myself, my research, and my future. I could not ask more from you. Thank you very much.

Y por último, y más importante. Gracias a toda mi familia y a mis amigos. Masa, cielo, gracias por tu apoyo, tu comprensión, y tu cariño. No podría haber

hecho esto sin ti. Soy la persona más afortunada del mundo. Te quiero. Mamá, eres mi inspiración. Eres la persona más luchadora que conozco y eso me ha marcado profundamente. Esta tesis no sería ni la mitad de buena sin tus enseñanzas. Y por cierto, Gerard, Lorenzo, Manu, Marc, Nacho, Iván, Xavi, que no me olvido de vosotros. Sois unos cracks, y os admiro muchísimo. Gracias por vuestro apoyo durante todo este camino. Espero que os guste mi tesis.

Abstract

This thesis presents a comprehensive examination of topological deep learning through two complementary approaches: utilizing topological tools to analyze and improve traditional neural networks, and testing and developing novel architectures for learning on high-order topological domains such as simplicial or cellular complexes.

The thesis is organized into three thematic blocks. First, we establish the theoretical foundations of TDA and TDL, providing a comprehensive literature review of existing methodologies. Second, we exploit persistent homology —a fundamental TDA tool that quantifies multi-scale topological features— to analyze neural network activations and their relationship to neural network performance and generalization. We use the differentiability theory of persistent homology to develop topological regularization methods that demonstrably improve neural network performance on selected problems.

In the third block, we address the challenges of learning on high-order domains such as simplicial and cellular complexes. We introduce MANTRA, a novel topological dataset specifically designed to evaluate the capacity of TDL methods to leverage high-order structural information. Furthermore, we develop the Cellular Transformer, an adaptation of the transformer architecture to cellular complexes that addresses some of the limitations of message passing neural networks, such as their general inability to capture long-range interactions.

This thesis advances both the theoretical understanding of neural networks through a topological lens and the practical capabilities of learning algorithms on topological structures. The empirical results obtained in this thesis demonstrate that topology provides valuable insights into the geometric processes underlying deep learning while enabling more effective feature extraction from complex data representations. Thus, the research presented here expands the state-of-the-art in topological deep learning, establishing a foundation for more interpretable, topologically-aware neural architectures while opening promising avenues for future research at the intersection of algebraic topology and deep learning.

Resum

Aquesta tesi presenta un estudi exhaustiu de l'aprenentatge profund topològic mitjançant dos enfocaments complementaris: l'ús d'eines topològiques per analitzar i millorar les xarxes neuronals tradicionals, i el desenvolupament i l'anàlisi de noves arquitectures per a l'aprenentatge automàtic en dominis topològics d'ordre superior, com ara els complexos simplicials o els complexos cel·lulars.

La tesi s'estructura en tres blocs. En el primer bloc, establim els fonaments teòrics de l'anàlisi de dades topològica (ADT) i l'aprenentatge profund topològic (APT) i realitzem una revisió bibliogràfica exhaustiva de les metodologies existents. En el segon bloc, utilitzem l'homologia persistent, una eina fonamental de l'ADT, per analitzar les activacions de les xarxes neuronals i la seva relació amb el rendiment i la generalització de la xarxa i per desenvolupar mètodes de regularització topològica que milloren de manera demostrable el rendiment de les xarxes neuronals en problemes seleccionats.

En el tercer bloc, tractem l'ús de l'aprenentatge automàtic en dominis d'ordre superior, com ara els complexos simplicials o els cel·lulars. En primer lloc, desenvolupem MANTRA, un nou conjunt de dades topològic dissenyat específicament per avaluar la capacitat dels mètodes d'APT per aprofitar la informació estructural d'ordre superior contingut en els diferents complexos d'ordre superior. En segon lloc, desenvolupem el Transformer Cel·lular, una adaptació de l'arquitectura Transformer als complexos cel·lulars que resol algunes de les limitacions fonamentals de les xarxes neuronals de pas de missatges en complexos cel·lulars, com ara la seva incapacitat general per capturar interaccions de llarg abast entre cel·les del mateix complex.

En resum, aquesta tesi avança tant en la comprensió teòrica de les xarxes neuronals, estàndar i d'ordre superior, a través d'una òptica topològica com en el desenvolupament d'aquestes xarxes. Els resultats obtinguts demostren que la topologia proporciona informació valuosa sobre els processos geomètrics subjacents a l'aprenentatge profund i que les xarxes d'ordre superior obtenen un rendiment superior a les xarxes estàndard quan l'estructura topològica de les dades és utilitzada correctament durant el procés d'inferència neuronal. Així doncs, la recerca presentada en aquesta tesi amplia l'estat de l'art en l'aprenentatge profund topològic, establint una base per a arquitectures neuronals més interpretables i conscients de la topologia, alhora que obre vies prometedores per a la investigació futura en la intersecció de la topologia algebraica i l'aprenentatge profund.

Contents

List of Figures

List of Abbreviations

DL	Deep Learning
ML	Machine Learning
TDL	Topological Deep Learning
TDA	Topological Data Analysis
TSP	Topological Signal Processing
GDL	Geometric Deep Learning
MPNN	Message Passing Neural Network
SOTA	State of the Art
PH	Persistent Homology
GNN	Graph Neural Network
HONN	High-Order Neural Network
TOGL	Topological Graph Layer
PGDL	Predicting Generalization in Deep Learning
MST	Minimum Spanning Tree
WL	Weisfeiler–Leman
FCFNN	Fully Connected Feedforward Neural Network
MLP	Multi-Layer Perceptron
GCN	Graph Convolutional Network
GAT	Graph Attention Network
UniMP	Unified Message Passing
TAG	Topology Adaptive Graph Convolutional Network
SAN	Simplicial Attention Network
SCCN	Simplicial Complex Convolutional Network
SCCNN	Simplicial Complex Convolutional Neural Network
SCN	Simplicial Complex Network
CellMP	Cellular Message Passing
DECT	Differentiable Euler Characteristic Transform

CT . . . . . Cellular Transformer

AMCC . . . . . Augmented Molecular Cellular Complex

PCA . . . . . Pairwise Cellular Attention

GCA . . . . . General Cellular Attention

LapPE . . . . . Laplacian Positional Encoding

RWPE . . . . . Random Walk Positional Encoding

TopoSlepiansPE . Topological Slepians Positional Encoding

BSPe . . . . . Barycentric Subdivision Positional Encoding

HodgeLapPE . . . Hodge Laplacian Positional Encoding

1. Introduction

Contents

1.1 Motivation

Machine learning (ML) is a central branch of artificial intelligence (AI) dedicated to developing general-purpose algorithms capable of identifying patterns within data distributions and learning how to use these patterns to make accurate inferences about the underlying properties of the data. Among the different branches of machine learning, deep learning (DL) focuses on the use of artificial neural networks (ANN), computational models that were originally inspired by the structure and behavior of biological neural networks.

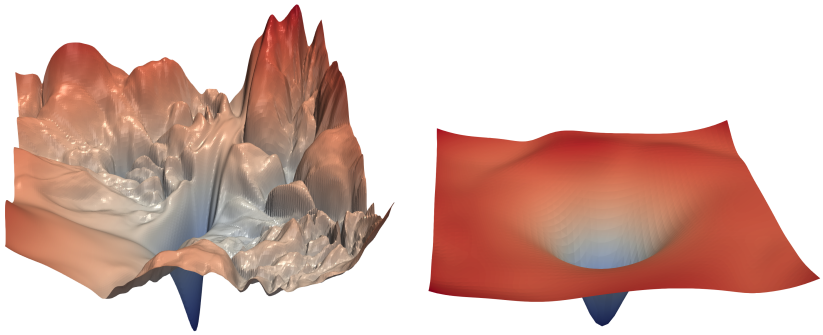
The origin of ANNs can be traced back to [1], where McCulloch and Pitts developed the first mathematical model of biological neural networks as computational graphs composed of nodes, representing simple artificial neurons, which could represent a set of logic propositions extending propositional logic, denominated *temporal* logic propositions, where the temporality came from the order in which the neurons are *fired* in the model. Later on, notable scientists such as Marvin Lee Minsky or Frank Rosenblatt, inspired by McCulloch and Pitts's work, continued developing models of artificial networks capable of solving problems such as Shannon's maze [2] or distinguishing a square from a circle printed on paper [3, 4]. Since then, ANNs research flourished, not without difficulties in its early stages, surviving two major *AI winters* between the 1970s and 1990s [5], where the AI funding suffered major cuts and the industry collapsed due to inflated expectations

that far exceeded the computational power and algorithmic capabilities available at the time. The eventual convergence of algorithmic breakthroughs such as backpropagation, introduced in the 80s, modern GPU architectures, and increasing availability to vast datasets, led to a revolution of AI and ANNs, paving the way for the 2020s, a decade defined by the dominance of massive-scale models leading the AI field and contributing to the progression on some of the humanity's most challenging problems such as autonomous driving [6, 7], automatic math discovery [8, 9], or nuclear fusion [10], among others.

However, the proliferation of large-scale neural networks has been accompanied by significant challenges. For example, the high complexity of state-of-the-art neural networks has made comprehending essential properties of networks, such as their interpretability, generalization ability, or suitability for specific problems [11, 12], an unsolved research problem. Not only that, but also with the introduction of more complex data sets for these networks, new challenges on how to leverage the relevant information from them effectively have arisen. A prominent example of these challenges can be found in basic topological tasks such as predicting the Betti numbers of simplicial complexes coming from manifold triangulations, where, as we will see in Chapter 6, state-of-the-art methods are not able to capture topological invariants associated to the input data, yielding performances equivalent to the performances of random-guessing models.

During the last years, Topological Deep Learning (TDL) [13], a specialized subfield within the broader framework of Geometric Deep Learning (GDL) [14], has arisen as a promising discipline using tools from algebraic topology and combinatorics to offer (partial) solutions to the aforementioned problems. Fundamentally, it incorporates principles and methodologies from Topological Data Analysis (TDA) [15] and Topological Signal Processing (TSP) [16, 17], which offer a framework for gaining insights into the *global shape* of data, in a broad sense.

The notion of *shape* is ubiquitous in contemporary deep learning research, manifesting across multiple theoretical frameworks and applications including the manifold hypothesis [18], which posits that high-dimensional data concentrates



(a) without skip connections **(b)** with skip connections

Figure 1.1: Comparative visualization of loss surfaces for ResNet-56 [21] as presented in [19, Figure 1]. Loss surfaces are low-dimensional projections mapping the high-dimensional parameter space to loss values. Figure (a) depicts the loss surface without skip connections, exhibiting a highly-chaotic, mountain-like surface that makes minimization with iterative algorithms hard. In contrast, Figure (b) shows the loss surface with skip connections, characterized by a smoother, more convex geometry that facilitates gradient-based optimization. This visualization reveals how critical is the shape, in this case of loss surfaces, when choosing architectural modifications, directly impacting convergence behavior and ultimately network performance.

near lower-dimensional manifolds; loss landscapes [19], which characterize the optimization geometry of parameter spaces and their corresponding performance metrics; or the intrinsic geometry of learned representations [20], which captures the fundamental organizational principles of latent feature spaces, among others. These examples merely represent a subset of relevant research directions, underscoring the significance of methodologies emerging from the topological deep learning community and explaining the growing attention this field has attracted within the broader deep learning research ecosystem lately.

This thesis aims to both (1) *understanding* the capacity of classical and topology-based neural network models; and, (2) *improving* this capacity either by using TDA tools or by developing new architectures that leverage the combinatorial and topological structure of the input in a better way than current architectures.

Topological data analysis is a discipline that adapts methods from algebraic topology for the quantitative analysis of (usually discrete) data sets. The origins of TDA date back to the 1990s [22, 23], specifically to the works of Patrizio Frosini (1992) [24] on size functions and Vanessa Robins (1999) [25] on the inference of the topology of attractors in dynamical systems using finite sets of data. Both

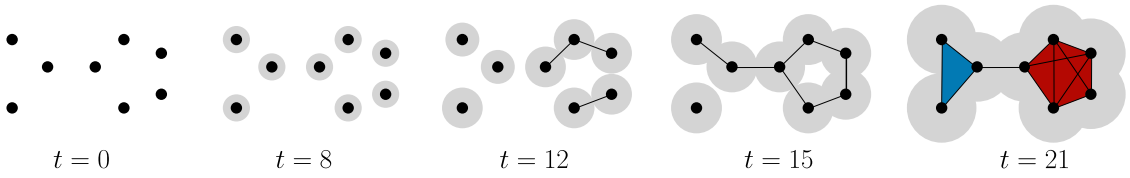


Figure 1.2: Čech filtration at time values $t \in \{0, 8, 12, 15, 21\}$ for a point cloud P equipped with the Euclidean distance d in the plane. For $t < 0$, the simplicial set $\check{\mathrm{C}}_t(P, \mathbb{R}^2, d)$ is the empty set. For $t \in \{12, 15\}$, only edges are added as there are no three balls of radius t centered at the points in P with common intersection. For $t = 21$, one triangle and two tetrahedra are added to the filtration. Eventually, for all t after a threshold, $\check{\mathrm{C}}_t(P, \mathbb{R}^2, d)$ becomes a simplex of dimension equal to the number of points of P minus one, because all balls of radius t higher than the threshold centered at the points in P have a common intersection. More information about the Čech filtration can be found in Section 2.1.6.

works paved the way for persistent homology, the main tool of TDA, which was established as a unified framework by Edelsbrunner, Letscher and Zomorodian [23, 26] in the early 2000s and subsequently generalized to persistence modules for the first time by Zomorodian and Carlsson [27].

Persistent homology quantifies multi-scale homological features in data which may not even have associated a meaningful topological structure, such as finite sets of points, usually endowed with a dissimilarity function. Unlike classical homology, which characterizes the homological structure of a single and well-defined space by counting its different n-dimensional *holes*[1] for each possible dimension n such as connected components, loops, or voids, persistent homology tracks how these topological features appear and disappear across a filtration, which is a nested sequence of spaces obtained by varying a parameter. This crucial distinction favours the use of persistent homology against traditional homology, especially for data with unknown topology, as it offers a way to attach a flexible topological structure to data whenever the data do not have it naturally, and more fine-grained topological invariants taking into account the stability and significance of topological features across multiple scales of observation.

During the first part of the thesis, we use persistent homology to dilucidate some relationships between neural network activations and their performance. Moreover,

[1] A precise definition of homology is given in Section 2.1.4.

thanks to the differentiability theory of persistent homology [28, 29], we use this information to build topological regularization terms that improve neural network performances in some toy problems. Additionally, we study the limits in terms of graph-learning expressivity[2] of persistent homology-based learning algorithms both in theory, for general filtrations, and in practice, for specific filtrations and datasets.

The nested sequences of spaces used by persistent homology consist of inclusion sequences of abstract simplicial complexes, which are combinatorial generalizations of geometric simplicial complexes, which are sets of simplices satisfying certain conditions. Simplices can be seen as generalizations of triangles for any dimension, being points, edges, triangles, and tetrahedra the representatives of simplices for dimensions zero, one, two, and three, respectively. Simplices have a hierarchical relationship where the boundary of an n-dimensional simplex is a set of $(n-1)$-dimensional simplices: the boundary of tetrahedra is composed of triangles, while the boundary of triangles is composed of edges, and so on. The hierarchical relationship of simplices and its natural extension to simplicial complexes is known as the *incidence structure* of simplices or simplicial complexes, respectively.

The incidence structure is a fundamental tool in TDL, as it is required to perform most computations, including the computation of (persistent) homology, but not limited to it. In particular, incidence structures of simplicial complexes allow us to define notions of closeness and neighborhood in simplicial complexes, which makes it possible to extend the message passing neural network (MPNN) framework for graph learning to the realm of higher-order domains, including (abstract) simplicial complexes and some powerful generalizations such as (abstract) cellular complexes or combinatorial complexes [30]. Message passing neural networks and neural network learning on high-order domains are the main topic of the second part of this thesis, where we analyze the capabilities of current state-of-the-art TDL architectures and propose a new architecture based on the transformer model for graphs [31, 32]. Our model addresses, in theory, some of the drawbacks of the MPNN framework, such

[2]There are many definitions of graph-learning expressivity. Through the text, we use graph-learning expressivity in the sense of the Weisfeiler–Leman test. A concise introduction of graph expressivity can be found in Chapter 2.

as the capacity of learning long range interactions in the input data [33] or the limited expressivity [34, 35], without the need of complicated engineering solutions.

In summary, this thesis exploits the fundamental connection between the concept of shape as described by geometric and topological methods and deep learning techniques, employing and developing tools from the TDL framework to study and enhance neural networks working on both classical and non-classical domains such as images, graphs, or simplicial complexes.

Since I started this thesis, it is my belief that topology is an invaluable tool for understanding and improving deep learning and its methods. On one hand, it is hard to imagine humans fully understanding the underlying geometric processes happening in deep learning methods. However, it is not that hard to imagine humans understanding what is happening *globally* in these complex systems, which is precisely the focus of the field of topology. On the other hand, topology already provides tools for extracting, modeling, and leveraging information from real-world data, as demonstrated by the many successful applications of TDA in sciences [36–40]. It is likely that successful neural networks in high-order domains such as simplicial complexes need to fully leverage the hidden topological and high-order structural patterns, and this endeavour most likely involves developing architectures and strategies that explicitly take these patterns into consideration. Thus, I sincerely hope that this thesis serves as a step forward in the field and as an inspiration for future researchers attempting to discover the intrincate patterns of this new tool that is revolutionizing our reality: deep learning.

1.2 Contributions and structure of the thesis

The thesis is divided into three thematic blocks consisting of seven chapters in total plus a final concluding chapter. The three parts are the following:

(1) **Preliminaries**. This block consists of an introduction (Chapter 1), the necessary background on TDL to understand the thesis (Chapter 2), and an extended literature review of methods in TDL to analyze and improve

neural networks, neural network architectures for high-order domains such as simplicial or cellular complexes, and persistent homology-based neural networks for graph domains. The literature review is also contained in our monograph [41] which was developed entirely within the scope of this PhD thesis.

(2) **Persistent homology for deep learning**. This block consists of two chapters, the first one dedicated to the use of (differentiable) persistent homology to the analysis and improvement of *classical* neural networks (Chapter 4), and the second one dedicated to the expressivity analysis of persistent homology based graph neural networks. Both chapters present original research developed exclusively within the scope of this dissertation. Specifically, in Chapter 4 we extend the work done in [42, 43] to arbitrary-size neural architectures by bootstrapping topological feature vectors computed from activation samples. Using the dataset from the PGDL global competition, we uncover a correlation between the generalization gap of a network and these topological features. Then, we leverage this correlation to design novel topological regularization terms that improves generalization. In Chapter 5, we provide a theoretical analysis of the expressivity of persistent homology for graph learning, establishing a new full characterization of persistent homology expressivity in terms of the Weisfeiler-Leman test for graph isomorphism and demonstrating experimentally that persistent homology achieves competitive performance on several expressivity benchmarks unexplored until now.

(3) **High-order learning**. This block consists of another two chapters, the first one dedicated to the development of a novel topological dataset to evaluate the performance state-of-the-art TDL methods for learning in high-order domains that require to leverage information from the high-order structures of the data to be successful (Chapter 6), and the second one dedicated to the novel adaptation of transformer architectures to the realm of cellular complexes

(Chapter 7). As in the previous block, both chapters present original research developed during the writing of this dissertation.

1.3 Publications and software contributions

The original research presented in this dissertation has resulted in several peer-reviewed publications, preprints available on arXiv, and open-source software contributions. The following lists summarize these contributions.

Published works

- Rubén Ballester, Xavier Arnal Clemente, Carles Casacuberta, Meysam Madadi, Ciprian A. Corneanu, and Sergio Escalera. "Predicting the generalization gap in neural networks using topological data analysis". In: *Neurocomputing* 596 (2024), p. 127787. URL: https://www.sciencedirect.com/science/article/pii/S0925231224005587. Related to: Chapter 4.
- Rubén Ballester, Carles Casacuberta, and Sergio Escalera. "Decorrelating neurons using persistence". In: *Proceedings of the 2nd NeurIPS Workshop on Symmetry and Geometry in Neural Representations*. Ed. by Sophia Sanborn, Christian Shewmake, Simone Azeglio, and Nina Miolane. Vol. 228. Proceedings of Machine Learning Research. PMLR, 16 Dec 2024, pp. 164–182. URL: https://proceedings.mlr.press/v228/ballester24a.html. Related to: Chapter 4.
- Rubén Ballester and Bastian Rieck. "On the Expressivity of Persistent Homology in Graph Learning". In: *Proceedings of the Third Learning on Graphs Conference*. Ed. by Guy Wolf and Smita Krishnaswamy. Vol. 269. Proceedings of Machine Learning Research. PMLR, 26–29 Nov 2025, 42:1–42:31. URL: https://proceedings.mlr.press/v269/ballester25a.html. Related to: Chapter 5.
- Rubén Ballester, Ernst Röell, Daniel Bin Schmid, Mathieu Alain, Sergio Escalera, Carles Casacuberta, and Bastian Rieck. "MANTRA: The Manifold Triangulations Assemblage". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=X6y5CC44HM. Related to: Chapter 6.

- Melih Barsbey, Rubén Ballester, Andac Demir, Carles Casacuberta, Pablo Hernández-García, David Pujol-Perich, Sarper Yurtseven, Sergio Escalera, Claudio Battiloro, Mustafa Hajij, and Tolga Birdal. "Higher-Order Molecular Learning: The Cellular Transformer". In: *ICLR 2025 Workshop on Generative and Experimental Perspectives for Biomolecular Design*. 2025. URL: https://openreview.net/forum?id=GW3h79mxrF. Related to: Chapter 7.
- Rubén Ballester, Carles Casacuberta, and Sergio Escalera. *Topological Data Analysis for Neural Networks*. Springer Briefs in Computer Science. Springer Cham, 2025. URL: https://link.springer.com/book/9783032082824. Related to: Chapter 2.

Preprints

- Rubén Ballester, Pablo Hernández-García, Mathilde Papillon, Claudio Battiloro, Nina Miolane, Tolga Birdal, Carles Casacuberta, Sergio Escalera, and Mustafa Hajij. *Attending to Topological Spaces: The Cellular Transformer*. 2024. arXiv: 2405.14094 [cs.LG]. URL: https://arxiv.org/abs/2405.14094. Related to: Chapter 7.

Open-source software contributions

- **TopoX software package**. Contributed to TopoModelX by partially developing the Implementation of HOAN Mesh Classification model. Contributed to TopoNetX by finding bugs and adding small fixes. https://github.com/pyt-team/TopoModelX/pull/104. https://github.com/pyt-team/TopoNetX/issues?q=author%3ArballebaX. Contributions led to two publications:

  (1) Mathilde Papillon, Mustafa Hajij, Audun Myers, Helen Jenne, Johan Mathe, Theodore Papamarkou, Aldo Guzmán-Sáenz, Neal Livesay, Tamal Dey, Abraham Rabinowitz, Aiden Brent, Alessandro Salatiello, Alexander Nikitin, Ali Zia, Claudio Battiloro, Dmitrii Gavrilev, German Magai, Gleb Bazhenov, Guillermo Bernardez, Indro Spinelli, Jens Agerberg, Kalyan Nadimpalli, Lev Telyatninkov, Luca Scofano, Lucia Testa, Manuel Lecha, Maosheng

Yang, Mohammed Hassanin, Odin Hoff Gardaa, Olga Zaghen, Paul Hausner, Paul Snopoff, Rubén Ballester, Sadrodin Barikbin, Sergio Escalera, Simone Fiorellino, Henry Kvinge, Karthikeyan Natesan Ramamurthy, Paul Rosen, Robin Walters, Shreyas N. Samaga, Soham Mukherjee, Sophia Sanborn, Tegan Emerson, Timothy Doster, Tolga Birdal, Abdelwahed Khamis, Simone Scardapane, Suraj Singh, Tatiana Malygina, Yixiao Yue, and Nina Miolane. "ICML 2023 Topological Deep Learning Challenge: Design and Results". In: *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*. ed. by Timothy Doster, Tegan Emerson, Henry Kvinge, Nina Miolane, Mathilde Papillon, Bastian Rieck, and Sophia Sanborn. Vol. 221. Proceedings of Machine Learning Research. PMLR, 28 Jul 2023, pp. 3–8

(2) Mustafa Hajij, Mathilde Papillon, Florian Frantzen, Jens Agerberg, Ibrahem AlJabea, Rubén Ballester, Claudio Battiloro, Guillermo Bernárdez, Tolga Birdal, Aiden Brent, Peter Chin, Sergio Escalera, Simone Fiorellino, Odin Hoff Gardaa, Gurusankar Gopalakrishnan, Devendra Govil, Josef Hoppe, Maneel Reddy Karri, Jude Khouja, Manuel Lecha, Neal Livesay, Jan MeiÃŸner, Soham Mukherjee, Alexander Nikitin, Theodore Papamarkou, Jaro Prílepok, Karthikeyan Natesan Ramamurthy, Paul Rosen, Aldo Guzmán-Sáenz, Alessandro Salatiello, Shreyas N. Samaga, Simone Scardapane, Michael T. Schaub, Luca Scofano, Indro Spinelli, Lev Telyatnikov, Quang Truong, Robin Walters, Maosheng Yang, Olga Zaghen, Ghada Zamzmi, Ali Zia, and Nina Miolane. "TopoX: A Suite of Python Packages for Machine Learning on Topological Domains". In: *Journal of Machine Learning Research* 25.374 (2024), pp. 1–8. URL: `http://jmlr.org/papers/v25/24-0110.html`

- **Minor update in Giotto-TDA**. Updated `pybind` package to latest version to make Giotto-TDA compatible with newer versions of Python.
  `https://github.com/giotto-ai/giotto-tda/pull/689`

2. Background

Contents

2.1 Topological Data Analysis

This section provides the essential background in topological data analysis (TDA) for the remainder of the dissertation. We introduce simplicial and cellular complexes together with their incidence structures, which we use to define homology, persistent homology (PH), Hodge Laplacians and related transformations, and topological summaries. We also discuss the barycentric subdivision transformation and review basic notions of manifolds and triangulated manifolds, including low-dimensional triangulability and classification. Persistent homology is used in Chapters 4 and 5 to study the relationship between neural-network generalization and PH-based topological summaries, and to analyze the expressivity of PH-driven learning algorithms. Hodge Laplacians are used in Chapters 5 and 7 both to design

filtrations for PH-based learning methods and to develop positional encodings for our transformer architecture on cellular complexes. Manifold triangulations are used in Chapter 6 to construct the MANTRA dataset and to evaluate the capacity of topological deep learning methods to leverage higher-order structural information. Barycentric subdivision is used in Chapter 6 to generate new triangulations of manifolds and in Chapter 7 to develop positional encodings for our transformer architecture on cellular complexes. For a more extensive background in general topology, see Munkres [52]; for a deeper introduction to computational topology, see Edelsbrunner and Harer [53].

2.1.1 Simplicial complexes

An *abstract simplicial complex* is a collection K of non-empty finite subsets of a set P such that $\{p\}$ is in K for all $p \in P$ and such that $\sigma' \in K$ whenever $\sigma' \subseteq \sigma$ with $\sigma \in K$. Elements of P are called *vertices* and elements of K are called *simplices*. If $\sigma' \subseteq \sigma$, then σ' is called a *face* of σ. A simplex σ has *dimension* d if its cardinality is $d+1$, and the dimension of a finite simplicial complex K is the maximum dimension of its simplices. From now on, we omit the term "abstract" for shortness.

Every undirected simple graph $G = (V, E)$ with set of vertices V and set of edges E can be viewed as a 1-dimensional simplicial complex with $P = V$ and a simplex $\sigma_e = \{v, w\}$ for every edge $e \in E$ joining v and w.

An *affine simplex* of dimension d is a convex hull of $d+1$ affinely independent points $\{p_0, \ldots, p_d\}$ in a Euclidean space $\mathbb{R}^n$ for some $n \geq 1$, that is,

$$\Delta(p_0, \ldots, p_n) = \left\{ \sum_{i=0}^{d} c_i p_i \text{ such that } \sum_{i=0}^{d} c_i = 1 \text{ and } c_i \geq 0 \text{ for all } i \right\}.$$

A *geometric realization* of a finite simplicial complex K is the union of a collection of affine simplices Δ_σ in the same ambient Euclidean space, one for each simplex $\sigma \in K$, where σ is mapped bijectively to the vertices of Δ_σ, and where any non-empty intersection of two affine simplices Δ_σ and Δ_τ of the collection is another affine simplex $\Delta_{\sigma \cap \tau}$ in the collection. Any two geometric realizations of a simplicial complex K are homeomorphic through a face-preserving map.

The *barycentric subdivision* of a simplicial complex K is the simplicial complex $\mathrm{Sd}(K)$ obtained by setting its d-dimensional faces to be sequences of strict inclusions $\sigma_0 \subset \sigma_1 \subset \cdots \subset \sigma_d$ of simplices of K. It then follows that K and $\mathrm{Sd}(K)$ have homeomorphic geometric realizations [54, Proposition 1.13].

Given a simplicial complex K, an *incidence* relation can be defined between its simplices, by declaring that σ_1 is *incident* to σ_2 if either $\sigma_1 \subseteq \sigma_2$ or $\sigma_2 \subseteq \sigma_1$. If K is equipped with an order on its set of vertices, then the *signed incidence* between a d-simplex $(v_0, \ldots, v_d)$ with $v_0 < \cdots < v_d$ and one of its faces $(v_0, \ldots, \hat{v}_i, \ldots, v_d)$ is $(-1)^i$, where $\hat{v}_i$ means that v_i is omitted. If two simplices are not incident, then their signed incidence is zero.

2.1.2 Cellular complexes

Cellular complexes generalize the concept of simplicial complex to more flexible structures. A *cellular complex* is a topological space X that can be decomposed as a union of disjoint subspaces called *cells*, where each cell σ is homeomorphic to $\mathbb{R}^k$ for some integer $k \geq 0$, called the *rank* of σ. Additionally, for every cell σ, the difference $\overline{\sigma} \smallsetminus \sigma$ is contained in the union of finitely many cells of lower rank, where $\overline{\sigma}$ denotes the closure of σ. The *dimension* of a finite cellular complex is the maximum of the ranks of its cells. The set of cells of rank k in a cellular complex X is denoted by X_k. The *n-skeleton* of X is the cellular complex spanned by $X_0, \ldots, X_n$, for $0 \leq n \leq \dim X$.

A *characteristic map* for a cell σ of rank k is a map from the Euclidean unit closed ball of dimension k into $\overline{\sigma}$ whose restriction to the open ball is a homeomorphism. A cellular complex is called *regular* if each cell σ admits a characteristic map which is itself a homeomorphism from the closed ball to $\overline{\sigma}$. For example, a decomposition of a circle as the union of a 0-cell and a 1-cell is not regular, while one with two 0-cells is regular. Geometric realizations of abstract simplicial complexes are regular cellular complexes. Cellular complexes generalize simplicial complexes as their cells are not constrained to be simplices.

The restriction of a characteristic map for a cell σ of rank k to the boundary of the unit closed ball, i.e., the sphere S^{k-1}, is called an *attaching map* for the cell σ. Thus, the image of an attaching map for a cell σ of rank k is contained in the $(k-1)$-skeleton of X.

As in the case of simplicial complexes, an incidence relation can be defined between cells by inclusion of their closures. Moreover, signed incidence is defined between cells of consecutive ranks, assuming that an orientation has been chosen on each cell. If σ is a cell of rank k with attaching map f, and τ is a cell of rank $k-1$, then the *signed incidence* between σ and τ is the degree of the map $S^{k-1} \to S^{k-1}$ obtained from f by collapsing, within the $(k-1)$-skeleton of X, all cells except τ. If X is regular, then signed incidences are equal to ± 1, or zero if τ is not contained in the image of f.

Combinatorial cellular complexes. Representing cellular complexes for their use in computational tasks requires a purely combinatorial definition, similar to the definition of an abstract simplicial complex. The mathematical literature contains different approaches to such a combinatorial definition, notably [55–58]. Nevertheless, there is no conventional choice of formalism for this purpose. In this dissertation, we use the approach outlined in [49], which is restricted to finite 2-dimensional regular cellular complexes, in view of their use in Chapter 7.

Hence, we define a *combinatorial regular cellular 2-complex* (for shortness, a *cellular complex*, if no ambiguity can arise) as a triple $X = (X_0, X_1, X_2)$ of finite ordered sets, where elements $v \in X_0$ are called *nodes*, *vertices*, or 0-*cells*, elements $e \in X_1$ are called *edges* or 1-*cells*, and elements $\sigma \in X_2$ are called *faces* or 2-*cells*, equipped with an *incidence* relation, which is defined as follows. We associate to each edge $e \in X_1$ an ordered pair of distinct vertices $v_1 < v_2$, and say that v_1 and v_2 are *incident* to e. Then we denote $e = (v_1, v_2)$, and the oppositely oriented edge is denoted by $-e = (v_2, v_1)$. A collection of edges $e_1, \ldots, e_m$ form a *closed path* if there is a set of distinct vertices $v_1, \ldots, v_m$ such that $\pm e_i = (v_i, v_{i+1})$ for $1 \leq i \leq m$ and $v_{m+1} = v_1$. We associate to each face $\sigma \in X_2$ an ordered sequence of edges

$(e_1, \ldots, e_{m(\sigma)})$ that form a closed path without self-intersections and constitute the edges *incident* to σ. We assume that $m(\sigma) \geq 3$ to ensure that the pair (X_0, X_1) is a (loopless, simple, directed) graph. The subscript k of each set X_k is called its *rank*.

Signed incidences between cells of consecutive ranks are encoded into *incidence matrices*. The first incidence matrix B_1 has (i, j) entry equal to -1 if the j-th edge e_j starts at the i-th vertex v_i, 1 if e_j ends at v_i, and 0 otherwise. The entries of the second incidence matrix B_2 are the incidence numbers between faces and edges, where the *incidence number* of a face σ with an edge e is the sign of $\pm e$ if it belongs to a closed path of edges incident to σ, and 0 otherwise. These two matrices satisfy $B_1 B_2 = 0$, as shown in [59].

Incidence allows us to define *adjacencies* between cells. We say that two cells σ and σ' of the same rank k are *upper adjacent* if they are incident to a common cell τ of rank $k+1$, and *lower adjacent* if they are incident to a common cell τ of rank $k-1$. We define upper and lower *adjacency matrices* A_k^{up} and A_k^{down} for $k = 1, 2$ to represent connections between cells of the same rank. In these matrices, an (i, j) entry equal to 1 indicates that cells i and j (both of rank k) are upper adjacent or lower adjacent, respectively.

Hodge Laplacian. Hodge Laplacians are generalizations of graph Laplacians to higher-dimensional spaces such as simplicial or cellular complexes. They are defined using incidence matrices, and they are a key tool in topological signal processing. Given a cellular complex X, the Hodge Laplacian of rank n is defined as

$$L_n = B_{n+1} B_{n+1}^T + B_n^T B_n, \tag{2.1}$$

where B_n is the incidence matrix of dimension n. The matrix L_n defines a self-adjoint operator on the vector space of functions on X with values in $\mathbb{R}$ with many interesting properties such as the connection between the kernel of L_n and the n-th homology group of X, described in Section 2.1.4. Usually, $B_{n+1} B_{n+1}^T$ and $B_n^T B_n$ are called *upper* and *lower* Laplacians, and are denoted by L_n^{up} and L_n^{down}, respectively. For more details on Hodge Laplacians, see [60].

2.1.3 Manifolds

An *n-dimensional manifold* is a second-countable Hausdorff topological space M such that every point of M is contained in some open set U, called a *chart*, equipped with a homeomorphism into an open subset of $\mathbb{R}^n$, see [61, § 36]. This definition does not include manifolds with boundary, which are not considered in this dissertation. A manifold (without boundary) is called *closed* if its underlying topological space is compact.

A collection of charts covering a manifold M is an *atlas* of M. A manifold M is called *orientable* if it admits an atlas with compatible orientations on its charts. For a closed connected n-dimensional manifold M, orientability is determined by its n-th Betti number β_n, which is nonzero if and only if M is orientable. The definition of Betti numbers is recalled in Section 2.1.4.

A *triangulation* of a manifold M is a simplicial complex whose geometric realization is homeomorphic to M. In [62], it is proved that every surface admits a triangulation (which can be chosen to be finite if the surface is compact), and that any two such triangulations admit a common refinement. Moise [63] proved that the same facts are true for 3-dimensional manifolds. For dimensions greater than 3, however, there are examples of manifolds that cannot be triangulated [64, 65]. The triangulability property of manifolds up to dimension three facilitates the computational manipulation of these manifolds, making them better-suited than higher-dimensional manifolds for computational analyses.

Classification. Closed connected surfaces can be classified, up to homeomorphism, as given by the following list: (i) the two-dimensional sphere S^2; (ii) a connected sum of tori T^2; (iii) a connected sum of projective planes $\mathbb{R}P^2$. The *genus* of a surface M is defined as zero if $M \cong S^2$ and equal to g if M is a connected sum of g tori or g projective planes. Thus the homeomorphism type of M is determined by its orientability and genus.

The *Euler characteristic* of a finite triangulation of a manifold M is the alternating sum of the numbers of simplices of each dimension. It does not depend

on the choice of a triangulation, and it is equal to the alternating sum of the Betti numbers of M, as shown in [66]. The Euler characteristic of a closed connected surface M of genus g is equal to $2-2g$ if M is orientable and $2-g$ if M is not orientable.

The underlying graph of a finite triangulation of a closed surface M determines the Euler characteristic $v-e+t$. This is due to the fact that, in any triangulation of M, each edge bounds precisely two triangles, so $3t=2e$. Therefore, the underlying graph of a triangulation of a closed surface M determines the homeomorphism type of M up to orientability. As shown in [67], the torus and the Klein bottle admit triangulations with the same underlying graph. The existence of different manifolds sharing the same underlying graphs on selected triangulations motivates the necessity of topological neural networks capturing topological properties beyond the ones captured by regular graph neural networks, and the development of our MANTRA dataset in Chapter 6 to test the capacity of these topological neural networks.

For manifolds of dimension 3, classification up to homeomorphism is theoretically possible due to the Geometrization Theorem [68]. However, the complexity of performing homeomorphism classification is harder than in dimension 2, involving more sophisticated tools than the ones needed in the two-dimensional case. For example, in contrast with dimension 2, the Euler characteristic does not carry any information about the homeomorphism type in dimension 3, since if M is any odd-dimensional closed manifold then $\chi(M)=0$ by Poincaré duality [66, § 3.37]. Due to the difficulty of performing full three-dimensional manifold classification, in Chapter 6 we only select a small number of manifold families of dimension 3, whereas for dimension 2 we perform a full classification of manifold triangulations up to 10 vertices.

2.1.4 Simplicial homology

Each finite simplicial complex K with a prescribed order in its set of vertices has an associated family of *homology groups* $H_n(K)$ for $n \in \mathbb{N}$, defined as follows. An *n-chain* is a finite sum $c=\sum_i a_i\sigma_i$ of n-dimensional simplices σ_i of K with

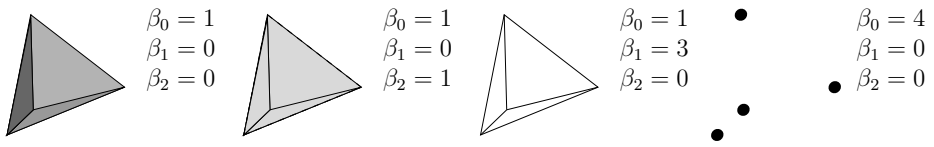


Figure 2.1: From left to right, four simplicial complexes K_1, K_2, K_3, and K_4 with their respective Betti numbers β_0, β_1, and β_2. The n-th Betti number indicates the number of n-dimensional holes in a simplicial complex. Here K_1 is a solid tetrahedron with $\beta_0 = 1$, $\beta_1 = 0$, and $\beta_2 = 0$, since K_1 has only one connected component, no unfilled cycles, and no empty cavity enclosed by 2-faces; K_2 is a hollow tetrahedron with $\beta_0 = 1$, $\beta_1 = 0$, and $\beta_2 = 1$ (the difference with K_1 is that the triangles of K_2 enclose a cavity); K_3 is the underlying graph, with $\beta_0 = 1$, $\beta_1 = 3$, and $\beta_2 = 0$, since there is no cavity and there are three linearly independent cycles; K_4 consists of four vertices and has $\beta_0 = 4$, $\beta_1 = 0$, and $\beta_2 = 0$, since there are four connected components and no cycles nor cavities.

coefficients a_i in the ring of integers $\mathbb{Z}$. More generally, the coefficients a_i can be elements of any associative ring R with 1, in which case the collection of n-chains forms a left R-module.

The *boundary* of an ordered n-simplex $\sigma = (v_0, \ldots, v_n)$ is the alternating sum

$$\partial\sigma = \sum_{i=0}^{n} (-1)^i (v_0, \ldots, \hat{v}_i, \ldots, v_n), \tag{2.2}$$

where $\hat{v}_i$ means that v_i is omitted. Hence (2.2) is the sum of all the faces τ of σ with coefficients equal to the signed incidences of σ and each τ. The boundary operator is linearly extended to n-chains, that is, if $c = \sum_i a_i \sigma_i$ then $\partial c = \sum_i a_i \partial\sigma_i$. The pair $C_\bullet(K) = (\{C_n(K)\}_{n=0}^{\dim K}, \partial)$ consisting of the chain groups and the boundary map ∂ is called the *chain complex* (with integer coefficients) of K. If coefficients are chosen in a ring R, then $C_\bullet(K)$ is a chain complex of R-modules. In particular, if R is a field $\mathbb{F}$, then $C_\bullet(K)$ is a chain complex of $\mathbb{F}$-vector spaces.

An n-chain c is an *n-cycle* if $\partial c = 0$. The *n-th homology group* of K, denoted $H_n(K)$, is defined as a quotient of the abelian group of n-cycles modulo the subgroup of n-boundaries, that is, n-chains of the form ∂c for some $(n+1)$-chain c. These are indeed n-cycles, since $\partial\partial c = 0$ for all c.

The homology groups with integer coefficients $H_n(K)$ are finitely generated abelian groups and therefore can be decomposed as a direct sum

$$H_n(K) = \mathbb{Z}^{\beta_n(K)} \oplus \mathbb{Z}_{q_1} \oplus \cdots \oplus \mathbb{Z}_{q_t}, \tag{2.3}$$

where $\beta_n(K)$ is the n-th *Betti number* of K and $q_1, \ldots, q_t$ are prime powers. The sum $\mathbb{Z}_{q_1} \oplus \cdots \oplus \mathbb{Z}_{q_t}$ is the *torsion subgroup* of $H_n(K)$.

If coefficients in a field $\mathbb{F}$ are used to define n-chains, then $H_n(K)$ becomes an $\mathbb{F}$-vector space. For convenience, we will work with such vector spaces in most of what follows. If no coefficient field is explicitly selected, we will assume that the one used is the field $\mathbb{F}_2$ of two elements.

The dimension of $H_n(K)$ as an $\mathbb{F}$-vector space is denoted by $\beta_n(K;\mathbb{F})$. When the simplicial complex K and the field $\mathbb{F}$ are clear from the context, we simply write β_n. Although the Betti numbers β_n depend on the coefficient field $\mathbb{F}$, their alternating sum does not, and it is equal to the Euler characteristic $\chi(K)$.

The zeroth Betti number β_0 counts the number of connected components of K, and the first Betti number β_1 is equal to the number of linearly independent cycles in the 1-skeleton of K. For $n \geq 2$, generators of $H_n(K)$ are interpreted as n-dimensional "cavities" in K if the coefficient ring is $\mathbb{Z}$ or $\mathbb{Q}$. For a more detailed discussion of simplicial homology, see [53].

2.1.5 Persistence modules

An $\mathbb{R}$-indexed *persistence module* over a field $\mathbb{F}$ is a family $\mathbb{V} = (V_t)_{t\in\mathbb{R}}$ of $\mathbb{F}$-vector spaces equipped with $\mathbb{F}$-linear maps $f_{s,t}\colon V_s \to V_t$ for $s \leq t$, such that $f_{s,t} \circ f_{r,s} = f_{r,t}$ if $r \leq s \leq t$ and $f_{t,t} = \mathrm{id}$ for all t.

Given a filtration of simplicial complexes $(K_t)_{t\in\mathbb{R}}$ ordered by inclusion, the family $(H_*(K_t))_{t\in\mathbb{R}}$ is a persistence module, where $H_*(K_t)$ denotes the direct sum of $H_n(K_t)$ for $n \geq 0$, and coefficients in $\mathbb{F}$ are meant. The $\mathbb{F}$-linear maps $f_{s,t}\colon H_*(K_s) \to H_*(K_t)$ are induced by the inclusions $K_s \subseteq K_t$ if $s \leq t$.

Persistence modules are discussed in detail in [69]. We denote the persistence module given by the n-th homology H_n of a filtration of simplicial complexes $(K_t)_{t\in\mathbb{R}}$ and coefficient field $\mathbb{F}$ by $\mathbb{V}_n^{\mathbb{F}}(K)$. We drop the superscript denoting the field when it is clear from the context.

A persistence module $\mathbb{V}$ is of *finite type* if V_t is finite-dimensional for all t and, moreover, there is a finite set $\{t_0, \ldots, t_k\} \subset \mathbb{R}$ such that $V_t = 0$ if $t < t_0$

and $f_{s,t}$ is an isomorphism whenever the half-open interval $(s,t]$ is contained in the complement of $\{t_0,\ldots,t_k\}$ in $\mathbb{R}$.

A vector $v \in V_b$ is said to be *born* at a parameter value b if it is not in the image of $f_{s,b}$ for any $s < b$, and a vector $v \in V_s$ *dies* at a parameter value d if $f_{s,d}(v) = 0$ while $f_{s,t}(v) \neq 0$ for $s \leq t < d$. As explained in [69, Theorem 1.4], the lifetime intervals $[b,d) \subset \mathbb{R}$ of a full collection of (arbitrarily chosen) basis elements of a persistence module $\mathbb{V}$ of finite type represent $\mathbb{V}$ up to isomorphism.

The intervals $[b,d)$ form a *multiset*, since they can be repeated, so every interval is given with a multiplicity. This multiset is called a *barcode* of $\mathbb{V}$. We denote by $\bar{\mathbb{R}}$ the extended real line with a point at infinity, and assume that $d \in \bar{\mathbb{R}}$ since death values are infinite in the case of vectors $v \in V_s$ for which $f_{s,t}(v) \neq 0$ for all t.

2.1.6 Persistence diagrams

Barcodes are more efficiently represented by means of *persistence diagrams*, which are multisets of points in a coordinate plane with a point (b,d) with $b < d$ and $d \in \bar{\mathbb{R}}$ for each interval $[b,d)$ in the barcode of a persistence module $\mathbb{V}$. We denote by $D(\mathbb{V}) = \{(b_i,d_i)\}_{i\in I}$ the persistence diagram of a persistence module $\mathbb{V}$, where I is the multiset of intervals $[b_i,d_i)$ in its associated barcode; see Figure 2.2.

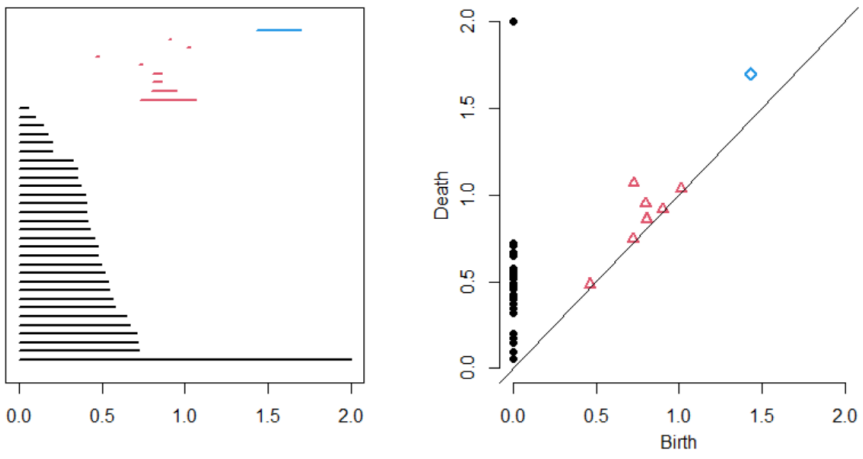


Figure 2.2: A barcode and its associated persistence diagram: each interval $[b,d)$ in the barcode is represented as a point (b,d) in a coordinate system. The barcode corresponds to a Vietoris–Rips filtration (Section 2.1.7) of a point cloud with 30 points sampled from the surface of a 3D sphere of radius 1. Black: H_0; red: H_1; blue: H_2.

In the case of a filtration $(K_t)_{t\in\mathbb{R}}$ of simplicial complexes, the persistence diagram of $(H_*(K_t))_{t\in\mathbb{R}}$ describes the evolution of homology generators along the filtration. Thus, a representative n-cycle ζ can be associated with each point (b, d) in homological degree n, so that b is the birth parameter of ζ and d is the value at which ζ becomes a boundary.

We briefly mention an extension of the usual persistence modules computed with homology that also produce persistence diagrams, known as *zigzag persistence*, introduced in [70]. Zigzag persistence modules are analogous to persistence modules, but the inclusions are not necessarily given by the order of the real numbers indexing the vector spaces. Zigzag persistence modules share a good amount of properties with ordinary persistence modules, such as their unique representation by persistence diagrams or their stability [71, 72], under mild assumptions.

2.1.7 Filtrations

There are several ways to construct filtrations of simplicial complexes given a point cloud or a weighted graph. In this dissertation, a *point cloud* is a finite set P equipped with a symmetric function $d\colon P \times P \to \bar{\mathbb{R}}$ such that $d(x, x) \leq d(x, y)$ for all $x, y \in P$, called a *dissimilarity*.

The *Vietoris–Rips filtration* associated with a point cloud (P, d) is defined as

$$\mathrm{VR}_t(P, d) = \{\sigma \subseteq P \,:\, \sigma \neq \emptyset,\, \mathrm{diam}(\sigma) \leq t\}\,,$$

where $\mathrm{diam}(\sigma)$ is the supremum of $d(x, y)$ for $x, y \in \sigma$.

For practical purposes, there are situations in which we may want to limit the dimension of Vietoris–Rips simplicial complexes to a maximum value $k_{\max}$. In this case, we only take simplices up to dimension $k_{\max}$. We denote dimension-limited Vietoris–Rips filtrations by

$$\mathrm{VR}_t^{k_{\max}}(P, d) = \{\sigma \in \mathrm{VR}_t(P, d) \,:\, \dim(\sigma) \leq k_{\max}\}\,.$$

If the point cloud P is a subset of a metric space (X, d), then another popular family of simplicial complexes is the *Čech filtration*

$$\check{C}_t(P, X, d) = \left\{\sigma \subseteq P \,:\, \sigma \neq \emptyset,\, \textstyle\bigcap_{x\in\sigma} \bar{B}(x, t/2) \neq \emptyset\right\},$$

for $t \geq 0$, where $\bar{B}(x, \varepsilon) = \{y \in X : d(x, y) \leq \varepsilon\}$ denotes the closed ball centered at x of radius ε. We assume that $\check{C}_t(P, X, d) = \emptyset$ if $t < 0$. The indexing parameter has been chosen so that $\check{C}_t(P, X, d) \subseteq \mathrm{VR}_t(P, d)$ for all t. In fact, $\check{C}_t(P, X, d)$ and $\mathrm{VR}_t(P, d)$ have the same edges for all t. As an example, Figure 1.2 shows simplicial complexes for different values of t of a Čech filtration for a point cloud with eight points and the Euclidean distance.

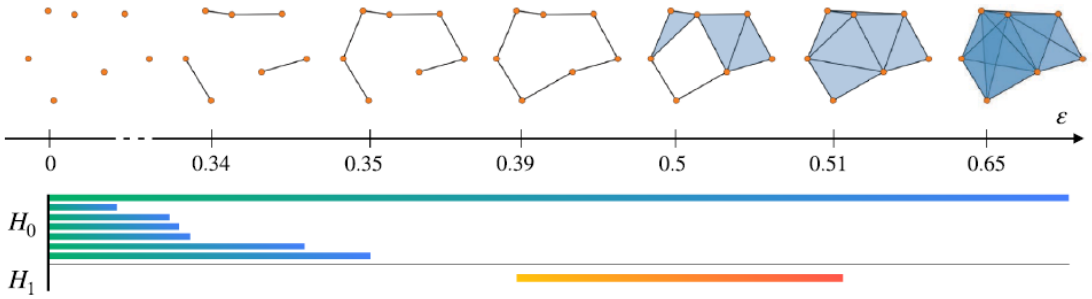


Figure 2.3: Barcode from a Vietoris–Rips filtration of a point cloud. Source: [73].

2.1.8 Weighted graphs

Similar ideas can be applied to weighted graphs. Let (G, w_V, w_E) be a weighted graph, where $w_V \colon V(G) \to \mathbb{R}$ and $w_E \colon E(G) \to \mathbb{R}$ are weight functions for the vertices and the edges of G, respectively. By requiring that $w_V(v) \leq w_E(e)$ for all $v \in V(G)$ and all $e \in E(G)$ incident to v, the weighted graph G can be treated as a point cloud. Points correspond to the vertices of G and distances between points are given by the weight functions w_V and w_E, where the lack of an edge between two vertices is encoded as an infinite distance. Thus the distance from a point to itself need not be zero. Formally,

$$d(v, w) = \begin{cases} w_V(v) & \text{if } v = w, \\ w_E(\{v, w\}) & \text{if } \{v, w\} \in E(G), \\ \infty & \text{otherwise.} \end{cases} \tag{2.4}$$

This allows one to compute a Vietoris–Rips filtration in the same way as for point clouds. We denote the Vietoris–Rips filtration of a weighted graph (G, w_V, w_E) by $\mathrm{VR}(G, w_V, w_E)$, and we denote by $\mathrm{VR}^{k_{\max}}(G, w_V, w_E)$ the dimension-limited version.

Sometimes, weighted graphs are provided only with weights on their edges, that is, a function w_V is not provided. In such cases, there is no canonical way to define a distance between a vertex and itself. One solution is to define w_V for a vertex $v \in V(G)$ as the minimum weight of all its incident edges, and another possible solution is to define w_V as the global minimum weight among all edge weights.

Vietoris–Rips and Čech filtrations add simplices with lower diameter first, that is, the lower the values of t, the lower the diameter of the simplices inside VR_t and $\check{C}_t$. However, it is usual in the graph realm to add edges with higher weights first. Thus, by giving a descending order $e_1, \ldots, e_n$ of the edges by their weights, where $n = |E(G)|$, the edge e_i is added at $t = i - 1$ to the filtration for all $i \in [n]$. In this case, the vertices can be added either at $t = 0$, or at the moment the first incident edge enters the filtration, or at $t = w_V(v)$ if the vertices are weighted by a function w_V that satisfies $w_V(v) \geq w_E(e)$ for all vertices v and all edges incident to v. We can replicate the three approaches also using Vietoris–Rips filtrations. For the first two, we do this by defining the following dissimilarity functions:

$$d_{\downarrow}^{0}(v,w) = \begin{cases} 0 & \text{if } v = w, \\ i-1 & \text{if } \{v,w\} = e_i, \\ \infty & \text{otherwise,} \end{cases}$$

$$d_{\downarrow}(v,w) = \begin{cases} \min\{i : v \in e_i\} & \text{if } v = w, \\ i-1 & \text{if } \{v,w\} = e_i, \\ \infty & \text{otherwise.} \end{cases}$$

For the third one, by defining $\mathfrak{S} = E(G) \cup \{\{v\} : v \in V(G)\}$ and $w \colon \mathfrak{S} \to \mathbb{R}$ as

$$w(\{v,w\}) = \begin{cases} w_V(v) & \text{if } v = w, \\ w_E(\{v,w\}) & \text{otherwise,} \end{cases}$$

and giving a descending order $s_1, \ldots, s_{|\mathfrak{S}|}$ of $\mathfrak{S}$ induced by w, we use the dissimilarity function

$$d_{\downarrow}^{V}(v,w) = \begin{cases} i-1 & \text{if } \{v,w\} = s_i, \\ \infty & \text{if } \{v,w\} \notin \mathfrak{S}. \end{cases}$$

For points (b,d) in persistence diagrams associated with Vietoris–Rips filtrations, b and d are values of the corresponding dissimilarity function for some pair of

points in the point cloud. This means that points of Vietoris–Rips persistence diagrams coming from functions $d_{\downarrow}^{0}$, $d_{\downarrow}$, and $d_{\downarrow}^{V}$ are tuples of indices of edges in the graph. Therefore, an alternative persistence diagram $D^{w}(\mathbb{V}_k(\mathrm{VR}(\cdot)))$ containing the weights of the edges can be obtained simply by taking the weight values of the edges associated with the indices in the persistence diagram. That is,

$$D^{w}(\mathbb{V}_k(\mathrm{VR}(V(G),d))) = \{(w_E(e_{i+1}), w_E(e_{j+1})) : (i,j) \in D(\mathbb{V}_k(\mathrm{VR}(V(G),d)))\}$$

for $d \in \{d_{\downarrow}^{0}, d_{\downarrow}\}$, and

$$D^{w}\Big(\mathbb{V}_k\Big(\mathrm{VR}\Big(V(G), d_{\downarrow}^{V}\Big)\Big)\Big) = \{(w(s_{i+1}), w(s_{j+1})) : (i,j) \in D(\mathbb{V}_k(\mathrm{VR}(V(G),d)))\}\,,$$

where we define $w_E(e_\infty) = w(s_\infty) = \infty$. We refer to this persistence diagram as the *weighted persistence diagram* of the persistence modules of the Vietoris–Rips family of filtrations for graphs induced by $d_{\downarrow}^{0}$, $d_{\downarrow}$ or $d_{\downarrow}^{V}$. This also holds for dimension-limited Vietoris–Rips filtrations.

2.1.9 Sublevel set filtrations

Vietoris–Rips filtrations are a particular type of filtrations included in a broader set of filtrations called *sublevel set filtrations*.

Sublevel set filtrations are simplicial complex filtrations $(K_t)_{t\in\mathbb{R}}$ induced by a pair (K, f) where K is any simplicial complex and $f\colon K \to \mathbb{R}$ is a map such that $f(\sigma) \leq f(\tau)$ if $\sigma \subseteq \tau$ for simplices $\sigma, \tau \in K$. The complex K_t for any t in the filtration is given by $K_t = f^{-1}((-\infty, t])$. Thus, given a point cloud P with dissimilarity d, the Vietoris–Rips filtration can be seen as the sublevel set filtration $f(\sigma) = \max_{p_1,p_2\in\sigma} d(p_1,p_2)$ on the simplicial complex $K = \mathcal{P}(P) \setminus \{\emptyset\}$ given by the power set of P minus the empty set.

In general, there are many useful filtrations that can be employed in persistent homology for finite sets of points or graphs. In this section, we have presented only some of them. For a broader perspective on point cloud and graph filtrations, we refer the reader to the works [46, 53, 74].

2.1.10 Wasserstein distances

For many purposes, it is useful to have a notion of distance between persistence diagrams. The most frequently used distances between persistence diagrams are the bottleneck distance and Wasserstein distances.

The *bottleneck distance* between two persistence diagrams D_1 and D_2 is

$$W_\infty(D_1, D_2) = \inf_{\eta: D_1^\Delta \to D_2^\Delta} \sup_{x \in D_1^\Delta} \|x - \eta(x)\|_\infty \tag{2.5}$$

where D_1^Δ and D_2^Δ denote the persistence diagrams D_1 and D_2 supplemented with their diagonals, that is, $D_1^\Delta = D_1 \cup \Delta$ and $D_2^\Delta = D_2 \cup \Delta$, where $\Delta = \big\{(x, x) : x \in \bar{\mathbb{R}}\big\}$, and $\eta\colon D_1^\Delta \to D_2^\Delta$ are bijections of multisets, where points in Δ have countably infinite multiplicity.

The q-th *Wasserstein distance* W_q is defined as

$$W_q(D_1, D_2) = \inf_{\eta: D_1^\Delta \to D_2^\Delta} \Big(\textstyle\sum_{x \in D_1^\Delta} \|x - \eta(x)\|_\infty^q \Big)^{1/q}$$

although a p-norm $\|\cdot\|_p$ may be used instead of the supremum norm $\|\cdot\|_\infty$, in which case we denote the corresponding Wasserstein distance by $W_q^p(D_1, D_2)$.

2.1.11 Persistence summaries

Persistence diagrams are often not used directly for data analysis due to their lack of a suitable structure for statistical inference. Instead, persistence summaries are used, which are real-valued functions or vector-valued functions derived from persistence diagrams. The following are examples of numerical persistence summaries. The *total persistence* of a persistence diagram $D = \{(b_i, d_i)\}_{i \in I} \cup \{(b_j, \infty)\}_{j \in J}$ is the sum of the lifetimes of all the non-infinity points:

$$\mathrm{TP}(D) = \sum_{i \in I} (d_i - b_i),$$

and, for $p \geq 1$, the *p-norm* is a generalization of total persistence given by

$$\|D\|_p = \Big(\sum_{i \in I} (d_i - b_i)^p \Big)^{1/p}.$$

The most frequently used vector-valued summaries are Betti curves, persistence landscapes, and persistence images, which are described next.

Given a persistence diagram $D = \{(b_i, d_i)\}_{i\in I} \cup \{(b_j, \infty)\}_{j\in J}$, the *Betti curve* of D is the graph of the function $\beta\colon \mathbb{R} \to \mathbb{N}$ defined as

$$\beta(t) = \#\{i \in I : b_i \leq t \leq d_i\}.$$

If the diagram D comes from persistent n-th homology H_n of a filtered simplicial complex $(K_t)_{t\in\mathbb{R}}$ with coefficients in a field $\mathbb{F}$, then $\beta(t)$ counts how many homology generators remain alive at parameter value t. In other words, $\beta(t) = \beta_n(K_t)$.

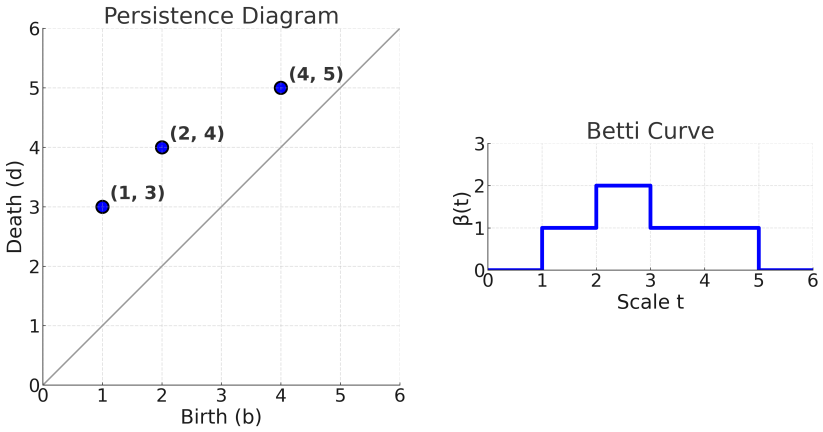


Figure 2.4: Betti curve drawn from a persistence diagram.

A *landscape* [75] of a persistence diagram D is a sequence of piecewise linear functions $\lambda_k\colon \mathbb{R} \to \mathbb{R}$ for $k \geq 1$, defined as

$$\lambda_k(t) = \max\nolimits_{i\in I}^{k} \left\{ f_{(b_i,d_i)}(t) \right\},$$

where $\max^k$ returns the k-th maximum value of a multiset, and

$$f_{(b,d)}(t) = \max\left(0, \min\left(b + t, d - t\right)\right)$$

is a *tent* function with apex $(\frac{1}{2}(b + d), \frac{1}{2}(d - b))$, as depicted in Figure 2.5.

Persistence images were defined in [76]. A surface is generated by centering 2-dimensional Gaussian distributions at each non-infinity point of a given persistence diagram $D = \{(b_i, d_i)\}_{i\in I} \cup \{(b_j, \infty)\}_{j\in J}$, as follows:

$$\rho_D(x, y) = \sum_{i\in I} f(b_i, d_i - b_i)\, \phi_i(x, y)$$

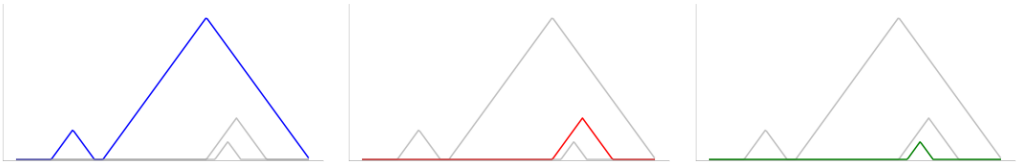


Figure 2.5: Three consecutive levels λ_1, λ_2 and λ_3 of a persistence landscape.

for $(x, y) \in \mathbb{R}^2$, where f is a continuous nonnegative weighting function vanishing along the horizontal axis, and ϕ_i is a normal distribution centered at $(b_i, d_i - b_i)$, for each $i \in I$, with a fixed variance σ^2. Then, a *persistence image* is obtained by integrating ρ_D over each pixel. Apart from the previous persistence summaries, in this dissertation we also use averages and standard deviations of lives and midlives, persistence entropies, persistence pooling vectors, and complex polynomials, which are defined in the rest of this section.

The *life* or *lifetime* of a point (b, d) in a persistence diagram is defined as $d - b$, while the *midlife* is $(b + d)/2$.

Persistence entropy was introduced in [77] as an adaptation of the concept of entropy used in information theory, which, according to [78], provides a measure of the uncertainty of some random variable. The *entropy* of a persistence diagram P is defined as

$$\epsilon(P) = -\sum_{(b,d)\in P} ((d-b)/L)\, \log_2((d-b)/L), \tag{2.6}$$

where $L = \sum_{(b,d)\in P} (d - b)$. If one defines a discrete random variable that picks points (b, d) from P weighted according to their life, then the persistence entropy corresponds to the entropy of this random variable. This choice of weights is based on the assumption that points near the diagonal carry less information. More details on persistence entropy can be found in [79].

Persistence pooling vectors were introduced in [80] in order to improve a max-pooling procedure using TDA. This approach consists of analyzing only the most important points in a given persistence diagram, where importance is weighted according to the difference $d - b$. We define the n-th persistence pooling vector

as the vector in descending order of the n maximum life values. If the persistence diagram has less than n points, then the void vector components are set to 0.

Complex polynomials were introduced in [81] as persistence summaries mapping persistence diagrams into polynomials with coefficients in the field $\mathbb{C}$ of complex numbers whose roots are the images of persistence diagram points under a well-chosen mapping from $\mathbb{R}^2$ to $\mathbb{C}$. In this thesis, we use the transformation T defined in [81].

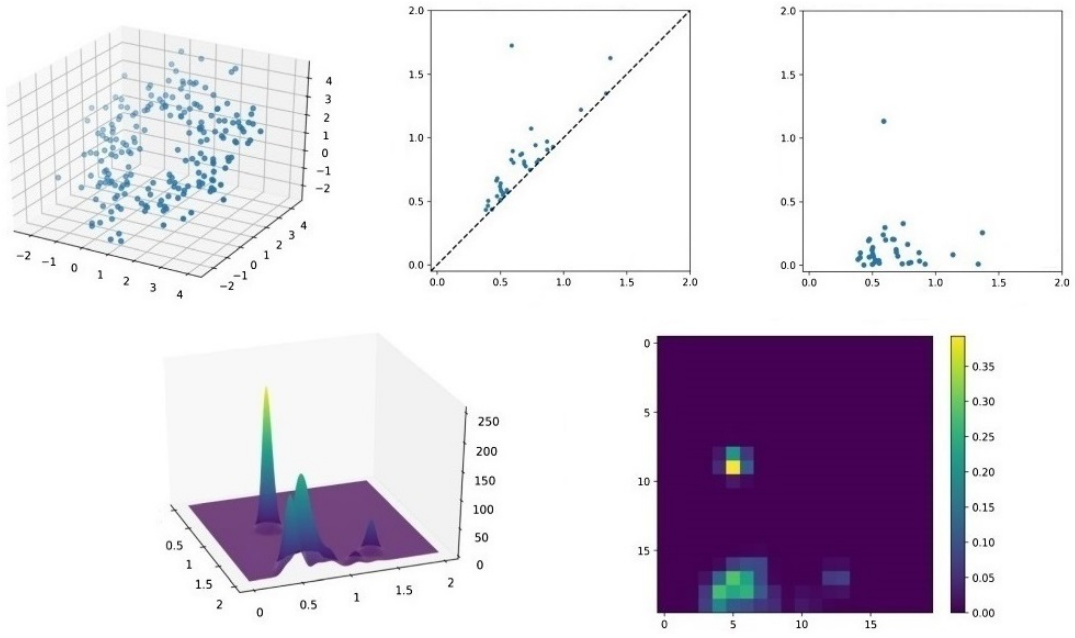


Figure 2.6: Persistence image from a point cloud. The transformation $T(x, y) = (x, y-x)$ is applied to the persistence diagram, sending the diagonal to the horizontal axis, before picking a Gaussian distribution with fixed variance centered at each point. Source: [82].

2.2 Deep Learning

In this section, we introduce the elementary concepts of deep learning, focusing on fully connected feedforward neural networks (FCFNNs), graph neural networks, and high-order neural networks, used throughout the dissertation. FCFNNs are the main focus in Chapter 4, while graph neural networks are used in Chapters 5, 6 and 7, and high-order neural networks are used in Chapters 6 and 7. The books [83, 84] together with the survey [85] provide a deeper introduction of most concepts developed here.

2.2.1 Computational tasks

Prediction tasks represent a core category of computational problems concerned with the inference of information from available data in machine learning. In prediction tasks, data points are modeled as realizations of random variables defined on the Cartesian product of two sets $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ denotes the input domain of the task and $\mathcal{Y}$ denotes the set of labels associated to the input space $\mathcal{X}$.

In prediction tasks, the goal is to identify a map $f\colon \mathcal{X} \to \mathcal{Y}$ that minimizes a *risk* value $\mathcal{R}(f) = \mathbb{E}_{(x,y)\sim\mathbb{P}_{(X,Y)}}\left[\mathcal{L}\left(f, x, y\right)\right]$, where $\mathbb{P}_{(X,Y)}$ is the joint distribution associated to the data on $\mathcal{X} \times \mathcal{Y}$. This quantity represents the expected penalization determined by a *loss function*

$$\mathcal{L}\colon \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R},$$

where $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ denotes the set of all maps from $\mathcal{X}$ to $\mathcal{Y}$. The term $\mathcal{L}(f, x, y)$ constitutes the specific penalization value assigned to the function f for a given data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$. A minimum of this expectation might not exist, in which case one looks for a function f arbitrarily close to attaining the infimum of $\mathcal{R}(f)$.

Due to the hardness of finding such an f (if it exists) in the set $\mathcal{M}\left(\mathcal{X}, \mathcal{Y}\right)$, the problem is often reduced to finding a function f whose risk is arbitrarily near to the infimum of risks in a smaller set of functions called a *hypothesis set*. In modern practice, state-of-the-art methods typically employ hypothesis sets parameterized by deep neural networks.

A *neural network* is a function in a family of parameterized functions $\mathcal{N}_\Theta = \{\mathcal{N}_\theta\colon \mathcal{X} \to \mathcal{Y} : \theta \in \Theta\}$, where Θ is the parameter space. In the context of neural networks, the hypothesis set is conventionally defined as the set $\mathcal{N}_\Theta$. This family is expected to possess desirable properties such as the capacity to universally approximate a large class of functions [84, Chapter 3] or to overcome the curse of dimensionality [14, Chapter 2.2], making the risk minimization feasible.

One of the main difficulties in finding an optimal function f according to the risk minimization criterion is that the data distribution $\mathbb{P}_{(X,Y)}$ is often unknown, and we only have access to a finite sample $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ from $\mathcal{X} \times \mathcal{Y}$, that we

assume independent and identically distributed from $\mathbb{P}_{(X,Y)}$. Therefore, instead of trying to minimize directly the risk $\mathcal{R}(f)$, one tries to minimize the *empirical risk* for a subset $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}$, called *training dataset*, given by

$$\widehat{\mathcal{R}}_{\mathcal{D}_{\text{train}}}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(f, x_i, y_i\right), \tag{2.7}$$

with the hope that minimizing the empirical risk also reduces the real risk.

In some situations, it is advantageous to minimize the empirical risk using a modified formulation. This may involve replacing the original loss $\mathcal{L}$ with a surrogate loss $\mathcal{L}^{\text{surr}}$, or replacing the function f by a surrogate function f^{surr} depending on f. These substitutions, which can be applied independently or jointly, are often necessary when employing gradient-based algorithms, particularly if the original formulation is non-differentiable or yields gradients that are zero almost everywhere.

Training algorithms are the processes that associate a function from a hypothesis set $\mathcal{F}$ with a dataset $\mathcal{D}$ sampled from the data distribution $\mathbb{P}_{(X,Y)}$ attempting to minimize the risk $\mathcal{R}(f)$ as described above. Formally, a *training algorithm* is a map

$$\mathcal{A}\colon \bigcup_{m\in\mathbb{N}} (\mathcal{X} \times \mathcal{Y})^m \longrightarrow \mathcal{F} \tag{2.8}$$

which assigns a specific map within the hypothesis set $\mathcal{F}$ to a finite dataset $\mathcal{D}$. In deep learning for prediction tasks, the output $\mathcal{A}(\mathcal{D})$ is typically computed by minimizing the empirical risk over $\mathcal{D}_{\text{train}}$ using iterative gradient-based optimization in an hypothesis set composed of neural networks.

Classification and regression constitute two canonical examples of *prediction* tasks. In classical machine learning, these tasks operate on a Euclidean input space $\mathcal{X} = \mathbb{R}^d$ for some $d \geq 1$. However, modern applications increasingly include structured non-Euclidean input domains, such as graphs or cell complexes, as discussed in Chapters 5 to 7. We say that a prediction task is a *regression task* if $\mathcal{Y} = \mathbb{R}$, and we say it is a *classification task* if $\mathcal{Y}$ is a finite set. In the latter case we assume, without loss of generality, that $\mathcal{Y} = \{1, \ldots, k\}$ with $k \geq 2$, and that the marginal distribution of $\mathcal{Y}$ satisfies $\operatorname{supp}(\mathbb{P}_Y) = \mathcal{Y}$. If $k = 2$, then the problem is termed *binary* classification.

Generative tasks are not as straightforward to define as prediction tasks, and formal definitions depend on the specific problem formulation or the solving method employed. Broadly, the objective is to produce synthetic examples within an ambient space $\mathcal{X}$ that are indistinguishable from samples drawn from a target data distribution $\mathbb{P}_X$ supported on $\mathcal{X}$. Approaches typically fall into two categories: models that explicitly approximate the probability distribution $\mathbb{P}_X$, enabling direct sampling, and models that learn a mechanism to generate new examples without explicitly characterizing an approximated data distribution. As these tasks are not the central focus of this work, we omit a more granular formalization and refer the reader to [86, Section 1.2.1] for a detailed explanation of generative models.

Reinforcement learning tasks are also considered in one of the articles analyzed in Chapter 3. In reinforcement learning, an *agent* (e.g., a robot) interacts with an environment over discrete time steps aiming to optimize a given notion of *rewards* generated by its actions. In deep reinforcement learning, neural networks are used to control the behaviour of the agent in various ways. These methods include approximating the value for the agent of being in a specific state or taking a particular action within a state [87], as well as extracting and transforming relevant information from the agent's environment [88], among other techniques. Similar to the previous tasks, agents typically undergo a training phase, where they learn to behave effectively in the environment, and an inference phase, where they interact with the environment without learning from it. For a comprehensive introduction to reinforcement learning, we refer the reader to [89]. For a broader introduction to machine learning tasks, we refer the reader to [90, Section 5.1.1].

2.2.2 Fully connected feedforward neural networks

Fully connected feedforward neural networks (FCFNN) are neural networks that have a basic structure in which most of the current models can be expressed. A FCFNN is a parameterized function $\mathcal{N}$ induced by the following two elements, comprising the FCFNN structure.

(i) An *architecture* $a(\mathcal{N}) = (N, \varphi)$, where $N = (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$ and

$$\varphi = \left(\varphi^{(l)}\right)_{l=1}^{L}, \qquad \varphi^{(l)} \colon \mathbb{R}^{N_l} \to \mathbb{R}^{N_l}$$

is a tuple of non-linear differentiable functions. Here differentiability is meant except perhaps on a set of zero measure. The number L counts the *layers* of the network, while N_0, N_L, and N_l for $l \in \{1, \dots, L-1\}$ are the numbers of neurons of the input, output, and l-th hidden layers, respectively, and $\varphi^{(l)}$ is the *activation function* of layer l. Usually, the activation functions $\varphi^{(l)}$ are chosen to be functions which apply a surrogate function $\varphi_e^{(l)} \colon \mathbb{R} \to \mathbb{R}$ component-wise, like the *rectified linear unit* (ReLU) function $\mathrm{ReLU}(x) = \max\{0, x\}$. If not specified otherwise, we assume that activation functions are calculated by applying surrogate functions component-wise.

(ii) A set of parameters

$$\theta(\mathcal{N}) = \left(\theta^{(l)}\right)_{l=1}^{L}, \qquad \theta^{(l)} = \left(W^{(l)}, b^{(l)}\right) \in \mathbb{R}^{N_l \times N_{l-1}} \times \mathbb{R}^{N_l}$$

for $l \in \{1, \dots, L\}$, where entries in the matrix $W^{(l)}$ are called *weights* and those in the vector $b^{(l)}$ are called *biases*.

The FCFNN architecture induces a directed acyclic graph from which the FCFNN function is obtained using the set of parameters. The directed graph defined by the architecture, denoted by $G(\mathcal{N})$, is given by $L+1$ disjoint ordered sets $V_l = \{v_l^1, \dots, v_l^{N_l}\}$, $l = 0, \dots, L$, with the property that every vertex in V_l has an edge pointing to it from all the vertices in V_{l-1}, for $l \in \{1, \dots, L\}$. The vertices of this graph are called *neurons*.

Given initial values for neurons $\{v_0^j\}_{j=1}^{N_0}$, often represented by an input vector $x \in \mathbb{R}^{N_0}$ where we set $v_i^j = x_j$, the values of the rest of the neurons are computed recursively by taking into consideration the values of the neurons pointing to them. Specifically, the function $\phi_{\mathcal{N}} \colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ defined on top of the directed graph,

which represents the FCFNN, is given by the recursive formula

$$\begin{aligned}\phi_{\mathcal{N}}^{(0)}(x) &= x,\\ \phi_{\mathcal{N}}^{(l)}(x) &= \varphi^{(l)}\left(W^{(l)}\phi_{\mathcal{N}}^{(l-1)}(x) + b^{(l)}\right) \quad \text{for } l \in \{1,\dots,L\},\\ \phi_{\mathcal{N}}(x) &= \phi_{\mathcal{N}}^{(L)}(x),\end{aligned} \tag{2.9}$$

where the values $\phi_{\mathcal{N}}^{(l)}(x)_i$ are associated to the vertices v_l^i. Given an input vector x, we call the values $\phi_{\mathcal{N}}^{(l)}(x)_i$ the activations of their corresponding vertices. A visual example of a FCFNN is shown in Figure 2.7.

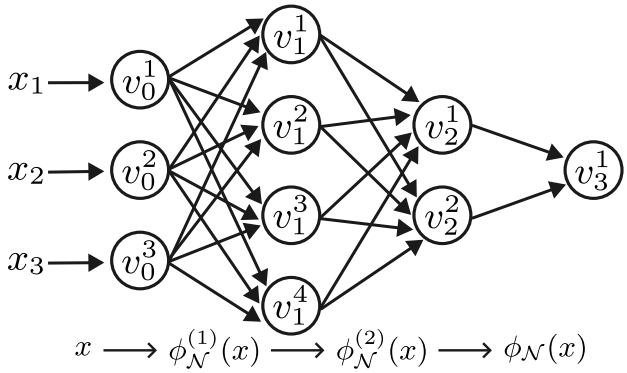


Figure 2.7: A graphical representation of a fully connected feedforward neural network $\mathcal{N}$ with $L = 3$ and $N = (3, 4, 2, 1)$. The input values x_i are associated with the vertices v_0^i of the first (input) layer and then transformed sequentially by a set of maps. In this representation, each edge indicates that the value of the source vertex is used for the computation of the value of the target vertex. Values for vertices are computed sequentially from the first layer to the last. A transformation from layer $l-1$ to layer l is a composite of an affine transformation and the activation function $\varphi^{(l)}$ applied component-wise.

Given an architecture $a = (N, \varphi)$ for a FCFNN and a subset of *admissible* parameters $\Theta \subseteq \prod_{l=1}^{L} \mathbb{R}^{N_l \times N_{l-1}} \times \mathbb{R}^{N_l}$, we denote the hypothesis set of all the neural networks represented by the architecture a with parameters in Θ as

$$\mathcal{F}_{a,\Theta} = \{\mathcal{N} : a(\mathcal{N}) = a,\, \theta(\mathcal{N}) \in \Theta\}\,.$$

If Θ is the space of all possible parameters, we simply write $\mathcal{F}_a$. Also, when it is clear that we speak about neural networks in a specific hypothesis set $\mathcal{F}_{a,\Theta}$, we denote the neural network with architecture a and parameters $\theta \in \Theta$ as $\mathcal{N}_\theta$. Moreover, when the usage of FCFNNs is clear from the context, we denote the hypothesis set as $\mathcal{N}_\Theta$, as stated before.

Fixing a dataset $\mathcal{D}$, an architecture a, and a set of possible parameters Θ, the composition of the empirical risk with the neural network functions in $\mathcal{F}_{a,\Theta}$ can be seen as a function $\widehat{\mathcal{R}}_{\mathcal{D}_{\text{train}}}(\theta)$ of the parameters θ. Usually, training algorithms for neural networks, given an initial set of parameters $\theta^{(0)} \in \Theta$, generate a discrete weight trajectory $\left(\theta^{(i)}\right)_{i=1}^{N}$ by minimizing the empirical risk $\widehat{\mathcal{R}}_{\mathcal{D}_{\text{train}}}(\theta)$ iteratively with respect to the parameters θ until some stopping criteria are satisfied. The output of the algorithm is thus the neural network with architecture a and parameters given by one of the weights $\theta^{(i)}$ generated in the trajectory, usually the last one, $\theta^{(N)}$. In this work, we assume that neural networks are trained in this way.

2.2.3 Convolutional neural networks

Convolutional neural networks (CNNs) are a special family of FCFNNs designed to work with data that have a grid-like structure, such as images. CNNs are one of the most common type of networks used in computer vision. They implement convolutional and pooling transformations, that are special functions between layers that impose restrictions on the set of weights and on the activation functions. A layer l whose output is the result of a convolutional or pooling transformation is called a convolutional or pooling layer, respectively.

A *convolutional layer* is a layer whose activations are the result of performing convolutions between the values associated to the neurons of its input layer rearranged as a grid and a filter tensor that imposes a set of equalities on the coefficients of the weight matrices before applying activation functions. Generally, a convolutional transformation is a transformation between the activations of V_{l-1} and V_l in such a way that V_{l-1} and V_l are ordered in grid structures of size $h^{(l-1)} \times w^{(l-1)} \times c^{(l-1)}$ and $h^{(l)} \times w^{(l)} \times c^{(l)}$, respectively. We say that a layer l arranged in this grid structure has *height* $h^{(l)}$, *width* $w^{(l)}$, and $c^{(l)}$ *channels*, where each channel is defined as the subgrid obtained by fixing one value in $\left[c^{l}\right]$ for the third dimension.

The output of a convolutional layer before applying its activation function φ is computed as follows. First, for each $j \in [c^{(l)}]$, $c^{(l-1)}$ discrete convolutions are performed for the activations of each channel $k \in [c^{(l-1)}]$ of size $h^{(l-1)} \times w^{(l-1)}$ of

the layer $l-1$. For this purpose, $c^{(l-1)}$ convolutional filters $C_{k,j}^{(l)}$ of size $k_h^{(l)} \times k_w^{(l)}$ are used. The values of the convolutional filters are called convolutional *filter weights*, and correspond with the parameters of layer l.

Depending on the application, bias terms can be added to the result of the convolutions. For a fixed j, the $c^{(l-1)}$ convolutions generate $c^{(l-1)}$ grids of size $h^{(l)} \times w^{(l)}$. These grids are summed to obtain the activations of the j-th channel of the layer l, obtaining a final grid of size $h^{(l)} \times w^{(l)} \times c^{(l)}$. Summarizing, a typical convolutional layer calculates the (r, s, j)-component of the l-th grid before applying the activation function as

$$\phi_{\mathcal{N}}^{(l)}(x)_{r,s,j} = b_j^{(l)} + \sum_{k=1}^{c^{(l-1)}} \sum_{u=1}^{k_h^{(l)}} \sum_{v=1}^{k_w^{(l)}} C_{k,j}^{(l)}(u,v) \phi_{\mathcal{N}}^{(l-1)}(x)_{r+u-1,s+v-1,k}$$

where the terms $b_j^{(l)}$ are the bias terms of the layer l for each channel $j \in [c^{(l)}]$.

Pooling transformations are similar operations aiming to reduce the dimensionality of their input layer. One of the most common pooling transformations is the *max pooling* transformation. Given the input layer of the pooling transformation structured as a grid of size $h^{(l-1)} \times w^{(l-1)} \times c^{(l-1)}$, the output of the max pooling transformation is a grid of size $h^{(l)} \times w^{(l)} \times c^{(l)}$ where $c^{(l-1)} = c^{(l)}$ and where each value for a fixed channel $j \in [c^{(l)}]$ is the maximum of a subgrid of size $k_h^{(l)} \times k_w^{(l)}$ for the same channel j of the input grid. The subgrid associated to each (r, s, j)-component of the output grid is chosen using a sliding window of size $k_h^{(l)} \times k_w^{(l)}$ that moves either $s_h^{(l)}$ positions vertically or $s_w^{(l)}$ positions horizontally at each step in such a way the full input grid is covered. A typical max pooling transformation calculates the (r, s, j)-component of the l-th grid as

$$\phi_{\mathcal{N}}^{(l)}(x)_{r,s,j} = \max_{u \in [k_h^{(l)}]} \max_{v \in [k_w^{(l)}]} \phi_{\mathcal{N}}^{(l-1)}(x)_{(r-1)\cdot s_h^{(l)}+u,\,(s-1)\cdot s_w^{(l)}+v,\,j}$$

In general, not all components of input values are used to compute the activations of a specific neuron in a convolutional or pooling layer. The region of the input, or more specifically, the set of input vector components, that affect the activations of a neuron is called its *receptive field*. A thorough introduction to CNNs can be found in [90, Chapter 9].

In general, we call *multi-layer perceptron* (MLP) a FCFNNs trained without imposing any *a priori* structure on the network's weights.

2.2.4 Decision regions

Fully connected feedforward networks define functions $\phi_{\mathcal{N}}\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$, which is well-suited whenever $\mathcal{X}$ and $\mathcal{Y}$ are subsets of Euclidean spaces. However, for classification tasks, one has $\mathcal{Y} = [k]$ for some $k \in \mathbb{N}$. In this case, one selects a projection function $\pi\colon \mathbb{R}^{N_L} \to [k]$ that projects the outputs of the neural network $\mathcal{N}$ into the finite set of points $[k]$ and uses $\pi \circ \phi_{\mathcal{N}}\colon \mathbb{R}^{N_0} \to [k]$ as a model. In this context, the *decision region* of a label $y \in [k]$ is the set of inputs x such that $\pi(\phi_{\mathcal{N}}(x)) = y$.

To train a neural network, it is common to choose surrogate loss functions $\mathcal{L}^{\text{surr}}$ that depend directly on the output of $\phi_{\mathcal{N}}$. In classification tasks, the most common configuration for a problem with $\mathcal{X}$ having $\mathbb{R}^d$ as an ambient space and $\mathcal{Y} = [k]$ is to choose neural networks with $N_0 = d$, $N_L = k$, projection

$$\pi(x_1, \ldots, x_k) = \operatorname{argmax}_{i \in [k]} x_i,$$

and surrogate loss function $\mathcal{L}^{\text{surr}}$ depending only on $\phi_{\mathcal{N}}$ like the *categorical cross entropy loss* CE, that, in its most basic form, looks like

$$\operatorname{CE}(\phi_{\mathcal{N}}, x, y) = -\log\left(\frac{\exp(\phi_{\mathcal{N}}(x)_y)}{\sum_{i=1}^{k} \exp(\phi_{\mathcal{N}}(x)_i)}\right).$$

For neural networks with $N_L = k$ and $\pi(x_1, \ldots, x_k) = \operatorname{argmax}_{i \in [k]} x_i$, there is an ambiguity when the argmax is not unique, i.e., when there are at least two indices $i, j \in [k]$ such that $x_i = x_j$. In this case, the projection function must define a deterministic way to choose one of the equal-valued indices. Given a neural network $\mathcal{N}$ as the previous one, the points $x \in \mathcal{X}$ for which there exists an ambiguity is called *decision boundary* of $\mathcal{N}$. Formally,

$$\mathcal{B}(\mathcal{N}) = \Big\{x \in \operatorname{Dom}(\phi_{\mathcal{N}}) : \exists\, i, j \in [N_L] \text{ such that } \phi_{\mathcal{N}}(x)_i = \phi_{\mathcal{N}}(x)_j = \max_{y \in [k]} \phi_{\mathcal{N}}(x)_y\Big\}. \tag{2.10}$$

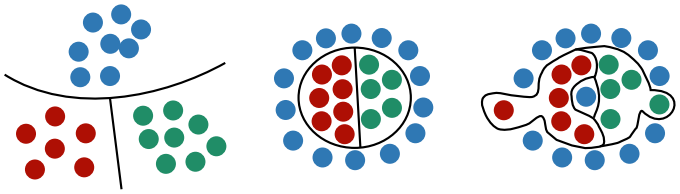


Figure 2.8: Decision regions and boundaries for three different classification problems ordered from left to right by increasing complexity, with three labels each. Black lines represent decision boundaries given by FCFNNs. Decision regions and their boundaries give valuable information on the neural network used in each case. For the two first problems, the decision regions and boundaries are "simple", in the sense that they seem to properly classify the inputs of the domain without visible outliers or "strange" regions. In the second problem, the blue decision region has one hole that does not exist in the first problem. In the third problem, the blue decision region has two connected components, and the red and green regions have protuberances that could have appeared due to outliers in the training data. The extra connected component is detected by the homology of the blue decision region.

Sometimes, decision regions are studied without their decision boundaries; in other words, $(\pi \circ \phi_{\mathcal{N}})^{-1}(y) \setminus \mathcal{B}(\mathcal{N})$ is used. Figure 2.8 shows examples of decision boundaries and regions for classification problems with three labels.

In this context, it is usual to interpret the outputs of the function $\phi_{\mathcal{N}}$ as (unnormalized) probabilities, indicating the likelihood that the input belongs to one of the classes represented by the outputs. Specifically, $\phi_{\mathcal{N}}(x)_i$ denotes the (unnormalized) probability that x belongs to the class $i \in [k]$. Therefore, π is chosen as argmax since it selects the class with the highest probability according to the output of the neural network.

For binary classification problems, i.e., $\mathcal{Y} = [2]$, the usual neural network architectures, projection functions, and decision boundaries are slightly different. On the one hand, the number of neurons in the last layer is set to one, that is, $N_L = 1$. On the other hand, the usual projection function is given by $\pi(x) = 2$ if $x > b$ and $\pi(x) = 1$ if $x \leq b$, with the usual value of b being zero. In this configuration, the decision boundary is defined as

$$\mathcal{B}(\mathcal{N}) = \{x \in \mathrm{Dom}(\phi_{\mathcal{N}}) : \phi_{\mathcal{N}}(x) = b\}. \tag{2.11}$$

2.2.5 Graph and high-order neural networks

While feedforward neural networks are designed to work with Euclidean spaces, *graph neural networks* (GNN) and *high-order neural networks* (HONN) are designed to handle data represented as (annotated) graphs or other high-order domains such as simplicial or cellular complexes.

In the case of GNNs, we use *annotated graphs*. Annotated graphs are generalizations of weighted graphs where, instead of associating maps w_V and w_E with codomain $\mathbb{R}$ to the graph, we associate maps, called *feature maps*, $f_V \colon V(G) \to \mathbb{R}^{d_V}$ and $f_E \colon E(G) \to \mathbb{R}^{d_E}$ where d_V and d_E depend on the data and task, and associate to each vertex or edge a *feature vector*.

In the more general case of HONNs, we use *annotated simplicial complexes* and their generalization, *annotated cellular complexes*. Given a simplicial or cellular complex K, its annotated version is a pair $(K, \{f_i\}_{i=0}^{\dim K})$ where $\dim K$ is the dimension of the simplicial complex or the cellular complex, f_i are the feature maps for simplices or cells of dimension i, and where $f_i \colon K_i \to \mathbb{R}^{d_{K_i}}$ for arbitrary d_{K_i} depending on the data.

For this dissertation, we see GNNs and HONNs as neural networks $\mathcal{N}_\theta \colon \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are either the class of annotated finite graphs, the class of annotated finite simplicial complexes, or the class of annotated cellular complexes, possibly with different feature map codomains. As in the case of feedforward neural networks, GNNs and HONNs may need a projection function π to address some tasks such as graph or complex classification.

Message passing neural networks

Most of the models used in the literature for graphs and higher-order structures are based on the message-passing paradigm. For graph and simplicial or cellular complexes, these models *pass messages* between *neighboring* nodes, simplices, or cells, in the graph or complex, updating their feature vectors based on the features of their neighbors. Let K be an annotated complex or a graph seen as a simplicial

complex with feature maps $\{f_i\}_{i=0}^{\dim K}$. A *message-passing layer* updates the features of a simplex or cell σ using the following steps [85]:

(1) *Selection of neighborhoods*: We first start by defining a map from simplices or cells σ in any simplicial or cellular complex K in the input space $\mathcal{X}$ to sets $\{\mathcal{N}_i(\sigma)\}_i^N$ of *neighbouring* simplices or cells within the same K. By convention, neighborhoods $\mathcal{N}_i$ are defined using the same procedure for all the simplices sharing the same dimension and contain only simplices sharing the same dimension. For example, adjacent or incident simplices are two types of neighbourhoods that can be defined in an arbitrary simplicial/cellular complex depending on the dimension of the considered simplices Thus, for $\mathcal{X}$ being the set of annotated graphs, we could define $\mathcal{N}_0(v)$ as the set of adjacent vertices to v, and $\mathcal{N}_0(e)$ and $\mathcal{N}_1(e)$ as the sets of incident vertices and edges sharing one vertex, respectively.

(2) *Message computation*: For each set of neighboring simplices $\mathcal{N}_i(\sigma)$, we compute messages $\{\mathrm{m}_{\tau\to\sigma}\}_i$ from the features of the simplices in $\mathcal{N}_i(\sigma)$ and the features of σ, that is

$$\mathrm{m}_{\tau\to\sigma} = \mathrm{M}_{\mathcal{N}(x)}(f_{\dim\tau}(\tau), f_{\dim\sigma}(\sigma), \theta_1),$$

where θ_1 are the learnable parameters of the layer, usually shared across the neighbourhood index i and the simplex dimension $\dim\sigma$.

(3) *Intra-aggregation*: The messages are aggregated to obtain a single message for each neighborhood $\mathcal{N}_i(\sigma)$, that is

$$\mathrm{m}_{\mathcal{N}_i(\sigma)} = \mathrm{Agg}_{\mathcal{N}_i(\sigma)}(\{\mathrm{m}_{\tau\to\sigma}\}_{\tau\in\mathcal{N}_i(\sigma)}),$$

where $\mathrm{Agg}_{\mathcal{N}_i(\sigma)}$ is a permutation-invariant aggregation function, for example, a sum, mean, or any other function that aggregates the messages.

(4) *Inter-aggregation*: The aggregated messages for the neighborhoods are then aggregated together to obtain a single message for the simplex σ, that is

$$\mathrm{m}_\sigma = \mathrm{Agg}_\sigma(\{\mathrm{m}_{\mathcal{N}_i(\sigma)}\}_i),$$

where Agg_σ is a permutation invariant aggregation function again.

(5) *Update*: The message m_σ is used to update the features of the simplex σ, i.e.,

$$f_{\dim \sigma}(\sigma) = \mathrm{Update}(f_{\dim \sigma}(\sigma), \mathrm{m}_\sigma, \theta_2),$$

where θ_2 is usually shared across simplex dimensions.

Finally, a *message passing neural network* is defined as the composition of several message passing layers. As with feedforward neural networks, we define a set of admissible parameters Θ where the parameters of the networks belong. For graphs, GCN [91] and GAT [92] are examples of message-passing networks. For simplicial complexes, SAN [93] and SCN [94] use (upper and lower) higher-order Laplacians to define neighborhoods, SCCN [95] uses (co)adjacency and incidence structures, and SCCNN [96] uses all together.

Non-message passing neural networks

Although the message passing paradigm is predominant in the literature, there are other state-of-the-art models that do not follow it, such as graph transformers [32], which are based on the standard transformer architecture [31], state-space topological models [97], or TDA-based networks [98]. In this thesis, we extend the graph transformer to the setting of cellular (and simplicial) complexes and we analyze the expressivity of TDA-based approaches.

Graph transformers. [31, 32] Unlike message passing approaches that rely on local neighborhoods, graph transformers consist of a composition of an input preprocessing layer and a set of transformer layers that utilize self-attention mechanisms to update vertex feature maps. Let G be an annotated graph with node feature maps $f_V \colon V(G) \to \mathbb{R}^{d_V}$. Assume that the vertices $V(G)$ are given any fixed order $V(G) = \{v_1, \ldots, v_n\}$ and express the feature map f_V as a matrix $F_V \in \mathcal{M}(n, d_V, \mathbb{R})$ concatenating the different feature vectors as rows respecting

the order of their associated nodes. In its most basic form, a graph transformer layer updates the feature maps as

$$\begin{aligned} F_V^{\text{int}} &= \text{Norm}(\text{Attn}(F_V) + F_V), \\ F_V^* &= \text{Norm}(F_V^{\text{int}} + \text{FFN}(F_V^{\text{int}})), \end{aligned} \tag{2.12}$$

with

$$Q = F_V\theta_Q, \quad K = F_V\theta_K, \quad V = F_V\theta_V, \text{Attn}(F_V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.13}$$

where $\theta_Q, \theta_K \in \mathcal{M}(d, d_k, \mathbb{R})$, $\theta_V \in \mathcal{M}(d, d, \mathbb{R})$, FFN is a feedforward neural network, Norm is a normalization layer such as LayerNorm [99] or RMSNorm [100], and where F_V^* denotes the updated feature vectors of the updated feature maps. In more advanced versions, $\text{Attn}(F_V)$ is replaced by a multi-head attention $\text{MultiHead}(F_V)$, that concatenates the output of functions $\text{Attn}(F_V)$ with different learnable projections Q, K, V receiving different splits from the same input. $\text{Attn}(F_V)$ is also often extended by considering a learnable bias matrix B such that

$$\text{Attn}(F_V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B\right)V, \tag{2.14}$$

where $B_{i,j}$ only depends on the feature vectors $f_V(v_i)$, $f_V(v_j)$ and on the relationship between v_i and v_j in the graph, for example, taking into account their shortest path distance.

The input preprocessing layer is a crucial step of graph transformers, as it modifies the initial data to take into consideration the structure of the graph and the relative positions of the nodes inside it. The most usual way to design input preprocessing layers consist of, given an annotated graph G, generate a new node feature map $p_V \colon V(G) \to \mathbb{R}^{d_p}$ known as the *positional encoding map* containing the extra structural information of the graph added to each node, and then, combine the original feature map f_V and the positional encoding map p_v into an updated feature map f_V^*, usually by summing both feature maps or by concatenating them.

Two of the most successful positional encoding maps are the Laplacian Positional Encodings (LapPE) [101] and the Random Walk Positional Encoding (RWPE) [102].

LapPE uses the d_p eigenvectors smallest non-zero eigenvalues of the normalized graph Laplacian matrix $L = I - D^{-1/2}AD^{-1/2}$, where D denotes the degree matrix and A is the adjacency matrix, to encode node positions. Concretely, given such d_p eigenvectors $\phi_1, \ldots, \phi_{d_p} \in \mathbb{R}^n$, the feature vector for the vertex v_i consists of the concatenation of the i-th coordinates of the d_p different eigenvectors, that is,

$$\text{LapPE}(v_i) = (\phi_1(i), \phi_2(i), \ldots, \phi_{d_p}(i)), \tag{2.15}$$

where $\phi_j(i)$ denotes the i-th coordinate of the vector ϕ_j. If the number of eigenvectors of the graph is lower than d_p, then the vector starts with the eigenvector values and then it is filled with zeros to get a d_p-dimensional vector.

RWPE, on the other hand, uses d_p-step random walks to encode the position of the vertices inside the graphs. The feature vector for the vertex v_i is given by

$$\text{RWPE}(v_i) = (\text{RW}_{i,i}, \ldots, \text{RW}_{i,i}^{d_p}),$$

where $\text{RW} = AD^{-1}$ is the random walk operator, which transitions uniformly from a vertex to any of its adjacent vertices.

Topological Graph Neural Networks. Persistent homology computes topological feature maps that can be utilized for tackling prediction tasks on annotated graphs, for example, by processing these maps with FCFNNs. However, persistent homology relies on the specification of weight functions w_V and w_E for vertices and edges, respectively. Defining appropriate weights often poses a significant challenge, as optimal choices are typically graph-dependent and domain-specific. *Topological Graph Neural Networks* [98] address this challenge by introducing *topological graph layers* (TOGL), that leverage learnable filtrations to generate contextually meaningful persistence diagrams for feature extraction, which can be combined with other neural network modules for downstream learning tasks. Specifically, given an annotated graph G with initial node feature map f_V, TOGL employs k distinct learnable node weight functions $w_V^i \colon V(G) \to \mathbb{R}$ inducing k edge weight maps $w_E^i(v_1, v_2) = \max(w_V^i(v_1), w_V^i(v_2))$. From these weight functions, TOGL computes l

persistence diagrams, $l \geq k$, that are fed to a (learnable) embedding function from which new feature maps for the graph G are obtained. Composing several TOGL layers enables the network to adaptively discover optimal filtrations during training, thereby circumventing the need for hand-crafted weight functions while enhancing the model's capacity to capture complex structural patterns and enabling a flexible integration of topological features into standard neural pipelines.

Expressivity of graph learning algorithms

Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic*, denoted by $G \simeq G'$, if there is a bijective function $\varphi\colon V \to V'$ that preserves adjacency, i.e., $(u, v) \in E$ if and only if $(\varphi(u), \varphi(v)) \in E'$. The problem of figuring out whether two graphs are isomorphic or not is referred to as the *graph isomorphism problem*. Presently, there is no known algorithm that solves this problem in polynomial time and efficient algorithms exist only for special families of graphs [103, 104]. Thus, all efficient graph isomorphism tests are perforce limited with respect to their expressivity, i.e., there exist classes of non-isomorphic graphs that they cannot distinguish. We use the term *graph expressivity* to denote two different concepts: first, the ability of a method to distinguish non-isomorphic (undirected) graphs, and second, the ability of a method to capture specific graph properties. We formalize these two definitions below.

In the first case, we define expressivity in terms of the capacity to distinguish non-isomorphic graphs. Let $f_1\colon \mathcal{G} \to \mathcal{Y}_1$ and $f_2\colon \mathcal{G} \to \mathcal{Y}_2$ be two functions with common domain $\mathcal{G}$ denoting the set of all finite graphs and arbitrary codomains $\mathcal{Y}_1$, $\mathcal{Y}_2$ such that for two isomorphic graphs $G, G' \in \mathcal{G}$ we have that $f_1(G) = f_1(G')$ and $f_2(G) = f_2(G')$. We say that f_1 is *at least as expressive* as f_2 if for any pair of non-isomorphic graphs G, G' we have that

$$f_2(G) \neq f_2(G') \text{ implies } f_1(G) \neq f_1(G').$$

In the second case, we define expressivity in terms of the ability to detect graph properties. Let $f_1, f_2, t\colon \mathcal{G} \to \mathcal{Y}$ be functions for some fixed codomain $\mathcal{Y}$.

We say that f_1 is *at least as expressive* as f_2 according to property t if for any graph G we have that

$$f_2(G) = t(G) \text{ implies } f_1(G) = t(G).$$

In both cases, if f_1 is at least as expressive as f_2 and f_2 is at least as expressive as f_1, we say that f_1 and f_2 are equally expressive. A more complete characterization of expressivity can be found in Li and Leskovec [105, Definition 5.5].

The Weisfeiler-Leman graph isomorphism test family

Weisfeiler-Leman (1-WL), also known as *colour refinement*, constitutes a simple method for addressing the graph isomorphism problem. It is the backbone of graph expressivity research; readers are referred to Morris et al. [106] for a comprehensive survey of 1-WL and its higher-order variants.

Formally, given a graph $G = (V, E)$, 1-WL proceeds by iteratively calculating a node colouring feature map $C_i^{(1)} \colon V \to \mathbb{N}$, where colors are represented as natural numbers. The output of this function depends on the colours of the neighbours of a given node. For a vertex v at iteration $i > 0$, we have

$$C_i^{(1)}(v) := \texttt{RELABEL}\Big(\Big(C_{i-1}^{(1)}(v), \{\!\{C_i^{(1)}(u) \mid u \in \mathcal{N}(v)\}\!\}\Big)\Big), \tag{2.16}$$

where `RELABEL` refers to an injective function that maps the tuple of colours to a unique colour, and $\mathcal{N}(v)$ refers to the neighbors of v, i.e., its adjacent vertices. The algorithm is initialised by either using existing labels or the degree of vertices.[1] After a finite number of steps, the colour assignments generated using Equation (2.16) stabilise. If two graphs give rise to different colour sequences, the graphs are guaranteed to be non-isomorphic. The 1-WL test is computationally easy and constitutes an upper bound for the expressivity of many graph neural network (GNN) architectures [107, 108]. In other words, if 1-WL cannot distinguish two non-isomorphic graphs, these expressivity-limited GNNs will also not be able to distinguish them.

[1]Note that Equation (2.16) does not recognise an ordering of labels. Initialising 1-WL with a constant value thus leads to the *same* colouring —up to renaming— after the first iteration.

On the other hand, it is known that persistent homology is at least as expressive as 1-WL because there is a filtration that distinguishes all the graphs 1-WL can distinguish [98, Theorem 4]. In fact, Theorem 14 demonstrates that the 1-WL algorithm fails to detect even the most fundamental topological property of graphs: the number of connected components. This limitation stands in contrast to the capabilities of (persistent) homology, which can readily capture such basic topological features, encouraging the use of a combination of GNNs with (learnable) topological feature maps for graph learning tasks.

While 1-WL is effective to solve most instances of the graph isomorphism problem, it fails to differentiate graphs based on features like cycle information or triangle density. For this reason, 1-WL was extended to label *tuples* of nodes (as opposed to only labelling single nodes), leading to different hierarchies of WL algorithms depending on the length of the tuples. In this thesis, we focus on the variant known as the *folklore Weisfeiler–Leman algorithm* [106], which is described below.[2] It can be shown that there are non-isomorphic graphs that cannot be distinguished by k-FWL, but that can be distinguished by $(k+1)$-FWL. A well-known family of such non-isomorphic, non-distinguishable graphs are commonly known as CFI graphs [110]. k-FWL operates on k-tuples of vertices; for iteration $i = 0$, two tuples $\mathbf{v} = (v_1, \ldots, v_k)$ and $\mathbf{w} = (w_1, \ldots, w_k)$ are assigned the same colour if the map $v_j \mapsto w_j$ induces a graph isomorphism between the subgraphs induced by $\mathbf{v}$ and $\mathbf{w}$, respectively. For subsequent iterations with $i > 0$, we relabel the tuples similar to 1-WL, i.e.,

$$C_i^{(k)}(\mathbf{v}) := \texttt{RELABEL}\Big(\Big(C_{i-1}^{(k)}(\mathbf{v}), \Big\{\!\!\Big\{ C_i^{(k)}(\phi_1(\mathbf{v}, u)), \ldots, C_i^{(k)}(\phi_k(\mathbf{v}, u)) \mid u \in \mathcal{N}(v) \Big\}\!\!\Big\}\Big)\Big), \tag{2.17}$$

where $\phi_j(\mathbf{v}, u) := (v_1, \ldots, v_{j-1}, u, v_{j+1}, \ldots, v_k)$ refers to the function that replaces the jth element of the k-tuple $\mathbf{v}$ with u. This induces a neighbourhood relation between tuples and just as in the case of 1-WL, we run the algorithm until the

[2] There are also other variants, for instance the *oblivious Weisfeiler–Leman algorithm*. It slightly differs in the way tuples are being relabelled, but a paper by Grohe [109] shows that the variant is essentially as powerful as k-FWL (with a minor shift in indices). The reader is referred to Morris et al. [106] and the references therein for an extended discussion of these aspects.

assigned colours of tuples stabilise for one graph. As in the previous case, if the colour sequences of two graphs differ, the graphs are non-isomorphic.

3. Literature review

Contents

This literature review is divided into two sections: (1) Homology and persistence in neural network analysis; (2) High-order neural networks and topological graph learning.

In the first section, we focus on the subset of TDL literature concerned with the analysis of weights and activations of neural networks using (persistent) homology and with high-order neural networks. This section is also mostly related with Chapter 4, where we study the use of persistent homology to analyze the correlations between the activations of neurons in a neural network and their generalization capabilities using persistent homology. This section is related with our book [111] published during the development of this dissertation, from which we extracted most of the content.

In the second section, we focus on high-order neural architectures and topological graph learning methods. This review provides essential context for Chapter 5, Chapter 6, and Chapter 7, which respectively explore persistent homology-based learning algorithms, topological and standard graph learning, and high-order neural networks. The latter taking the highest relevance in Chapters 6 and 7, where

we switch the focus from persistent homology-based learning algorithms to more standard architectures in TDL. For completeness, we also review concisely the use of TDL in structural biology and drug design, due to its relationship with Chapter 7. For a more comprehensive literature review of TDL, including applications to general machine learning, we refer the reader to [41, 85, 112].

3.1 Homology and persistence in neural network analysis

Weights are arguably one of the most important parts of neural networks, as they determine their functions alongside architectures. Therefore, selecting the appropriate weights when training is key to ensuring that neural networks perform effectively. On the other hand, activations are the result of computations performed by a neural network on input data, and therefore are essential to understand the behavior of neural networks and their properties. In this section, we review the uses of persistent homology to extract topological features of internal representations of neural networks and link them with different properties of the networks, such as their generalization capabilities. We divide this section into six blocks, each representing a space of internal representations in which TDA is applied, as follows:

(1) Activations in the complete neural network graph;
(2) Activations for each layer;
(3) Weights in the complete neural network graph;
(4) Weights layer by layer;
(5) Activations whose dissimilarities depend on weights;
(6) Generic spaces.

3.1.1 Activations in the complete neural network graph

Persistent homology of activation vectors has been successfully used to analyze many aspects of deep neural networks, such as their generalization or interpretability. In an early study within this subsection, [42] analyzed how diverse topological information extracted from the set of neuron activations of feedforward neural networks was

correlated with their generalization capabilities. The neuron activations were studied for the complete graph simultaneously, and the activation vector $a_{v_i^l}$ for each neuron v_i^l was taken as

$$a_{v_i^l} = \left(\phi_{\mathcal{N}}^{(l)}(x_1)_i, \ldots, \phi_{\mathcal{N}}^{(l)}(x_n)_i\right),$$

for a fixed set of inputs $\mathcal{D} = \{x_i\}_{i=1}^n$ to the neural network $\mathcal{N}$, in this article a subset of the training dataset.

More specifically, [42] introduced *functional graphs*, with the objective of studying topological structures induced by correlations between neuron activations. The functional graph $\mathfrak{F}_{\mathcal{N}}$ of a neural network $\mathcal{N}$ and subset of vertices P with non-constant activation values is the weighted, complete graph that includes a set P of neurons of $\mathcal{N}$ as its vertices, with edge weights given by the *correlation dissimilarity*

$$w_E(\{v, w\}) = 1 - |\mathrm{corr}(a_v, a_w)|\,,$$

where corr is the sample Pearson correlation and a_v and a_w are the activation vectors of neurons v and w, respectively. The graph $\mathfrak{F}_{\mathcal{N}}$ can be seen as a point cloud (P, d) where $d(v, w) = d(w, v) = w_E(\{v, w\})$ for $v \neq w$ and $d(v, v) = 0$ for all $v \in P$, from where standard filtrations on point clouds can be used.

In their first experiments, [42] studied how the number $S(n)$ of n-dimensional simplices of altered Vietoris–Rips simplicial complexes $\mathrm{VR}_t^*(P, d)$, containing all the simplices of $\mathrm{VR}_t(P, d)$ except for isolated vertices, varied during several training processes of a LeNet neural network architecture [113] for a given parameter t satisfying

$$\frac{|\{\sigma = \{v, w\} \in \mathrm{VR}_t^*(P, d)\}|}{|\{\sigma = \{v, w\} \subseteq P : |\mathrm{corr}(a_v, a_w)| > 0\}|} \simeq 0.25.$$

In the experiments, it was found that, the higher the area under the curve of $S(n)$, the better the generalization capabilities of the neural network. In particular, it was shown that, for such training procedures, the function $S(n)$ is capable of distinguishing between the three main regimes given during training: underfitting, generalization, and overfitting. That is, the training process starts with a small area under $S(n)$ (underfitting regime), then $S(n)$ grows and achieves its maximum value (generalization regime), and then decreases again (overfitting regime).

These results motivated more involved experiments to find out the relationship between the topological properties of the activations of neural networks and their generalization capacity. For this purpose, [42] computed persistence diagrams $D_k = D(\mathbb{V}_k(\mathrm{VR}(P,d)))$ for $k \in \{1,2,3\}$ and a normalized version of their Betti curves

$$\beta_k(t) = \Big| \{(b,d) \in D_k : t \in [b,d)\} \Big|,$$

during several training procedures of a LeNet neural network, with the expectation that persistent homology contained more information than just counting the number of simplices.

During the first epochs, it was observed that the highest values of these curves moved from the left part of the domain to the right for $k \in \{1,2,3\}$. However, when the training entered into the overfitting regime, the maximum values of the normalized Betti curves moved again to the left part.

Due to the behaviour of the normalized Betti curves, [42] proposed an early stopping of the training whenever it started moving the maximum values of Betti curves to the left again. Also, [42] used normalized Betti curves to detect sets of adversarial examples, if any.

The observed evolution of Betti curves suggested that the topological structure of the functional graph of the activations of neural networks is highly related with their generalization capacity. This was further studied by [43], who used persistence diagrams $D(\mathbb{V}_k(\mathrm{VR}(P,d)))$ in dimensions $k \in \{0,1\}$ to predict the generalization gap, that is, the difference between training and test accuracies, and by us, in Chapter 4.

[43] proposed to predict the generalization gap by performing a linear regression with independent variables chosen to be two persistence summaries, namely average persistence λ and average midlife μ, given by

$$\lambda(D) = \frac{1}{|D|} \sum_{(b,d)\in D} d - b, \qquad \mu(D) = \frac{1}{|D|} \sum_{(b,d)\in D} \frac{d+b}{2},$$

respectively. This approach seemed to work on controlled computer vision scenarios.

A more specific study of one-dimensional persistence diagrams computed from neural network activations and their correlations was conducted by [114]. This study observed that the number of points in one-dimensional persistence diagrams and their 2-norms were correlated with distribution shifts in the input data and the training process of the network. Specifically, for image datasets, the number of points and the 2-norms of the persistence diagrams of the same neural network decreased as the dataset used to compute the activation vectors was increasingly corrupted by Gaussian blur or random pixel switching. Similarly, during training, the number of points and the 2-norms increased as the training process advanced until overfitting, where the number of points with high persistence stabilized suggesting an early stopping criterion without the need for a validation dataset, as in [42].

A very similar approach was used to detect trojaned neural networks. [115] proposed to build zero- and one-dimensional Vietoris–Rips persistence modules from activations as in the previous works, yet replacing the Pearson correlation coefficient in absolute value by a more general correlation coefficient measure. For such persistence modules, [115] found a significant difference between clean and trojaned neural network persistence diagrams for both synthetic and real-world experiments, where the real-world experiments comprised training 70 ResNet18 using a clean MNIST dataset and another 70 with a trojaned MNIST counterpart.

The average death time of zero-dimensional persistence diagrams was particularly significant for segregating persistence diagrams from clean and trojaned neural networks. Averages for trojaned networks were significantly lower than those of their clean counterparts. Manual inspection also revealed that trojaned models exhibited edges connecting shallow and deep layers within cycle representatives of high-persistence points in persistence diagrams, a phenomenon absent in clean models.

With this in mind, [115] trained basic FCFNN models to detect trojaned convolutional neural networks trained on MNIST, CIFAR10 and the IARPA/NIST TrojAI competition [116] datasets using persistence summaries from the persistence diagrams of the networks as input. These models resulted in higher performance than those for other state-of-the-art trojan detection algorithms.

3.1.2 Activations for each layer

While in previous works the topology of activations was studied globally, [117] studied the evolution of activations layer by layer in FCFNNs for binary classification problems in an extensive set of experiments. For data samples $\mathcal{D}_i$ from one of the classes $i \in \{1, 2\}$ at a time, the evolution of the persistent homology of $\mathcal{D}_i$ through the different layers was analyzed. Activation vectors a_x^l given by

$$a_x^l = \left(\phi_{\mathcal{N}}^{(l)}(x)_1, \ldots, \phi_{\mathcal{N}}^{(l)}(x)_{N_l}\right)$$

were computed for each example $x \in \mathcal{D}_i$ and every layer l in an FCFNN $\mathcal{N}$, representing the activations of each layer for the different input values. Then, persistent homology was calculated from Vietoris–Rips filtrations of the point clouds $\mathcal{D}_i^l = \left\{a_x^l : x \in \mathcal{D}_i\right\}$ for each layer l.

The experiments were divided into two different scenarios: synthetic and real datasets. For the first scenario, the datasets were point samples from spaces with a known topology. For the second scenario, the datasets were simplified and binarized classification versions of several known datasets, such as MNIST. In both cases, FCFNNs were trained with different architectures, different training parameters, and almost zero training error.

For synthetic datasets, the evolution of Betti numbers was studied for different homological dimensions of Vietoris–Rips complexes of fixed value t. The dissimilarity used to build these complexes was chosen to be the graph geodesic distance on k-nearest neighbors graph applied to the point clouds $\mathcal{D}_i^l$. For a fixed class i and a fixed neural network $\mathcal{N}$, the parameters t and k were selected in such a way that $\beta_0(\mathrm{VR}_t(\mathcal{D}_i))$, $\beta_1(\mathrm{VR}_t(\mathcal{D}_i))$ and $\beta_2(\mathrm{VR}_t(\mathcal{D}_i))$ were equal to the first three (known) Betti numbers of the space from which the dataset $\mathcal{D}_i$ was sampled. For the second scenario, however, persistence diagrams of the activations for each layer were analyzed, circumventing the challenge of selecting optimal parameters t and k in real datasets.

In the synthetic scenario, [117] observed a decay of the different Betti numbers across the layers. This decay was slower for zero-dimensional Betti numbers in

networks implementing completely smooth activation functions than for networks implementing ReLU-like (ReLU and leaky ReLU) activation functions. The rate of decay correlated with the topological complexity of the input data. Specifically, datasets sampled from more intricate input spaces exhibited slower decay of Betti numbers compared with those from simpler spaces.

For neural networks with a constant number of neurons in the hidden layers, narrow layers —that is, those with few neurons per layer— appeared to simplify the topology faster than their wider counterparts. On the other hand, bottleneck architectures, i.e., architectures with decreasing numbers of neurons per layer, seemed to force larger topological changes in the data than networks with a constant number of neurons per layer.

Depth did not seem to influence the way the topological simplifications are distributed across the layers, since initial layers did not perform substantial topological simplifications, in general. Thus, reducing depth only had the effect of concentrating topological simplifications in the last layers. This topological simplification was also observed in the persistent diagrams of activations for the real datasets in simpler networks than the ones used for the synthetic scenario, where the number of points and their persistence across all dimensions diminished through layers.

In this work, Betti numbers were used to quantify the *topological complexity* of activations in a given layer. Persistence landscapes also allow one to define topological complexities over point clouds. Specifically, given a persistence landscape λ coming from a point cloud, one way to measure its complexity is to measure the area under its curves λ_i. The sum of these areas defines an inner product

$$\langle \lambda, \lambda' \rangle = \sum_{i=1}^{\infty} \int_{-\infty}^{\infty} \lambda_i(t) \lambda_i'(t)\, dt,$$

that induces a norm $\|\lambda\| = \sqrt{\langle \lambda, \lambda \rangle}$. The higher the norm, the higher the topological complexity of the point cloud.

[118] used landscape complexities to contradict the previous results by [117]. In this work, evolution of the topology of activations through the different layers of FCFNNs for synthetic and real data was studied again. However, activations

and distances between them were preprocessed and slightly modified depending on the experiments performed to build a point cloud. From such preprocessed point clouds, Vietoris–Rips persistence diagrams and their landscapes were computed using the Euclidean distance between vectors for each layer and in different homological degrees.

The results for synthetic data were similar to the previous ones: for FCFNNs with 11 layers with ReLU activation functions except for the output one (that used the identity function), trained to perfect accuracy with 100 different initialization weights, topological complexities were observed to decay through the last layers, in which the best weight configurations decayed faster on average than their worse counterparts. However, for the first layers, the topological complexities increased layer by layer. More strikingly, for FCFNNs trained to near-perfect accuracy with 7 layers distributed following a bottleneck architecture with ReLU activation functions except for the output one, topological complexities were observed to increase in the last layers on average and only decreased in the first layers.

Another promising approach, deeply related to the study of decision regions, is the analysis of the topology of activations of the output layer. The fundamental hypothesis is similar to what was found in [117]: the easier the topology of the last layer, the more robust, and thus the better, the neural network.

In this regard, [119] studied the zero- and one-dimensional Vietoris–Rips persistence diagrams of last-layer activations, taken as in the previous articles, using the Euclidean distance. Although they did not prove the main hypothesis and could not relate network performance with output layer topology, it was a first step towards this direction.

Intuitively, the topology of activations for inputs of the same class must be similar. Based on this hypothesis, [120] proposed a way to measure the *quality* of convolutional filters for individual channels of convolutional layers in neural networks for classification problems. In this work, the inputs are squared images, and one assumes that the widths and heights of convolutional layer outputs are also equal. To compare the topology of the activations produced by the convolutional

filters of interest, undirected weighted fully connected graphs are built. For a layer l, channel $c \in [c^{(l)}]$, and input sample x, the filter graph $C_x^{l,c}$ is generated such that $V(C_x^{l,c}) = [h^{(l)}]$ and

$$w_E(\{i,j\}) = \max\left(\phi_{\mathcal{N}}^{(l)}(x)_{i,j,c},\, \phi_{\mathcal{N}}^{(l)}(x)_{j,i,c}\right),$$

for $i,j \in [h^{(l)}]$, where $\phi_{\mathcal{N}}^{(l)}(x)_{i,j,c}$ is the (i,j,c) output value of $\mathcal{N}$ for the l-th convolutional layer given x.

Instead of directly comparing persistence diagrams from the previous graphs for the different data classes, [120] took an alternative probabilistic approach based on the distribution of a specific topological summary for each label. Specifically, [120] computed for each graph the infimum of the support of the associated Betti curves coming from the persistence module $\mathbb{V}_1(\mathrm{VR}^1(V(C_x^{l,c}), d_\downarrow^0))$, which is directly related to the time in which the first *non-trivial cycle*, i.e., not given by a combination of triangles, appears in the filtration. We denote this minimum by $b_1^{\inf}(x,l,c)$.

The values $b_1^{\inf}(x,l,c)$ induce discrete random variables $B_1(l,c,y)$ for each label y of the classification problem with sample space Ω_y, i.e., the subset of samples from the training dataset $\mathcal{D}_{\text{train}}$ with common label y. These random variables have probability distributions

$$P_{1,l,c,y}(n) = \frac{b_n}{|\Omega_y|}, \qquad b_n = \sum_{x \in \Omega_y} \mathbb{1}_{\left\{x' \,:\, b_1^{\inf}(x',l,c)=n\right\}}(x),$$

that capture information about the similarity of the topology of the activations given by the convolutional filters of interest in the data distributions of the different labels of the classification problem. This similarity can be further quantified using the entropies of the probability distributions, given by

$$H'_{1,l,c,y} = -\sum_{n=0}^{\infty} P_{1,l,c,y}(n) \log P_{1,l,c,y}(n). \tag{3.1}$$

The values given by (3.1) are called *feature entropies*, and are the topological summaries proposed by [120] to measure the quality of convolutional filters. Feature entropies are invariant to weight reescaling, a desirable property to measure the

quality of the convolution operations, as reescalings of weights have no substantial impact on the network performance in general.

One problem of this approach is that the infimum value $b_1^{\inf}$ may not exist in some cases. For layers and channels in which this happens many times, the entropy approaches zero, although entropy is not really informative as it is affected by the non-existence of infimum values. In such cases, entropy is modified as

$$H_{1,l,c,y} = \begin{cases} H'_{1,l,c,y} & \text{if } \varepsilon_{1,l,c,y} \geq p, \\ (1 - \varepsilon_{1,l,c,y}) \log |\Omega_y| & \text{otherwise,} \end{cases}$$

where $\varepsilon_{1,l,c,y}$ is the percentage of images in class y having birth times, and p is the minimum percentage of images that we admit to use the real entropy (in the article, $p = 0.1$).

[120] demonstrated the effectivity of the entropy measure to obtain information on the *quality* of the convolutions and of the entire neural network for VGG-16 models trained on the ImageNet dataset. They observed that, for well-trained neural networks, feature entropy continually decreased as the layers went deeper, while for random weights this decreasement was absent. On the other hand, they observed that, during training, the feature entropies of the last convolutional layer decreased, and were highly correlated with the evolution of the cross entropy training loss, suggesting that feature entropies are good indicators of the generalization of networks. Additionally, randomness of the weights was also detected by comparing the feature entropies of trained and randomly initialized networks. Finally, for the last convolutional layers, models with better generalization were correlated with low feature entropies.

A similar study was conducted by [121], who analyzed the evolution of Betti curves induced by persistence diagrams computed from convolutional filter weights during training. For a fixed set of convolutional filters $\{C_i\}_{i \in I}$ of the network with the same size $h \times w$ (not necessarily from the same layer), convolution filters were transformed into vectors of size $h \cdot w$ and then used as points in a point cloud $(P, \|\cdot\|_2)$ to compute persistence diagrams of dimensions zero and one, from which Betti curves for these dimensions were extracted.

The evolution of Betti curves during training was studied for two different convolutional architectures and several datasets, including grayscale images, color images, and noise images. The networks consisted of three or four convolutional layers, depending on the architecture, and two fully connected layers. Additionally, the sizes of the convolutional filters were fixed to 3×3 across all the layers. The subset of convolutional filters used to compute persistence diagrams included all filters except those from the first convolutional layer.

For the grayscale and color datasets, the Betti curves of zero- and one-dimensional persistence diagrams were observed to shift rightward as training iterations (and performance) increased. For one-dimensional Betti curves, which exhibited a bell-shape, the maximum value decreased as training progressed. For the noise datasets, however, Betti curves remained stable across iterations, suggesting that learning occurred when changes in Betti curves happened during training, as there are no patterns to learn in random noise.

3.1.3 Weights in the complete neural network graph

Recall that the input values influence the output of a neural network via the different input-output paths available in the neural network graph. The influence of each path is characterized by the magnitudes of the weights associated with the edges in the path, which modify the magnitudes of the activations, and thus their relevance, at each step. [122] proposed to build neural network graph filtrations taking into account the influence of the different paths in the network.

To formalize their approach, let $\mathcal{N}$ be a neural network, and $v_l^i, v_{l+1}^j \in V(G(\mathcal{N}))$ be two connected neurons in the directed graph $G(\mathcal{N})$. We define the *relevance* of the directed edge (v_l^i, v_{l+1}^j) connecting v_l^i and v_{l+1}^j in the output calculation as the linearly rectified weight of the edge, normalized by the sum of the linearly rectified weights of edges pointing to v_{l+1}^j, that is,

$$R_{v_l^i, v_{l+1}^j} = \frac{\mathrm{ReLU}\left(W_{i,j}^{(l+1)}\right)}{\sum_{k \neq i} \mathrm{ReLU}\left(W_{k,j}^{(l+1)}\right)}.$$

This edge-level relevance can be extended to define relevance between vertices connected by a path in $G(\mathcal{N})$. For any two vertices v and w connected by a path, the influence from v to w is characterized as the maximum product of relevances of edges among all paths between v and w, whenever $v \neq w$, and 1 otherwise:

$$\tilde{R}_{v,w} = \begin{cases} \max_{(v,p_1,\ldots,p_n,w)\in P_{v,w}} R_{v,p_0} \left(\prod_{i=1}^{n-1} R_{p_i,p_{i+1}}\right) R_{p_n,w} & \text{if } v \neq w, \\ 1 & \text{otherwise,} \end{cases}$$

where $P_{v,w}$ is the set of all paths starting at v and ending at w in $G(\mathcal{N})$.

Building upon these definitions, the proposed graph filtrations were defined as filtrations $(K_i)_{i=1}^n$ with a monotonically decreasing sequence of indices $(t_i)_{i=1}^n$ given by

$$K_i^p = K_{t_i}^p = \begin{cases} V(G(\mathcal{N})) & \text{if } p = 0, \\ \{\{v_{k_0}\ldots,v_{k_p}\} : \tilde{R}_{v_{k_i},v_{k_j}} \geq t_i \text{ for } k_i > k_j\} & \text{if } p \geq 1, \end{cases}$$

where p indicates the dimensions of the simplices of K_i^p, and the vertices $V(G(\mathcal{N}))$ are ordered in such a way that if the layer number of v_i is lower or equal than the layer number of v_j then $i \geq j$.

The filtration $(K_i)_{i=1}^n$ captures the influence of the different paths between neurons in the neural network graph and can be easily extended to a persistence module where the simplicial complex assigned to time $t \in \mathbb{R}$ is $K_{\lfloor t \rfloor}$ for $0 \leq t \leq n$, K_0 for $t < 0$, and K_n for $t > n$.

For simple networks trained on MNIST and CIFAR-10, the distribution of points of one-dimensional persistence diagrams computed from persistence modules coming from the previous filtration appeared to be correlated with several properties of the network and the dataset, such as problem difficulty or network expressivity. For example, the appearance of points near the diagonal of the persistence diagrams was associated with a shortage of data of specific labels in the training dataset.

In addition, persistence diagrams seemed to be robust with respect to the initial values of the neural network weights, yielding each architecture similar persistence diagrams for different initializations after training. However, a more thorough study of such persistence modules and their associated persistence diagrams is needed to understand their relationship with the generalization capabilities of a neural network, as [122] remarked.

In a follow-up work, [123] presented an extension of the relevance quantity for convolution and pooling transformations that led to persistence modules better suited to analyze convolutional neural networks. Specifically, [123] used this extension to study the overfitting of CNNs.

[123] observed that, in simple scenarios, distributions of points in one-dimensional persistence diagrams were correlated with the overfitting of convolutional networks. Specifically, as the dropout rate increased for different training instances within the same architecture, a corresponding increase in the number of points near the diagonal and in overfitting was also observed. Furthermore, it was found that a higher total number of points in the persistence diagram correlated with reduced underfitting in neural networks.

One drawback of their method is that straightforward persistence diagrams were found not to be entirely suitable for comparing models with different architectures in this case. To address this difficulty, [123] proposed a novel approach: pruning the networks of interest using a magnitude-based strategy [124] before generating persistence diagrams to compare them so that the persistence diagrams were normalized with respect to the size of the network. After selecting several groups of neural networks to be compared, an overall high correlation between the number of points near the diagonal of normalized persistence diagrams and overfitting of the networks of the different groups was observed, validating their previous results in a more general setting.

Pruning using persistent homology was further developed in [125] using a slightly modified relevance function given by

$$R_{v,w} = \frac{\left|W_{i,j}^{(l+1)}\right|}{\sum_{k\neq i}\left|W_{k,j}^{(l+1)}\right|},$$

[125] proposed a pruning algorithm that was shown to be competitive with respect to the global magnitude algorithm [124] in terms of final accuracy of the pruned networks. The algorithm was as follows:

(1) Sort points (b, d) of one-dimensional persistence diagrams by their value $b + d$ in ascending order;

(2) For each point, choose a cycle representative c of the point (b, d);

(3) Select the edges contained in the representatives in the previous step until you reach a desired number of edges to retain;

(4) Prune the edges not selected in the previous step.

A similar approach to the one taken by [43] was proposed by [126]. [126] argued that the dataset has too much influence on the activations and thus their correlations do not capture all the relevant information about the neural network related with generalization. Instead, they proposed to analyze zeroth persistence diagrams $D(V_0(\mathrm{VR}(P, d)))$ induced by subsets P of neural network vertices and dissimilarities given by the Euclidean distance between HVS scores [127] S_v associated to each vertex v of P defined as

$$S_v = \sum_{(v,w)\in E(G(\mathcal{N}))} \pi_{w,v} \cdot \delta_w,$$

$$\pi_{w,v} = \frac{|W_{w,v}|}{\sum_{(u,w)\in E(G(\mathcal{N}))} |W_{w,u}|}, \qquad \delta_w = \begin{cases} 1 & \text{if } w \in V_L, \\ S_w & \text{otherwise,} \end{cases}$$

where $W_{w,v}$ is the weight corresponding to the edge $(v, w) \in E(G(\mathcal{N}))$ and V_L is the set of neurons of the output layer.

To study the relationship of these diagrams with the generalization gap, [126] trained very simple FCFNNs with two hidden layers for a variety of datasets from the UCI Machine Learning repository [128], taking P as the 90% of neurons with highest relevance score for each network. For the experiments, the generalization gap was compared with the average persistence value of the diagram generated at each iteration of the training procedure. Linear regression models were fitted with the generalization gaps and average persistences during training as dependent and independent variables, respectively. The R^2 values of these linear regressions were slightly greater than the ones for the same linear regression models using the persistence diagrams computed in our own results of Chapter 4. However, the simplicity in the experiments performed, both in the experimental pipeline and in the networks used, makes that further experimentation is needed to evaluate

if this approach generalizes to more complex scenarios and also improves our methods in them.

3.1.4 Weights layer by layer

Layerwise topological complexities have also been studied as a function of the weights. [129] proposed to study the so-called neural persistence. Given a non-output layer l of a neural network $\mathcal{N}$, its *neural persistence* is defined as

$$\mathrm{NP}_l(\mathcal{N}) = \|D\|_p = \Big(\textstyle\sum_{(b,d)\in D} |d-b|^p \Big)^{1/p},$$

where $D = D^w\Big(\mathbb{V}_0\Big(\mathrm{VR}^1\Big(V(G_l), d_\downarrow^V\Big)\Big)\Big)$ is the zeroth persistence diagram of the weighted complete bipartite graph G_l with vertices $V_l \sqcup V_{l+1}$ and weights given by $w_V(v) = w_{\max}$ and $w_E(\{v_l^i, v_{l+1}^j\}) = |W_{j,i}^{(l+1)}|/w_{\max}$, where $W_{j,i}^{(l+1)}$ is the weight associated to the edge connecting the vertices v_l^i and v_{l+1}^j, as in Figure 2.7, and $w_{\max} = \max_{l,i,j} |W_{i,j}^l|$ is the maximum weight in absolute value among all the weights of the network.

To compare the neural persistence of different layers from the network, [129] proposed to normalize neural persistence by dividing it by an upper bound of the neural persistence,

$$\mathrm{NP}_l^+(\mathcal{N}) = w_{\max}^{-1} \left(\max_{i,j} \left|W_{i,j}^{(l+1)}\right| - \min_{i,j} \left|W_{i,j}^{(l+1)}\right| \right) (N_l - 1)^{1/p},$$

obtaining the normalized neural persistence $\widetilde{\mathrm{NP}}_l(\mathcal{N})$ for each layer l. Averaging normalized neural persistences over all the layers of the network, a global topological complexity measure $\overline{\mathrm{NP}}(\mathcal{N})$ is obtained.

For simple FCFNN networks trained on MNIST, neural persistence was able to distinguish clearly between properly and badly trained networks, for which the values of the neural persistence of properly trained ones were consistently higher than for their badly trained counterparts. Furthermore, it was observed that different regularization techniques augmented the mean neural persistence with respect to the values of regular trained networks, suggesting that for a fixed architecture, the higher the neural persistence, the better the generalization capabilities of the network.

This was exploited as an early stopping criterion for training neural networks, where the training process is completed when the mean neural persistence $\overline{\mathrm{NP}}(\mathcal{N})$ stops increasing significantly. This early stopping criterion was found to be competitive with other early stopping criteria but without the need for a validation dataset.

[130] deepened more into the properties of neural persistence. A key contribution of their work was observing a close relationship between the variance of the learned weights of deep learning models and the neural persistence, questioning the value of the latter with respect to this simpler measure to study the properties of the neural network, arguing that this variance may be similarly useful for the applications showcased by [129].

3.1.5 Activations whose dissimilarities depend on weights

Both weights and activations can be used together to study the topology of neural networks. In particular, they can be used to inspect differences in the internal workings of neural networks for different input values. Given an input sample x and a neural network $\mathcal{N}$, [131] proposed to analyze the zeroth persistence module $\mathbb{V}_0\Big(\mathrm{VR}^1\Big(V(G_x), d_{\downarrow}^0\Big)\Big)$ for G_x the undirected weighted graph induced by the directed neural network graph $G(\mathcal{N})$ with weights given by the formula

$$w_E(\{v_l^i, v_{l+1}^j\}) = \Big|W_{j,i}^{(l+1)} \phi_{\mathcal{N}}^{(l)}(x)_i\Big|.$$

The weight function captures how activations are distributed through the neural network, as in the previous works. However, in this case, activations are weighted by the weights of the neural network, which do not necessarily depend on the example x, and which have an effect of normalization on the influence of the input. Each persistence module is intended to capture information of the network under the influence of input x. In this way, the differences between persistence modules for the same network and different inputs can be used to gain insight into the dynamics and representations used by the neural network to compute its function.

To measure the differences between persistence modules coming from the same network $\mathcal{N}$ and different inputs x, x' quantitatively, [131] proposed to calculate

a dissimilarity based on the differences between the cycle representatives of the points of the two persistence diagrams $D_x = D\Big(\mathbb{V}_0\Big(\mathrm{VR}^1\Big(V(G_x), d^0_{\downarrow}\Big)\Big)\Big)$ and $D_{x'} = D\Big(\mathbb{V}_0\Big(\mathrm{VR}^1\Big(V(G_{x'}), d^0_{\downarrow}\Big)\Big)\Big)$, respectively.

More precisely, let $\{\alpha_i\}_{i=1}^{|D_x|}$ and $\{\beta_j\}_{j=1}^{|D_{x'}|}$ be representatives of the points of D_x and $D_{x'}$, respectively. Each representative cycle α_i and β_j is associated with a subgraph of the graph G_x and $G_{x'}$, respectively, denoted by T_i and T'_j. Let v_x and v'_x be two vectors with as many components as edges there are in the union of graphs

$$\left(\cup_{i=1}^{|D_x|} T_i\right) \cup \left(\cup_{j=1}^{|D_{x'}|} T'_j\right),$$

respectively, where v_x and v_y have one in the components corresponding to the edges that come from the union of graphs $\left(\cup_{i=1}^{|D_x|} T_i\right)$ and $\left(\cup_{j=1}^{|D_{x'}|} T'_j\right)$, respectively, and a zero otherwise. Then, the dissimilarity between the persistence modules generated from x and x' is defined as a weighted version of the Hamming distance between v_x and $v_{x'}$ using the persistence of the cycle representatives associated with the edges corresponding to the different components of the vectors as weights.

[131] demonstrated the efficacy of the aforementioned persistence modules and associated dissimilarity measures in analyzing neural networks through experiments on MNIST, FashionMNIST, and CIFAR-10 datasets, employing three simple convolutional architectures, including an AlexNet [132] variant for the CIFAR-10 dataset. Their findings were threefold:

(1) Persistence modules effectively captured distinctions between clean and adversarial or shifted examples when processed by the networks;

(2) Distances between persistence modules derived from image sets closely mirrored actual image distances;

(3) Support vector machines (SVMs) achieved high classification accuracy using persistence modules as input, with the dissimilarity measure serving as a kernel.

These results suggest that the topological information extracted by persistence modules encapsulates much of the data used by neural networks for classification. Moreover, this topological information appears sufficiently distinct to elucidate differences in the neural network's behavior across various classes.

The aforementioned persistence modules were further studied to detect adversarial examples by [133]. Their hypothesis was that only a small set of edges within the networks are used for inference of non-adversarial inputs, meaning that activations associated to most edges are irrelevant to the output, and that, for adversarial examples, the number of edges used for inference is larger. The idea behind the hypothesis is that adversarial examples attack input-output edge paths containing underused edges of the neural network to, with imperceptible modifications of the inputs, completely change the output.

Ideally, changes in the activations of neurons within underused paths lead to structural modifications in the activation patterns of the neural networks, consequently affecting their persistence modules. To better identify topological changes, [133] only added edges to the graph G_x that were underoptimized, defined as those with *low* weights in absolute value.

The previous hypothesis was reflected in the number of points of zeroth persistence diagrams points coming from persistence modules of adversarial and non-adversarial examples, where adversarial examples had more points on average than their non-adversarial counterparts. Furthermore, they successfully detected adversarial examples by training a support vector machine using the sliced Wasserstein kernel [134] between the persistence diagrams induced by these persistence modules for a variety of neural network types (LeNet, ResNet), datasets (MNIST, Fashion-MNIST, SVHN, CIFAR-10) and adversarial attacks.

Topological data analysis has also been used to analyze neural networks in reinforcement learning (RL) tasks. [135] studied the evolution of Betti numbers given by homology groups of complexes induced by graphs coming from RL neural networks during a *time* period either in inference or training. Specifically, homology groups for the directed flag [136] and Vietoris–Rips complexes were calculated from

adjacency graphs $G_x^{r,d}$ derived from the weighted activation graph G_x proposed by [131], where x was the input of the RL agent at a specific time. Each graph $G_x^{r,d}$ was built by taking the edges of G_x with weights higher than or equal to r and adding an edge between two vertices v_l^i and v_l^j if there existed a third vertex v_{l-1}^k such that

$$\left|(W_{i,k}^{(l)} - W_{j,k}^{(l)})\phi_{\mathcal{N}}^{(l-1)}(x)_k\right| < d,$$

meaning that the activations of vertices v_l^i and v_l^j are highly correlated for input x.

For inference experiments, Betti numbers up to dimension three for each time step were computed and smoothed using a moving average window. Transitions between actions of the agent and evolution of the Betti numbers of dimension three seemed to be correlated, while for the other dimensions this correlation was not conclusive. For training experiments, complex environments seemed to induce the development of higher-order Betti numbers during training. Also, with a similar number of neurons, the higher the Betti numbers in the last steps of training, the better the model worked.

Training was also studied through a matrix H with training steps as columns and neurons as rows. Each coefficient $H_{v,t}$ of H was derived from cocycle representatives of one-dimensional cohomology groups of Vietoris–Rips complexes at time t. Specifically, $H_{v,t}$ was the maximum length of 1-cocycles in the set $\mathcal{C}_v^t$ of cocycles whose support had edges containing the vertex v at time t.

The matrix H was seen to have higher values for neurons on the latest layers consistently for all the training steps, suggesting that deeper neurons have more relevance in the topological structure extracted from the graphs $G_x^{r,d}$. However, more experiments are needed to validate these results, as experiments were performed only for basic FCFNNs.

A combination of the methods proposed in [129] and in [131] was used to define a new prediction reliability score for neural networks in classification problems called *topological uncertainty*. Given a neural network $\mathcal{N}$ and a sample x_{in} for which we want to compute this score, the topological uncertainty of x_{in} is computed

from the set of persistence diagrams

$$D_x^l = D^w\Big(\mathbb{V}_0\Big(\mathrm{VR}^1\Big(V(G_x^l), d_V^0\Big)\Big)\Big),$$

for all $l \in [L]$ and $x \in \{x_{\text{in}}\} \cup \{x : (x, y) \in \mathcal{D}_{\text{train}}\}$, where L is the number of non-input layers of $\mathcal{N}$, and $\mathcal{D}_{\text{train}}$ is the training dataset, and G_x^l is the subgraph of G_x as in [131] induced by the vertices of the layer l and $l-1$ of $\mathcal{N}$ and all the possible edges connecting them in G_x.

In this case, edges are weighted as in G_x but vertices are weighted as -1. This is done to always have a fixed number of points in zeroth persistence diagrams equal to the number of vertices of G_x^l for any possible weight assignment to the edges and to have all birth values equal. It can be proved that, in this way, there is a bijection between the finite deaths of the persistence diagrams D_x^l and the multiset of weights of the maximum spanning tree of G_x^l, which we denote by $\mathfrak{W}_x^l$. These persistence diagrams are meant to capture information similar to that computed in [131], but more fine-grained, since they are computed for each pair of adjacent layers.

Maximum spanning tree weights are used to build class-specific weight distributions. The topological uncertainty measure for an example is then computed as the difference between its maximum spanning tree weights and the distribution of weights for its predicted class. Recall that $\left|\mathfrak{W}_x^l\right| = \left|\mathfrak{W}_{x'}^l\right|$ for any pair of inputs x and x'. Given a multiset of weights $\mathfrak{W}_x^l$, let us order them in descending order and denote them by

$$\mathfrak{W}_x^l = \left\{w_{x,1}^l, \ldots, w_{x,\left|\mathfrak{W}_x^l\right|}^l\right\}.$$

Weights induce a probability distribution for each subgraph G_x^l given by

$$\mu_x^l = \frac{1}{n}\sum_{i=1}^n \delta_{w_{x,i}^l},$$

where $n = \left|\mathfrak{W}_x^l\right|$ and $\delta_{w_{x,i}^l}$ denotes a Dirac measure at $w_{x,i}^l \in \mathbb{R}$. For the same layer, these probability distributions can be combined to obtain an *average topological distribution* across several samples. The average over points $x_1, \ldots, x_m$ is given by

$$\bar{\mu}^l = \frac{1}{\left|\mathfrak{W}_x^l\right|}\sum_{i=1}^{\left|\mathfrak{W}_x^l\right|} \delta_{\bar{w}_i}, \qquad \bar{w}_i = \frac{1}{m}\sum_{j=1}^m w_{x_j,i}^l.$$

Therefore, given the new sample x_{in}, the topological uncertainty measures the average difference between the average topological distributions $\bar{\mu}^l$ for each layer l calculated for the subset of the training dataset $\mathcal{D}_{\text{train}}$ with label equal to the predicted label for x_{in} with respect to the topological distributions $\mu^l_{x_{\text{in}}}$ of the input sample x_{in}, that is,

$$\begin{aligned} \text{TU}_x(\mathcal{N}) &= \frac{1}{L}\sum_{l=1}^{L} \text{d}\Big(\mu^l_{x_{\text{in}}}, \bar{\mu}^l_{\mathcal{D}_x}\Big), \\ \mathcal{D}_x &= \{x : (x,y) \in \mathcal{D}_{\text{train}} \text{ with } \pi \circ \phi_{\mathcal{N}}(x_{\text{in}}) = y\}, \end{aligned}$$

where $\bar{\mu}^l_{\mathcal{D}_x}$ is the average probability distribution over the set $\mathcal{D}_x$ and where

$$\text{d}(\mu_1, \mu_2) = \frac{1}{|\mathfrak{W}_1|}\sum_{i=1}^{|\mathfrak{W}_1|} |w_i^1 - w_i^2|$$

is a dissimilarity function between distributions coming from sets of weights $\mathfrak{W}_1 = \{w_1^1, \ldots, w_{|\mathfrak{W}_1|}^1\}$ and $\mathfrak{W}_2 = \{w_1^2, \ldots, w_{|\mathfrak{W}_1|}^2\}$ with the same number of points decreasingly ordered as before.

The lower the value of the topological uncertainty measure, the more reliable the prediction for x_{in}, since its internal behavior is more similar to the behaviour of the network for the examples in the training dataset with the same label. This measure was successfully used for model selection from a set of models trained on MNIST and Fashion-MNIST, where the model with the lowest average topological uncertainty for the new dataset is selected. It was also used for detecting out-of-distribution and shifted examples in several basic networks and datasets, including MUTAG [137], COX2, and MNIST datasets, considering only some set of (non-convolutional) layers to compute topological uncertainty.

3.1.6 Generic spaces

So far we have seen methods to compare differences between the topology, in a broad sense, of different neural network *representations*. Most of these methods are simply based on distances on either persistence diagrams or on some constructions coming from persistence modules.

A more direct approach is taken by [138], who propose a method to compare two Vietoris–Rips filtrations for the same set of points V and different distances d_1, d_2. The idea behind the method is to compare, given a specific threshold t, how the connected components of $\mathrm{VR}_t(V, d_1)$ and $\mathrm{VR}_t(V, d_2)$ are merged in $\mathrm{VR}_t(V, d_{\min})$ where $d_{\min}(x, y) = \min(d_1(x, y), d_2(x, y))$. In particular, they count how many connected components are merged at each threshold for all the possible thresholds, and derive a measure of dissimilarity from such countings.

This measure of dissimilarity is called *representation topology divergence* and is defined as the average of two total persistences, denoted by $\mathrm{RTD}_1(d_1, d_2)$ and $\mathrm{RTD}_1(d_2, d_1)$, calculated from one-dimensional persistence diagrams of the Vietoris–Rips filtrations of the point clouds $(V_{1,2}, d_{1,2})$ and $(V_{2,1}, d_{2,1})$ with vertices $V_{i,j} = \{v_a\}_{a=1}^{|V|} \cup \{v'_a\}_{a=1}^{|V|} \cup \{O\}$ and distances

$$
\begin{aligned}
d_{i,j}(v'_a, v'_b) &= \min(d_i(v_a, v_b), d_j(v_a, v_b)), \qquad & d_{i,j}(v_a, v'_a) &= d_{i,j}(O, v_a) = 0, \\
d_{i,j}(v_a, v'_b) &= d_{i,j}(v_a, v_b) = d_i(v_a, v_b), & d_{i,j}(v_b, v'_a) &= d_{i,j}(O, v'_a) = \infty,
\end{aligned}
$$

for $i \in [2]$, $j \in [2] \setminus \{i\}$, where v'_a is the node v_a duplicated in the point cloud and O is an abstract point that is useful to capture the differences between $\mathrm{VR}_t(V, d_i)$, $\mathrm{VR}_t(V, d_j)$ and $\mathrm{VR}_t(V, d_{\min})$.

Intuitively, the k-dimensional persistence diagram from the Vietoris–Rips filtration of these point clouds records the k-dimensional topological features that are born in $\mathrm{VR}_t(V, d_{\min})$ but not yet in $\mathrm{VR}_t(V, d_i)$, and the $(k-1)$-dimensional topological features that are dead in $\mathrm{VR}_t(V, d_{\min})$ but are not yet dead for $\mathrm{VR}_t(V, d_i)$.

The representation topology divergence has been used to analyze many aspects of neural networks. For example, [138] used it to analyze 400-dimensional embeddings of 10,000 words for 90 randomly selected architectures from the NAS-Bench-NLP [139], as well as differences between real and shifted data, and several properties of neural networks, among others. For studying neural network properties, they trained a VGG-11 and a ResNet-20 convolutional networks on CIFAR-10 and CIFAR-100 and compared the evolution of the activations of convolutional layers to the activations of the final trained network for the same layer.

They observed that the representation topology divergence between activations at each epoch during training and activations of the network after training decreased as the number of epochs trained increased, capturing the convergence to the final state of the network. They also found that the evolution of the representation topology divergence for both models correlated well with the disagreement of the predictions between them. Finally, they showed that the representation topology divergence could be a good indicator of diversity and disentanglement of models in an ensemble method or generative tasks, respectively, when used properly.

3.2 High-order neural networks and topological graph learning

In this section, we present a focused review of the literature on high-order neural networks and topological graph learning. Given the terminological ambiguity surrounding *high-order* across different research communities, we narrow our scope to architectures specifically designed for simplicial and cellular complexes in the case of non-transformer based models, intentionally excluding high-order formulations for conventional graphs, hypergraphs, and other algebraic structures such as sheaves. This focused delimitation aligns with the topological domains central to our thesis. In the case of transformer-based models, we also consider other high-order architectures, due to the importance of these architectures for Chapter 7. For a broader survey on high-order architectures, we refer the reader to Papillon et al. [85], which provides a comprehensive survey of message-passing neural networks in these, and more, topological domains. Our treatment here is deliberately more concise than the previous Section 3.1. This brevity is motivated by two essential factors: (1) the existence of [85], which already provides a comprehensive survey of most of the literature on message passing high-order neural architectures; and (2) the different focus of our work, which does not aim at extending previous SOTA mechanisms, but rather at developing attention and transformers operating on topological domains and creating a new benchmark for high-order neural networks. Thus, we review

existing methodologies primarily to contextualize our contribution rather than to exhaustively catalog the field.

3.2.1 Higher-order neural networks

Message passing forms the foundational paradigm for most simplicial neural networks, with SNN [140] and SCCONV [141] representing two of the earliest contributions to this field. SNN introduced, to the best of our knowledge, the first simplicial convolutional layer, which leverages Fourier transforms on simplicial complex feature maps to define convolutions across feature map spaces of specific dimensions.

The simplicial convolutional layer operates by transforming feature maps associated with a given dimension k of the simplicial complex using the following combination of a non-linear map φ and a linear combination of powers of the Laplacian matrix:

$$\varphi\left(\sum_{i=0}^{N}\theta_i L_k^i F\right), \tag{3.2}$$

where L_k^i denotes the i-th power of the k-dimensional Laplacian matrix of the simplicial complex, F represents the matrix whose rows contain the feature maps for each k-dimensional simplex, arranged according to the indexing scheme used in constructing the Laplacian matrix, and $\{\theta_i\}_{i=0}^{N}$ are learnable weight matrices. Similar feature maps update functions were proposed by [142], where the update function was given by the formula

$$\varphi\left(A \odot \tilde{L}_k^{+} F\theta\right),$$

where $\odot$ is the Hadamard product, $A = \max(A_k^{\text{up}}, A_k^{\text{down}})$, θ is a learnable matrix, and $\tilde{L}_k^{+}$ is the sifted inverted Laplacian of dimension k, given by the Moore–Penrose pseudoinverse of the shifted Laplacian, i.e., $\tilde{L}_k^{+} = (I + L_k)^{+}$.

On the other hand, SCCONV proposes an alternative convolutional approach that explicitly accounts for hierarchical interactions between simplices of adjacent dimensions. While initially formulated for simplicial complexes of maximum dimension 2, this methodology can be generalized to higher-dimensional structures.

If we denote the feature map matrix for k-dimensional simplices as F_k, the SCCONV convolutional layer updates can be expressed, in a simplified notation, as:

$$\begin{aligned} F_0 &= \varphi_0\left(\tilde{B}_1 F_1 \theta_{0,1} + \tilde{A}_0 F_0 \theta_{0,0}\right), \\ F_1 &= \varphi_1\left(\tilde{B}_2 F_2 \theta_{1,2} + \tilde{A}_1 F_1 \theta_{1,1} + \tilde{B}_1^T F_0 \theta_{1,0}\right), \\ F_2 &= \varphi_2\left(\tilde{A}_2 F_2 \theta_{2,2} + \tilde{B}_2^T F_1 \theta_{2,1}\right). \end{aligned} \tag{3.3}$$

In this formulation, $\tilde{B}_k$ and $\tilde{B}_k^T$ represent normalized boundary and coboundary matrices of dimension k, derived from their original counterparts B_k and B_k^T, $\tilde{A}_k$ denotes normalized and generalized adjacency matrices of dimension k, constructed using the upper and lower adjacency concepts defined in Section 1.1, and $\theta_{i,j}$ are learnable parameter matrices that govern the transformations between feature spaces of different dimensions, enabling the network to capture cross- and same-dimensional dependencies within the simplicial complex and its feature maps.

[95] introduces two distinct convolutional architectures for simplicial complexes that combines characteristics of [140] and [141]. The first (SCN) operates independently on each feature map dimension k with the update function

$$\varphi\left(\tilde{A}_k F_k \theta_k\right),$$

where $\tilde{A}_k$ is a normalized adjacency matrix that simultaneously incorporates upper adjacencies, lower adjacencies, and self-connections. The second approach (SCCN) implements a joint convolution across all dimensions of the simplicial complex

$$\varphi\left(\tilde{A} F \theta\right),$$

where $\tilde{A}$ is a normalized version of a matrix A, structured as a block matrix. This matrix A contains dimension-specific adjacency matrices along the diagonal, with boundary matrices and their transposes positioned on the second superdiagonal and subdiagonal, respectively:

$$A = \begin{bmatrix} \alpha A_0 & \beta B_0 & 0 & \cdots & 0 \\ \beta B_0^T & \alpha A_1 & \beta B_1 & \ddots & \vdots \\ 0 & \beta B_1^T & \alpha A_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta B_n \\ 0 & \cdots & 0 & \beta B_n^T & \alpha A_n \end{bmatrix},$$

where α and β are learnable hyperparameters controlling the relative importance of adjacency and boundary information and where each A_k is defined as $A_k = \max(A_k^{\text{up}}, A_k^{\text{down}})$.

[143] introduces the Simplicial Convolutional Networks (SCoNe) architecture for simplicial complexes of dimension at least two. SCoNe primarily operates on 1-dimensional feature maps throughout the network, with 0-dimensional feature maps generated only at the final layer through the application of the boundary operator to the 1-dimensional features, followed by a linear projection. The core convolutional update function incorporates both upper and lower adjacency information via the Hodge Laplacian components $B_2B_2^T$ and $B_1^TB_1$, respectively. Each component is parameterized separately, and the update also includes a self-connection term. Specifically, the update function is given by:

$$\varphi\left(B_2B_2^TF_1\theta_1 + F_1\theta_2 + B_1^TB_1F_1\theta_3\right),$$

where φ is an activation function and $\{\theta_i\}_{i=1}^3$ are learnable matrix parameters. This formulation allows SCoNe to effectively capture interactions between one-dimensional feature maps depending on both, the features, and the incidence structure of the simplicial complex.

[144] introduces the Simplicial Convolutional Neural Network (SCNN), which extends previous approaches for k-dimensional feature maps by employing multiple parallel convolutions followed by aggregation within each layer. The SCNN architecture defines simplicial filters as matrices with the form

$$H_{\epsilon,\theta^1,\theta^2} = \epsilon I + \sum_{l_1=1}^{L_1} \theta_{l_1}^1 (B_1^TB_1)^{l_1} + \sum_{l_2=1}^{L_2} \theta_{l_2}^2 (B_2B_2^T)^{l_2},$$

where ϵ, $\{\theta_{l_1}^1\}_{l_1=1}^{L^1}$, and $\{\theta_{l_2}^2\}_{l_2=1}^{L_2}$ are (learnable) parameters, I is the identity matrix, and the powers of $B_1^TB_1$ and $B_2B_2^T$ capture multi-hop lower and upper adjacency relationships, respectively. For each dimension k, an SCNN layer utilizes a set of filter matrices $\{H_{\epsilon,\theta^1,\theta^2}^{f,g}\}_{f,g=1}^T$ to generate T distinct intermediate representations according to

$$F_k^f = \varphi\left(\sum_{g=1}^{T} H_{\epsilon,\theta^1,\theta^2}^{f,g} F_k^g\right), \quad f \in \{1,\dots,T\},$$

where φ is an activation function and $F_k^g = F_k$ for all $g \in \{1, \ldots, T\}$ in the initial layer. Through successive application of these operations across multiple layers, the SCNN produces T final representations $\{F_k^f\}_{f=1}^T$ that are subsequently aggregated into a final output representation tailored to the specific task. In a similar vein, [96] extends the concept of simplicial filters to incorporate feature maps from adjacent dimensions $k-1$ and $k+1$ when computing new feature representations at each layer for dimension k. Unlike the SCNN architecture, however, this approach does not utilize multiple parallel convolutions within the same layer. A generalization of message passing on simplicial complexes taking into account higher-order interactions between the same and adjacent dimensions is proposed in [145], where some expressivity results about simplicial complex message passing neural networks are also provided.

BSCnets [146] were among the first convolutional architectures to introduce interactions between simplices of non-adjacent dimensions. For a simplicial complex K, let F be the concatenation of feature maps F_k for $k = 0, \ldots, \dim K$. The BSCNet convolutional layer is then defined by the update function

$$\left(\tilde{L}^B F \theta_1\right) \theta_2,$$

where θ_1 and θ_2 are learnable parameter matrices, and $\tilde{L}^B$ is obtained through

$$\tilde{L}^B = \operatorname{softmax}\left(\operatorname{ReLU}\left(L^B\right)\right),$$

where L^B is the Adaptive Hodge Laplacian of K, structured as a block matrix:

$$L^B = \begin{bmatrix} L_0 & S_{0,1} & \cdots & S_{0,\dim K} \\ S_{0,1}^T & L_1 & \cdots & S_{1,\dim K} \\ \vdots & \vdots & \ddots & \vdots \\ S_{0,\dim K}^T & S_{1,\dim K}^T & \cdots & L_{\dim K} \end{bmatrix}.$$

Here, L_k represents the k-dimensional Hodge Laplacian of K, and $S_{i,j}$ denotes the similarity matrices between i-dimensional and j-dimensional simplices, computed using L_i and L_j. This formulation allows for flexible application to specific dimension subsets by selecting appropriate feature maps and corresponding blocks in L^B (e.g., restricting to dimensions $k = 1, 2$ or $k = 2, 3$).

More complex architectural designs have also been adapted from other domains such as graph or language learning. For example, the concurrent papers [147] and [93, 148], and [149] generalize the attention mechanisms originally proposed in [150] for graphs to simplicial complexes, [97] adapts the mamba state-space framework [151] to simplicial complexes, [152] extends the skip connection neural networks [153, 154] to simplicial complexes, [155] adapts the equivariance framework to simplicial complex neural networks dealing with feature maps containing geometric information such as coordinates, and [156] extends the encoder-decoder pattern using message passing simplicial complex architectures to learn euclidean simplicial representations.

However, some original work has also been developed specifically for simplicial complexes without direct equivalents in other fields. Two prominent examples are [157] and [158], where new architectures are developed for geometric realizations of abstract simplicial complexes with embedding information and for directed simplicial complexes, a novel concept developed in the same paper, respectively.

However, some original work has also been developed for the realm of simplicial complexes that does not have a direct equivalent in other fields. Two prominent examples of this original work are [157] and [158], where new architectures dealing with geometric realizations of abstract simplicial complexes and their embedding information and with directed simplicial complexes are developed.

Most of the previous architectures can be generalized to the realm of cellular complexes. This is the case of [35], which generalizes [159] to cellular complexes. Similarly, Hajij et al. [160] generalizes message passing to cellular complexes, while Giusti et al. [161] generalizes attention to cellular complexes. Finally, [162], defines a general layer to transform feature maps of taking into consideration arbitrary dimensions (not necessarily adjacent) to update a specific dimension k.

High-order transformers

Transformer models which go beyond pairwise relations represent a natural progression from graph-based transformers. The most prominent category of such higher order transformers operates on hypergraphs. In many instances, they have also

adopted the self-attention mechanism [163–166]. Beyond graphs and hypergraphs, transformers operating on topological domains are scarce. To our knowledge, only two higher-order transformers operating on simplicial complexes have been proposed [167, 168]. The first approach, however, does not consider higher order features directly, but rather leverages higher order relations to improve features on nodes. The second approach [168], although being a fairly general object to define higher-order structures, focuses primarily on graph learning. It proposes two architectures: one operating on tuples of nodes (i.e., cliques) within the graph, which may not naturally appear in the clique complex of the graph, and another applying a general attention mechanism to all simplices in a lifted graph simultaneously, disregarding the distinct nature of data across dimensions (e.g., properties of atoms in nodes vs. properties of bonds in edges). The latter was only tested on nodes and edges, excluding higher-order elements.

3.2.2 Topological graph learning

In the context of graph learning, the use of topological data analysis, and particularly persistent homology has contributed many new insights and tools. In the context of architectures, persistent homology has been used as pooling layers [169, 170], readout layers [171], and regular layers [98, 172, 173]. In the context of expressivity and graph neural network property analysis, [98] also proved that persistent homology is at least as expressive as 1-WL and [174] extended this result and our results of Chapter 5 for persistent homology for vertex- and edge-based filtrations.[1]

Topological deep learning in structural biology and drug design. [175] and [176] advanced virtual screening by employing multiparameter persistence homology to create topological fingerprints, surpassing traditional canonical methods like SMILES and Morgan fingerprints as well as VAEs and GNNs, and achieving notable performance improvements on benchmark datasets including Cleves-Jain and DUD-E Diverse. [177] presented MIFS, an adaptive multipath information

[1] Our results were extended after our publication [46] associated to Chapter 5 was published.

fused self-supervised framework that employs a topologically-driven molecular encoder, Mol-EN, to capture atom-to-atom, bond-to-atom, and group-to-atom information propagation pathways and integrates a topological contrastive loss based on molecular scaffolds, resulting in significantly enhanced drug discovery performance over traditional encoders and existing self-supervised methods across 14 benchmark datasets, including those from the ZINC database. [178] developed PersGNN, a hybrid deep learning framework that integrates persistent homology-based topological data analysis with graph neural networks to effectively capture both local and global structural features of protein structures, resulting in enhanced protein function prediction performance compared to traditional GNNs and baseline models across multiple benchmark datasets. TopologyNet, a deep convolutional and multi-task neural network that utilizes element-specific persistent homology to transform complex three-dimensional biomolecular structures into multichannel topological representations, demonstrated accurate predictions of protein-ligand binding affinities and protein stability changes upon mutation [179].

4. Functional graph topology

Contents

4.1 Predicting the generalization gap

Understanding the generalization capacity of a neural network is one of the most important questions in deep learning. Unfortunately, while the fundamental procedures of training neural networks are well understood, being able to tell why one network is better at generalizing than another still poses a great challenge. Good performance of a deep neural network (DNN) depends fundamentally on its architecture and its neuron functions and parameters. These yield an approximation of the desired function (prediction or regression) based on neuron interactions —the better the approximation, the better the generalization. However, with the high quantity of neurons and connections of deep neural networks (sometimes of the order of millions), understanding which interactions between neurons are improving or damaging a model is a hard problem. Developing new mathematical tools that capture the effect of these interactions on the output of the networks is key for increased understanding of network generalization.

A DNN that generalizes will perform well on test data on which it has not been trained. This is usually measured by the generalization gap, which is defined as

the difference between the accuracy in training versus test datasets. Although the two accuracies are correlated to a certain extent, studying training performance alone can be misleading. Several papers show how neural network performances on unseen examples can differ with respect to their training performances due to many reasons [11, 180, 181]. *To what extent is it possible to predict the generalization gap without testing a model?* In a practical sense, a measure of generalization that does not require a test dataset eliminates the responsibility of maintaining and curating such a dataset.

The issue of finding a generalization measure has been explored extensively and a recent challenge, called the *Predicting Generalization in Deep Learning* (PGDL) challenge [182], on the topic provides an excellent framework for algorithmic benchmarking. However, the most competitive participant methods rely on internal representations of independent layers, discarding more global structures that may be created across the network [183, 184] or even discarding structure altogether [185].

In this chapter, we present an approach to predict the generalization gap from persistence diagrams based on neuron interactions in deep neural networks of any size for the networks and tasks of the challenge. For this purpose, we use the functional graphs introduced in [43] and described in Section 3.1. We extend the set of persistence summaries used in [43] to regress the generalization gap, demonstrating that an optimized combination of these summaries yields competitive results when predicting the generalization gap in the context of the challenge. Moreover, we show that persistence summaries separate neural network architectures into clusters related with their generalization capacity. The associated code for this section can be found at `https://github.com/rballeba/PredictingGeneralizationGapUsingPersistentHomology`.

4.1.1 Experiments and results

In the first part of this subsection, we describe our experimental setups and comment on computational complexity. In the second part, we evaluate our approach and discuss results.

Experiments

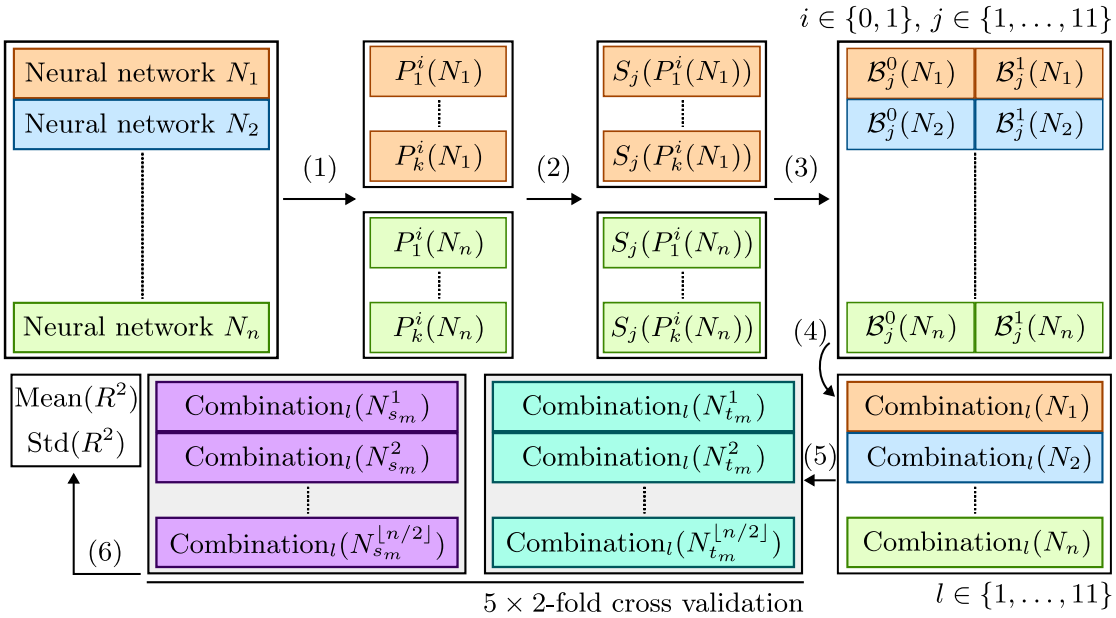


Figure 4.1: Experimental evaluation pipeline. Given a specific PGDL task as described in Section 4.1.1, let $\{N_i\}_{i=1}^n$ be the set of neural networks associated with the task. (1) Generation of k different persistence diagrams per DNN and dimension $i \in \{0,1\}$ using sampling in CIFAR10/SVHN datasets as described in Section 4.1.1. In our case, $k = 20$. (2) Computation of persistence summaries S_j for each persistence diagram. (3) Bootstrapping for each dimension and each summary computed from the same DNN. The bootstrapped summary S_j for dimension i and neural network N is denoted by $\mathcal{B}_j^i(N)$. (4) Generation of the eleven different combinations of bootstrapped persistence summaries described in the experimental procedure of Section 4.1.1. (5) A 2-fold cross-validation partition into sets with the same cardinality is calculated five times. Each time, for each combination of summaries $l \in \{1,\ldots,11\}$, two linear models to predict the generalization gap are trained on one of the partition sets and tested on the other, obtaining a R^2 score for each model on the test set. (6) We compute the mean and standard deviation of the resulting R^2 scores. Next, using the same partition sets, we train linear models with the generalization measures of the three winners of the PGDL competition, and we compare our best performing methods with their methods using 5×2-fold cross-validation statistical tests.

Datasets. The PGDL challenge dataset consists of eight different tasks, each composed of several neural network architectures trained to provide different generalization gaps on a particular dataset. In our experiments, we use only the first two tasks from the dataset, which were public when the competition was launched.

The first task consists of 96 VGG-like [186] neural networks, namely one for each combination of the following hyperparameters: each network is designed with

either two or six convolutional layers and one or two dense layers; either 256 or 512 filters in the last convolutional layer; a dropout probability, chosen to be 0 or 0.5; a number of convolutional blocks in each convolutional layer, either one or three; the weight decay during training, set up to be 0 or 0.001; and the batch size, varying between 9, 32, and 512. Each network was trained on the CIFAR10 dataset [187], consisting of 60,000 color images of size 32×32 split into ten classes that represent vehicles (airplanes, automobiles, ships and trucks) and animals (birds, cats, deers, dogs, frogs, and horses).

The second task is composed of 54 neural networks with *network in network* architectures [188], with a varying number of convolutional layers. Specifically, each network is chosen with either six, nine, or twelve convolutional layers and trained on the SVHN dataset [189], a digit classification benchmark dataset that contains 600,000 color 32×32 images of printed digits (from 0 to 9) cropped from pictures of house number plates. Other hyperparameters of Task 2 networks are the dropout probability, chosen among 0, 0.25, and 0.5; the weight decay, either 0 or 0.001; and the batch size, varying between 32, 512, and 1024.

Experimental procedure. Our experimental procedure is depicted in Figure 4.1, while the hyperparameters of our method are detailed as follows. Initially, we generate 20 distinct persistence diagrams for dimensions zero and one for each neural network, using the sampling methods outlined in Section 4.1.1. Subsequently, for each persistence diagram, we calculate persistence summaries as introduced in Section 2.1.11. This results in 20 distinct instances of each persistence summary for each network and homology dimension. Following this, we perform bootstrap analyses on each set of 20 values of persistence summaries derived from the same network and homology dimension. The bootstrapping process involves creating 1,000 bootstrap samples of size 20 selected with replacement from the various persistence summaries.

We combine bootstrapped persistence summaries to use them as predictor variables of the generalization gap with a linear regression for both tasks. The list

of persistence summaries that we test is the following: (1) Persistence pooling of 10 elements; (2) average lives and average midlives; (3) average births and average deaths; (4) average and standard deviation of births and deaths; (5) persistence entropy; (6) complex polynomials with 10 coefficients.

We also test concatenations of (2), (3) and (4) with their element-wise squared versions, and a concatenation of (3) with its element-wise logarithmic version, as well as a concatenation of (2) and (3), original and squared. All combinations are considered in homological dimension zero, homological dimension one, and a concatenation of both.

We compare our models with linear regressions trained from the three generalization measures that won the PGDL competition: BrAIn, Interpex, and Always Generalize.

Evaluation metrics. We train linear regression models with the previous combinations of persistence summaries and the three state-of-the-art generalization measures to predict the generalization gap of neural networks. To measure and compare their performance, we use a 5×2-fold cross-validation statistical test, as recommended in [190], with the coefficient of determination R^2 as performance metric. The coefficient of determination R^2 is computed as the proportion of the variation in the dependent variable that can be predicted from the independent variables, and it is calculated as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{4.1}$$

where y denotes the ordered set of actual values, $\hat{y}$ denotes the ordered set of predicted values, and $\bar{y}$ denotes the mean of y. This coefficient ranges from 0 to 1 in the training dataset but can be outside that range in unseen data. When the score is 1, the model perfectly predicts the values of y. A score of $R^2 = 0$ is obtained when one uses a horizontal line at the average of the set of y-values as a model. If a model performs worse than this (which usually indicates that the choice of model itself was ill-advised), then the numerator of (4.1) can grow arbitrarily large, and

thus R^2 can be negative. If an R^2 value is negative, then the prediction is worse than ignoring the input and predicting the average of the sample. This can actually happen when the training set yields a model that does not generalize in the test set.

The 5×2-fold cross-validation statistical test validates if there are significant differences between two models tested in a common dataset. The null hypothesis of this test is that, for a fixed-size random drawn training dataset, two learning algorithms have the same R^2 score on a randomly drawn test dataset. We compare linear models pairwise for each task.

Winners of the challenge. We compare our methodology to the three winners of the aforementioned challenge. The winners of the competition, the teams Interpex [184], Always Generalize [185], and BrAIn [183], presented generalization measures based on (1) data augmentation; and (2) on the analysis of intermediate representations. The Interpex team proposed a generalization measure based on neuroscience ideas that uses the Davies–Bouldin index [191] to quantify the consistency of internal representations of neural networks, the Always Generalize team proposed to measure the robustness of neural networks against data-augmented datasets, and the BrAIn team proposed a measure based on properties of a graph constructed from the internal representations of a neural network..

Reducing computational complexity

Computational complexity. Computing topological summaries with the complete set of activations calculated from the entire training dataset is unfeasible due to the high computational time and memory complexities of obtaining activation vectors and persistence diagrams. If $|\mathcal{D}|$ denotes the number of input samples for a dataset $\mathcal{D}$ and $|V|$ is the number of nodes in a neural network $\mathcal{N}$, then the set of activation vectors of nodes in $\mathcal{N}$ for the dataset $\mathcal{D}$ has cardinality $|\mathcal{D}| \times |V|$. Assuming that we have a standard current feedforward neural network like VGG16, that has about 8,000 neurons [192] only for fully connected layers, a standard dataset like CIFAR10 [187] with 50,000 training examples, and a double precision floating point format to represent each number, one would need at least 3 GB

only to store the activations of fully connected layers. Additionally, although zero dimensional persistent homology can be calculated in $\mathcal{O}\left(|V|^2 \cdot A^{-1}(|V|^2)\right)$ using the algorithm proposed in [26] where A^{-1} is the notoriously slowly growing inverse of the Ackermann function [193, Chapter 21], persistent homology in higher dimensions is harder to compute. The complexity of algorithms for computing persistent homology for dimension greater than or equal to one is $O(n^3)$ if n is the number of simplices of the Vietoris–Rips complex and Gaussian elimination is used to find ranks of matrices of boundary operators, or $O(n^\omega)$ where ω is the exponent of matrix multiplication (currently 2.37) if sparsity of boundary matrices is taken into account, as in [194]. In its turn, the number of simplices n depends cubically on the number $|V|$ of vertices of the functional graph if persistence diagrams are drawn only in homological dimension one, which requires determination of simplices up to dimension two.

In practice, this limits persistence diagram computations to a few thousand vertices. In order to alleviate these problems in neural networks with a large set of neurons, we introduce sampling strategies for both the input dataset and the functional graphs.

Sampling the input space. We compute activation vectors a'_v for a fixed subsample $\mathcal{D}' \subseteq \mathcal{D}$. In order to justify that this subsampling does not affect the results of the analysis, it is enough to verify that $\operatorname{corr}(a'_{v_i}, a'_{v_j})$ is sufficiently close to $\operatorname{corr}(a_{v_i}, a_{v_j})$ for all pair of vertices $\{v_i, v_j\}$ in the funtional graph, and that small variations in the correlation coefficients produce small changes in the persistence diagrams. This claim is justified by the fact that, if X and Y are random variables with non-null variance and X^n and Y^n denote sequences of n samples from X and Y respectively, then the sample correlation of X^n and Y^n converges in probability to the correlation between X and Y by the law of large numbers and the continuous mapping theorem [195].

In practice, $\mathcal{D}'$ is fixed to a uniform sample of 2,000 elements from the original training dataset, an experimentally selected size that is large enough to obtain sufficient precision.

Sampling the functional graph. Because of computational limitations, in the case of modern DNNs less than 1% of the nodes —a priori, a statistically insignificant sample size— can be included in the persistent homology calculation. To alleviate this, we sample nodes according to a notion of importance, following ideas introduced in [196] adapted to neurons on a neural network instead of inputs of the dataset. Thus, let $\mathcal{D}'$ be some selected subsample of the training dataset. The *importance score* of a node $v \in V$ is defined as

$$I_v(\mathcal{D}') = \left|\left\{x \in \mathcal{D}' : v = \arg\max_{w \in V} |\phi_{\mathcal{N}}^w(x)|\right\}\right|, \tag{4.2}$$

where $\phi_{\mathcal{N}}^w(x)$ is the activation of the vertex w of the neural network $\mathcal{N}$ for the example x, and arg max returns only one vertex in case of tie between multiple vertices —in our case, we use the tie breaking strategy implemented by the NumPy library [197]. Hence $I_v(\mathcal{D}')$ indicates the amount of inputs from $\mathcal{D}'$ for which the activation of v is the largest (or tied-to-largest) among all nodes. Note that a majority of nodes v will have $I_v(\mathcal{D}') = 0$. This is equivalent to excluding these nodes from analysis, which is undesirable —not only because it is unclear how this will affect the application of TDA, but also because the amount of nodes with $I_v(\mathcal{D}') \neq 0$ might be low enough to severely constrain the size of a subsample. Thus, from I we construct a probability distribution P on V, artificially inflated to make sure that every element of V appears with nonzero probability. This probability $P(v)$ is defined as

$$\frac{I_v(\mathcal{D}')}{|\mathcal{D}'|+1} \text{ if } I_v(\mathcal{D}') > 0, \text{ and } \frac{1}{(|\mathcal{D}'|+1)\cdot|\{u \in V : I_u(\mathcal{D}') = 0\}|} \text{ otherwise.} \tag{4.3}$$

Specifically, we sample 3,000 nodes (without repetition) according to this probability distribution, and restrict our analysis to these nodes. This sampling is non-deterministic, and thus can be repeated a number of times to obtain n different subsamples $V_1, \ldots, V_n$. Applying the same transformations on the n resulting functional graphs we obtain n different persistence diagrams per network. Then, we use bootstrapping over the n summaries (see Section 2.1.11) combining them into a single one. This last representation aims to approximate the persistence summary that would be obtained without sampling.

Table 4.1: Top three combinations of persistence summaries per task according to their respective mean of R^2 test values in the 10 experiments of the 5×2-fold cross-validation statistical test. **ASD:** Average and standard deviation of births and deaths. **ASDSQ:** Average and standard deviation of births and deaths, concatenated with the corresponding squared values; see Section 4.1.1.

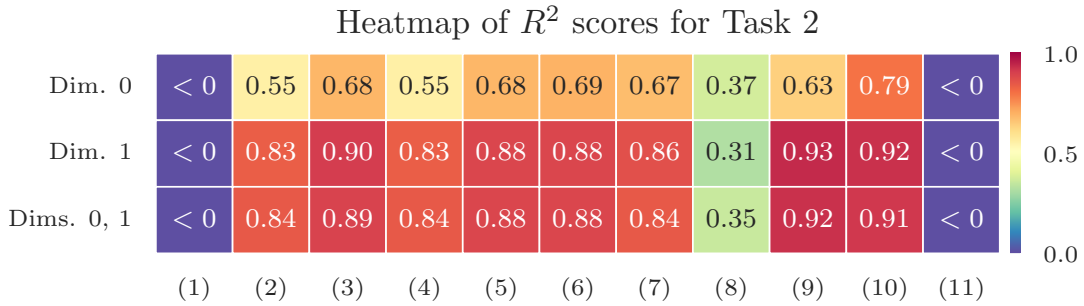
Task 1		
Top TDA summaries	Best dim	R^2 score
ASDSQ	0 and 1	0.5601 ± 0.13
ASDSQ	1	0.4321 ± 0.12
ASD	1	0.3720 ± 0.14
Task 2		
Top TDA summaries	Best dim	R^2 score
ASD	1	0.9337 ± 0.01
ASD	0 and 1	0.9198 ± 0.02
ASDSQ	1	0.9166 ± 0.03

4.1.2 Discussion

The combinations of persistence summaries that yielded the top three mean R^2 scores for the generalization gap prediction experiments are shown in Table 4.1. Basic statistical descriptors related to births and deaths of homology generators obtained highest scores overall, validating the results obtained in [198], in which simple vectorizations consisting of elementary statistical descriptors of persistence diagrams were the persistence summaries that obtained the best performances as input in a variety of image classification tasks. In particular, the vectors composed of averages and standard deviations of births and deaths (and their squares) were those that obtained the best R^2 scores in both tasks. Figure 4.2 shows the average performance of the entire list of summaries. These results suggest that the generalization gap is mostly linked with the average position and dispersion of points in persistence diagrams. Summaries based on alleged predominance of larger lifetime values, such as persistence entropy or persistence pooling vectors, showed a lower predictive value.

Overall, results are more conclusive for Task 2 than for Task 1, and more significant in homological dimension 1, although some of the best R^2 scores are

(a) Task 1 heatmap



(b) Task 2 heatmap

Figure 4.2: Mean R^2 test values after the 10 experiments of the 5×2-fold cross-validation statistical test for tasks 1 and 2 for the combinations of persistence summaries described in the experimental procedure of Section 4.1.1. Rows correspond to homological dimensions H_0, H_1, and a concatenation of both. Column numbers represent the following combinations of persistence summaries: (1) persistence pooling of 10 elements; (2) average lives and midlives; (3) average lives and midlives, original and squared; (4) average births and deaths; (5) average births and deaths, original and squared; (6) average births and deaths with a logarithmic model; (7) concatenation of combinations 3 and 5; (8) persistence entropy; (9) average and standard deviation of births and deaths; (10) average and standard deviation of births and deaths, original and squared; (11) complex polynomials with 10 coefficients.

achieved using a combination of dimensions 0 and 1 for both tasks. It should also be noticed that R^2 scores grow when squares of summaries are added to the model, suggesting departure from linearity.

Explainability. The distribution of points in persistence diagrams is determined by correlations between neuron activation vectors. Generators of the zero-homology group H_0 of a Vietoris–Rips simplicial complex at filtration level t correspond to connected components of a functional graph in which every edge has a weight smaller than or equal to t, hence a correlation coefficient of $1 - t$ in absolute value

among the neurons in the group. Hence, for $t = 0$ there is one generator for each group of neurons that share correlation coefficients equal to ± 1. Points $(0, d)$ in zero-dimensional persistence diagrams arise whenever two (or more) connected components merge in the filtration at time d, and therefore they correspond to non-zero edge weights of a minimum spanning tree of the network's functional graph. High weights in a minimum spanning tree imply that the overall correlations between neurons are low. The lower the correlation between neurons, the higher the number of nonlinearly related features learned by the neural network, and hence the stronger the real expressive power of the network. In conclusion, a combination of a high average of death values with a low standard deviation in a zero-dimensional persistence diagram,indicating lower activation correlations between neurons, is a plausible indication of an increased expressive power of the neural network, that should lead to better generalization capabilities and thus a smaller generalization gap.

Points in one-dimensional persistence diagrams correspond to cycles of the network's functional graph that are not filled by regions in the Vietoris–Rips complex. Thus a one-dimensional generator appears in the filtration at time t whenever there is a cyclically ordered group of neurons sharing correlations greater than or equal to $1 - t$ with their neighbours, which can be interpreted as a group of neurons that have learned similar features. The earlier a cycle is born, the higher the correlations among the neurons in the cycle, and the higher the death value of a cycle, the higher the differences between the features learned by non-neighbouring neurons in the cycle. Therefore, higher lifetime values may be associated with an increased number of different features learned by groups of jointly operating neurons. Thus, the higher the deaths in the one-dimensional persistence diagram of the functional graph of a neural network, the more expressive power the neural network may have, and thus the better it may generalize.

Clustering. The interpretations described in the previous subsection are consistent with what is shown in Figure 4.3. In this figure, each row represents a different

task, each column represents a different persistent summary, and each point in a cell corresponds to a neural network for the given task. The two rows, upper and lower, represent Task 1 and Task 2, respectively. The first and third columns represent the average of deaths of zero- and one-dimensional persistence diagrams, whereas the second and fourth columns represent the standard deviation of deaths of zero- and one-dimensional persistence diagrams, respectively. In the first and second rows, neural networks are clustered according to the number of convolutional blocks and the number of convolutional layers that each network contains.

Figure 4.3 suggests that persistence summaries detect very neatly the clusters of neural networks in each task. Naturally, the generalization gap is strongly influenced by the depth of the networks, which is almost determined by the number of convolution blocks and layers. When the number of convolutions is fixed, we see a consistent behavior: the higher the average deaths and the lower the standard deviations, the better the network's performance. This discovery has the potential of being used for network regularization. Support for this assertion was provided in [45] through the successful implementation of regularizers designed to raise the average number of deaths while reducing standard deviations.

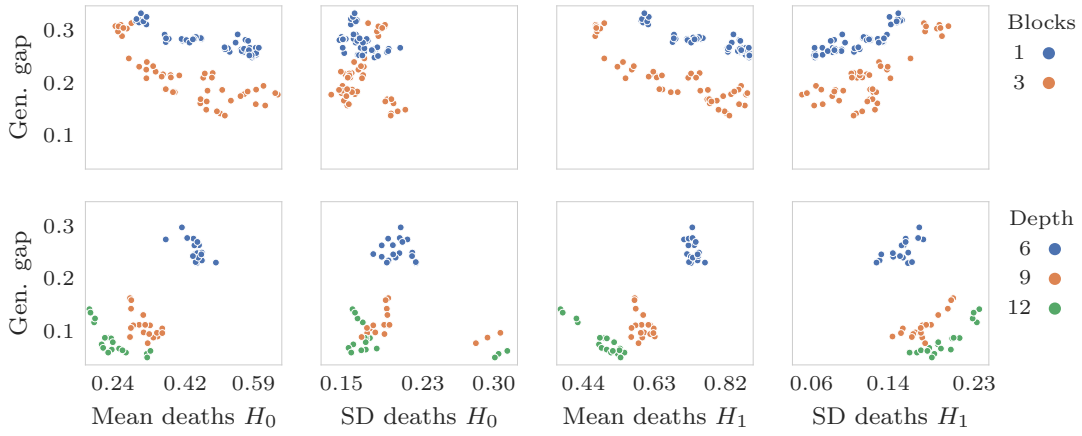


Figure 4.3: Averages and standard deviations of deaths for persistence diagrams in dimension 0 (first two columns) and dimension 1 (last two columns) for Task 1 (first row) and Task 2 (second row). For Task 1, points represent 96 VGG-like neural networks trained on the CIFAR10 dataset; blue and orange points represent neural networks with one and three convolutional blocks, respectively. For Task 2, points correspond to 54 *network in network* architectures trained on the SVHN dataset; blue, orange, and green points represent neural networks with six, nine, and twelve convolutional layers, respectively.

We further analyzed if persistence diagrams for individual labels in a classification task were different between them, in order to gain insight about what was influencing TDA methods and functional graphs the most. We computed persistence diagrams in dimensions 0 and 1 per different neural network and per label. The datasets used to recreate functional graphs were restrictions of the test set to each label. Details and figures can be found in Section A.1. Similar results were seen when comparing these persistence diagrams with the original ones. The majority of class-dependent persistence diagrams whose DNNs obtained extreme accuracies, i.e., highest and lowest, were analogous to the diagrams in the class-independent case. This shows that functional graphs are robust to unbalanced datasets in terms of the number of samples per label.

Table 4.2: Comparison of our best performing summaries with state of the art: Average and standard deviation of R^2 scores for Task 1 and Task 2 computed from linear models trained in the ten cases of the 5×2-fold cross-validation.

	Task 1	Task 2
Interpex	-0.0518 ± 0.06	0.9500 ± 0.01
Always Generalize	0.9715 ± 0.01	0.8893 ± 0.02
BrAIn	0.4520 ± 0.08	0.7180 ± 0.04
Ours	0.5601 ± 0.13	0.9337 ± 0.01

Table 4.3: Statistical p-values of the pairwise 5×2-fold cross-validation significance test proposed in [190], with the coefficient of determination R^2 as performance metric. The null hypothesis is that, for a fixed-size randomly drawn training dataset, the two linear models trained with our combinations of persistence summaries or with the winning generalization measures of the PGDL competition have the same R^2 score on a randomly drawn test dataset. Boldface p-values are lower than 0.05. **ASD1:** Average and standard deviation of births and deaths of dimension one. **ASDSQ01:** Average and standard deviation of births and deaths, concatenated with their squared values, for dimensions zero and one. Their R^2 scores are shown in Table 4.1.

Task 1	Interpex	Always Generalize	BrAIn	ASDSQ01
ASDSQ01	**0.00**	**0.01**	0.23	
ASD1	**0.03**	**0.00**	0.55	0.13
Task 2				
ASDSQ01	0.37	0.80	0.37	
ASD1	0.19	**0.00**	**0.01**	0.51

Persistence summaries. Results show that linear models of persistence summaries can predict the generalization gap, since we obtained competitive results in both tasks, as seen in Table 4.1 and Table 4.2. However, the fact that a summary based on a combination of non-linear transformations of persistence features yielded the best score for Task 1 suggests that more complex models can have better capacity to relate persistence summaries to the generalization gap.

When it comes to ranking summaries, persistence pooling and complex polynomials produced the lowest R^2 scores overall, as shown in Figure 4.2. For persistence pooling, one possible explanation of its low performance is that it relies on lifetimes of points that live the longest, in contrast to the most effective summaries, which are based on average location and dispersion of the whole set of points in a persistence diagram. Similarly, truncated complex polynomials are not sufficiently accurate measures of the location and aggregation of the collection of all points in persistence diagrams. The fact that persistence entropy achieves non-optimal R^2 scores for Task 2 in Figure 4.2 is consistent with the interpretation that the distribution of points near the diagonal in one-dimensional persistence diagrams is substantial for generalization gap prediction.

State-of-the-art comparison. Table 4.2 shows a comparison of the results of our best performing linear models based on persistence summaries with state-of-the-art methods. In this table, the R^2 scores describe the ability of each linear model to predict the generalization gap with respect to the coefficient of determination. Table 4.3 shows the pairwise p-values between the linear models induced by our best performing combinations of persistence summaries, shown in Table 4.1, and the linear models induced by the winning generalization measures of the PGDL dataset.

We obtain the second-best mean R^2 scores for both tasks, after Always Generalize and Interpex in the first and second ones, respectively. However, assuming that two methods are significantly different whenever their pairwise p-value is lower than 0.05 in Table 4.3, there is no significant difference between the R^2 scores of the linear models of our best combination of persistence summaries in Task 2 and the linear

models induced by the generalization measure of the Interpex team. Additionally, our models are significantly better than those for the Interpex generalization measure in Task 1 and than the ones for the generalization measures of Always Generalize and BrAIn in Task 2. These results suggest that persistence summaries are a promising tool to develop robust models to predict the generalization gap.

4.1.3 Conclusions

We have defined a framework that can be used to explore interpretability of DNNs based on topological properties of their functional graphs. This relaxes the problem of understanding the internal representations of a neural network to, in a broad sense, understanding their *shape*. Regarding generalization, we have shown examples of how one can interpret DNN neuron interactions based on their correlations by means of persistence diagrams. Moreover, we proved that the generalization gap can be consistently predicted using topological persistence summaries extracted from functional graphs, with a competitive prediction accuracy on two different computer vision problems. The most successful summaries were those related with the average location and dispersion of points in persistence diagrams. Hence, it is not true in our case that points near the diagonal in persistence diagrams are irrelevant, as often claimed in TDA studies.

Limitations. A practical limitation of persistent homology comes from its computational complexity —sampling methods are not necessarily optimal and information might be lost in sampling processes for datasets and for neurons. Transformations of persistence diagrams into summaries may also cause a loss of information; however, this seems unavoidable if one wants to obtain easy-to-compute generalization measures.

Future work. Although we found strong patterns relating persistence summaries with generalization gaps (Figure 4.3), broader experimentation is required to see if these patterns are consistent among other kinds of networks and machine learning

tasks, and also to make more explicit which features of the networks are involved in the TDA-driven clustering effect that we have observed.

The mere definition of functional graphs raises a question: which is the optimal metric to compare neurons given an architecture? There might be better alternatives to linear correlation between activation vectors; for instance, Spearman correlation was used in co-activation graphs for a similar purpose in [199].

Another problem is to find an optimal neuron sampling strategy. This is related to the problem of finding the most relevant neurons in a DNN graph. Persistence summaries suggest that grouping neurons in terms of their activation structure is feasible for DNNs. However, understanding which functional phenomena are being captured into such communities of nodes needs further study. This could lead to the discovery of new architectural properties useful to develop better networks.

Figure 4.3 shows that, fixing the depth of a neural network, there is a consistent association between a lower generalization gap and a higher average of death values together with a small dispersion in the persistence diagrams of the network's functional graph in dimensions zero and one. This finding has the potential to improve the performance of a given architecture during training by means of a regularization term that maximizes averages of deaths while minimizing standard deviation, using the framework for differential calculus on persistence diagrams discussed in [28, 29]. This work plan was undertaken in [45].

4.2 Decorrelating neurons using persistence

In the previous section, we have seen that correlations between activations play an important role in the generalization capabilities of neural networks. In particular, we have seen that topological summaries associated with lower activation correlations can be associated with an increased generalisation capacity of the network. Moreover, evidence from neuroscience indicates that correlation among human neurons is a significant factor in the brain's ability to encode and process information [200, 201]. This was pointed out in [202], where it was observed that the generalisation error of a deep network is monotonic with respect to the correlation between weight

matrices of neurons or filters, suggesting that decreasing this correlation can be beneficial to improve the generalisation capacity of a network. Furthermore, in [203], overfitting of neural networks was reduced by decorrelating their neuron activations. Overall, these studies suggest that a reduced correlation between neuron activations could improve the robustness of a network.

However, neuroscience also suggests that redundancy appears naturally in brain circuits and is useful to perform certain computations [204, 205]. For this reason, aggressively minimising correlations between all activations or weights may be detrimental for the performance of a neural network.

In this section, we propose a way to minimise only the most relevant high correlations between neurons using network functional graphs. Concretely, we propose two novel regularisation terms that minimise only some of the highest correlations of the most relevant neurons in a specific training batch based on an importance measure inspired by the activation criterion for neural network pruning presented in [206]. This approach allows for some redundancy in the neural network, avoiding possible detrimental effects of purely uncorrelated neurons.

To minimize the high correlations, both regularization terms maximize the weights of a minimum spanning tree of the network functional graph, that coincides with the death values of the persistence diagrams computed in Section 4.1, in a different way. Although both regularisation terms share the same objective, each of them outperforms the other in specific networks.

4.2.1 Methodology

The cut property of minimum spanning trees (MST) states that each MST of a graph contains, for each of its vertices, at least one edge with the minimum weight among its incident edges. In the case of functional graphs, their MSTs contain for each neuron $u \in V$ an incident edge to u with weight value

$$\min_{v \in V', u \neq v} (1 - |\mathrm{corr}_{\mathcal{D}}(u, v)|) = 1 - \max_{v \in V', u \neq v} |\mathrm{corr}_{\mathcal{D}}(u, v)|$$

among, possibly, other edges representing high correlations in the functional graph. By the equivalence of MSTs weights and zero-persistence diagrams, the latter contains at least the highest correlation achieved by each node plus a set with high correlation values among the functional graph vertices.

As in Section 4.1, computing MSTs of complete functional graphs, i.e., functional graphs containing all the vertices of the neural network, is not feasible, as computing a MST of a functional graph has a complexity of $\mathcal{O}(e \cdot \alpha(e, v))$ [207, Theorem 1.1], where $\alpha(e, v)$ is the functional inverse of Ackermann's function [208] and e and v are the number of edges and vertices, respectively, with $e = \binom{v}{2}$ because the graph is a clique. For this reason, we consider, as in Section 4.1, a subset $V_{\mathcal{B}}$ of neurons to compute the regularization terms for each batch $\mathcal{B} = \{(x_1, y_1), \ldots, (x_{|\mathcal{B}|}, y_{|\mathcal{B}|})\}$ during training. In particular, for the cases in which we cannot set $V_{\mathcal{B}} = V(G_{\mathcal{N}})$, we sample $V_{\mathcal{B}}$ using an *importance sampling algorithm*, different from the one presented in Section 4.1 for each batch. In this case, we take the top percentage P of most important neurons of each layer given the batch, except for the last layer, where we take all the neurons, where P is a hyperparameter depending on the size of the neural network. This is because the last layer showed to contain relevant information with respect to generalisation in other works like the one by [209]. For our experiments with very large neural networks, we set P to 0.5% due to practical hardware limitations.

The *importance* of a neuron given a batch is set to the average quantity of absolute activation achieved by the neuron, and it is inspired by the activation criterion for pruning presented in [206, Section 2.2]. The higher this value for a neuron is, the more relevance we allot to the neuron. More precisely, denote $\bar{\phi}^v_{\mathcal{N},\mathcal{B}} = |\mathcal{B}|^{-1} \sum_{i=1}^{|\mathcal{B}|} |\phi^v_{\mathcal{N}}(x_i)|$ and let $V_l = \{v_i^l\}_{i=1}^{n_l}$ be the neurons of the l-layer of $\mathcal{N}$ in any descending order of their $\bar{\phi}^v_{\mathcal{N},\mathcal{B}}$ values. In our case, the order is given by the `argsort` function of TensorFlow. We use

$$V_{\mathcal{B}} = \bigcup_l V_{\mathcal{B},l} \quad \text{where } V_{\mathcal{B},l} = \begin{cases} \{v_1, \ldots, v_{\lfloor 0.005\, n_l \rfloor}\} \text{ if } l \neq L, \\ V_l \text{ otherwise,} \end{cases}$$

where l iterates over all possible layers of $\mathcal{N}$ and L is the number of the last layer.

Since regularisation terms are minimised by network training algorithms, to maximise a function $f(\theta)$ we minimise its opposite function $-f(\theta)$. We propose two regularisation terms that maximise persistence diagram values in different ways:

$$\mathcal{T}_1(\theta) \triangleq -\sum_{y \in D(V_{\mathcal{B}})} y, \tag{4.4}$$

$$\mathcal{T}_2(\theta) \triangleq -\alpha \bar{D}(V_{\mathcal{B}}) + \beta \sigma(D(V_{\mathcal{B}})), \tag{4.5}$$

where $D(V_{\mathcal{B}}) = D(\mathrm{VR}^0(\mathfrak{F}_{\mathcal{N}}(V_{\mathcal{B}})))$ is the zero-dimensional Vietoris–Rips persistence diagram of the functional graph of $\mathcal{N}$ given the subset of vertices $V_{\mathcal{B}}$, $\alpha, \beta \in \mathbb{R}_{\geq 0}$ are weight parameters, θ is the set of parameters of the network, and

$$\bar{D}(V_{\mathcal{B}}) = \frac{1}{|D(V_{\mathcal{B}})|} \sum_{y \in D(V_{\mathcal{B}})} y, \qquad \sigma^2(D(V_{\mathcal{B}})) = \frac{1}{|D(V_{\mathcal{B}})|} \sum_{y \in D(V_{\mathcal{B}})} (y - \bar{D}(V_{\mathcal{B}}))^2$$

are the mean and variance of the values of $D(V_{\mathcal{B}})$. The regularisation term $\mathcal{T}_1$ given by Equation (4.4) maximises the sum of values in the persistence diagram, while the regularisation term $\mathcal{T}_2$ given by Equation (4.5) focuses on how the entries of the persistence diagram are distributed, minimising their dispersion and maximising their average value. In our case, we pick $\alpha = \beta = 1/2$ since we treat mean and dispersion with the same strength.

Theorem 1. Let $c, n \geq 2$ and let $d \colon \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ be $d(x, y) = 1 - |\mathrm{corr}(x, y)|$ where corr denotes correlation. There is an open dense subset $\mathfrak{D}_{c,n} \subseteq \mathbb{R}^{cn}$ such that the functions

$$\mathcal{T}_1(x_1, \ldots, x_c) \triangleq -\sum_{y \in D(X,d)} y, \qquad \mathcal{T}_2(x_1, \ldots, x_c) \triangleq -\alpha \bar{D}(X, d) + \beta \sigma(D(X, d))$$

are $\mathcal{C}^\infty$ on $\mathfrak{D}_{c,n}$ for all $\alpha, \beta \in \mathbb{R}_{\geq 0}$, where $X = \{x_1, \ldots, x_c\}$ and $\bar{D}(X, d)$ and $\sigma(D(X, d))$ denote average and standard deviation of the zero-dimensional persistence diagram $D(X, d)$.

A proof of this result is provided in Section A.2.2. By the chain rule, our regularisation terms are well defined as soon as the neuron activations of the set of neurons $V_{\mathcal{B}} = \{\nu_1, \ldots, \nu_c\}$ in the batch $\mathcal{B} = \{x_1, \ldots, x_n\}$ form a vector

$$\mathbf{v} = (\nu_1(x_1), \ldots, \nu_1(x_n)), \ldots, (\nu_c(x_1), \ldots, \nu_c(x_n)) \in \mathbb{R}^{cn}$$

such that $\mathbf{v} \in \mathfrak{D}_{c,n}$ and such that the neuron activations are obtained in a differentiable way. Experimentally, we need not control when this vector $\mathbf{v}$ is inside $\mathfrak{D}_{c,n}$ thanks to the fact that $\mathfrak{D}_{c,n}$ is a dense set. However, we note that ignoring points where non-differentiability may occur in the domain could introduce errors in some iterations during training, as it may also happen in fact with ReLU [210].

4.2.2 Experiments and results

We first describe the experimental setup that we use to validate the hypothesis stated in the previous section. This is done in Section 4.2.2. Then, we present and discuss the results in Section 4.2.2. [1].

Experimental setup

We perform two blocks of basic proof-of-concept experiments to demonstrate the plausibility of the hypotheses formulated in Section 4.2.1. For each block, we train several neural networks following a common architecture with different regularisation terms, including our proposed ones, and without regularisation terms. For the first block, we use multi-layer perceptron models whereas we use VGG-like models for the second one. The networks of the first block are trained in the MNIST dataset whereas the networks of the second one are trained in CIFAR-10. In both blocks, we explore the same set of weights for the regularisation terms. Finally, to compare the accuracies of our proposed regularisation terms to the other alternatives, we use the Friedman statistical test with its Nemenyi post-hoc. Further details of the experiments are provided through this section.

Multi-layer perceptron experiments. In the first block of experiments, we examine our regularisation terms in a simplified problem. We train three different multi-layer perceptron architectures with 1,000 hidden neurons, labelled `0`, `1`, and `2` using the MNIST dataset [211]. Networks `0` and `1` share the same fully connected architecture. However, network `1` is trained using dropout with a 50% probability

[1]The code used in the experiments of this chapter is available at `https://github.com/rballeba/DecorrelatingNeuronsUsingPersistence`

of dropping a hidden neuron at each iteration. Specifically, architectures `0` and `2` have a trapezium shape consisting of a sequence of hidden layers of 450, 350, and 200 neurons for the first network, and of 300, 250, 200, 150, and 100 for the second one, respectively. In this block of experiments, we do not aim to achieve state-of-the-art performance but to test our approach in a simple scenario where no sampling of neurons is needed to compute persistence diagrams.

PGDL experiments. In the second block of experiments, the objective is to see if the method scales well to more complex datasets and models. We train eight different neural network architectures from the PGDL dataset introduced in 4.1. The eight different neural network architectures we take belong to the first task, which is composed of VGG-like neural networks trained in the CIFAR10 dataset [187]. The architectures we selected are the ones that correspond to the numbers `20`, `21`, `22`, `23`, `148`, `149`, `150`, and `151` from the dataset. Architectures `20`, `21`, `148`, and `149` are the same as the architectures `22`, `23`, `150`, and `151`, but with a layerwise dropout probability of 0.5, respectively. The difference between models `22` and `23` is the width of their convolutions, where architecture `22` has convolution widths of 256 and architecture `23` has convolution widths of 512. Finally, the architectures `150` and `151` are the same as the architectures `22` and `23` but with one more dense layer.

Training procedures. In these experiments, we train different architectures with regularisation terms weighted with several values. To train the networks, we replicate approximately the training performed by the PGDL dataset used in the second block of experiments.

For both blocks of experiments we split the data into training, validation, and test datasets. For the MNIST dataset, we split the original training dataset into new training and validation datasets with 80% and 20% of the original data, respectively. Finally, we use the original test dataset for testing. For the CIFAR10 dataset, we split the original training dataset into new training and validation datasets, where we choose 1000 examples of each class randomly for the validation

dataset and we place the remaining examples into the training dataset. Again, we reuse the original test dataset.

For the training procedure, we train for a maximum of 1200 epochs with early stopping after 20 epochs without improvement in accuracy and with a batch size of 256. The algorithm used for training is the usual stochastic gradient descent (SGD) with momentum 0.9. For the first block of experiments, we use an adaptive learning rate $\alpha_i \triangleq \alpha_0 \cdot (0.95)^{i/3520}$ where i is the iteration where the learning rate α_i is used and $\alpha_0 = 0.01$. For the second block of experiments, we use a fixed learning rate of 0.001, for which we obtained similar accuracies to the ones given by the original trainings of the PGDL neural networks.

Let $\mathrm{CCE}(\theta)$ denote the categorical cross entropy loss for a fixed neural network, clear from the context, with a set of parameters θ. For each of the networks described before we perform several trainings with the different regularisation terms that we study, weighed by different values. In particular, each training minimises a loss function

$$\mathcal{L}(\theta) = \mathrm{CCE}(\theta) + \omega\mathcal{R}(\theta), \tag{4.6}$$

where $\omega \in \{10^{-6}, 10^{-5}, 10^{-4}, 0.001, 0.01, 0.1, 1.5, 10, 100\}$ represents one of the possible weight values used in the experiments, and $\mathcal{R}(\theta)$ represents one of the regularisation terms. We also train the models without any regularisation term, i.e., with $\omega = 0$.

We use the full and sampled version of our regularisers $\mathcal{T}_1$ and $\mathcal{T}_2$ for the first and second blocks of experiments, respectively, due to the small size of the networks of the first block and to the large size of the networks of the second one. To see if reducing only some correlations between neurons is better than minimising all of them, we also study the regularisation term

$$\mathcal{C}(\theta) = \frac{1}{|\mathfrak{C}|} \sum_{(x,y)\in\mathfrak{C}} |\mathrm{corr}_{\mathcal{B}}(x, y)|, \tag{4.7}$$

where $\mathfrak{C} = \{(x, y) \in V_{\mathcal{B}} \times V_{\mathcal{B}} : x \neq y \text{ and } \mathrm{corr}_{\mathcal{B}}(x, y) \neq 0\}$, and $V_{\mathcal{B}}$ is defined as in Section 4.2.1. Note that for the first block of experiments we consider all the

non-input neurons and for the second block of experiments we perform the same sample of neurons due to the complexity of computing all the possible pairwise correlations for each iteration of the training. Finally, we also train the networks with l_1 and l_2 regularisation terms [212].

Evaluation procedure. To evaluate the performances of the regularisation terms compared, we use a Friedman test with the Nemenyi post-hoc, as proposed in [213], and we report the test accuracies for each regularisation term and network. To obtain test accuracies, we choose, for each term and network, the weight that maximises the validation accuracy after training. Then, we compute the test accuracy using the selected weight. For the training procedures without regularisation terms, we compute test accuracies directly.

Results and analysis

Table 4.4 contains test accuracies for all the networks and regularisation terms studied in our experiments. Each cell shows the test accuracy of the regularisation term with the weight that obtained the best validation accuracy in its column. The Friedman test with null hypothesis that all the algorithms are equivalent [213] gives a p-value of 0.00001, so we reject the null hypothesis. Therefore, we perform a Nemenyi post-hoc test, obtaining the p-value matrix shown in Table 4.5. The null hypothesis of this test is that there is no difference between the accuracies yielded by the two training approaches. A critical difference diagram for the Friedman and Nemeny statistical tests is shown in Section A.2.3.

When comparing test accuracies individually, $\mathcal{T}_1$ and $\mathcal{T}_2$ outperform the other training methods for all the networks except for model `0`. In fact, $\mathcal{T}_1$ and $\mathcal{T}_2$ obtain the best test accuracies in five of the networks each, respectively, out of eleven total networks. Both of them are significantly better than training without regularisation term, according to the Nemenyi p-value matrix, showing p-values of 0.018 and 0.002 for $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

Regarding the differences between minimising all the correlations and minimising only the highest ones, we see that $\mathcal{T}_1$ and $\mathcal{T}_2$ obtain low p-values against $\mathcal{C}$.

Table 4.4: Test accuracies for different training procedures and networks. Each row represents training with a regularisation term except for the first row, that represents no regularisation. $\mathcal{T}_1$: First topological regularisation term (4.4); $\mathcal{T}_2$: Second topological regularisation term (4.5); l_1: Lasso regression regularisation term; l_2: Ridge regression regularisation term; $\mathcal{C}$: Regularisation term minimising all pairwise correlations in $V_{\mathcal{B}}$ (4.7); $\emptyset$: no regulariser. Each column corresponds to a network in the PGDL dataset. Best accuracies per network are bolded.

	MNIST and MLP			PGDL and VGG-like							
	0	1	2	20	21	22	23	148	149	150	151
$\emptyset$	**0.929**	0.501	0.636	0.681	0.680	0.685	0.682	0.672	0.677	0.675	0.680
$\mathcal{T}_1$	0.928	**0.547**	**0.883**	0.687	**0.705**	0.675	0.700	0.688	**0.704**	0.678	**0.698**
$\mathcal{T}_2$	0.923	0.540	0.879	**0.691**	0.701	**0.688**	**0.706**	**0.689**	0.698	**0.688**	0.695
l_1	0.914	0.536	0.870	0.682	0.680	0.682	0.683	0.677	0.675	0.685	0.678
l_2	0.919	0.531	0.878	0.681	0.688	0.686	0.683	0.680	0.680	0.681	0.679
$\mathcal{C}$	0.923	0.530	0.881	0.679	0.687	0.680	0.686	0.678	0.690	0.683	0.674

Table 4.5: Nemenyi *p*-value matrix. Cells contain p-values of the Nemenyi post-hoc test (p-values < 0.05 are bolded). The meaning of column labels is specified in Table 4.4.

	$\emptyset$	$\mathcal{T}_1$	$\mathcal{T}_2$	l_1	l_2	$\mathcal{C}$
$\emptyset$		**0.018**	**0.002**	0.900	0.785	0.900
$\mathcal{T}_1$			0.900	**0.036**	0.380	0.159
$\mathcal{T}_2$				**0.006**	0.122	**0.036**
l_1					0.900	0.900
l_2						0.900

In particular, for $\mathcal{T}_2$ and $\mathcal{C}$ we obtain a p-value of 0.036, that validates the hypothesis that minimising only the highest correlations is significantly different than minimising all the correlations for a sampled set of relevant neurons in our experiments.

As for classical regularisation terms, both $\mathcal{T}_1$ and $\mathcal{T}_2$ obtain p-values lower than or equal to 0.05 with respect to l_1, making our regularisation terms better than l_1. The p-values of $\mathcal{T}_1$ and $\mathcal{T}_2$ with respect to l_2 are lower than the other p-values for l_2, although not as low as for l_1. This, together with the fact that l_2 never obtains the best accuracies, suggests that our regularisers are better than l_2, although more experiments are needed to confirm this claim. Overall $\mathcal{T}_1$ and $\mathcal{T}_2$ perform better than other training approaches in our experiments.

4.2.3 Limitations and future work

The computational cost of determining persistence diagrams can pose limitations in practical scenarios. To extend the applicability of our regularisation terms, improvements to the algorithms for computing zero-dimensional persistence should be developed. For example, advances in the implementation of these algorithms in a distributed manner, as discussed in [214, 215], could be relevant.

A crucial bottleneck in our pipeline is the computation of pairwise correlations. With a large number of neurons, it is impractical to compute correlations for all neurons at each training step. The effect of selecting a range of different percentages of neurons for sampling in large networks should be tested, since neuron selection may greatly impact the performance of regularisation terms. To enhance the computational efficiency of regularisation focused on high correlations, one approach could involve implementing weighted dropout based on the average correlation of each neuron with others, discarding at each step of the training procedure the nodes with lowest overall average correlation. Selectively training only neurons with higher average correlations may yield a positive effect by forcing them to learn independent functions.

We emphasise that our experiments were performed in simple tasks as a proof of concept of our methodology, and that further work is needed to understand how correlations affect generalisation capacity, especially in complex scenarios. For example, [216] found that their regulariser was more effective in suboptimal hyperparameter settings. A similar phenomenon might occur with our regularisers.

As correlations within neurons are independent of dataset labels, our approach is well-suited for application in unsupervised or semi-supervised learning.

4.2.4 Conclusions

In this section, we introduced regularisation terms that minimise high correlations between the most important neurons given a training batch, by maximising the values of their zero-dimensional persistence computed with the dissimilarity function $d(u, v) = 1 - |\text{corr}_{\mathcal{D}}(u, v)|$. The use of persistent homology in [42, 43] and in the

previous section was intended to assess trained models as a post-hoc method. The present article adds evidence that the association between activation correlations and generalisation gap can be exploited to build regularisers with the aim of enhancing generalisation.

Our regularisation terms outperformed classical regularisation terms and improved performance compared to minimising all pairwise correlations of important neurons in the MNIST and CIFAR10 datasets with MLP and VGG-like architectures, respectively. These findings support the hypothesis that neuron correlations play a role in the generalisation capacity of neural networks, consistently with previous studies such as [202, 203]. Additionally, we demonstrated that, when minimising higher correlations using persistent homology, several loss functions that are used with the same objective can yield different performances, suggesting that, for differentiable persistence descriptors, the choice of a loss function is a crucial step in the process.

Our results also show that topological regularisation terms can be used to improve the performance of neural networks not only by considering the final representations of the data, but also by looking at intermediate representations as well. In summary, our findings provide insight into the relevance of topological data analysis and neuron correlations on the generalisation capacity of neural networks, as well as their potential for future advances.

5. Expressivity of TDA-based graph networks

Contents

Graph learning is a highly-active research domain in machine learning, fuelled in large parts by the *geometric deep learning* [14, 217] paradigm as well as the resurgence of new neural network architectures for handling graph data. Methods from computational topology, by contrast, have not yet been applied in this domain at large scales. Even though a large amount of prior work employs topological features to solve graph learning tasks [98, 173, 218–226], a formal investigation relating expressivity in graph learning and topological machine learning is still lacking.[1] This chapter shows new theoretical and empirical results about the *expressivity* of topological graph learning methods. Here, we understand expressivity as a general concept to signify which graph properties can be captured by a method. This includes being aware of certain substructures in graphs [227], for instance, but also being able to distinguish large classes of non-isomorphic graphs [228–230]. While graph neural networks have demonstrated substantial gains in this area, our paper focuses on topology-based algorithms, aiming to provide a better understanding of their theoretical and empirical properties.

[1] A notable exception is recent work by Immonen, Souza, and Garg [174], which analyses the expressivity of topological methods for graph classification tasks.

The main contribution of the chapter is a full characterisation of the expressivity of *persistent homology* in terms of the Weisfeiler–Leman hierarchy [107, 108]. We prove that persistent homology is *at least as expressive* as a corresponding Weisfeiler–Leman test for graph isomorphism. Moreover, we show that there exist graphs that cannot be distinguished using k-FWL, the *folklore Weisfeiler–Leman algorithm* [106], for a specific dimension k but that can be distinguished by persistent homology (with or without access to k-cliques in the graph). Along the way, we also prove new properties of filtrations hinting at their ability to capture information about graph substructures. Finally, we complement our theoretical expressivity discussions by an experimental suite that highlights the capabilities of different filtrations for (1) distinguishing certain types of graphs, (2) predicting characteristic graph properties, and (3) serving as a baseline for classification tasks.[2]

5.1 Properties of filtrations

Filtrations for simplicial complexes are usually obtained from general maps $F\colon K \in \mathcal{K} \mapsto f_K \in \mathbb{R}^K$ where $\mathcal{K}$ is the set of all finite simplicial complexes. In the scenarios where we construct simplicial complexes from graphs, we can promote F to a function $\mathcal{F}\colon G \in \mathcal{G} \mapsto (K_G, f_{K_G}) \in \mathcal{K} \times \mathbb{R}^{\mathcal{K}}$ that assigns to each graph a simplicial complex with vertex set equal to the set of vertices of the graph and a filtration function for this simplicial complex. We say that $\mathcal{F}$ is *equivariant* if for each isomorphism $\varphi\colon G \to G'$ we have that φ induces a simplicial complex isomorphism between K_G and $K_{G'}$ and $f_{K_G} = f_{K_{G'}} \circ \varphi$. If $K_G = G$ for all $G \in \mathcal{G}$, then we recover the previous definition of equivariance for the graph case. A crucial result of the previous maps $\mathcal{F}$ is that, if they are equivariant, then the persistence diagrams of the filtrations they induce are invariant under graph isomorphism.

Proposition 2. For $\mathcal{F}$ equivariant and two isomorphic graphs G and G', the persistence diagrams of f_{K_G} and $f_{K_{G'}}$ coincide.

[2]The code of this chapter is available at `https://github.com/aidos-lab/PH_expressivity`.

Proposition 2 shows that it is impossible to 'adversarially' pick an equivariant filtration generator function $\mathcal{F}$ that leads to a dissimilarity between two non-isomorphic graphs. Dropping this condition incurs a substantial loss of expressive power. The proposition thus guarantees that persistent homology is compatible with equivariant function learning, pointing towards the utility of hybrid models that leverage different types of structural properties of graphs. To finish this discussion of general properties, we remark that the calculation of persistent homology carries information about the *diameter* (the length of the longest shortest path) and *girth* (the length of the shortest cycle) of a graph.

Proposition 3. Given *any* filtration f of a graph G with a single connected component such that the values of f of the endpoints of edges are *strictly lower* than the values of their corresponding edges, Algorithm 1, used to compute β_0, yields an upper bound of $\operatorname{diam}(G)$.

Proposition 4. Given *any* filtration f of a graph G, Algorithm 1 yields an upper bound of the girth of G.

While the theoretical upper bounds are not tight, our empirical analysis in Section 5.3.3 shows that persistent homology and its algorithms captures more than 'just' topological information about a graph. Our work thus corroborates recent results in the setting of point clouds, where persistence diagrams permit inferring additional properties about the input data [231–233].

5.2 The Weisfeiler–Leman hierarchy

Having discussed the properties of specific filtrations, we now analyse the expressivity of persistent homology in the context of the Weisfeiler–Leman hierarchy of graph isomorphism tests. To this end, we categorise our filtrations (and their corresponding generators) into two distinct classes:

(1) Filtrations operating on clique complexes derived from graphs;
(2) Filtrations that consider all possible cliques of the vertex set up to a specified maximum dimension.

In the first class, we incorporate only those cliques present in the original graph, constrained to a predetermined maximum dimension. Conversely, in the second class, we include *all* feasible cliques up to a specified dimension, irrespective of their presence in the original graph structure.

As a generalisation of previous work [98], we can show that *any* k-FWL colouring can be reproduced with one equivariant filtration of type (2), thus proving that persistent homology is *at least* as expressive as k-FWL. We achieve this by showing that zero-dimensional persistence diagrams can *recover* any isomorphism-invariant real-valued function $f\colon \mathcal{G} \to \mathbb{R}$ on the space of finite graphs by setting, for a given graph G, a constant filtration with value equal to the function value in G, i.e., $f(G)$.

Theorem 5. For $k \geq 2$, there exists an equivariant filtration generator $\mathcal{F}$ of type (2) such that its zero-dimensional persistence diagrams are at least as expressive as k-FWL.

Both Theorem 5 and Theorem 11 may not be completely satisfactory because they only show the *existence* of such a filtration, but make no claims about the theoretical expressivity of filtrations of type (1) filtrations. Hence, it is particularly interesting to understand the expressivity of the Vietoris–Rips filtration [234], which is a popular choice in the context of persistent homology. Given a graph $G = (V, E)$ equipped with the shortest-path distance d_G, the Vietoris–Rips filtration generator yields the pair $(K, f_{\mathbf{V}})$, where K is the set of all non-empty subsets σ of V such that $\operatorname{diam}(\sigma) \neq \infty$ and $f_{\mathbf{V}}$ is the Vietoris–Rips filtration function given by $f_{\mathbf{V}}(\sigma) = \max_{u,v\in\sigma} d_G(u, v)$. Notice that the Vietoris–Rips filtration generator is equivariant. This is because graph isomorphisms preserve distances between vertices, making the induced simplicial morphisms preserve diameters and filtration values, and inducing an simplicial complex isomorphism.

Vietoris–Rips persistent homology cannot be directly compared to the WL hierarchy, as no method is at least as expressive as the other. For example, Vietoris–Rips persistence diagrams of dimension zero can capture the number of connected components of graphs, while the 1-WL hierarchy cannot always capture them. See Theorem 14 for a proof of this fact. By contrast, the 1-WL test can capture the

difference between a path graph and a cycle graph both of length 3, while Vietoris–Rips persistent homology cannot. The fact that both methods are not comparable in terms of expressivity suggests that persistent homology methods based on Vietoris–Rips filtrations and WL/message passing methods are complementary, since many standard graph neural networks are exactly as expressive as 1-WL [107, Theorem 2]. Regarding filtrations of Type (1), we do not expect cliques and high-dimensional persistent homology to be crucial in distinguishing pairs of non-isomorphic graphs. On the one hand, equivariant graph filtration generators of type (1) are already *at least as expressive* as 1-WL, thus they can distinguish almost all non-isomorphic graph pairs [235, Theorem 3.3]. On the other hand, to prove that persistent homology for filtrations of type (1) is at least as expressive as k-FWL for $k \geq 2$, we would need to distinguish non-isomorphic CFI pairs for all $k \geq 2$. However, by definition of CFI graphs, they do not contain cliques of size three or greater, see Lemma 12, meaning that most of the expressive power of these filtrations can only arise from the values in the original graph structure and low-dimensional persistent homology. We would ideally want to extend Theorem 5 to state that persistent homology is *strictly* more expressive than k-FWL, although this is not as straightforward as for the case $k = 1$. Currently, we can provide one such counterexample, described in Table 5.1, consisting of the 4×4 rook's graph and the Shrikhande graph. With an appropriate filtration, persistent homology can distinguish these two graphs *without* requiring more than vertices and edges, whereas 2-FWL is unable to distinguish them. We leave a general result for future work.

5.3 Experiments

The previous sections discussed the theoretical properties of filtrations. Here, we analyse their *empirical* performance as a complement to the theoretical discussion to demonstrate the high expressivity of persistent homology in graph tasks. We first analyse the *expressivity* of five different well-known filtrations by letting them distinguish non-isomorphic graphs. This is followed by experiments on graph

property prediction. Please refer to Sections B.4 and B.5 for additional expressivity and graph-classification experiments.

Experimental setup. We use different data sets containing strongly-regular graphs [236], minimal Cayley graphs, as well as benchmark data sets for graph-learning tasks [237, BREC]. In the following, we will use five different filtrations for each graph:

(1) A *degree* filtration (denoted by **D**), i.e., $v \mapsto \deg(v)$. The degree filtration is the most basic non-trivial filtration of a graph, showing nevertheless surprising empirical performance in graph classification tasks [222, 238, 239].

(2) A filtration based on the eigenvalues of the *undirected graph Laplacian* (denoted by **L**), i.e., $v \mapsto \lambda_v$, where λ_v indicates the eigenvalue of the undirected graph Laplacian corresponding to vertex v. The graph Laplacian is known to capture characteristic properties of a graph; in the context of persistent homology it is often used in the form of a *heat kernel signature* [218].

(3) A filtration based on the *Ollivier–Ricci curvature* [240] (denoted by O) in the graph, setting $v \mapsto -1$ and $(u, v) \mapsto \kappa(u, v)$, with κ denoting the Ollivier–Ricci curvature

$$\kappa(u, v) := 1 - \mathrm{W}_1(\mu_u^\alpha, \mu_v^\alpha), \tag{5.1}$$

where W_1 denotes the first Wasserstein distance,[3] and $\mu_u^\alpha, \mu_v^\alpha$ denote *probability measures* based on a lazy random walk in the graph, i.e., $\mu_u^\alpha(u) := \alpha$, indicating the probability of staying at the same vertex, $\mu_u^\alpha(v) := (1-\alpha)^{1}/_{\deg(u)}$ for a neighbour v of u, and $\mu_u^\alpha(\cdot) := 0$ otherwise. The probability measures in Equation (5.1), i.e., μ_u^α, may be adjusted; recent work investigates the utility of this perspective [243, 244]. We set $\alpha = 0$ for our subsequent experiments, thus obtaining a non-lazy random walk, and leave the investigation of the impact of other values for future work.

[3] This metric is also known as the *Earth Mover's Distance* [241]. The Wasserstein distance is a fundamental concept in optimal transport; the monograph by Villani [242] contains a comprehensive introduction to this topic.

(4) A filtration based on the *augmented Forman–Ricci curvature* [245] (denoted by $\mathbf{F}$), where we again set $v \mapsto -1$ and $(u, v) \mapsto \mathcal{F}(u, v)$, with $\mathcal{F}(u, v) := 4 - \deg(u) - \deg(v) + 3\,|\mathcal{N}(u) \cap \mathcal{N}(v)|$.

(5) A *Vietoris–Rips* filtration (denoted by $\mathbf{V}$) over the metric space defined by the shortest-path distance between its nodes [234].

We exclude colouring filtrations, as used in Theorem 5, from our analysis due to their practical infeasibility. For each input graph G, this approach would require generating $k!$ filtrations and constructing simplicial complexes containing all cliques up to dimension k. After picking a filtration (except for the Vietoris–Rips one), we expand the graph by filling in all $(k+1)$-cliques, with filtration value for a clique σ given recursively by the maximum filtration value of its proper subcliques, i.e., $\sigma \mapsto \max_{\tau \subsetneq \sigma} f(\tau)$, and calculating persistent homology up to dimension k. Hence, for $k = 1$, we leave the graph 'as-is,' making use of connected components and cycles only.[4] Our persistent homology calculations result in a set of persistence diagrams for each graph, which we compare in a pairwise manner using the bottleneck distance described by Equation (2.5). We consider two graphs to be different whenever the distance between their persistence diagrams is $> 1 \times 10^{-8}$, i.e., above machine precision. This setup has the advantage that no additional classifier is required; when dealing with data sets of pairs of non-isomorphic graphs, we may thus simple *count* the number of non-zero distance pairs, showing the utility of persistent homology as a powerful baseline for graph expressivity analysis. We omit comparisons between persistent homology and isolated filtration values, as persistent homology is at least as expressive as filtration values alone. This is a consequence of the *pairing lemma*, which ensures all filtration values, with multiplicities, are distributed as births or deaths in persistence diagrams up to the dimension of the simplicial complex. Hence, the filtration values can always be reconstructed from the persistence diagrams.

Table 5.1: Success rate (↑) for distinguishing pairs of *strongly-regular graphs* when using different filtrations at varying expansion levels of the graph (denoted by k). 2-FWL cannot distinguish between any of these pairs.

Data	$k=1$					$k=2$					$k=3$				
	Filtration														
	D	O	F	L	V	D	O	F	L	V	D	O	F	L	V
16622	0.00	**1.00**	0.00	0.00	0.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
251256	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**
261034	0.00	0.00	0.00	0.00	0.00	0.20	0.20	0.20	0.76	0.20	0.93	0.93	0.93	**0.98**	0.93
281264	0.00	0.83	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
291467	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**
351668	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.95	0.95	0.95	**0.99**	0.95
351899	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**
361446	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.02	0.83	0.02	0.92	0.92	0.92	**0.99**	0.92
401224	0.00	0.00	0.00	0.00	0.00	0.93	0.93	0.93	0.99	0.93	0.94	0.94	0.94	**0.99**	0.94

5.3.1 Strongly-regular graphs and minimal Cayley graphs

We start our investigation by analysing *strongly-regular graphs*, which are are known to be extremely challenging to distinguish. 2-FWL, for instance, cannot distinguish *any* of these graphs [106, Section 3.3]. Table 5.1 summarises the performance of our selected filtrations. We first observe that for $k = 1$, i.e., for the original graph without any cliques, few pairs of graphs can be distinguished by the five filtrations. Notably, a curvature-based filtration *is* sufficient to distinguish the two graphs in the `16622` data set, colloquially known as the 4×4 rook's graph and the Shrikhande graph. Distinguishing between these two graphs is usually said to require knowledge about cliques [159], but it turns out that a suitable filtration is sufficient. However, the empirical expressivity of curvature-based filtrations appears limited for $k = 1$, improving only for higher-order clique complexes. The Laplacian filtration, by contrast, exhibits strong empirical performance for $k = 2$ on almost half of the data sets, increasing to near-perfect performance for $k = 3$ in almost all data sets. Vietoris–Rips and degree filtrations obtain the exact same success rates for every k and data set, exhibiting the lowest success rates for $k = 1$ and $k = 2$, and the same ones to the curvature filtrations for $k = 3$. It is clear that

[4]Notice the shift in dimension: a k-simplex has $k+1$ vertices, meaning that persistent homology in dimension k contains information about $(k + 1)$-cliques.

Table 5.2: Success rate (↑) for distinguishing pairs of *minimal Cayley graphs* when using five different filtrations at varying expansion levels of the graph (denoted by k). Values for 1-WL are shown as a baseline.

Data		$k=1$					$k=2$				
		Filtration									
	1-WL	D	O	F	L	V	D	O	F	L	V
`cay12`	0.67	0.67	0.71	0.95	0.86	0.00	0.95	0.95	0.95	**1.00**	0.95
`cay16`	0.83	0.83	0.42	0.83	0.58	0.00	0.83	0.92	0.83	**1.00**	0.94
`cay20`	0.61	0.61	0.46	0.61	0.79	0.00	0.61	0.79	0.61	**1.00**	0.89
`cay24`	0.65	0.65	0.82	0.86	0.98	0.00	0.83	0.93	0.86	**1.00**	0.93
`cay32`	0.76	0.76	0.81	0.76	0.90	0.00	0.76	0.94	0.76	**1.00**	0.90
`cay36`	0.69	0.69	0.87	0.84	0.99	0.00	0.84	0.95	0.84	**1.00**	0.94
`cay60`	0.69	0.69	0.90	0.78	1.00	0.00	0.77	0.95	0.78	**1.00**	0.97
`cay63`	0.49	0.49	0.89	0.73	0.88	0.00	0.73	0.93	0.73	**1.00**	0.96

knowledge about higher-order cliques helps in driving performance here. Notice that in contrast to other algorithms [159], no additional embedding of the graphs is required; we are comparing 'raw' persistence diagrams directly.

As an additional class of complex graphs, we analyse *minimal Cayley graphs*, i.e., Cayley graphs that encode a group with a minimal generating set. Minimal Cayley graphs are still a topic of active research in graph theory, with several conjectures yet to be proven [246, 247]. Specifically, isomorphisms of Cayley graphs have been extensively studied [248–251] and can be associated to isomorphisms of subsets of groups and interesting questions about their structure; see [248, Section 3]. We follow the same experimental setup as described above but also show the performance of 1-WL, calculated via a subtree Weisfeiler–Leman graph kernel [252]. Table 5.2 shows the results. We observe that the Laplacian filtration is trivially able to distinguish between all these graphs for $k = 2$ and has a strong performance for $k = 1$; this is not surprising since spectra of graphs are known to be characteristic [253]. The performance of the curvature filtrations also points towards the utility of this formulation in practice. We also observe that Vietoris–Rips filtrations are the ones that most benefit from the availability of higher-order information, with a significant improvement in performance, going from a 0%

Table 5.3: Success rate (↑) for distinguishing pairs of instances of the *BREC data set* when using different filtrations at varying expansion levels of the graph (denoted k). Due to combinatorial constraints, we did not calculate the Vietoris–Rips filtration for $k = 4$. Legend and number of graphs per category: `B` (Basic, 60), `R` (Regular, 100), `E` (Extension, 100), `C` (CFI, 100), `4, 20` (4-Vertex Condition), `D` (Distance-Regular, 20) graphs, respectively and `A` (average over full data set, 400 graphs).

	$k=1$					$k=2$					$k=3$					$k=4$			
Data	*Filtration*																		
	D	O	F	L	V	D	O	F	L	V	D	O	F	L	V	D	O	F	L
B	0.03	0.93	0.87	**1.00**	0.00	0.78	**1.00**	0.98	**1.00**	0.52	0.83	**1.00**	0.98	**1.00**	0.58	0.83	**1.00**	0.98	**1.00**
R	0.00	0.42	0.32	0.00	0.00	0.39	0.54	0.50	0.48	0.39	0.85	0.93	0.91	0.93	0.85	0.89	**0.97**	0.95	**0.97**
E	0.07	0.76	0.44	0.94	0.00	0.26	0.92	0.59	**1.00**	0.11	0.29	0.92	0.59	**1.00**	0.16	0.29	0.92	0.59	**1.00**
C	0.03	0.03	0.03	**0.06**	0.03	0.03	0.03	0.03	**0.06**	0.03	0.03	0.03	0.03	**0.06**	0.03	0.03	0.03	0.03	0.06
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
D	0.00	0.00	0.00	**0.05**	0.00	0.00	0.00	0.00	**0.05**	0.00	0.00	0.00	0.00	**0.05**	**0.05**	0.00	0.00	0.00	0.05
A	0.03	0.44	0.33	0.40	0.01	0.29	0.52	0.43	0.54	0.21	0.47	0.67	0.58	0.70	0.40	0.48	0.68	0.59	**0.71**

success rate for $k = 1$ to $> 90\%$ for $k = 2$.

5.3.2 BREC data set

BREC [237] is a novel graph expressivity data set focused on providing a robust, challenging benchmark for graph isomorphism detection, containing particularly difficult graph classes. The data set consists of 400 graphs, divided into 6 different categories. Table 5.3 provides an overview of them and shows our experimentally-observed success rates. We observe that the Vietoris–Rips filtration is the worst performing filtration for all tested k values, followed by the degree filtration. The most informative curvature-based filtration is the Ollivier–Ricci one, which obtained higher or equal success rates than the Forman–Ricci curvature filtration in all the subsets of data, with strictly higher average success rates for all k values. The most effective filtration overall is the Laplacian filtration for $k = 4$, surpassing almost all the algorithms, including both graph neural networks and classical methods, described in Wang and Zhang [237, Table 2]. The only exception was the N_2 algorithm [254] that obtained a success rate of 74.5%, compared to the 71% obtained by the Laplacian filtration for $k = 4$. Given the fact that N_2 requires knowledge of the isomorphism class of *all* 2-hop-induced subgraphs of a graph, the computational complexity makes it infeasible to apply for many graph sizes in practices [254,

Table 5.4: Accuracy (↑) when predicting the properties of graphs in the `ogbg-molhiv` molecular graph data set [255] using different filtrations at varying expansion levels of the graph (denoted by k). Column R contains the average probability of successfully predicting a property at random over all possible values of the property in the data, with the probability of choosing a label being proportional to the number of graphs with that label.

Data		$k=1$					$k=2$				
		Filtration									
	R	D	O	F	L	V	D	O	F	L	V
Diameter	0.02	0.09	**0.11**	0.05	0.08	0.07	0.10	0.08	0.06	0.09	-
Girth	0.04	0.00	0.11	0.34	0.46	**0.48**	0.14	0.21	0.33	0.45	-
Radius	0.03	0.16	**0.21**	0.06	0.15	0.14	0.20	0.18	0.10	0.17	-

Section 4.3], whereas the Laplacian filtration remains computable. Overall, these results underscore the high expressivity of persistent homology, making it a strong baseline for graph-learning tasks. We refer to Section B.4.2 for additional results.

5.3.3 Predicting graph properties

As our final expressivity experiments, we assess the capability of persistent homology to predict graph properties. Here, we focus on the *diameter*, the *radius*, and the *girth*.

Predicting the diameter of random graphs. We predict the diameter of Erdős–Rényi and Watts–Strogatz graphs. For the Erdős–Rényi graphs, we generate $N = 100$ graphs with $n = 100$ vertices and $p = 0.1$. This edge probability corresponds to the critical connectivity regime, for which closed-form solutions of the diameter distribution are not readily available [256]. We assess the utility of persistent homology by specifying a regression task;[5] to this end, we vectorise the persistence diagrams for each filtration using *Betti curves* [238, 239], a simple curve-based topological representation. While this representation is technically a function, we represent it as a histogram of 10 bins and train a *ridge regression classifier* via leave-one-out cross-validation to predict the diameter of each graph.

[5]Following Hartmann and Mézard [256], we take the diameter of an Erdős–Rényi graph, which might consist of different connected components, to be the largest diameter of all connected components of the graph.

We deliberately focus only on zero-dimensional and one-dimensional persistent homology, i.e., we leave $k = 1$. Using the *mean absolute error* (MAE) for evaluation, we find that Ollivier–Ricci curvature performs best (0.057), followed by the Laplacian spectrum (0.061), and the degree filtration (0.065). We observe similar patterns when calculating $N = 100$ Watts–Strogatz graphs of type $(100, 5, 0.1)$, i.e., we keep the same number of vertices and the same edge rewiring probability p, but connect each node with its 5 nearest neighbours in a ring neighbourhood. Using the same classifier, we again find that Ollivier–Ricci curvature achieves the lowest MAE (0.851), followed by the degree filtration (0.865), and the Laplacian spectrum (1.072).

Predicting graph properties of the `ogbg-molhiv` [255] data set. We predict the maximum radius and diameter among the radii and diameters of the connected components of each graph as well as their girth. See Figure B.2 for a visualisation of the distribution of these properties across all graphs in the data set. We use a random forest regression model and *persistence images* [76] as its input, computed from the persistence diagrams of the graphs calculated as in the previous sections. The model is trained on the usual train and test splits of the data set, with the final prediction obtained by rounding to the nearest integer.[6] Overall, this is a challenging task, with 24, 15, and 5 labels having more than 100 examples for the diameter, radius and girth, respectively. Thus, the average probabilities of guessing the correct label by random choice is 0.02, 0.03, and 0.04. However, we find that persistent homology, with a suitable filtration, can predict the maximum radius, the maximum diameter, and the girth of the graphs on unseen examples (i.e., on the test data set) in approximately 11%, 20%, and 48% of the cases, respectively, exhibiting substantially-improved results over a random baseline. For predicting radii and diameters, we find that the Ollivier–Ricci curvature outperforms the other filtrations for $k = 1$ but gets worse for $k = 2$, where the degree filtration performs the best. It is surprising that the Vietoris–Rips filtration is *not* able to predict the radii of the graphs in this data set accurately ($\approx 14\%$ accuracy), being the only

[6] We predict a girth of ∞ in case the output value is larger than the number of nodes in the graph.

filtration that works with explicit distances. However, this filtration proves most effective for predicting the girth, having an accuracy of almost 50% as compared to the 4% of the baseline. These results complement prior work on predicting properties of point clouds [231, 233], showing that topology-based graph-learning approaches carry a large degree of additional information about graphs.

5.4 Conclusions

We discussed various aspects of the computation and provided evidence of the advantageous properties of persistent homology in the context of graph learning. Our primary theoretical insight is that persistent homology is *at least as expressive* as a corresponding 1-WL or k-FWL test, in some cases surpassing their discriminative power. Experiments underscore these theoretical expressivity properties, while also demonstrating that persistent homology is able to capture additional properties of a graph. In light of the performance differences among the different filtration functions in our experiments, we suggest that future work should focus on elucidating properties of classes of such functions, aiming to strike a balance between expressivity and efficiency. Another limitation involves the calculation *per se*, which requires costly clique-finding operations and is thus not scalable to large, dense graphs. If node features are present, alternative geometrical-topological approaches could potentially be used [157, 257–260], necessitating additional research.

A compelling open question concerns the characterization of filtration classes that make k-dimensional persistent homology strictly more expressive than k-FWL (if any). Follow-up research could also focus on identifying other properties (next to the diameter and girth) that can be captured by persistent homology in graph learning. Such research has both theoretical and empirical components; a first step would be a formalisation of which substructures are captured by models [227, 228, 244]. Moreover, the success of the Laplacian filtration at the experimental tasks may hint at new filtrations based on spectral graph theory that provide a trade-off between utility and computational efficiency. We find that this research direction is overlooked by the computational topology research community, with

most of the expressivity/stability results focusing on describing the stability of distance-based filtrations under perturbations [261, 262], and few works focusing on graphs [263]. Based on our experiments, we thus envision that persistent homology will constitute a strong baseline for graph-learning applications. As previous work shows, even topology-inspired approaches, making use of concepts such as filtrations, can approximate the performance of highly-parametrised models at a fraction of the computational cost [238]. All insights obtained using such topological methods hint at the overall utility of graph-structural information for graph learning tasks, but it is not clear whether current graph benchmark data actually exhibit such structures [264]. We thus hope that persistent homology and related techniques will also find more applications in *hybrid models*, which are able to incorporate geometrical-topological information about graphs. This is an emerging research topic of crucial relevance since there are now numerous graph data sets that combine geometrical information (node coordinates) with topological information [229].

Our theoretical analysis of the properties of persistent homology for graph learning tasks show the potential and benefits of a topology-based perspective. We are confident that additional computational topology concepts will enrich and augment machine learning models, leading to new insights about their theoretical and empirical capabilities. This chapter is but a first attempt at elucidating the theoretical utility of computational topology in a graph learning context; advancing the field will require many more insights.

6. MANTRA

Contents

Success in machine learning is commonly measured by a model's ability to solve tasks on benchmark datasets. While researchers typically devote a large amount of time to build their models, less time is devoted to data and its curation. As a consequence, graph learning is facing some issues in terms of reproducibility and wrong assumptions, which serve as obstructions to progress. An example of this was recently observed while analyzing long-range features: additional hyperparameter tuning resolves performance differences between message-passing (MP) graph neural networks on one side and graph transformers on the other [265]. In a similar vein, earlier work pointed out the relevance of better taxonomies for existing datasets [266], as well as the need for strong baselines, highlighting the fact that *structural* information is not exploited equally by all models [267]. Recently, new analyses even showed that for some benchmark datasets, as well as their associated tasks, graph information may be detrimental for the overall predictive performance [268, 269], raising concerns about the future of graph learning as a research field [270].

These troubling trends concerning data are accompanied by increased interest in leveraging higher-order structures in data, with new models, usually called *topological models*, extending graph-learning concepts to *simplicial complexes*, i.e., generalizations of graphs that incorporate higher-order relations, going beyond the dyadic relations captured by graphs [157, 159, 257, 271–273]. Some topological

models already incorporate state-of-the-art mechanisms for learning such as message-passing [274], but adapted to higher-order domains, sometimes outperforming their original counterparts in graph datasets. However, as pointed out in a recent position paper [13], there is a dire need for "higher-order datasets", i.e., datasets that contain non-trivial higher-order structures. Indeed, the scarcity of such datasets impedes the development of reliable benchmarks for assessing (1) the utility of higher-order structures present in data, and (2) the performance of new models that leverage them, thus potentially eroding trust in topological models among the broader deep learning community.

Some of the currently-available higher-order datasets belong to the realm of networks, complex systems, and science. In [275], it is presented a rich collection of such datasets, comprising nineteen complex networks enhanced with higher-order information. Similar works have also employed higher-order structures in data; for instance, [276] used clique complexes on top on graphs coming from brain imaging data. Similarly, [277] proposed modeling neural data with simplicial complexes by constructing clique, concurrence (and its dual), and independence complexes on the data. However, most of these datasets are either *annotated* or *derived* from simpler data like graphs or time series. In the case of annotated data, it is unclear whether current non-higher-order (graph) neural networks or algorithms can extract information contained in higher-order structures using only annotations on vertices and edges. Similarly, for datasets obtained from simpler data, it is also uncertain whether non-higher-order algorithms can recover the higher-order structural information by reconstructing the processes used to generate these relationships explicitly. To the best of our knowledge, the only publicly-available purely higher-order dataset is the "Torus" dataset proposed in [278], which consists of a small number of unions of tori triangulations. Due to the nature of the dataset, the only varying topological property among the samples is the number of connected components of each union, making hard to assess the true capacity of the models to learn and exploit higher-order structures. The lack of higher-order datasets is also remarked upon in a recent benchmarking paper for topological models [279],

which restricted itself to existing graph datasets that were subjected to a variety of *topological liftings*, i.e., methods for endowing graph datasets with higher-order structures [280, 281]. However, it remains unclear whether standard graph neural network architectures can also learn and take advantage of the information provided by the topological liftings, as they are solely based on the graph structure.

To address these issues, we present MANTRA, the **man**ifold **tr**iangulations **a**ssemblage, which constitutes the first instance of a large, diverse, and intrinsically higher-order dataset, consisting of triangulations of combinatorial 2-manifolds and 3-manifolds. Along with the data, we provide a list of potential tasks, as well as a preliminary assessment of the performance of existing methods, both graph-based and higher-order-based, on the dataset. We focus on a subset of tasks concerned with the classification of simplicial complexes according to some topological labels, where we can interpret the success of a model by its effectiveness in extracting higher-order topological information. However, these tasks are by no means *exhaustive*, and we believe that the generality offered by MANTRA encourages the emergence of even more demanding tasks. Some of these tasks, such as the prediction or approximation of the Betti numbers from topological data, have been previously studied in learning [282] and non-learning [283] contexts. A noteworthy aspect of MANTRA is the conspicuous *absence* of any intrinsic vertex or edge features such as coordinates or signals. We believe that this absence renders tasks more topological, as models can only rely on topology, instead of non-topological information contained in features. Moreover, as manifold triangulations are directly related to the topological structure of the underlying manifold, we study to which extent higher-order models are *invariant* to transformations of a triangulation that preserve the topological structure of the associated manifold.

6.1 Dataset specification

MANTRA contains 43,138 and 250,359 simplicial complexes corresponding to triangulations of closed connected two- and three-dimensional manifolds, respectively, with varying number of vertices, obtained originally by Frank H. Lutz and compiled

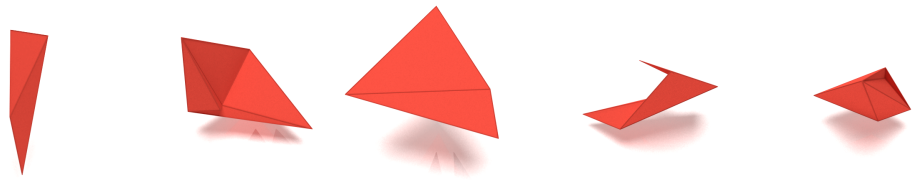


Figure 6.1: Geometric realizations of some manifold triangulations included in MANTRA. The precise coordinates of vertices in Euclidean space are not geometrically significant; what matters is the topology of the resulting polyhedra. Hence, MANTRA is a *purely combinatorial dataset*.

in [284]. Manifolds have many applications: the configuration space of a robotic arm can be seen as a manifold (e.g., a torus or hyperbolic space, see [285]); 3D shapes in geometry processing are triangulated manifolds [286]; physical fields in climate forecasting naturally live on a sphere [287], and the manifold hypothesis argues that high-dimensional data often lies in or close to lower-dimensional manifolds [288]. Throughout the text, we use the term *surface* to refer to a two-dimensional manifold. A *triangulation* of a manifold M is a pair consisting of a simplicial complex K and a homeomorphism between M and the geometric realization of K. For brevity, we use the term triangulation to refer exclusively to the simplicial complex K. See Appendix C for precise definitions and further information.

Table 6.1: Number of triangulations by manifold dimension (2-$\mathcal{M}$: 2-manifolds; 3-$\mathcal{M}$: 3-manifolds) and number of vertices $|V|$ in a triangulation.

$\lvert V\rvert$	2-$\mathcal{M}$	3-$\mathcal{M}$
4	1	0
5	1	1
6	3	2
7	9	5
8	43	39
9	655	1,297
10	42,426	249,015
Total	43,138	250,359

Triangulations of surfaces and 3-manifolds encode higher-order topological information that cannot be inferred solely from their underlying graphs. Indeed, there exist non-homeomorphic surfaces with identical graph structures. Specifically, for $n > 7$, the complete graph with n vertices triangulates both a connected sum of tori and a connected sum of projective planes, which are non-homeomorphic [67]. Figure 6.1 contains examples of geometric realizations of MANTRA triangulations. Table 6.1 contains the distribution of triangulations in terms of their number of vertices. Each triangulation contains a set of labels based on its dimension. Common labels are the number of vertices of the triangulation, the first three Betti numbers β_0, β_1, β_2, and torsion in homology with integer coefficients. For

triangulations of a Klein bottle K, a real projective plane $\mathbb{R}P^2$, a 2-dimensional sphere S^2, or a torus T^2, the homeomorphism type is included explicitly as a surface label. We additionally specify the top Betti number β_3 and the homeomorphism type, which can be a 3-sphere S^3, a product $S^2 \times S^1$ of a 2-sphere and a circle, or a Möbius-like S^2-bundle along S^1, denoted by $S^2 \tilde{\times} S^1$. An exploration of the distributions of labels is made in Appendix C.1.

We make the dataset and benchmark code available via two repositories:

(1) `https://github.com/aidos-lab/MANTRA`

(2) `https://github.com/aidos-lab/mantra-benchmarks`

These repositories contain ((1)) the raw and processed datasets, and ((2)) the code to reproduce all our results. Detailed hyperparameter settings can be found in Appendix C.4. Step-by-step instructions on how to set up and execute the benchmark experiments are attached in the `README` file of the repository. Docker images and workflow, together with package dependencies are included to ensure a unique environment across different machine configurations. Finally, random seeds were used to split the datasets in each run.

The dataset is available in two formats, namely a *raw version* and a *PyTorch Geometric processed version*. The raw version currently consists of a pair of compressed files `2_manifolds.json.gz` and `3_manifolds.json.gz`, containing a JSON list with the triangulations of the corresponding dimension. Each object of the JSON list consists of a set of the following fields, depending on the dimension of the associated triangulation:

Field	Type	Description
`id`	`str`	This attribute refers to the original ID of the triangulation as used by [284] when compiling the triangulations. This facilitates comparisons to the original dataset if necessary.
`triangulation`	`list`	A doubly-nested list of the facets of the triangulation.
`n_vertices`	`int`	The number of vertices in the triangulation.
`name`	`str`	Homeomorphism type of the triangulation. Possible values are `''`, `'Klein bottle'`, `'RP^2'`, `'S^2'`, `'T^2'` for surfaces, where `''` indicates that the explicit homeomorphism type is not available. For 3-dimensional manifolds, possible values are `'S^2 twist S^1'`, `'S^2 x S^1'`, `'S^3'`.
`betti_numbers`	`list`	A list of Betti numbers of the triangulation, computed using $R = \mathbb{Z}$, i.e., integer coefficients.
`torsion_coefficients`	`list`	A list of the torsion subgroups of the triangulation. Possible values are `''`, `'Z_2'`, where an empty string `''` indicates that no torsion is present in that dimension.
`genus`	`int`	For surfaces, contains the genus of the triangulation.
`orientable`	`bool`	For surfaces, specifies if the triangulation is orientable or not.

The `PyTorch Geometric` [289, PyG] version is available as a `Python` package that can be installed using the command `pip install mantra-dataset`. Each example of the dataset is implemented as a PyG `Data` object, containing the same attributes as JSON objects in the raw version. The main difference with the data in the raw version is that numerical values are stored as PyTorch tensors. Both formats, raw and processed, are versioned using the Semantic Versioning 2.0.0 convention [290] and are also available via Zenodo,[1] thus ensuring *reproducibility* and clear tracking of the dataset evolution.

Dataset limitations. In the present version of MANTRA, triangulations are restricted to two- and three- dimensional complexes up to 10 vertices. This may pose a limitation concerning the transferability of our findings to datasets with significantly higher number of vertices per sample, such as fine-grained mesh datasets.

[1] `https://doi.org/10.5281/zenodo.14103581`

While extending the dataset beyond 10 vertices is theoretically possible, it poses substantial storage and computational challenges due to the exponential growth in possible triangulations —for example, there are over 11 million surfaces for triangulations of 11 vertices— and the unavailability of complete enumerations of triangulations for more than 13 vertices, which may potentially lead to incomplete datasets and skewed label distributions. Note that this does not preclude the addition of triangulations with other properties, for instance certain minimality properties (like vertex-transitive triangulations). Additionally, focusing solely on two- and three-dimensional manifolds excludes higher-dimensional triangulations and data, which remain active areas of research. Nevertheless, we believe that MANTRA provides a valuable benchmark for testing higher-order models on the most common types of higher-order structured data, that is, graphs, surfaces, and volumes. Finally, we want to highlight the fact that MANTRA does not (yet) encompass the full spectrum of properties present in real-world data—for example, the large simplicial complex sizes found in complex networks or the geometric information contained in some datasets like the existence of vertex coordinates in meshes. Therefore, although we consider MANTRA a valuable dataset for testing the capabilities of higher-order models, it should be studied in conjunction with other conceptually diverse datasets to better study the capabilities of models.

6.2 Experiments

In this section, we assess twelve state-of-the-art simplicial complex- and graph-based architectures, on various topological prediction tasks such as Betti number homeomorphism type classification, and orientability detection. Our experiments confirm that simplicial complex-based neural networks almost always achieve better results than graph-based ones in extracting the topological invariants mentioned above. However, we also find that the performance of the assessed models may be suboptimal for being called topological models. In particular, we discover that all model performances deteriorate when applying barycentric subdivisions to

the original test datasets, suggesting that the tested models are unable to learn topologically invariant functions.

Sections 6.2.1 and 6.2.2 present the comprehensive experimental design for MANTRA, outlining the key scientific questions addressed. Section 6.2.3 provides a detailed analysis of the experimental results.

6.2.1 Main experiments

Leveraging the extensive set of labels and triangulations available, our experiments are designed to address the following critical research questions:

Q1 To what extent are higher-order models needed to perform inference tasks on higher-order domains like simplicial complexes? Are graph-based models enough to successfully capture the full set of combinatorial properties present in the data?

Q2 Do current neural networks, both graph- and simplicial complex-based, capture topological properties in data? Are they able to predict basic topological invariants such as Betti numbers of simplicial complexes?

Q3 How invariant are state-of-the-art models to transformations that preserve topological properties of data?

The difference between **Q1** and **Q2**, **Q3** is subtle. Combinatorial information is related to the structure of the data input values, in our case, simplicial complexes, while topological information is related to properties that are invariant under *topological transformations* of the data. For example, in prediction tasks involving molecules, we expect combinatorial information to be more relevant than topological features, since the structure of a molecule is crucial in predicting properties of the molecule. Of course, both types of information are intertwined: to properly compute topological properties of data, we need to consider their combinatorial structure , as the input as explained in Sections 2.1.3 and 2.1.4. To answer the above questions, we benchmarked twelve models: five graph-based [289], using only zero- and one-dimensional simplices of complexes, and seven simplicial complex-based

models, four from the TopoModelX library [51] and three extra cellular models, using the full set of simplicial complexes in

T1 Predicting the Betti numbers β_i for triangulated surfaces and 3-dimensional manifolds.

T2 Predicting the homeomorphism type of triangulated surfaces.

T3 Predicting orientability of triangulated surfaces.

To address the high proportion of surfaces without explicitly assigned homeomorphism type, we duplicated the experiments on both the full set of surfaces and the subset of surfaces with known type. Throughout the paper, we denote by 2-$\mathcal{M}^0$, 2-$\mathcal{M}^0_H$, and 3-$\mathcal{M}^0$ the full set of surfaces, the set of surfaces with known homeomorphism type, and the full set of 3-manifolds, respectively.

Models. The graph-based models benchmarked are the Multi-Layer Perceptron (MLP), the Graph Convolutional Network [91, GCN], the Graph Attention Network [92, GAT], the Unified Message Passing graph transformer [291, UniMP], and the Topology Adaptive Graph Convolutional Network [292, TAG]. The simplicial complex-based benchmarked models are the Simplicial Attention Network [93, SAN], three convolution-based simplicial neural networks previously benchmarked in [279] and introduced in [95, SCCN], [96, SCCNN], and [94, SCN], respectively, the cellular message passing from [35, CellMP], the Differentiable Euler Characteristic Transform [257, DECT], and our Cellular Transformer (CT), introduced in Chapter 7 to alleviate message passing-based shortcomings. Note that except for the MLP model, the graph and cellular transformers, and the DECT, all models implement some variant of a (higher-order) message-passing paradigm [30, 85]. More information about the models can be found in Appendix C.3.

Features. All twelve models assume that simplicial complexes are equipped with feature vectors on top of a subset of the simplices. The feature vectors for graph-based models and DECT are either: (1) scalars randomly generated, (2) degrees of each vertex, (3) degree one-hot encodings of each vertex. For the rest of simplicial complex-based models, the feature vectors are either: (1) eight-dimensional vectors

generated randomly, (2) number of upper-adjacent neighbors (upper-connectivity index) of each simplex of dimensions lower than the dimension of the simplicial complex and number of lower-adjacent neighbors (lower-connectivity index) for simplices of the same dimension as the simplicial complex. By definition, two simplices are upper-adjacent, and both are upper-adjacent neighbors of the other, if they share a coface of one dimension higher. Similarly, two simplices are lower-adjacent if they share a face of one dimension lower.

Training details. In total, our experiments span 240 training results across various tasks, feature generation, and models. To ensure fairness, all configurations use the same learning rate of 0.01 and the same number of epochs of 6; we observe that graph-based models already overfit after a single epoch, though. Hyperparameters for graph-based models were mostly extracted from the default examples from PyTorch Geometric, while hyperparameters for simplicial complex-based models were set to values similar to the ones from the TopoBenchmarkX paper [279], specially for the already benchmarked SCCN, SCCNN, and SCN models. Hyperparameter details can be found in Appendix C.4. To mitigate the effects of training randomness, we re-ran each experiment five times and considered both the best and the mean (together with standard deviation) performance obtained across these runs for each model and initialisation of features. Due to the high imbalance in the datasets for most labels, we performed stratified train/validation/test splits for each task individually, with 60/20/20 percentage of the data for each split, respectively. Splits were generated using the same random seed for each run, ensuring that the same splits are used across all configurations. All models were trained using the Adam optimizer.

Loss and metric functions. Each task (**T1**, **T2**, **T3**) was treated as a classification task during testing. We report the area under the ROC curve (AUROC) [293] as performance metric, which is standard for imbalanced classification problems, on all tasks except for predicting β_0, where we report accuracy due to the fact that we only have the label 1, as all our triangulations correspond to connected manifolds.

For both the homeomorphism type and orientability tasks, we train the models using the standard cross-entropy loss for classification problems. We also experimented with weighting the cross-entropy loss to penalize mispredictions in under-represented classes more heavily, but we did not obtain improvements. To avoid increasing the computational complexity of our experiments, we chose not to implement more involved methods for handling the class imbalances and leave this issue for future work. For Betti number prediction, we approached training as a multivariate regression task, since Betti numbers can theoretically be arbitrarily large. Our loss function in this case was the mean squared error, and the Betti number prediction was obtained by rounding the model outputs to the nearest integer.

6.2.2 Barycentric subdivision experiments

The previous experiments try to answer **Q1** and **Q2**: if performances are good for simplicial complex-based graph-based ones, then we can conclude that higher-order models are needed to perform inference tasks on domains with higher-order and topological information. By contrast, if performances are good for graph-based models then we can conclude that graph models are enough to capture the full set of combinatorial and topological features present in MANTRA's dataset, questioning the need for higher-order models. However, **Q3** is more subtle. Although it is closely related to **Q2**, **Q3** emphasizes the *invariance* of the models to transformations that preserve the topological properties of the input data, a desirable property for high-order models known as remeshing symmetry [13]. For example, if a model is well-trained with a dataset containing only triangulations up to a certain number of vertices, we can expect the model to perform correct predictions in new examples that also have at most the maximum number of vertices seen in the training dataset. However, what happens if we try to predict from a *refinement* of a manifold triangulation? For instance, barycentric subdivisions increase the (combinatorial) distances between the original vertices in a triangulation, and this can be harmful for networks relying on the MP algorithm, since distances determine how many layers are needed to propagate information from one vertex to another. In fact,

[98] showed that MP-based graph neural networks with a small number of layers struggled to obtain good performances on synthetic datasets where the number of cycles and connected components played a crucial role.

To answer **Q3**, we performed an additional evaluation of the models trained on surface tasks with known homeomorphism type for the experiments described in Section 6.2.1. Particularly, for each task, we evaluated the performance of the trained models on a dataset obtained by performing one barycentric subdivision on each triangulation in the original test dataset, and then we compared the performances of the models on both datasets, original and subdivided. Throughout the text, we denote the subdivided test dataset as $2\text{-}\mathcal{M}_H^1$. We did not analyze barycentric subdivisions of 3-dimensional manifolds due to computational constraints. Also, for these experiments, we leave out the DECT model from the analysis, since the DECT is invariant with respect to barycentric subdivision by construction if used appropiately.

6.2.3 Analysis

Our analysis reports *aggregated results* and focuses primarily on the comparison between graph-based models ($\mathcal{G}$) and simplicial complex-based models ($\mathcal{T}$). Comprehensive results are available in Appendix C.5. Table 6.2 presents the mean and standard deviation of the maximum performance achieved by each combination of feature vector initialization and model type across the 5 runs of each task for both graph-based ($\mathcal{G}$) and simplicial complex-based ($\mathcal{T}$) model families, including performances on the barycentric subdivisions of the test triangulations for each experiment run in the set of surfaces with known homeomorphism type, as described in Section 6.2.2. Notably, our experiments suggest that higher-order MP-based and transformer models, including our proposed cellular transformer, are *not invariant* relative to topological transformations and therefore cannot be considered topological in the strictest sense of the term: higher-order models predicting better than random in any task suffer from a performance degradation when testing

Table 6.2: Predictive performance of graph- and simplicial complex-based models on surface and 3-manifold tasks. Results for the full set of surfaces ($2\text{-}\mathcal{M}^0$), for the set of surfaces with known homeomorphism type ($2\text{-}\mathcal{M}^0_H$), and for the full set of three-manifolds ($3\text{-}\mathcal{M}^0$) are reported. Additionally, performance metrics for the barycentric subdivision of the test set on the models trained on $2\text{-}\mathcal{M}^0_H$, i.e., $2\text{-}\mathcal{M}^1_H$, are included; see Section 6.2.2 for details. For each family of models, $\mathcal{G}$ (graph-based) and $\mathcal{T}$ (simplicial complex-based), we report the mean and standard deviation of the maximum performance achieved across five runs by each combination of feature vector initialization and model contained in the family. The tasks reported are prediction of β_0, β_1, β_2, β_3, prediction of the homeomorphism type, and prediction of orientability. For all tasks except for prediction of β_0, we report the AUROC metric. For β_0, we report accuracy. Best average result among both families for each task is in bold. Note that the reported averages and standard deviations are not calculated from individual model performances across different random seeds. Instead, for each model, we selected its best performance achieved across all seeds for each experiment. Then, we aggregated these best performances within each category—graph-based and simplicial complex-based models—to compute the averages and standard deviations reported in the table.

		Accuracy			*AUROC*		
Dataset	Model Family	β_0	β_1	β_2	β_3	Homeo. Type	Orientability
$2\text{-}\mathcal{M}^0$	$\mathcal{G}$	$\mathbf{1.00 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00		0.47 ± 0.01	0.50 ± 0.00
	$\mathcal{T}$	0.73 ± 0.39	$\mathbf{0.68 \pm 0.16}$	$\mathbf{0.59 \pm 0.10}$		$\mathbf{0.69 \pm 0.18}$	$\mathbf{0.56 \pm 0.07}$
$2\text{-}\mathcal{M}^0_H$	$\mathcal{G}$	$\mathbf{1.00 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00		0.49 ± 0.01	0.50 ± 0.00
	$\mathcal{T}$	0.57 ± 0.44	$\mathbf{0.25 \pm 0.03}$	$\mathbf{0.52 \pm 0.02}$		$\mathbf{0.66 \pm 0.13}$	$\mathbf{0.52 \pm 0.02}$
$2\text{-}\mathcal{M}^1_H$	$\mathcal{G}$	$\mathbf{0.47 \pm 0.51}$	0.22 ± 0.00	0.50 ± 0.00		0.49 ± 0.04	0.50 ± 0.00
	$\mathcal{T}$	0.21 ± 0.38	$\mathbf{0.25 \pm 0.02}$	$\mathbf{0.51 \pm 0.01}$		$\mathbf{0.60 \pm 0.10}$	$\mathbf{0.51 \pm 0.01}$
$3\text{-}\mathcal{M}^0$	$\mathcal{G}$	$\mathbf{1.00 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	0.14 ± 0.00		0.14 ± 0.00
	$\mathcal{T}$	0.78 ± 0.41	$\mathbf{0.25 \pm 0.04}$	$\mathbf{0.13 \pm 0.03}$	$\mathbf{0.16 \pm 0.03}$		$\mathbf{0.15 \pm 0.02}$

on the subdivided examples, as shown by the full set of results in Sections C.5.1 to C.5.2 and tables C.9 and 6.2.

Weaknesses in the MP-based models are not a recent phenomenon, as highlighted by oversmoothing [294] and oversquashing [295, 296], and the MP paradigm has required numerous fixes since its existence (inlcuding, but not limited to, virtual nodes, feature augmentation, and graph lifting). More recently, [278] argued that, in many cases, higher-order MP-based models cannot distinguish combinatorial objects based on simple topological properties, and has devised another MP variant to compensate for this.

Graph-based ($\mathcal{G}$) vs. simplicial complex-based ($\mathcal{T}$) models. Table 6.2 together with the full results of Appendix C.5 show that simplicial complex-based models consistently obtain better or equivalent performances predicting non-trivial

topological properties of triangulated manifolds, meaning β_1, β_2, β_3, orientability, and homeomorphism type. We note that graph models *always* correctly detect the connectivity of triangulations in 2-$\mathcal{M}^0$, 2-$\mathcal{M}^0_H$, and 3-dimensional manifolds, thus predicting β_0 exactly, while topological models consistently fail to predict connectivity, except for the CT, DECT, and SCCN architectures in our experiments. The fact that some higher-order message passing networks cannot accurately predict connectivity was also found, and theoretically proved, in [278, Proposition 4.3]. Moreover, although simplicial complex-based models obtain better results overall, these are far from being highly accurate, with averages below 70 for all tasks, a high performance variance across the models in some tasks, and tasks for which simplicial complex-based models obtain a performance similar to a random-guessing strategy. Nonetheless, the best performances obtained by specific simplicial complex-based models, as described in the full results of Appendix C.5, are promising, achieving excellent AUROC results in some tasks, such as homeomorphism type prediction, where the CT and SCCN models obtained average AUROCs of 91 and 83 and 85 and 80 for the full and known homeomorphism type surface datasets, respectively, and Betti number prediction for the full set of surfaces, where the CT and SCCN models obtained an average AUROC of 93 when predicting the first Betti number. Overall, the results suggest that higher-order models, and particularly our proposed Cellular Transformers, are indeed necessary to capture topological and higher-order characteristics of data, although several current models are not yet able to do so effectively, partially answering questions **Q1** and **Q2**. Such results were expected, given that one-dimensional structures are insufficient, in principle, to fully characterize the topology of two- or three-dimensional triangulated manifolds, as stated at the beginning of Section 6.1. However, it is plausible that graph-based networks can accurately classify approximately 50% of homeomorphism types of surfaces, since the underlying graph of a triangulation determines the Euler characteristic, which in turn defines the homeomorphism type up to orientability (see Section 2.1.3).

Orientability. Predicting orientability turns out to be the most difficult task for graph- and simplicial complex-based models, obtaining performances equivalent to random guessing in most cases for surfaces and performances worse than random for 3-manifolds. Moreover, we do not find significative differences between the performances of predicting the Betti number β_2 and orientability for surfaces as a binary problem. This is consistent with the similar results obtained for predicting β_2 and orientability type as a binary classification problem for surface datasets.

Barycentric subdivisions. Table 6.2 shows that the performance of all models decreases when subdividing the triangulations of the test dataset if the models were performing better than random guessing, indicating that the models are not learning the invariance of topological properties with respect to subdivisions transformations that leave topological properties invariant. This is a crucial property that any model dealing with topological domains should have, as real data is often highly variable in terms of combinatorial information and representation, but not in terms of their topology. This phenomenon is particularly evident in mesh datasets, where combinatorial structure varies with resolution. Meshes typically comprise more triangles, yet all or nearly all input data represent triangulations of connected closed surfaces, regardless of resolution. In fact, [13] posit the capacity of high-order models to capture this invariance, denoted *remeshing symmetry*, as one of the reasons for using topological deep learning models. Our preliminary experimental results challenge this claim, opening the door to a new line of research based on the invariance of input transformations that leave topological properties of the input data unaltered.

Experimental limitations. Although our results challenge the efficiency of state-of-the-art higher-order models to predict topological properties of data and open the door to exciting new research avenues, they must be interpreted with care. For example, we mostly tested message-passing networks in our experiments, leaving aside interesting proposals such as higher-order state-space models [97], combinatorial complex networks [30, 35], topological Gaussian processes [297] or equivariant

higher-order neural networks [155]. Due to computational limitations, training procedures were limited to 6 epochs, model hyperparameters were not necessarily selected optimally, and barycentric subdivisions experiments were limited to one subdivision. A significant computational bottleneck arose from the implementations of simplicial complex-based models, which processed data noticeably slower than their graph counterparts as observed in Table C.5, highlighting the need for more efficient implementations of high-order methods. Despite these limitations, we believe that each of the three stated questions should be investigated individually, with a broader set of experiments and ablations to be fully answered.

6.3 Conclusions

We proposed MANTRA, a higher-order dataset of manifold triangulations that is (1) *diverse*, containing triangulations of surfaces and three-dimensional manifolds with different topological invariants and homeomorphism types (2) *large*, with over 43,000 triangulations of surfaces and 250,000 triangulations of three-dimensional manifolds, and (3) *naturally higher-order*, as the triangulations are directly related to the topological structure of the underlying manifold. Using MANTRA, we observed that existing models, both graph-based and higher-order-based, struggle to learn topological properties of triangulations, such as the orientability of two-dimensional manifolds, which was the hardest topological property to predict for surface triangulations, suggesting that new approaches are needed to leverage higher-order structure associated with the topological information in the dataset. However, we also observed that current higher-order models outperform graph-based models in our benchmarks, substantiating the promises of this new trend of higher-order machine-learning models. Regarding invariance, we observed that barycentric subdivision deeply affects the performance of the models, suggesting that current state-of-the-art models *fail to be invariant* to transformations that preserve the topological structure of data, opening an interesting research direction for future work. In the case of MP-based models, this could be potentially related to sensitivity of message-passing to the distances between simplices in simplicial

complexes. Another interesting research direction for barycentric subdivisions is their application as inputs to graph neural networks. The induced graph of a barycentric subdivision represents each simplex of the original complex as a vertex, with edges encoding face relationships on the original complex. This structure provides an effective representation of simplicial complexes for graph-based neural architectures, potentially facilitating the processing of higher-order topological information. We hope that MANTRA will serve as a benchmark for the development of new models leveraging higher-order and topological structures in data, and as a reference for the development of new higher-order datasets.

7. Cellular transformers

Contents

Many complex scientific entities, including proteins, brain connectomes, and molecular structures, are more accurately represented through high-order structures such as simplicial or cellular complexes rather than traditional graph representations. This high-order perspective is particularly crucial in computational chemistry and drug discovery, where the chemical space encompasses approximately 10^{60} potential molecules. In such vast combinatorial domains, the ability to effectively encode and leverage the intrinsic higher-order relationships within molecular representations, including ring systems, non-covalent bonds, long-range dependencies and non-trivial connectivity patterns, can significantly determine the success or failure of learning tasks [155, 298].

Traditional methods for molecular modeling often rely on graph-based representations and models such as MPNNs or graph transformers [102, 299]. Despite their success, graph-based approaches suffer from limitations in capturing critical chemical interactions. In particular, while graph transformers enjoy improved higher-order feature utilization through the attention mechanisms, as they create new feature representations for the graphs taking into account all the features and the graph structure at the same time, they still rely on a graph-based representation that does not natively encode multi-scale molecular structures such as fused rings

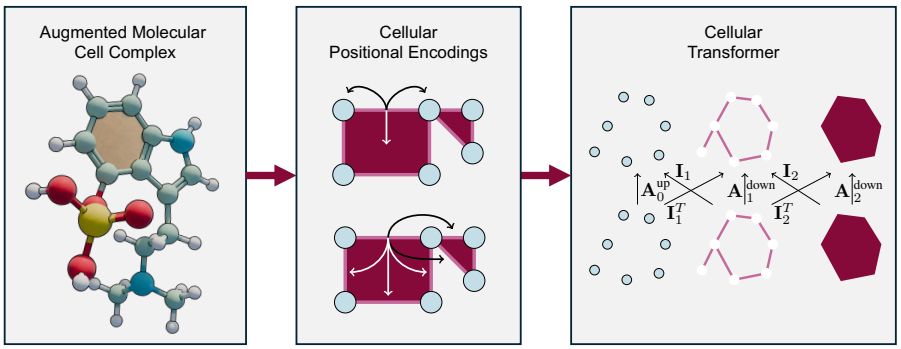


Figure 7.1: Pipeline of molecular tasks using the Cellular Transformer. First, we construct an augmented molecular cellular complex (AMCC) from the molecular graph. Then, we use the Cellular Transformer to extract features from the AMCC. Finally, we use the extracted features to predict a property or solve a general task involving the molecule.

or higher-dimensional topological motifs. This lack of topological awareness leads to suboptimal feature extraction and generalization.

In this chapter, we introduce the *Cellular Transformer* (CT), a novel transformer-based HONN that extends attention-based learning beyond graphs by leveraging cellular complexes. CT consists of a high-order attention mechanism incorporating topological information by interacting multiple structural levels and by using novel positional encodings designed natively for cellular complexes. Our CT enables more expressive feature learning without requiring graph rewiring, virtual nodes, or domain-specific heuristics, which allows us to tackle graph learning tasks successfully, outperforming state-of-the-art in many datasets. Also, attention-based mechanisms potentially solve some of the MPNN limitations appearing in graph and high-order learning such as long-range dependency detection, oversmoothing, and oversquashing, due to the fact that they can attend to all the features at the same time and not only the ones in the local neighborhood.

We evaluate CT on molecular benchmarks due to their idoneity for high-order representation, as discussed before. To properly leverage the high-order interactions of our architecture, we also introduce a new molecular cellular complex representation that we call *augmented molecular cellular complexes* (AMCCs), an enriched molecular representation that captures topological motifs at the ring and bond level.

We observe that our model consistently outperforms graph message-passing and transformer-based architectures while requiring fewer heuristics or bells and whistles. Notably, we show that our approach captures molecular properties more effectively, particularly in datasets where higher-order structural relationships play a critical role. As depicted in Figure 7.1 Our results validate the power of topologically informed transformers for molecular property prediction and pave the way for further advancements in topological deep learning for chemistry and materials science.

7.1 The Cellular Transformer model

In this section, we present our general transformer architecture for cellular complexes. First, we discuss different approaches to perform attention on cells and define the cellular transformer layer. Then, we propose different positional encoding methods for the cellular transformer, that identify cells by their relative position in the cellular complex or by their *centrality* according to random walks, as in [102].

A *Cellular Transformer* (CT) is a neural network taking and outputing annotated cellular complexes and whose function is given by a composition of functions

$$\mathrm{CT} = \mathrm{R} \circ \mathrm{CT}_L \circ \cdots \circ \mathrm{CT}_1 \circ \mathrm{P},$$

where CT_l are CT layers for $l \in \{1, \dots, L\}$, P is a preprocessing layer as described in Section 7.1.3, and R is a readout layer that converts annotations on top of cells into an output prediction value. CT layers are dependent on the input annotated cellular complex. Given an annotated cellular complex $(K, \{f_i\}_{i=0}^{\dim K})$ a CT layer defines a function with signature

$$\mathrm{CT}_l \colon (\mathbb{R}^{d_{K_0}})^{K_0} \times \cdots \times (\mathbb{R}^{d_{K_{\dim K}}})^{K_{\dim K}} \to (\mathbb{R}^{d_{K_0}})^{K_0} \times \cdots \times (\mathbb{R}^{d_{K_{\dim K}}})^{K_{\dim K}}, \quad (7.1)$$

where $(\mathbb{R}^{d_{K_l}})^{K_l}$ are the spaces of possible feature maps of fixed length d_{K_l} for each dimension $l \in \{0, \dots, \dim K\}$. In our case, we set $d_{K_l} = h$ for all $l \in \{0, \dots, \dim K\}$, where h is a hyperparameter depending on the dataset. The actual implementation of Equation (7.1) depends on the type of attention mechanism chosen for the CT layer, as we will see in the following sections.

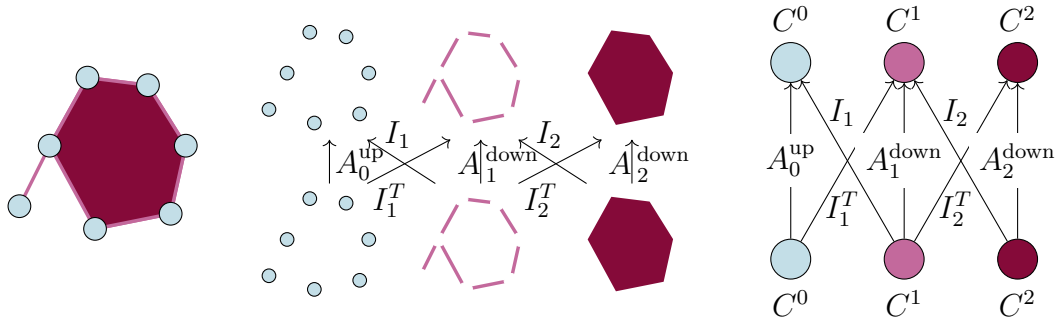


Figure 7.2: Attention tensor diagram illustrating the flow of signals between feature maps defined on 0-, 1-, and 2-cells. For pairwise attention, the neighborhood matrices indicate the bias N in the attention formula (7.1.1). For general attention, neighborhood matrices indicates how to build the bias matrix N by composition of smaller bias matrices $N_{k_s \to k_t}$ between dimensions.

7.1.1 Cellular attention

We now introduce two cellular attention mechanisms for transformer layers. Let $(K, \{f_i\}_{i=0}^{\dim K})$ be an annotated cellular complex. For the rest of the chapter, we denote by F_k the matrix of size $|K_k| \times d_k$ representing the values of the feature map f_k for the different simplices of K_k in any arbitrary (but fixed) order for each $k \in \{0, \dots, \dim K\}$.

The first mechanism, we call *pairwise cellular attention*, generalizes self- and cross-attention and depends on the dimensions of the cells. The second, *general cellular attention*, performs the attention over all cells ignoring their dimension.

Pairwise Cellular Attention (PCA). Given source and target ranks $0 \leq k_s, k_t \leq \dim K$ and feature maps F_{k_t}, F_{k_s}, the single-head attention from k_s to k_t is a map $(\mathbb{R}^{d_s})^{K_{k_s}} \times (\mathbb{R}^{d_t})^{K_{k_t}} \to (\mathbb{R}^{d_t})^{K_{k_t}}$ defined as

$$\mathcal{A}^{\bullet}_{k_s \to k_t}(F_{k_t}, F_{k_s}) = \text{softmax}(F_{k_t} Q_{k_s \to k_t} (F_{k_s} K_{k_s \to k_t})^\top \star \phi(N_{k_s \to k_t})) F_{k_s} V_{k_s \to k_t},$$

where $Q_{k_s \to k_t} \in \mathcal{M}(d_t^h, p, \mathbb{R})$, $K_{k_s \to k_t} \in \mathcal{M}(d_s^h, p, \mathbb{R})$, and $V_{k_s \to k_t} \in \mathcal{M}(d_s^h, d_t^h, \mathbb{R})$ are learnable query, key, and value real matrices with p a fixed hyperparameter shared by all transformer layers. The symbol $\bullet \in \{d, s\}$ indicates whether we perform dense or sparse attention, respectively. The symbol $\star$ is a sum or a Hadamard

product for dense or sparse attention, respectively. N is a neighborhood matrix, and ϕ is a function, possibly with learnable parameters.

The PCA mechanism performs pairwise attention between cells of arbitrary ranks according to a tensor diagram (see Section 7.1.2), and then aggregates the outputs received for the same rank. For our experiments, we set $\phi(N)$ to be a learnable embedding layer for dense attention and the identity o.w. Attention formulae performs query, key, and value projections without bias for simplicity. A bias term can be added to the projections, as in most transformers. Multi-head attention can also be performed by

(1) splitting the feature maps F_{k_s} and F_{k_t} into multiple feature maps $F^1_{k_s}, \ldots, F^m_{k_s}$ and $F^1_{k_t}, \ldots, F^m_{k_t}$ of smaller dimension;

(2) performing single-head attention for each pair of feature maps $F^i_{k_s}, F^i_{k_t}$;

(3) concatenating the outputs of the single-head attention for the different pairs into a full feature map of dimension d^h_t.

For a specific rank k_t, CT layers can produce multiple attention outputs from different rank sources k_s. In the CT layer, we adopt the standard prenorm design [300], where for each rank k_t, the outputs from the various rank sources k_s are added to form the final output for the rank k_t. The specific algorithm for the CT layer is detailed in Section D.3. In our experiments, we set $N_{k\to k}$ to be a distance matrix computed from the 1-skeleton of the barycentric subdivision distance matrix; see Section 7.1.3, restricted to the cells of dimension k in the dense case, and the the upper adjacency matrix for $k = 0$ and the lower adjacency matrix if $k > 0$ in the sparse case. Additionally, we define $N_{k_s\to k_t}$ to be the non-signed incidence matrix between dimensions k_s and k_t if $k_s > k_t$, and the transpose matrix otherwise. We now present the second mechanism which performs attention with all the cells at the same time, disregarding their rank, similar to what was proposed in [168]:

General Cellular Attention (GCA). We propose the following single-head GCA attention:

$$\mathcal{A}_g^\bullet(F) = \text{softmax}\left(FQ(FK)^\top \star \phi(N)\right) FV, \tag{7.2}$$

using the same notations as in Section 7.1.1, where F is the concatenation of all the feature maps for all dimensions and where cells share the same key, query, and value matrices. In this case, the prenorm transformer layer is performed as usual.

The algorithm corresponding to the GCA layer is detailed in Section D.3.1. As in our pairwise attention, multi-head attention can be performed by splitting the original feature maps into smaller feature maps, applying the general single-head attention to pairs of the smaller feature maps, and concatenating again into a single, big feature map. In our experiments, we let N be the following combination of the previous $N_{k_s \to k_t}$ matrices

$$N = \begin{bmatrix} N_{0\to 0} & N_{1\to 0} & \dots & 0 \\ N_{1\to 0}^\top & N_{1\to 1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & N_{\dim K \to \dim K} \end{bmatrix}. \tag{7.3}$$

7.1.2 Attention tensor diagrams for CTs

CTs involve interactions between feature maps of different ranks. Tensor diagrams [30] provide a graphical abstraction illustrating the flow of information on one CT layer. A tensor diagram portrays a CT Layer through the use of a directed graph, where the nodes represent feature map spaces for different ranks $0 \leq k \leq n$, n being the maximum allowed rank of cellular complexes processed by that CT layer. If the input cellular complex K is of lower dimension than n, the attention on ranks $k > \dim K$ are ignored. In turn, edges represent either the pairwise attentions performed in the CT layer together with the bias matrices $N_{k_s \to k_t}$, or simply the matrices used to build the matrix N from smaller matrices $N_{k_s \to k_t}$ as in Equation (7.3), for the general attention. A missing arrow from feature maps of rank k_s to feature maps of rank k_t implies a zero in the block of N corresponding to the matrix $N_{k_s \to k_t}$. Figure 7.2 illustrates the tensor diagram used in our experiments.

In cellular complex molecular modeling, incidence matrices in a molecular cellular complex enable cross-attention, while adjacency and Hodge-Laplacian matrices enable self-attention within our CT framework. In particular, cross-attention captures multiscale interactions between structural levels, such as nodes, edges, and cycles, while self-attention uses adjacency relations to refine feature aggregation within the same rank. This dual mechanism enhances molecular representations, capturing both local and global dependencies.

Tensor diagrams provide a graphical abstraction of these interactions, guiding the construction of the CT across feature map ranks. This aids in designing custom attention mechanisms within the transformer, ensuring effective encoding of both cross-rank and self-attention patterns.

7.1.3 Positional encodings on cellular complexes

Transformers do not leverage the input structure explicitly by default [31]. Positional encodings (PEs) help to overcome this problem by injecting positional and structural information about the input *tokens*. For sequences, the first positional encoding used sine and cosine functions depending on the position of the token in the sequence. For graphs, several positional encodings have been studied such as the eigenvectors of the graph Laplacian (LapPE) [301] and Random Walk Positional Encodings (RWPe) [102], where the latter were also adapted for simplicial complex transformers [302, 303].

Let $0 \leq k \leq \dim K$, where K is a cellular complex. *A cellular k-positional encoding of K_k* is a k-feature map e_k that captures some structural information about K_k within K.

Given feature maps $\{f_i \colon K_i \to \mathbb{R}^{d_i}\}_{i=0}^{\dim K}$ and positional encodings $\{e_i \colon K_i \to \mathbb{R}^{d_{e_i}}\}_{i=0}^{\dim K}$, the input for the first transformer layer is defined as a function

$$\begin{aligned} \mathrm{P} \colon (\mathbb{R}^{d_0})^{K_0} \times (\mathbb{R}^{d_{e_0}})^{K_0} \times \cdots \times (\mathbb{R}^{d_{\dim K}})^{K_{\dim K}} \times (\mathbb{R}^{d_{e_{\dim K}}})^{K_{\dim K}} \\ \longrightarrow (\mathbb{R}^{d_0})^{K_0} \times \cdots \times (\mathbb{R}^{d_{\dim K}})^{K_{\dim K}}, \end{aligned} \tag{7.4}$$

composed of the concatenation of smaller functions per rank

$$\mathrm{P}_k \colon (\mathbb{R}^{d_k})^{K_k} \times (\mathbb{R}^{d_{e_k}})^{K_k} \longrightarrow (\mathbb{R}^{d_k})^{K_k}, \tag{7.5}$$

where P_k combines the signals and the positional encodings for the rank k. Usual functions to perform this combination are

$$\mathrm{SumPE}(F_k, E_k) = F_k\theta_{\mathrm{in},k} + b_{\mathrm{in},k} + E_k\theta_{\mathrm{in,pe}} + b_{\mathrm{in,pe}}$$
$$\mathrm{ConcatPE}(F_k, E_k) = \mathrm{Concat}(F_k, E_k)\theta_{\mathrm{in,pe}} + b_{\mathrm{in,pe}},$$

where $\theta_\bullet$, $b_\bullet \in$ are learnable parameters and E_k is the concatenation of the positional encodings for the rank k. For this chapter, we use $\mathrm{P}_k = \mathrm{ConcatPE}$. Next, we introduce our novel positional encodings on cellular complexes: HodgeLapPE, BSPe, and CC-RWPe plus TopoSlepiansPE, a positional encoding based on the Topological Slepians [304].

Hodge Laplacian Positional Encoding (HodgeLapPE). A naive extension of LapPE introduced in Section 2.2.5 to cellular complexes involves using the unnormalized Hodge Laplacian matrix, instead of the graph Laplacian one. We deem this version *HodgeLapPE*. We use the unnormalized version because normalizing the Hodge Laplacian for dimensions greater than zero is not a trivial task [303]. HodgeLapPE are, however, not a good choice for high-order positional encodings *a priori* due to both a lack of normalization and the

Barycentric Subdivision Positional Encoding (BSPe). BSPe leverages the graph Laplacian eigenvectors of the 1-skeleton of the barycentric subdivisions of the cellular complexes, where the *barycentric subdivision* of a cellular complex K, denoted by $\Delta(K)$, is the order complex of its face poset [305], i.e., the abstract simplicial complex whose set of vertices is the set of cells of K and whose simplices are the totally ordered flags of cells of K. The 1-skeleton of the barycentric subdivision of K is a graph $G = (V, E)$ where V is the set of cells of K and where two vertices σ_1 and σ_2 are connected if one is a face of the other. Thus, the positional encoding of a cell σ is the Laplacian positional encoding of σ seen as a vertex in G.

This positional encoding respects the same theoretical advantages of the LapPE while assigning relative positions to all the cells *at the same time*, and thus relative positions take into account all the cells and not only the cells of a specific dimension.

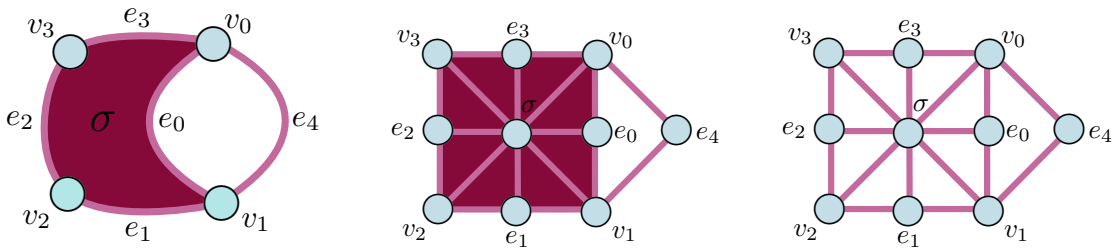


Figure 7.3: Left: A cellular complex $\mathcal{X}$. Center: Barycentric subdivision of $\mathcal{X}$. Right: 1-skeleton of the barycentric subdivision. Each original cell of $\mathcal{X}$ is represented by a node in the 1-skeleton.

The positional encodings satisfying this property are called *global*, in contrast to *local* PEs, where encodings are assigned independently for each dimension.

CC Random Walk PE (CC-RWPe). Similar to BSPe, we propose to extend RWPe to cellular complexes by taking the positional encodings given by the original RWPe for the 1-skeleton of the barycentric subdivision. We denote these *global* positional encodings as *CC-RWBSPe*. We also propose a more sophisticated, local approach, denoted *CC-RWPe*, extending the random walks from [303] to cellular complexes. The full development of the random walk matrix can be found in Section D.2. From the random walk matrix, PEs are taken as in RWPe for each rank of the cellular complex.

Topological Slepians PE (TopoSlepiansPE) [304]. Given an ordered multiset of rank k Slepians $\mathcal{S}_k = \{s_k^i \mid i = 1, \ldots, d_{\text{pe}}\}$, we let the positional encoding of a k-cell σ be $e_k(\sigma)_i = s_{k,\sigma}^i$, the coordinate corresponding to σ of the s_k^i Slepian. If the number of Slepians is lower than d_{e_k}, we pad $\mathcal{S}_k$ with zero vectors elements.

Topological Slepians are signals characterized by their maximal concentration within the topological domain (the cells) and their perfect localization in the corresponding dual domain (the frequencies [306]). This means that the non-zero coordinates of the different Slepians are concentrated only on some of the cells of the complex, with a specific spectral content, assigning a relative position to each cell. Topological Slepians have been shown to be particularly efficient for signal representation and sparse coding tasks [304], making them a valuable tool to devise

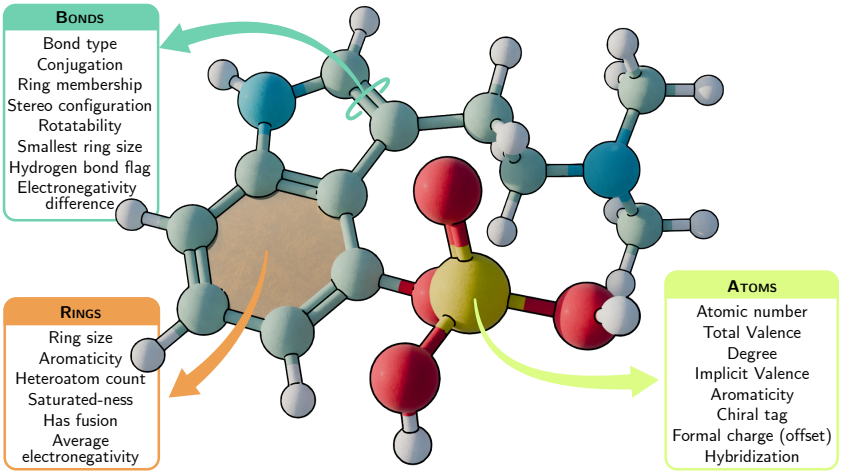


Figure 7.4: Our augmented molecular cellular complex enables our topological transformer to consume richer information than traditional graphs.

PE. In this work, we compute Slepians as in Section 4 of [304]. Although it comes with guaranteed localization properties, it is a local PE, and suffers from permutation variance and eigenvector sign variance, that the transformers need to learn.

7.1.4 Augmented molecular cellular complexes

Recent GDL and TDL approaches to molecular modeling, especially those leveraging graph transformers, have benefited greatly from the integration of ring-level or other higher-order structural motifs (e.g., functional groups, pharmacophores) [155, 307, 308]. A prime example is cyclohexyl rings versus straight aliphatic chains. Chemically, both structures are made of carbon atoms that are sp^3-hybridized and form single σ-bonds. Hence, while atoms and bonds are formally of the same type, ring closure in cyclohexane imposes unique structural constraints (e.g., the well-known chair conformation) that differ from those in a linear alkane.

As such, methods treating each carbon atom and each bond equally often overlook these subtle but crucial differences. Cellular complexes are a natural solution to this issue, as rings can be explicitly regarded as 2-cells (faces) with their peculiar (ring-level) features, and the topology of the complex inherently captures vital structural constraints. Thus, we introduce *augmented molecular cellular complexes* (AMCCs), novel variants of molecular cellular complexes in which atoms are nodes (0-cells), bonds are edges (1-cells), and rings are faces (2-cells). Here, *augmentation*

refers to a curated set of features for all the cells being more exhaustive w.r.t. to the ones in [155], and resulting in significantly better results, shown in our ablation study in Section 7.2.2. An example of an AMCC listing all the features we leverage is depicted in Figure 7.4. Our CT is then naturally able to process and learn for the rich representation offered by AMCCs, as confirmed by the numerical results in Section 7.2. We provide an overview of AMCC features in Section D.4.4.

7.2 Experimental evaluation

We extensively evaluate the proposed transformer, AMCCs and the positional encodings both on the MoleculeNet [309] benchmark as well as in the Graph Classification Benchmark dataset [310]. All our experiments use the AdamW optimizer. For classification tasks, we use the standard cross entropy loss, and for regression, we either use the mean squared error or the mean absolute error, depending on the dataset.

7.2.1 Graph classification benchmark dataset

We begin by evaluating our method on the Graph Classification Benchmark (GCB) [310] in its hard version. As this is a graph dataset, we obtain our cellular complexes by adding all the rings belonging to a cycle basis using the `TopoX` library. Table D.2 presents a comparison of our best topological model using RWBSPe encodings against the state-of-the-art architectures evaluated for the dataset in [310, 311].

Overall, even with a simple lifting, we outperform several graph-based methods in this dataset, and do so without the need for advanced techniques such as graph rewiring, virtual nodes, or learnable bias matrices in the attention mechanism. The results corroborate that leveraging high-order information can be useful even when the underlying domain is a graph and in such an uninformative case, our method falls back gracefully.

Table 7.1: Models and accuracy (↑) for the `GCB` dataset. The first seven model rows represent message-passing architectures, and the next six rows are classic machine learning algorithms.

Model	**Accuracy** (↑)
Graclus [312]	0.690 ± 0.015
NDP [313]	0.726 ± 0.009
DiffPool [314]	0.699 ± 0.019
Top-K [315]	0.427 ± 0.152
MinCutPool [316]	0.738 ± 0.019
ESC + RBF-SVM [317]	0.625 ± 0.046
ESC + L1-SVM [317]	0.722 ± 0.010
ESC + L2-SVM [317]	0.693 ± 0.016
Hist Kernel [318]	0.720 ± 0.000
Jaccard Kernel [318]	0.630 ± 0.000
Edit Kernel [318]	0.600 ± 0.000
Stratedit Kernel [318]	0.600 ± 0.000
CT (ours)	$\mathbf{0.754 \pm 0.017}$

Ablation studies We observe in Table 7.2 that, for six different settings, our CT outperforms the other tested ML models. We also observe that, in general, the PEs derived from the barycentric subdivisions and obtained for all ranks simultaneously (global PE) outperform the rest of the PEs that are computed for each rank in isolation (local PE). The only exception is the HodgeLapPE, that obtains the second- and fourth-best results. From the barycentric subdivision ones, the RWBSPe performs slightly better than the ones based on eigenvectors, which was expected due to the lack of unicity in the computation of the eigenbasis for the PE vectors. We also observe that in general the pairwise attention is superior to the general cellular attention, while the sparse attention dominates over the dense one, indicating that a strong bias is needed in this dataset for achieving the best results.

Table 7.2: Ablating attention and PEs on GCB.

Attn. \ **PE**	**BSPe**	**RWPe**	**HodgeLapPE**	**TopoSlepians**	**RWBSPe**
$\mathcal{A}^s_{k_s \to k_t}$	0.745 ± 0.010	0.740 ± 0.016	$\mathbf{0.745 \pm 0.013}$	0.703 ± 0.032	$\mathbf{0.754 \pm 0.017}$
$\mathcal{A}^s_g$	$\mathbf{0.745 \pm 0.006}$	0.728 ± 0.013	$\mathbf{0.751 \pm 0.006}$	0.723 ± 0.003	0.733 ± 0.010
$\mathcal{A}^d_{k_s \to k_t}$	0.707 ± 0.012	0.712 ± 0.012	0.712 ± 0.024	0.681 ± 0.011	0.698 ± 0.021

7.2.2 MoleculeNet

Baselines. Following [319], we now compare our work on various subsets[1] of MoleculeNet [309] against the strong graph baselines of GCN [320], GIN [321], SchNet [322], MGCN [323] and DMPNN [324], as well as the Pytorch Geometric implementation of GPS [299] modified to use our AMCC except the 2^{nd}-order features. We also include classical RF [325] and SVM [326] performances for reference.

Results on MoleculeNet. Table 7.4 presents our main results across various tasks where we report Avg. AUC-ROC for classification and root mean square error (RMSE) for regression. Our CT, leveraging the useful information provided by AMCC and barycentric subdivision PEs, can often surpass all methods, or remain on par. Notably, GPS, in the lack of geometric features and limited hyperparameter search, significantly underperforms. This highlights the capability of our CT in extracting strong topological signals and the importance of our higher-order AMCC. We provide further details in Section D.4 and now study our AMCC and PEs in comparison to [299] and the representation of [155] before reporting runtime.

When is AMCC beneficial? Table 7.3 compares our molecular representation, AMCC, as well as the one of ETNN [155] both for GPS and for our cellular transformer on the FreeSolv subset of MoleculeNet. Notably, while our richer higher-order information brings more benefits, even compared to a SotA molecule representation, in the case of our cellular transformer, it is also useful for graph transformers (GPS) which can only leverage information on nodes and edges. This is because AMCC extracts richer bond and node features previously not captured in the SotA.

[1] We exclude the QM7, QM8, and QM9 due to their dependency on geometric coordinates, which are not in the focus of this work.

Table 7.3: Evaluating different molecule models (AMCC and that of ETNN) and pairwise vs higher-order transformers on the FreeSolv dataset of MoleculeNet. We report root mean square errors.

Molecule Model		ETNN [155]	AMCC (ours)
Method	GPS [299]	0.778 ± 0.286	$\underline{0.759 \pm 0.043}$
	CT (ours)	0.776 ± 0.033	$\mathbf{0.662 \pm 0.045}$

Table 7.4: Evaluations across MoleculeNet benchmark datasets. All values are reported in percentage (%) with standard deviations over 3 runs.

Dataset	BBBP	Tox21	ClinTox	HIV	BACE	SIDER	MUV	FreeSolv	ESOL	Lipo
Molecules	2,039	7,831	1,478	41,127	1,513	1,427	93,087	642	1,128	4,200
Tasks	1	12	2	1	1	27	17	1	1	1
Metric	Average AUC-ROC (↑)							RMSE (↓)		
RF	71.4 ± 0.0	76.9 ± 1.5	71.3 ± 5.6	78.1 ± 0.6	$\underline{86.7 \pm 0.8}$	$\mathbf{68.4 \pm 0.9}$	63.2 ± 2.3	-	-	-
SVM	72.9 ± 0.0	$\mathbf{81.8 \pm 1.0}$	66.9 ± 9.2	$\underline{79.2 \pm 0.0}$	86.2 ± 0.0	$\underline{68.2 \pm 1.3}$	67.3 ± 1.3	3.14 ± 0.00	1.50 ± 0.00	0.82 ± 0.00
GCN	71.8 ± 0.9	70.9 ± 2.6	62.5 ± 2.8	74.0 ± 3.0	71.6 ± 2.0	53.6 ± 3.2	71.6 ± 4.0	2.87 ± 0.14	1.43 ± 0.05	0.85 ± 0.08
GIN	65.8 ± 4.5	74.0 ± 0.8	58.0 ± 4.4	75.3 ± 1.9	70.1 ± 5.4	57.3 ± 1.6	71.8 ± 2.5	2.76 ± 0.18	1.45 ± 0.02	0.85 ± 0.07
SchNet	$\underline{84.8 \pm 2.2}$	$\underline{77.2 \pm 2.3}$	71.5 ± 3.7	70.2 ± 3.4	76.6 ± 1.1	53.9 ± 3.7	71.3 ± 3.0	3.22 ± 0.76	1.05 ± 0.06	0.91 ± 0.10
MGCN	$\mathbf{85.0 \pm 6.4}$	70.7 ± 1.6	63.4 ± 4.2	73.8 ± 1.6	73.4 ± 3.0	55.2 ± 1.8	70.2 ± 3.4	3.35 ± 0.01	1.27 ± 0.15	1.11 ± 0.04
D-MPNN	71.2 ± 3.8	68.9 ± 1.3	$\mathbf{90.5 \pm 5.3}$	75.0 ± 2.1	85.3 ± 5.3	63.2 ± 2.3	$\underline{76.2 \pm 2.8}$	2.18 ± 0.91	$\underline{0.98 \pm 0.26}$	$\mathbf{0.65 \pm 0.05}$
GPS	60.4 ± 2.6	63.6 ± 0.6	58.8 ± 7.7	66.8 ± 1.2	72.4 ± 1.0	54.3 ± 0.1	68.3 ± 1.3	$\underline{0.99 \pm 0.04}$	1.14 ± 0.23	0.84 ± 0.4
CT (ours)	71.2 ± 0.5	74.4 ± 1.0	$\underline{86.6 \pm 6.0}$	$\mathbf{79.7 \pm 0.7}$	$\mathbf{86.8 \pm 2.7}$	60.5 ± 1.8	$\mathbf{78.9 \pm 2.0}$	$\mathbf{0.66 \pm 0.04}$	$\mathbf{0.87 \pm 0.03}$	$\underline{0.69 \pm 0.02}$

Topological positional encodings. We now evaluate the impact of different positional encodings on our cellular transformers in Table 7.5. While most methods perform similarly, we see that random walk driven PEs coupled with sum-attention performs the best for this dataset. This is also what we use in practice in Table 7.4.

Table 7.5: PE ablations for our CT on the FreeSolv subset.

Attention	PE	RMSE
$\mathcal{A}^d_{k_s \to k_t}$	BSPe	0.753 ± 0.024
$\mathcal{A}^d_{k_s \to k_t}$	HodgeLapEig	0.717 ± 0.042
$\mathcal{A}^d_{k_s \to k_t}$	RWBSPe	$\underline{0.702} \pm \underline{0.032}$
$\mathcal{A}^d_{k_s \to k_t}$	RWPe	$\mathbf{0.662 \pm 0.045}$
$\mathcal{A}^d_{k_s \to k_t}$	TopoSlepiansPE	0.810 ± 0.183

Runtime. Table 7.6 reports the average Multiply-Accumulate Operations across all ablations for each pair of architectures (CT and GPS) and molecular modeling strategy (ETNN and AMCC) on the FreeSolv subset. While our method is slower

than GPS due to the added dimensions, it is still within an acceptable range. Moreover, our AMCC does not incur added overhead compared to ETNN [155].

Table 7.6: Average MACs (Multiply-Accumulate Ops) per epoch. For GPS, we report the avg. for each PE (LapPE, RWPE). For CTs, the avg. is over each of our PEs and attention (pairwise, general).

Molecular Model	AMCC	ETNN
CT	$80,026,112 \pm 1,804,280$	$82,350,068 \pm 2265684$
GPS	$49,930,176 \pm 0$	$52,855,832 \pm 0$

7.3 Conclusions

In this chapter, we introduced the Cellular Transformer (CT), a high-order neural network that introduces attention mechanisms to high-order domains, proving its utility in molecular tasks by leveraging molecular strucures beyond binary relationships between atoms using cell AMCCs. With novel topological positional encodings, our approach captures higher-order molecular interactions without requiring heuristics or graph rewiring. Our results on MoleculeNet benchmarks demonstrate state-of-the-art or competitive performance across molecular property prediction tasks while maintaining architectural simplicity.

Limitations and future work. Our novel CTs excel at leveraging molecular topological structures but are less effective in directly modeling geometry. As a remedy, we aim to develop equivariant-CTs that incorporate physical conformations. Finally, we plan to learn the topological features, scaling to larger biomolecules, and extending TDL to scientific domains like materials science and drug discovery.

8. Discussion

Contents

In this thesis, we have systematically investigated multiple facets of topological deep learning, ranging from theoretical frameworks for understanding neural network generalization to the development of novel architectures capable of high-order learning. Our results demonstrate that topology offers powerful tools for understanding, designing, and enhancing neural networks beyond traditional Euclidean spaces. The work presented in this dissertation advances the field of TDL through the following key contributions:

(1) **Establishing new relationships between topology and neural network generalization:** Section 4.1 expanded the topological analysis of functional graphs initiated by [42, 43] by introducing dual sampling mechanisms in both data and neuron spaces, effectively addressing computational complexity barriers that previously limited application of TDA analysis on functional graphs to more general datasets and network architectures. Our analysis of the PGDL dataset revealed a striking near-linear correlation between the generalization capacity of the analyzed networks and specific topological descriptors derived from their functional graphs when fixing some of their hyperparameters. This finding allowed us to provide a moderately successful simple regression model for predicting generalization performance by means of topological descriptors of persistence diagrams computed from the functional graphs of the networks. The material of this chapter is based on joint work with Xavier Arnal Clemente, Carles Casacuberta, Meysam Madadi, Ciprian Corneanu, and Sergio Escalera.

(2) **Developing topology-informed regularization methods to enhance neural network training:** Drawing from our PGDL findings and established neurobiological principles regarding neural correlation in brain structures, Section 4.2 introduced novel regularization techniques leveraging persistent homology and minimum spanning trees to strategically decorrelate neural activations. These methods demonstrated consistent improvements in performance across multiple architectures and datasets, even outperforming some of the usual regularization terms (Lasso and Ridge) in DL. The material of this chapter is based on joint work with Carles Casacuberta and Sergio Escalera.

(3) **Advancing the theoretical understanding of the expressivity of persistent homology in graph analysis:** Chapter 5 made dual contributions to the area of graph learning expressivity analysis. First, we introduced the concept of equivariant filtrations and provided formal proofs that persistent homology computed with these filtrations can be as expressive as k-FWL algorithm for any $k > 0$. Additionally, we established some upper bounds for several fundamental graph properties in terms of graph-based persistent homology. Second, our comprehensive experimental evaluation for graph isomorphism detection and property prediction empirically demonstrated that PH-based methods can capture structural information beyond the capabilities of conventional graph neural networks, providing both theoretical and practical evidence for the representational power of TDA-based learning approaches. The material of this chapter is based on joint work with Bastian Rieck.

(4) **Creating a benchmark dataset for high-order learning:** Chapter 6 developed MANTRA, a dataset of manifold triangulations that inherently contains high-order structures that learning models must capture to achieve optimal performance. Through extensive empirical evaluation, we demonstrated that current SOTA models for graphs, simplicial complexes, and cellular complexes still struggle with topological tasks requiring high-order

learning. However, our results also revealed that high-order learning frameworks consistently outperform graph-based models, evidencing the need for high-order architectures and motivating future research on TDL models for problems involving high-order structures such as meshes and complex networks. The material of this chapter is based on joint work with Ernst Röell, Daniel Bīn Schmid, Mathieu Alain, Bastian Rieck, Carles Casacuberta, and Sergio Escalera.

(5) **Generalizing attention mechanisms to high-order data:** Chapter 7 introduced the Cellular Transformer, a novel approach that extends the attention mechanism to high-order domains, specifically cellular complexes, together with a novel set of positional encodings for the cellular data. Our experimental results demonstrated that, when applied to appropriate cellular representations of molecular data, the Cellular Transformer consistently outperforms traditional SOTA models across multiple MoleculeNet benchmarks. This architectural innovation bridges the gap between the capacity of transformers on usual learning tasks such as text and images and the structural complexity of high-order data representations. The material of this chapter is based on joint work with Melih Barsbey, Andac Demir, Carles Casacuberta, Pablo Hernández-García, David Pujol-Perich, Sarper Yurtseven, and Claudio Battiloro, Tolga Birdal, Sergio Escalera, and Mustafa Hajij.

8.1 Limitations and future work

As mentioned in the Introduction, the significance of topological methods in deep learning was expected: neural networks and data are rich in geometric structure, and topology is the mathematical language to describe the most fundamental properties of these structures. However, we also saw, especially in Chapters 3 and 6, that there is a long way to go to fully harness the power of topological methods in deep learning. From our work, we have identified several limitations and future research directions that we believe will be key to the success of TDL in the coming years.

(1) *Computational scalability of PH:* Persistent homology, while theoretically powerful, remains computationally demanding when applied to large-scale datasets, particularly in non-Euclidean settings. Recent advances, such as [327], which introduces Flood complex filtrations enabling the analysis of point clouds with millions of points in Euclidean spaces, and [328], which accelerates the computation of one-dimensional persistence diagrams of point clouds in Euclidean spaces, are beginning to establish TDA as a viable tool for large-scale data analysis. Nevertheless, significant challenges remain. Future work should prioritize the development of algorithmically efficient approaches for PH computation, potentially through:

- Parallelized and distributed algorithms specifically designed for topological calculations. Some recent works in this direction are [329] and [330], which introduces GPU enhancements for Ripser and quantum algorithms to compute persistent Betti numbers, respectively;
- Mathematically-grounded approximation techniques that preserve key topological features while reducing computational overhead, such as the use of bootstrapping and confidence sets for PH [331];
- Alternative topological invariants with more favorable computational profiles, such as Euler characteristics (transforms) [332, 333].

(2) *Extending topological interpretability to modern architectures:* Our experiments, like most work in topological DL focused on interpretability, have primarily addressed relatively fundamental neural network architectures and datasets. There remains a critical need to extend these approaches to contemporary state-of-the-art architectures including large-scale transformer models in multimodal or massive textual datasets for both predictive and generative tasks. Such extensions will be essential for establishing the broader validity and applicability of topological interpretability frameworks.

(3) *Expanding real-world applications of TDL:* While MANTRA represents an important step forward as a benchmark for high-order learning, both it and related datasets remain limited in scope. To advance TDL research, there is an urgent need for diverse, large-scale, real-world datasets with intrinsic high-order structure. Some examples of such datasets could include mesh datasets or complex biological networks including protein and molecular systems.

(4) *Advancing mechanistic interpretability through topological analysis:* While more speculative than the previous directions, we firmly believe that TDA can play a crucial role in mechanistic interpretability of neural networks. Works like [334, 335] and derivatives are a good starting point, but the topic is still in its infancy and there is a lot of room for exploration. A motivating example of where to apply TDL for mechanistic interpretability can be seen in [20], where a clear evolution of the topology of the *feature space* of a network depending on the evolution of some hyperparameters can be observed, where points are feature embeddings and distances are given between the dot product of the embeddings[1].

(5) *Development of better positional encodings for high-order data:* The success of the Cellular Transformer in our experiments suggests that TDL can play a crucial role in high-order data learning. However, the presented positional encodings are still far from being optimal, and are a straightforward extension of the positional encodings used in the graph case. Better positional encodings leveraging, for example, more sophisticated random walks on cellular complexes could be a good starting point for future work [336, 337].

(6) *Design of new topology-aware architectures:* MANTRA has shown the limitations of current SOTA architectures to tackle topological tasks. Topology-aware architectures have the potential to solve MANTRA and advance research

[1] In this context, a feature embedding f_i is the projection of a specific input value x_i where the model and the input and output spaces satisfies several constraints needed to interpret the mechanistic interpretation results from [20]. For a more detailed explanation of the feature geometry graph, see [20].

in (algebraic) topology, as it is being done with other mathematical fields such as geometry, knot theory, or algebra [338–340].

8.2 Broader impact and conclusions

Topological perspectives offer unique insights into complex systems across scientific domains such as biology and physics, and have proved fundamental to advancing our knowledge about the world surrounding us. By advancing our understanding on how topology interacts with deep learning on the architectural and data levels, we have contributed to the broader goals of demystifying the black box of neural networks and developing more principled and data-aligned deep learning models.

In the first case, we observed that it seems to be a close relationship between the behaviour and performance of neural networks and the topology of their functional graphs that transcends the specific architecture and dataset used, paving the way for a topological intepretability theory that is data- and architecture-agnostic.

In the second case, we have shown that the high-order learning frameworks which incorporate structural inductive biases that respect the topological nature of data, particularly the Cellular Transformer, can yield significant performance improvements. This principle —that architectural design should reflect the inherent structure of the problem domain— has broad implications for machine learning research and challenges the current trend of designing architectures that are agnostic to the problem domain.

In conclusion, TDL stands at a promising frontier of deep learning research, offering powerful tools for understanding, designing, and improving neural networks. While significant challenges remain, the rapid development of the field suggests a bright future where topological methods become standard components of the deep learning toolkit. We hope this thesis serves as both a contribution to the current state of the art and an inspiration for future researchers exploring this fascinating intersection of topology and machine learning.

Appendices

A. Supplemental material for Chapter 4

A.1 Predicting the generalization gap

This section contains an analysis per label of persistence diagrams in dimensions 0 and 1. The datasets that we used to recreate functional graphs were the restriction of the test sets to each label. We computed accuracy for each of these test subsets, and plotted persistence diagrams corresponding to those neural networks that achieved the maximum and minimum accuracies on test subsets per label for dimensions 0 and 1. The results can be seen in Figures A.1, A.2, A.3 and A.4. These results are consistent with what we found in persistence diagrams computed with the whole training dataset. Thus we see that distinction between inputs of different labels does not have a substantial influence on the distribution of points in persistence diagrams.

For a more convenient visualization, persistence diagrams in dimension 0 have been replaced with lifetime density curves, calculated by means of Gaussian kernels. Lifetime values are equal to death values for zero-homology generators.

It can be seen in Figure A.3 and Figure A.4 that increased accuracy values for Task 2 match with scattering of points downwards the diagonal of the persistence diagram in dimension 1 and with a lower average life in dimension 0. This pattern is apparently not consistent with other architectures, such as those used in Task 1. This is explained by the splitting of network types into clusters as observed in Figure 4.3, since for Task 2 the regression line for average deaths has negative slope in each cluster, while it has positive slope if clustering is not taken into account.

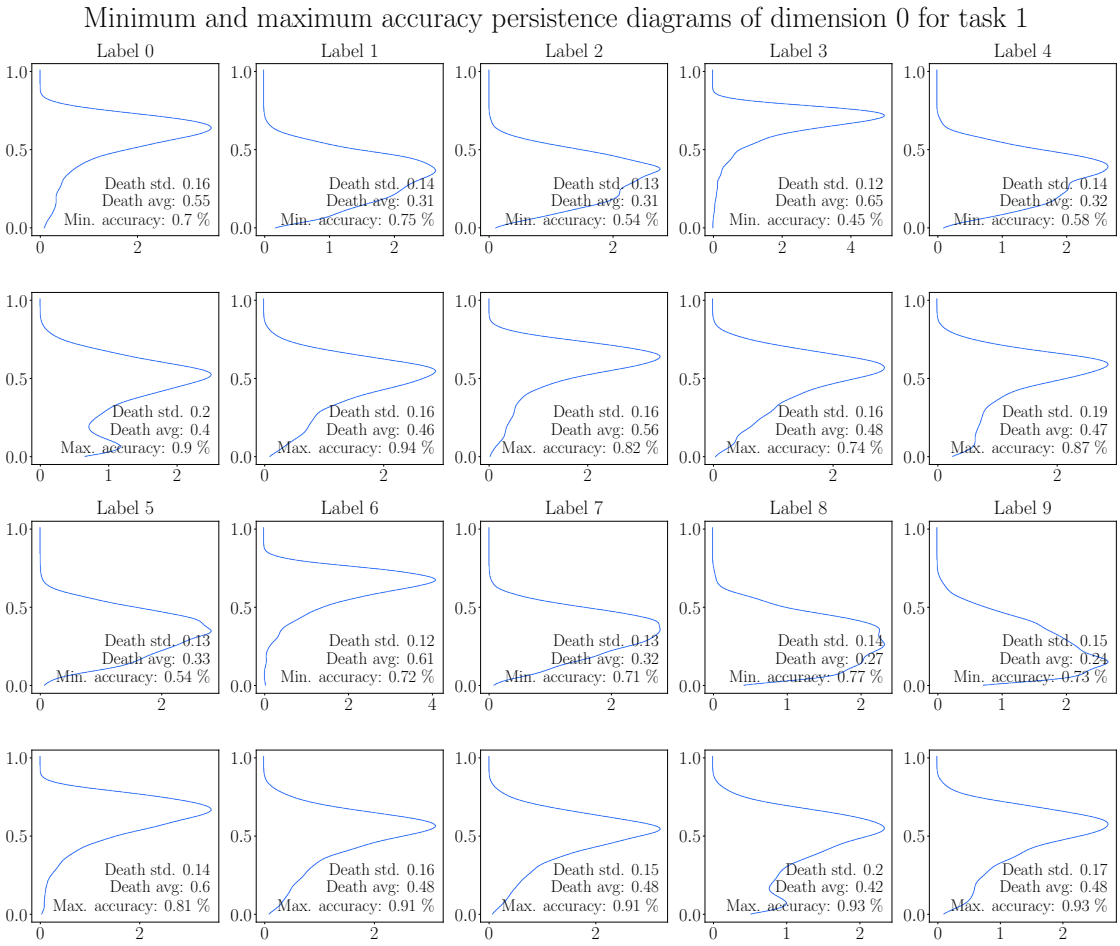


Figure A.1: Lifetime densities in persistence diagrams in homological dimension zero of 96 VGG-like neural networks with minimum and maximum accuracies on the test set per label for Task 1.

A.2 Decorrelating neurons using persistence

A.2.1 Equivalence between zero-dimensional persistence and minimum spanning trees

In this appendix, we prove that there is a bijection between the set of non-diagonal points with finite death parameter in the zero-dimensional persistence diagram of a finite set P equipped with a dissimilarity function $d\colon P \times P \to \mathbb{R}_{\geq 0}$ and the set of weights of a minimum spanning tree of the complete weighted graph whose vertices are the elements of P and whose weights are given by d. For this section, we consider a dissimilarities *dissimilarity* d which are symmetric and satisfy $d(p,p) = 0$ for all $p \in P$. An example is the Euclidean distance function when P is a set of

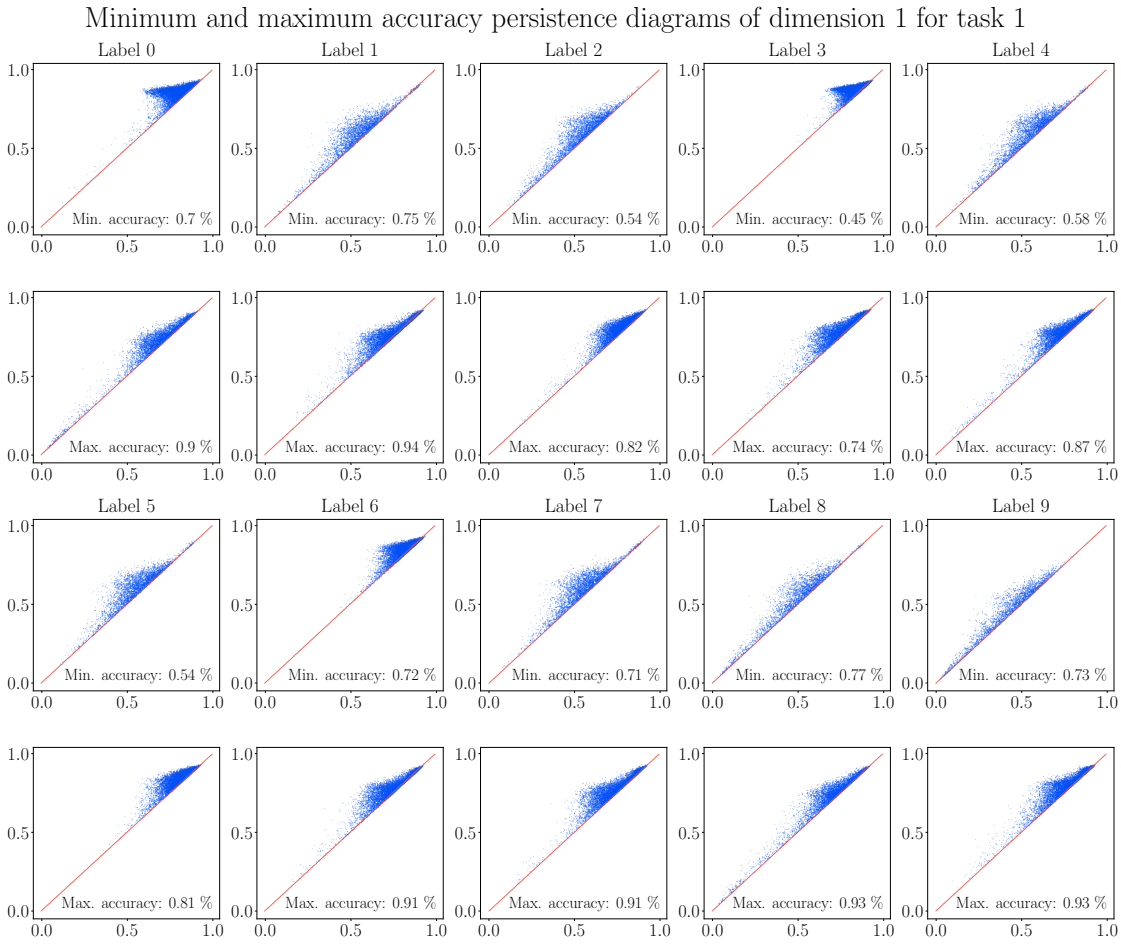


Figure A.2: Persistence diagrams in homological dimension one of 96 VGG-like neural networks with minimum and maximum accuracies on the test set per label for Task 1.

points in Euclidean space, and another example is the correlation dissimilarity used in this article when P is a set of neurons of a neural network.

For the following proofs, we compute simplicial homology with coefficients in the field of two elements.

For a connected, vertex-weighted graph $G = (V, E, w_V)$, we denote the multiset of weights of G by $\mathcal{W}_G = \{w_V(e) : e \in E\}$. A *minimum spanning tree* of G is a subgraph without cycles containing all the vertices with the minimum possible total edge weight. Kruskal's algorithm [341] finds a minimum spanning tree of every weighted graph G and shows that the multiset of weights of all minimum spanning trees of G coincide.

Theorem 6. Let (P, d) be a finite set and let $d\colon P \times P \to \mathbb{R}_{\geq 0}$ be a sym-

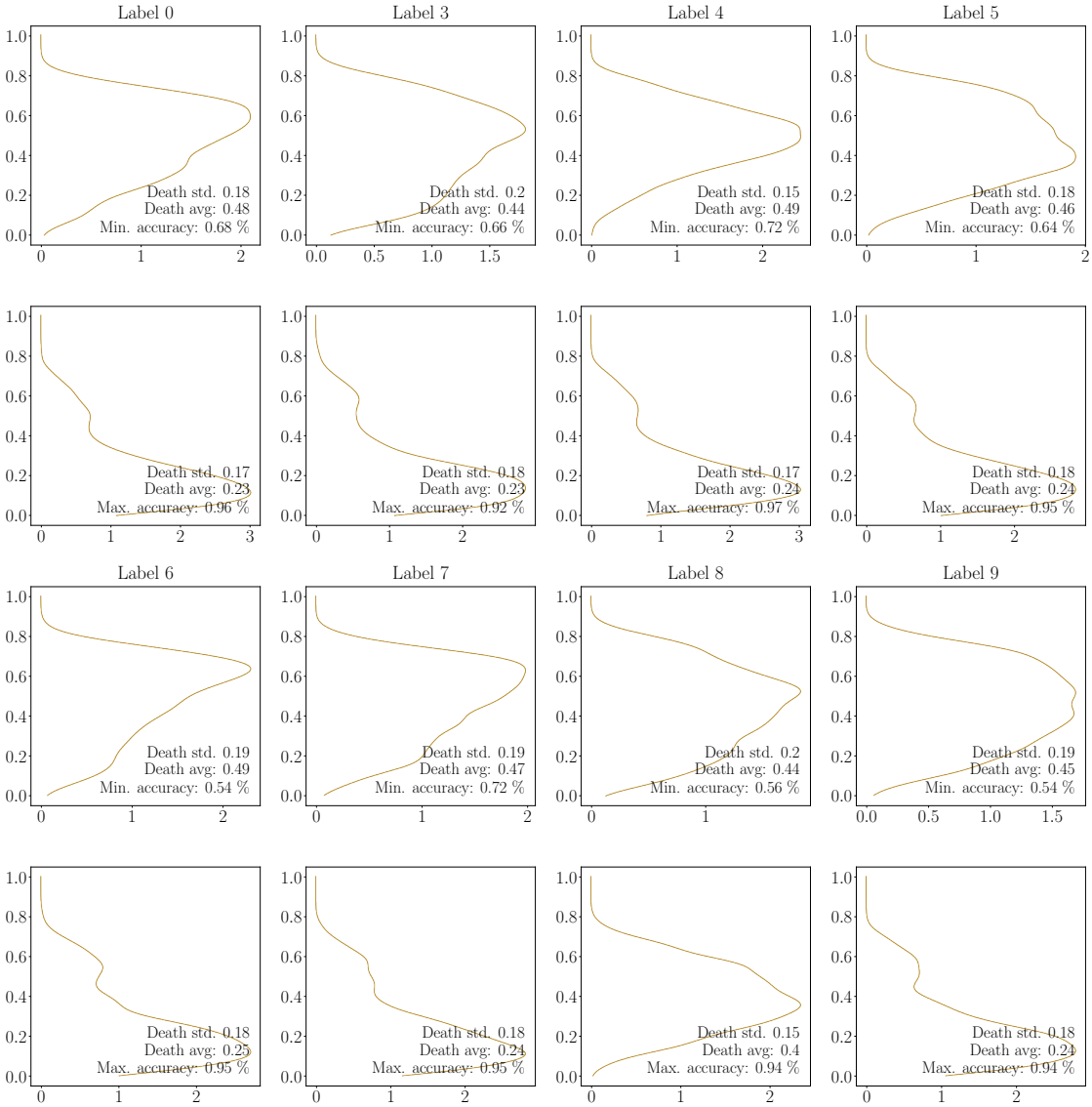


Figure A.3: Lifetime densities in persistence diagrams in homological dimension zero of 54 *network in network* architectures with minimum and maximum accuracies on the test set per label for Task 2.

metric function such that $d(p,p) = 0$ for all $p \in P$. Let G be the complete weighted graph with set of vertices P and weights given by d. Let $D_{<\infty} = \{(b_i, d_i) \in D(\mathbb{V}_0(\mathrm{VR}(P,d))) : b_i < d_i < \infty\}$ be the multiset of finite, non-diagonal points of the Vietoris-Rips zero-dimensional persistence diagram of $(Points, d)$. Then all points (b_i, d_i) in $D_{<\infty}$ have $b_i = 0$ and

$$\{d_i : (0, d_i) \in D_{<\infty}\} = \{w \in \mathcal{W}_T : w > 0\}$$

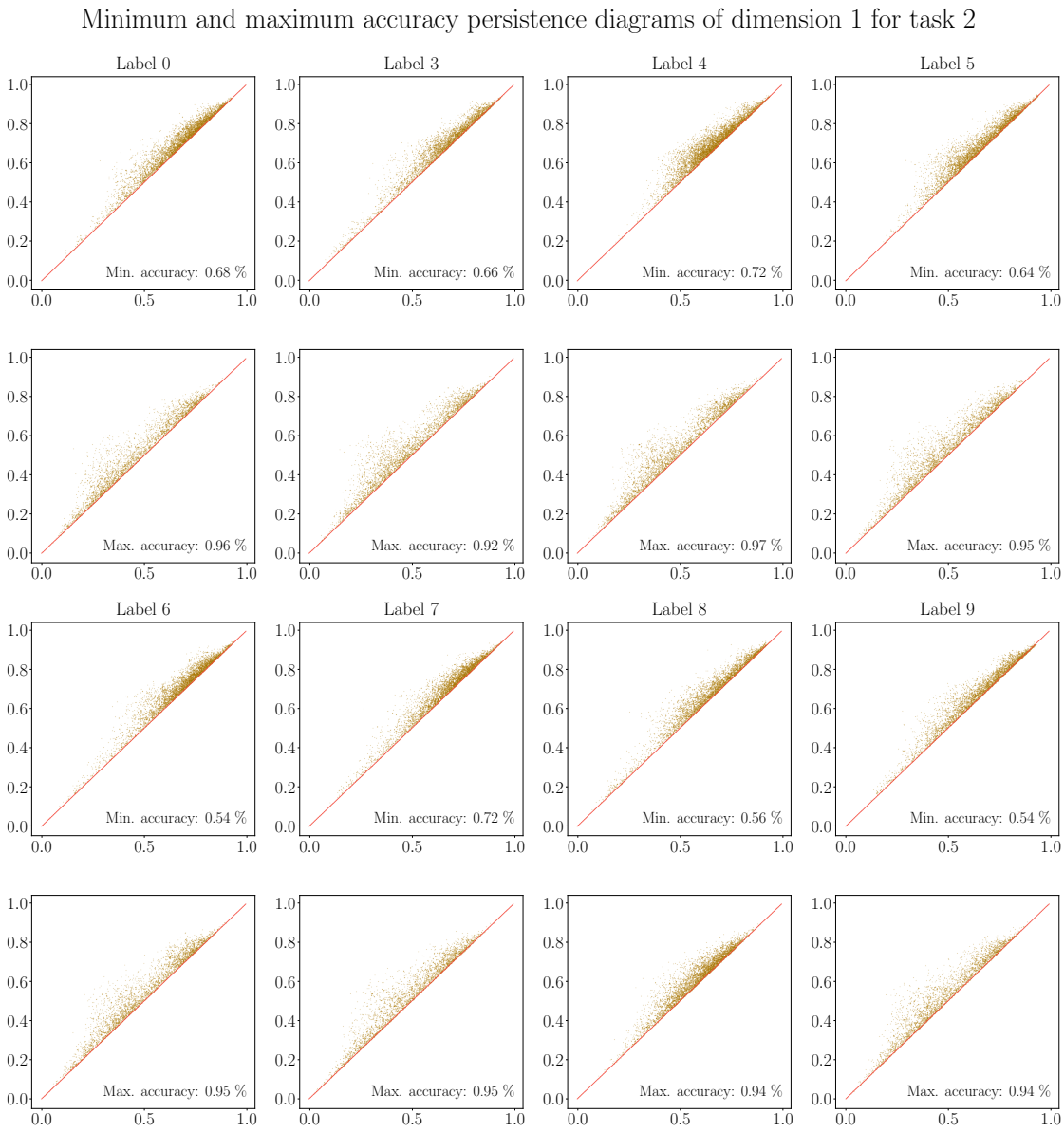


Figure A.4: Persistence diagrams in homological dimension one of 54 *network in network* architectures with minimum and maximum accuracies on the test set per label for Task 2.

for any minimum spanning tree T of G.

Proof. Note first that, since the function d takes non-negative values and $d(p,p) = 0$ for all $p \in P$, we have that $\mathrm{VR}_r(P,d) = \emptyset$ for $r < 0$ and $\{p\} \in \mathrm{VR}_0(P,d)$ for all $p \in P$. This means that all connected components of $\mathrm{VR}(P,d)$ are born at $r = 0$ and consequently all $(b_i, d_i) \in D(\mathbb{V}_0(\mathrm{VR}(P,d)))$ have $b_i = 0$. The corresponding death values d_i are filtration levels at which connected components merge. Moreover, $\mathrm{VR}(P,d)$ becomes connected eventually, since P is finite. Hence there is a single

point in $D(\mathbb{V}_0(\mathrm{VR}(P,d)))$ with $d_i = \infty$.

Write $D_{<\infty} = \{(0,d_1),\dots,(0,d_n)\}$ with $d_1 \leq \cdots \leq d_n$ without loss of generality, counted with the respective multiplicities. For each connected component C_i in $\mathrm{VR}_0(P,d)$, choose a minimum spanning tree T_i with total weight 0, and write the edges of T_i as $e^i_1,\dots,e^i_{k_i}$. Next, order the edges of the complete graph G in such a way that the first elements of the list are $e^1_1,\dots,e^1_{k_1},\dots,e^n_1,\dots,e^n_{k_n}$ in any order.

For each point $(0,d_i)$ in $D_{<\infty}$, there is at least one edge e_i in G with weight d_i connecting two previously separated connected components in the Vietoris-Rips filtration. Place the edges $e_1,\dots,e_n$ before all the edges in G with their same weights. If there exist $i \neq j$ such that $w(e_i) = w(e_j)$, order them arbitrarily and place them consecutively. Then, by applying Kruskal's algorithm, we obtain a minimum spanning tree T of G containing the edges

$$\left\{e^1_1,\dots,e^1_{k_1},\dots,e^n_1,\dots,e^n_{k_n},e_1,\dots,e_n\right\}.$$

Therefore, $\{w \in \mathcal{W}_T : w > 0\} = \{d_i : (0,d_i) \in D_{<\infty}\}$. Since the multisets of weights of all minimum spanning trees of G coincide, the equality is satisfied for any minimum spanning tree T of G, as claimed. □

A.2.2 Differentiability of functions on persistence diagrams

In this section, we prove Theorem 1 using methods and results from [28]. We consider finite ordered sets $P \subseteq \mathbb{R}^n$ with c elements, where $n \geq 2$ and $c \geq 2$. Each such set P corresponds to an element $(p_1,\dots,p_c) \in \mathbb{R}^{cn}$, where $p_i \in \mathbb{R}^n$ denotes the ith point of P.

Given points $p_1,\dots,p_c$ in $\mathbb{R}^n$, where $c \geq 2$, the *covariance* between p_i and p_j is

$$\mathrm{cov}(p_i,p_j) = \frac{1}{n}\sum_{k=1}^{n}(p_{i,k}-\bar{p}_i)(p_{j,k}-\bar{p}_j),$$

where $\bar{p}_i = \frac{1}{n}\sum_{k=1}^{n} p_{i,k}$. Since the covariance function is polynomial on the entries, the set

$$\mathcal{D}_{c,n} = \{(p_1,\dots,p_c) \in \mathbb{R}^{cn} : \mathrm{cov}(p_i,p_j) \neq 0 \text{ for all } i,j \in \{1,\dots,c\}\}$$

is open and dense in $\mathbb{R}^{cn}$.

Suppose given a dissimilarity $d\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$. In the case of persistence diagrams computed from filtrations, a persistence diagram in homological dimension k can be viewed as a function $B_k = D_k \circ f$ defined on $\mathbb{R}^{cn}$ where $D_k = D \circ \mathbb{V}_k$ taking values in the set **Diag** of families of points with multiplicities in the upper quadrant of $\mathbb{R}^2$ extended with points at infinity:

$$\mathbb{R}^{cn} \xrightarrow{f} \mathbb{R}^K \xrightarrow{D_k} \mathbf{Diag}. \tag{A.1}$$

Here we denote by $\mathbb{R}^K$ the set of all functions $f\colon K \to \mathbb{R}$, where K is the collection of nonempty subsets of $\{1, \ldots, c\}$, which we view as faces of a $(c-1)$-dimensional simplex. In the case of Chapter 4, the function f is the Vietoris–Rips filtration defined as

$$f(P)(\sigma) = \max_{i,j \in \sigma} d(p_i, p_j)$$

for $\sigma \in K$ and $P = \{p_1, \ldots, p_c\}$. The function D_k assigns to each function $f\colon K \to \mathbb{R}$ the corresponding Vietoris–Rips persistence diagram in homological dimension k, where f is treated as a filtering function on the faces of a $(c-1)$-dimensional simplex. For the whole section, we consider persistence diagrams containing the diagonal Δ as in Section 2.1.10, even when not denoting it explicitly with the δ superscript.

Differentiability of functions valued in **Diag** is defined in [28] as follows. For $m, \ell \in \mathbb{Z}_{\geq 0}$, consider the quotient map $Q_{m,\ell}\colon \mathbb{R}^{2m} \times \mathbb{R}^\ell \to \mathbf{Diag}$ sending each point

$$\tilde{D} = (x_1, y_1, \ldots, x_m, y_m, z_1, \ldots, z_\ell) \in \mathbb{R}^{2m} \times \mathbb{R}^\ell$$

to the diagram obtained by forgetting the order of the points:

$$Q_{m,p}(\tilde{D}) = \{(x_i, y_i)\}_{i=1}^m \cup \{(z_j, \infty)\}_{j=1}^\ell \cup \Delta^\infty.$$

Let $\mathcal{M}$ be a smooth manifold and let $B\colon \mathcal{M} \to \mathbf{Diag}$ be any map. For $x \in \mathcal{M}$ and $r \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$, the map B is said to be *r-differentiable* at x if there exists an open neighborhood U of x and there exist integers $m, \ell \in \mathbb{Z}_{\geq 0}$ and a map $\tilde{B}\colon U \to \mathbb{R}^{2m} \times \mathbb{R}^\ell$ of class $\mathcal{C}^r$ such that $B = Q_{m,\ell} \circ \tilde{B}$ on U. Similarly, for a smooth

manifold $\mathcal{N}$, a map $V\colon \mathbf{Diag} \to \mathcal{N}$ is said to be *r-differentiable* at a diagram D, where $r \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$, if for all $m, \ell \in \mathbb{Z}_{\geq 0}$ and all $\tilde{D} \in \mathbb{R}^{2m} \times \mathbb{R}^{\ell}$ such that $Q_{m,\ell}(\tilde{D}) = D$ the map $V \circ Q_{m,\ell}\colon \mathbb{R}^{2m} \times \mathbb{R}^{\ell} \to \mathcal{N}$ is $\mathcal{C}^r$ on an open neighborhood of $\tilde{D}$.

As proved in [28, Proposition 3.14], if a function $B\colon \mathcal{M} \to \mathbf{Diag}$ is r-differentiable at $x \in \mathcal{M}$ and another function $V\colon \mathbf{Diag} \to \mathcal{N}$ is r-differentiable at $B(x)$, then $V \circ B\colon \mathcal{M} \to \mathcal{N}$ is $\mathcal{C}^r$ at x as a map between smooth manifolds.

In what follows, we consider the projections for $i, j \in \{1, \ldots, c\}$,

$$\pi_{i,j}\colon \mathbb{R}^{cn} \to \mathbb{R}^n \times \mathbb{R}^n, \qquad \pi_{i,j}(p_1, \ldots, p_c) = (p_i, p_j).$$

Proposition 7. Let $d\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ be a dissimilarity which is $\mathcal{C}^r$ on an open set $U \subseteq \mathbb{R}^n \times \mathbb{R}^n$, where $r \geq 0$. Let $p = (p_1, \ldots, p_c) \in \mathbb{R}^{cn}$ such that $p \in \pi_{i,j}^{-1}(U)$ for all $i, j \in \{1, \ldots, c\}$. Suppose that $d(p_i, p_j) \neq d(p_k, p_l)$ when $\{i, j\} \neq \{k, l\}$, where $i, j, k, l \in \{1, \ldots, c\}$. Then the function $B_k = D_k \circ f$ defined in Equation (A.1) is r-differentiable at p.

Proof. Since $d(p_i, p_j) \neq d(p_k, p_l)$ for $\{i, j\} \neq \{k, l\}$, the values $d(p_i, p_j)$ for $i \neq j$ are strictly ordered. As the projections $\pi_{i,j}$ are $\mathcal{C}^\infty$ and d is $\mathcal{C}^r$ on U, and $\pi_{i,j}(p) \in U$, we infer that $d \circ \pi_{i,j}$ is $\mathcal{C}^r$ in p. Since, in particular, $d \circ \pi_{i,j}$ is continuous, there is a neighbourhood U' of p where the order of the values $d(p'_i, p'_j)$ remains the same. Then $f(p)$ and $f(p')$ induce the same preorder on the set of simplices of K for every $p' \in U \cap \bigcap_{i,j} \pi_{i,j}^{-1}(U)$. Hence, the hypotheses of [28, Theorem 4.7] hold and B_k is r-differentiable at p. □

In what follows, we denote, for a given dissimilarity d,

$$\mathfrak{D}_{c,n} = \{(p_1, \ldots, p_c) \in \mathcal{D}_{c,n} : d(p_i, p_j) \neq d(p_k, p_l) \text{ if } \{i, j\} \neq \{k, l\}\}.$$

We note that, if the dissimilarity $d(x, y) = 1 - |\mathrm{corr}(x, y)|$ is chosen, then $\mathfrak{D}_{c,n}$ is an open dense subset of $\mathbb{R}^{cn}$, since $d(p_i, p_j) = d(p_k, p_l)$ precisely when $|\mathrm{corr}(p_i, p_j)| = |\mathrm{corr}(p_k, p_l)|$, and the square of correlation is a rational function.

Proposition 8. Let $c, n \geq 2$, $k \in \mathbb{Z}_{\geq 0}$, and $d\colon \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ be the dissimilarity given by $d(x, y) = 1 - |\mathrm{corr}(x, y)|$. The function $B_k = D_k \circ f$ in (A.1) is ∞-differentiable on $\mathfrak{D}_{c,n}$.

Proof. Take $p = (p_1, \ldots, p_c) \in \mathfrak{D}_{c,n}$. By the definition of $\mathfrak{D}_{c,n}$, we have that $d(p_i, p_j) \neq d(p_k, p_l)$ for all $\{i, j\} \neq \{k, l\}$. Furthermore, the correlation is well-defined and $\mathcal{C}^\infty$ on $\pi_{i,j}(p) = (p_i, p_j)$ for all $i, j \in \{1, \ldots, c\}$, because $\operatorname{cov}(p_i, p_j) \neq 0$ for points in $\mathfrak{D}_{c,n}$. We also have that $|\operatorname{corr}(x, y)|$ is $\mathcal{C}^\infty$ on every (p_i, p_j), since the absolute value function is $\mathcal{C}^\infty$ on $\mathbb{R} \setminus \{0\}$. Therefore, d is $\mathcal{C}^\infty$ on $\pi_{i,j}(p) = (p_i, p_j)$ for all $i, j \in \{1, \ldots, c\}$ and thus the assumptions of Proposition 7 hold, implying that B_k is ∞-differentiable on $\mathfrak{D}_{c,n}$. □

Proof of Theorem 1 By Proposition 8, the function B_0 is ∞-differentiable on $\mathfrak{D}_{c,n}$, where $B_0(P)$ is the persistence diagram of P in homological dimension zero. Therefore, we only need to display functions $\tilde{\mathcal{T}}_i \colon \mathbf{Diag} \to \mathbb{R}$ that are ∞-differentiable on $B_0(\mathfrak{D}_{c,n})$ such that $\mathcal{T}_i = \tilde{\mathcal{T}}_i \circ B_0$ for $i \in \{1, 2\}$.

Here we view zero-dimensional persistence diagrams as consisting of points $(0, y)$, although we keep denoting them in the general form (b, d). When computing the average persistence and standard deviation of persistence of the points in $B_0(P)$, the number of points in the diagram is assumed to be equal to the number of edges of a minimum spanning tree for (P, d), that is, $|P| - 1$. Therefore, the functions $\tilde{\mathcal{T}}_i$ can be defined as

$$\tilde{\mathcal{T}}_1(D) \triangleq \frac{1}{c-1} \sum_{(b,d) \in D^*} (d - b), \qquad \tilde{\mathcal{T}}_2(D) \triangleq -\alpha \tilde{\mathcal{T}}_1(D) + \beta \sigma(D),$$

where

$$\sigma^2(D) \triangleq \frac{1}{c-1} \sum_{(b,d) \in D^*} ((d - b) - \tilde{\mathcal{T}}_1(D))^2$$

with $D^* = \{(b, d) \in D : d < \infty\}$. Points in the diagonal Δ are sent to zero by $d - b$ and are not taken into consideration in the sums, and neither are points at infinity.

In order to prove that the functions $\tilde{\mathcal{T}}_i$ are ∞-differentiable, take any $m, \ell \in \mathbb{Z}_{\geq 0}$ and $\tilde{D} \in \mathbb{R}^{2m} \times \mathbb{R}^\ell$ such that $Q_{m,\ell}(\tilde{D}) = D$. If we write

$$\tilde{D} = (x_1, y_1, \ldots, x_m, y_m, z_1, \ldots, z_\ell),$$

then the functions $\tilde{\mathcal{T}}_i \circ Q_{m,\ell}$ are given by

$$\tilde{\mathcal{T}}_1(Q_{m,\ell}(\tilde{D})) = \frac{1}{c-1} \sum_{i=1}^{m} (y_i - x_i), \qquad \tilde{\mathcal{T}}_2(Q_{m,\ell}(\tilde{D})) = -\alpha \tilde{\mathcal{T}}_1(Q_{m,\ell}(\tilde{D})) + \beta \sigma(\tilde{D}),$$

where

$$\sigma^2(\tilde{D}) \triangleq \frac{1}{c-1}\sum_{i=1}^{m}[(y_i - x_i) - \tilde{\mathcal{T}}_1(Q_{m,\ell}(\tilde{D}))]^2.$$

The functions $\tilde{\mathcal{T}}_i \circ Q_{m,\ell}$ are $\mathcal{C}^\infty$ on all their domain because they are compositions of $\mathcal{C}^\infty$ functions on a neighborhood of $\tilde{D}$. The only function that is not $\mathcal{C}^\infty$ in all its domain is the square root function, which is not differentiable at zero. However, for points $p \in \mathfrak{D}_{c,n}$ we have pairwise different distances, and consequently the persistence diagram $B_0(P)$ contains at least two different points, making $\sigma(\tilde{D}) \neq 0$ for a neighbourhood U of $\tilde{D}$ and thus making $\sigma(\tilde{D})$ a $\mathcal{C}^\infty$ function on U. Hence, $\tilde{\mathcal{T}}_i$ is ∞-differentiable. Therefore, as B_0 and $\tilde{\mathcal{T}}_i$ are ∞-differentiable in $\mathfrak{D}$ and $B_0(\mathfrak{D}_{c,n})$ respectively, the functions $\mathcal{T}_i$ are $\mathcal{C}^\infty$ on $\mathfrak{D}_{c,n}$. ■

A.2.3 Figures

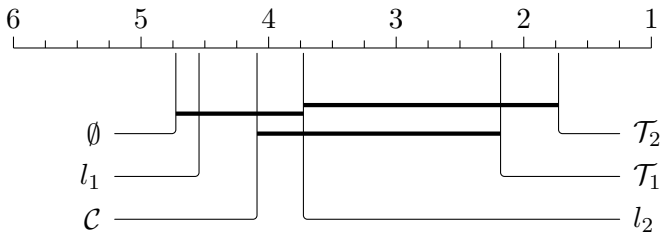


Figure A.5: Critical difference diagrams [213] for the Friedman and Nemenyi post-hoc statistical tests conducted in both blocks of experiments. The position of each training approach on the diagram corresponds to its average rank based on the test accuracies of the trained network —see Table 4.4. Lower ranks indicate that a regularisation term, or the training without regulariser, outperforms competitors with higher ranks. Regularisation terms are connected if the p-value obtained from the Nemenyi post-hoc test is greater than 0.05. See Table 4.5 for a description of the training approaches and the p-values.

B. Supplemental material for Chapter 5

B.1 Counting connected components

There are many different ways of computing the number of connected components β_0 from a simplicial complex. From them, we use Algorithm 1 for some proofs of Section B.2. Algorithm 1 is an algorithm based on the *union–find* data structure, also known as a disjoint set forest, that we will use in Section B.2. This data structure is built on the vertices of a graph and affords two operations, viz. `union` (or `merge`) and `find`. The `merge` operation assigns two vertices to the same connected component, while the `find` operation returns the current connected component of a vertex. Building such a data structure is reasonably easy in programming languages like `Python`, which offer *associative arrays*. The pseudo-code assumes that all operations are changing objects 'in place.' Notice that the `find` operation is implemented implicitly via a lookup in the `merge` function. A proper object-oriented implementation of a union–find data structure should have these two operations in its public interface.

B.2 Theorems and proofs

For the convenience of the reader, we restate all results again before providing their proof.

When working with filtrations in the subsequent proofs, it would be ideal to have filtrations that satisfy *injectivity* on the level of vertices, i.e., $f(v) \neq f(v')$ if $v \neq v'$. Such injective filtrations have the advantage of permitting gradient-based optimisation schemes [222]. The following lemma, first proved in Horn et al. [98], demonstrates that injectivity is not a strict requirement, though, as it is always

Algorithm 1 Using associative arrays to find connected components

```
1: function GET_CONNECTED_COMPONENTS(V, E)
2:     UF ← {}
3:     for v ∈ V do
4:         UF[v] ← v
5:     end for
6:     for e = (v, w) ∈ E do
7:         MERGE(UF, v, w)
8:     end for
9:     return {v | UF[v] = v}
10: end function
11: function MERGE(UF, v, w)
12:     if UF[v] ≠ UF[w] then
13:         UF[v] ← w
14:     end if
15: end function
```

possible to find an injective filtration function that is arbitrarily close (in the Hausdorff sense) to a non-injective filtration function.

Lemma 9. For all $\epsilon > 0$ and a filtration function f defined on the vertices, i.e., $f\colon V \to \mathbb{R}^d$, there is an injective function $\tilde{f}\colon V \to \mathbb{R}^d$ such that $\|f - \tilde{f}\|_\infty < \epsilon$.

Proof. Let $V = \{v_1, \ldots, v_n\}$ be the vertices of a graph and $\operatorname{im} f = \{u_1, \ldots, u_m\}$ be their images under f. Since f is not injective, we have $m < n$. We resolve non-injective vertex pairs iteratively. For $u \in \operatorname{im} f$, let $V' := \{v \in V \mid f(v) = u\}$. If V' only contains a single element, we do not have to do anything. Otherwise, for each $v' \in V'$, pick a new value from $\mathrm{B}_\epsilon(u) \setminus \operatorname{im} f$, where $\mathrm{B}_r(x) \subset \mathbb{R}^d$ refers to the open ball of radius r around a point x (for $d = 1$, this becomes an open interval in $\mathbb{R}$, but the same reasoning applies in higher dimensions). Since we only ever remove a finite number of points, such a new value always exists, and we can modify $\operatorname{im} f$ accordingly. The number of vertex pairs for which f is non-injective decreases by at least one in every iteration, hence after a finite number of iterations, we have modified f to obtain $\tilde{f}$, an *injective* approximation to f. By always picking new values from balls of radius ϵ, we ensure that $\|f - \tilde{f}\|_\infty < \epsilon$, as required. □

Proposition 2. For $\mathcal{F}$ equivariant and two isomorphic graphs G and G', the persistence diagrams of f_{K_G} and $f_{K_{G'}}$ coincide.

The proof of Proposition 2 is a direct consequence of the following lemma.

Lemma 10. Let $\mathbb{K}$ be any field. If $\mathcal{F}$ is equivariant and $G \simeq G'$, then the persistence modules over $\mathbb{K}$ obtained by applying the homology functor H_k to the chain complexes generated by the sublevel set filtrations f_{K_G} and $f_{K_{G'}}$ are isomorphic for any $k \geq 0$.

Proof. Let $\varphi\colon G \to G'$ be an isomorphism between G and G', and let $a \in \mathbb{R}$. We claim that φ induces a simplicial isomorphism between $K_a(G) = f_{K_G}^{-1}((-\infty, a])$ and $K_a(G') = f_{K_{G'}}^{-1}((-\infty, a])$. First, note that $\sigma \in K_a(G)$ implies $\varphi(\sigma) \in K_a(G')$. Then, the restriction φ_a of the map $\varphi\colon K_G \to K_{G'}$ to $K_a(G)$ is a well-defined simplicial complex morphism between $K_a(G)$ and $K_a(G')$. In particular, φ_a is an isomorphism. The injectivity of φ_a stems from the fact that φ is an isomorphism between simplicial complexes. The surjectivity of φ_a comes from the fact that $f_{K_G}(\varphi^{-1}(\tau)) = f_{K_{G'}}(\tau)$ for all $\tau \in K_a(G')$, and thus if $\tau \in K_a(G')$, then $\varphi^{-1}(\tau) \in K_a(G)$ and $\varphi_a(\varphi^{-1}(\tau)) = \tau$. Having prove that ϕ_a is an isomorphism, let $C_\bullet(K_a(G))$ and $C_\bullet(K_a(G'))$ be the chain complexes induced by the simplicial complexes $K_a(G)$ and $K_a(G')$, respectively. The previous function φ_a induces a chain map $C_\bullet(\varphi_a)$ between $C_\bullet(K_a(G))$ and $C_\bullet(K_a(G'))$ defined as the linear map $C_k(\varphi_a)$ sending an element $\sigma \in C_k(K_a(G))$ to $\varphi_a(\sigma)$ for $k \geq 0$. The proof that $C_\bullet(\varphi_a)$ is a chain map can be found in Nanda [54, Proposition 4.5]. Concretely, $C_k(\varphi_a)$ is an isomorphism because φ_a is a simplicial complex isomorphism that generates a one-to-one correspondence between the k-simplices of $K_a(G)$ and $K_a(G')$. As $C_\bullet(\varphi_a)$ is a chain isomorphism, $C_\bullet(\varphi_a)$ induces isomorphisms between the homology groups $H_k(C_\bullet(K_a(G)))$ and $H_k(C_\bullet(K_a(G')))$ for all $k \geq 0$. Moreover, these isomorphisms $C_\bullet(\varphi_a)$ constitute isomorphisms between the persistence modules given by $(H_k(C_\bullet(K_a(G))))_{a\in\mathbb{R}}$ and $(H_k(C_\bullet(K_a(G'))))_{a\in\mathbb{R}}$

for all $k \geq 0$. To prove it, we only need to show that the diagram

$$\begin{array}{ccc} H_k(C_\bullet(K_G(a_i))) & \xhookrightarrow{\quad i \quad} & H_k(C_\bullet(K_G(a_j))) \\ \downarrow{\scriptstyle C_k(\varphi_{a_i})} & & \downarrow{\scriptstyle C_k(\varphi_{a_j})} \\ H_k(C_\bullet(K_{G'}(a_i))) & \xhookrightarrow{\quad i \quad} & H_k(C_\bullet(K_{G'}(a_j))) \end{array}$$

commutes for $a_i \leq a_j$. This is a consequence of the definition of the isomorphisms $C_\bullet(\varphi_a)$. Note that $C_k(\varphi_{a_i})$ and $C_k(\varphi_{a_j})$ are the same map for the elements of $C_k(K_G(a_i))$ for all $k \geq 0$ and $a_i \leq a_j$ due to the fact that $\varphi_{a_i} = \varphi_{a_j}$ for $K_G(a_i)$ for $a_i \leq a_j$ by definition. Thus, given $[c] \in H_k(C_\bullet(K_G(a_i)))$, we have $(C_k(\varphi_{a_j}) \circ i)([c]) = [\varphi_{a_i}(c)] = (i \circ C_k(\varphi_{a_i}))([c])$ for all $a_i \leq a_j$ and $k \geq 0$, as we wanted to prove. □

Theorem 5. For $k \geq 2$, there exists an equivariant filtration generator $\mathcal{F}$ of type (2) such that its zero-dimensional persistence diagrams are at least as expressive as k-FWL.

Proof. The main idea involves harnessing the colours of k-tuples. Denote by $\mathcal{F}(G) = (K_G, f_{K_G})$ the output of our filtration generation. For a given graph G, we set K_G to be the simplicial complex of dimension zero with 0-simplices given by the vertices of G. Let C_G be the multiset of colours assigned to the graph G by the k-FWL algorithm. Without loss of generality, we assume that the k-FWL algorithm assigns colours that are always greater or equal than one.

Now, given a finite multiset S of colours, this is, a finite multiset of elements in $\mathbb{N} \cap [2, +\infty)$, we define the representation of S as the concatenation

$$\mathrm{R}(S) = c_1 \parallel \Big\| _{i=2}^{|S|} 0 \parallel r_9(c_i),$$

where $S = \{\!\{c_1, \ldots, c_{|S|}\}\!\}$ is the multiset of colours ordered in non-decreasing order, and $r_9(c_i)$ is the representation of the number c_i in bijective base-9 numeration. The map from the set of multisets of colours to the set of natural numbers given by R is injective because the representation of individual natural numbers in bijective base-9 is injective and because the digit zero does not appear in the base-9 representation of the numbers, allowing for the separation of the different colours, and for the recovery of the original multiset.

Set now f_{K_G} to be the constant filtration function that assigns to each simplex the value $\mathrm{R}(C_G)$. Thus, the filtration generator $\mathcal{F}$ is equivariant because the k-FWL algorithm outputs the same multiset of colours for isomorphic graphs $G \simeq G'$, and then the representation of the multiset of colours is the same for both graphs and yield the same constant filtration function.

Finally, take two non-isomorphic graphs G and G'. By definition of $\mathcal{F}$, the simplicial complexes K_G and $K_{G'}$ contain only vertices, yielding non-empty persistence diagrams only in dimension zero. Particularly, the zero-dimensional persistence diagrams of G and G' are multisets containing as many non-diagonal points as the number of vertices in the graphs of the form $(\mathrm{R}(C_G), +\infty)$ and $(\mathrm{R}(C_{G'}), +\infty)$, respectively. Since the graphs are non-isomorphic, the multisets and their representations are different and thus the persistence diagrams are different. □

Theorem 11. Given 1-WL colourings of two graphs G and G' that are different, there exists a filtration of G and G' such that their persistence diagrams in dimension 0 are also different.

Proof. Since the colourings are different, there is an iteration h of 1-WL such that the label sequences of G and G' are different. We thus have at least one colour—equivalently, one label—whose count is different. Let $\mathcal{L}^{(h)} := \{l_1, l_2, \dots\}$ be an enumeration of the finitely many hashed labels at iteration h. We can build a filtration function f by assigning a vertex v with label l_i to its index, i.e., $f(v) := i$, and setting $f(v, w) := \max\{f(v), f(w)\}$ for an edge (v, w). The resulting 0-dimensional persistence diagrams for G and G', denoted by $\mathcal{D}_0$ and $\mathcal{D}'_0$, respectively, now contain tuples of the form (i, j). Moreover, each vertex is guaranteed to give rise to *exactly* one such pair since each vertex creates a connected component in 0-dimensional persistent homology. Letting $\mu^{(i,j)}(\mathcal{D}_0)$ refer to the multiplicity of a tuple in $\mathcal{D}_0$, we know that, since the label count is different, there is *at least* one tuple (k, l) with $\mu^{(k,l)}(\mathcal{D}_0) \neq \mu^{(k,l)}(\mathcal{D}'_0)$. Hence, $\mathcal{D}_0 \neq \mathcal{D}'_0$. □

Prior to stating Corollary 13 concerning CFI graphs, we must first provide a precise definition of these structures. These definitions can also be found in Cai, Fürer, and Immerman [342, Section 6]. CFI graphs are constructed from a family of graphs $\{X_k\}_{k\geq 2}$. For each $k \geq 2$, we construct a pair of non-isomorphic CFI graphs (G_k, H_k) such that G_k and H_k cannot be distinguished by the $(k-1)$-FWL algorithm but can be distinguished by the k-FWL algorithm. Concretely, X_k is defined as the graph (V_k, E_k) where (1) $V_k = A_k \cup B_k \cup M_k$ such that $A_k = \{a_i\}_{i=1}^k$, $B_k = \{b_i\}_{i=1}^k$, and $M_k = \{m_S : S \subseteq \{1, \ldots, k\}, |S| \text{ is even}\}$, and (2) $E_k = \{(m_S, a_i) : i \in S\} \cup \{(m_S, b_i) : i \notin S\}$. Given any finite and connected graph G with at least degree two for all vertices, we can use the graphs X_k to build new graphs $X(G)$ and $\tilde{X}(G)$ that allows us to construct the CFI graphs. $X(G)$ is built as follows. For each vertex v of G, we replace v by a copy of $X_{\deg v}$, called $X(v)$. Then, for each edge $\{u, v\}$ of G, we associate to each of its endpoints u and v one of the pairs $\{a_i, b_i\}$ from $X(u)$ and $X(v)$, denoting the pairs as $\{a(\{u, v\}, u), b(\{u, v\}, u)\}$ and $\{a(\{u, v\}, v), b(\{u, v\}, v)\}$, respectively. Finally, for each of edges $\{u, v\}$ of G, we add the edges $\{a(\{u, v\}, u), a(\{u, v\}, v)\}$ and $\{b(\{u, v\}, u), b(\{u, v\}, v)\}$ to G, discarding the original edge. The graph $\tilde{X}(G)$ is constructed in the same way as $X(G)$, but we arbitrarily choose one edge $\{u, v\}$ of G and, instead of adding the previous two edges, we add the edges $\{a(\{u, v\}, u), b(\{u, v\}, v)\}$ and $\{a(\{u, v\}, v), b(\{u, v\}, u)\}$. Finally, G_k and H_k are given by $G_k = X(T_k)$ and $H_k = \tilde{X}(T_k)$ where T_k is a degree-three graph with separator size k. In the original construction, these graphs are also coloured, but we will not need this information for the proof of Corollary 13.

Lemma 12. For any finite and connected graph G, $X(G)$ and $\tilde{X}(G)$ have no cycles of length 3, and thus, no cliques of size 3 or more.

Proof. Let γ be a cycle of $X(G)$ of length three. First, note that there are no cycles of length 3 in the graphs X_k for any $k \geq 1$. This is because all edges go from M_k to a subset A_k or B_k. This implies that, in order to have a cycle, one needs at least 4 edges, passing twice by the set M_k. Therefore, the cycle γ must contain at least one edge e connecting two subgraphs $X(u)$ and $X(v)$ for u and v original vertices of G. The endpoints of this edge must be connected by design to two disjoint sets of

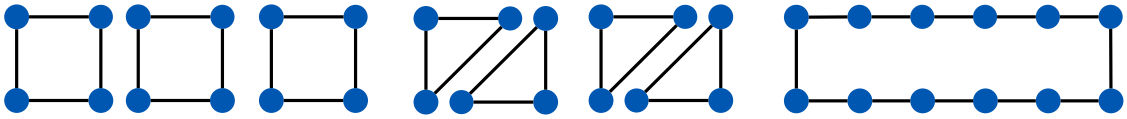


Figure B.1: The image shows three graphs composed of three disjoint cycles of length four, four disjoint cycles of length three, and a single cycle of length twelve. For 1-WL, all vertices are initially assigned the same colour, in this case blue. During the first iteration, the `RELABEL` function for each vertex receives the tuple $(\bullet, \{\!\{\bullet, \bullet\}\!\})$. Consequently, all vertices in the three graphs receive the same new colour, keeping intact the initial partitions, finishing the algorithm. Thus, the final multiset of colours is the same for all three cases (twelve blue colours), making 1-WL unable to distinguish between these three graphs.

type M_k of the subgraphs $X(u)$ and $X(v)$, respectively. However, if the cycle γ is of length three, this means that the two endpoints of e are connected to the same element, implying that both sets M_k are not disjoint and arriving at a contradiction. The proof for $\tilde{X}(G)$ is analogous. □

Corollary 13. For any $k \geq 2$, the CFI graphs G_k and H_k have no cycles of length 3, and thus, no cliques of size 3 or more.

Proof. This is a direct consequence of Lemma 12. □

Theorem 14. Let $n = ab$ with integers $a, b \geq 3$. Then, the 1-WL test cannot distinguish between a cycles of length b, b cycles of length a, and a single cycle of length n.

Proof. In all three cases, each graph consists of n vertices, with each vertex having a degree of two. Initially, all vertices across the three graphs are assigned the same colour c. During the first iteration, each node receives as input the tuple $(c, \{\{c, c\}\})$. Consequently, after this iteration, all vertices in the three graphs obtain the same colour assignment. Since all vertices have the same colour at the end of the first iteration, the algorithm terminates. This occurs because the partition of nodes generated by the colours after the first iteration is identical to the partition from the initial assignment. As a result, the 1-WL test fails to distinguish between the three cases. □

An example for $a = 3$, $b = 4$ is provided in Figure B.1. We end this section with two results concerning *other* graph properties that are being captured by persistent homology.

Proposition 3. Given *any* filtration f of a graph G with a single connected component such that the values of f of the endpoints of edges are *strictly lower* than the values of their corresponding edges, Algorithm 1, used to compute β_0, yields an upper bound of $\mathrm{diam}(G)$.

Proof. We can provide a procedure to obtain an upper bound d of $\mathrm{diam}(G)$ alongside the calculation of $\mathcal{D}_0$. To this end, we set $d = 0$. While calculating $\mathcal{D}_0$ with Algorithm 1, we check for each edge whether it is a destroyer, or a regular edge. If the edge is a destroyer, we increase d by one; otherwise, we do nothing. This procedure works because $\mathrm{diam}(G)$ is upper bounded by the diameter of the minimum spanning tree of G. Our estimate d counts the number of edges in such a tree. We thus have $\mathrm{diam}(G) \leq d$. The bound is tight for some graphs, such as line graphs. □

Proposition 4. Given *any* filtration f of a graph G, Algorithm 1 yields an upper bound of the girth of G.

Proof. Similar to Proposition 3, we can use the calculation of $\mathcal{D}_0$, the persistence diagram in dimension 0, of the filtration function to obtain an upper bound of the girth of the graph. We calculate $\mathcal{D}_0$ with Algorithm 1, checking once again for each edge whether it is a creator or destroyer. If the edge is a creator, we stop and set our upper bound of the girth g to be the number of vertices in the connected component that contains the endpoints of the edge. Since the addition of the respective edge is guaranteed to create a cycle —the edge is a creator— the cycle has to make use of at most as many vertices as the number of vertices in the connected component to which it belongs. Our estimate g thus constitutes an upper bound of the girth of the graph. □

Propositions 3 and 4 indicate that persistent homology captures more than 'just' topological information about a graph. We substantiate these theorems with empirical results concerning different filtrations and other graph properties in Section 5.3.3, thus showing how a topological perspective complements and enriches graph-learning tasks.

B.3 Topology and the Weisfeiler–Leman hierarchy

In the following, we want to briefly extend the argumentation of the expressivity of persistent homology with respect to the Weisfeiler–Leman hierarchy. Since 1-WL is oblivious to certain topological structures such as cycles [343], the existence of graphs with different Betti number counts proves that persistent homology is *strictly more expressive* than 1-WL. For example, consider a graph consisting of the union of two triangles, i.e.,  . This graph has $\beta_0 = \beta_1 = 2$ since it consists of two connected components and two cycles. If we change the connectivity slightly to obtain a hexagon, i.e.,  , we obtain a graph with $\beta_0 = \beta_1 = 1$. 1-WL is not able to distinguish between these graphs, but persistent homology can, since the Betti numbers of the graph are still encoded in the persistence diagram as essential features. Note that Theorem 11 does *not* apply to arbitrary filtrations since the theorem requires knowing the correct labels assigned by 1-WL. Finding filtration functions that are able to split graphs in a manner that is provably equivalent to 1-WL remains an open research question.

B.4 Additional expressivity experiments

This section contains additional expressivity experiments that had to be excluded from the main paper for reasons of space.

B.4.1 Additional results for connected cubic graphs

We start our supplementary experiments by distinguishing connected cubic graphs, i.e., 3-regular graphs. These graphs cannot be distinguished by 1-WL, but they

can be distinguished by 2-FWL [106, 344]. As such, they provide a good example of how different filtrations harness different types of graph information. Table B.1 shows the results. We first observe that the degree filtration is incapable of distinguishing graphs for $k = 1$. This is a direct consequence of the regularity—since the function is constant on the graph, persistent homology cannot capture any variability. This changes, however, when higher-order structures —triangles— are included for $k = 2$. We also observe that the Laplacian-based filtration for $k = 2$ exhibits strong empirical performance; in the absence of additional information, the spectral properties captured by the Laplacian help in distinguishing graphs. The Ollivier–Ricci curvature filtration is performing similarly well also for the same dimension, while it outperforms the Laplacian filtration for $k = 1$. In contrast to the Laplacian-based filtration, it does not require the calculation of eigenvalues, which may be prohibitive for larger graphs.[1] Finally, we see that the Forman–Ricci curvature filtration, a purely combinatorial quantity depending only on local counts, is consistently outperformed by the Ollivier–Ricci curvature for $k = 1$. However, for $k = 1$, the Forman–Ricci filtration outperforms the Laplacian filtration. Finally, the Vietoris–Rips filtration is incapable of distinguishing the graphs for $k = 1$, and performs similarly to the Forman–Ricci curvature for $k = 2$, suggesting that filtrations based on distances are not capable of capturing the graph structure in an optimal way.

B.4.2 Additional results for the BREC data set

Table B.2 shows BREC data set results, itemised by the value of k, while Table B.3 provides a summary and performance comparison of the persistent homology results with other baselines.

B.4.3 Additional figures for graph-property prediction tasks

Figure B.2 shows the distributions of graph properties that we predict in the main paper in Section 5.3.3.

[1] Ollivier–Ricci curvature requires solving optimal transport problems, for which highly efficient approximative solvers are available [345].

Table B.1: Success rate (↑) of distinguishing pairs of *connected cubic graphs* when using five different filtrations at varying expansion levels of the graph (denoted by k). Due to the regularity of each graph, $k = 3$ is omitted since the clique complex of the graph is exactly the same as for $k = 2$. These graphs can be distinguished by 2-FWL but not by 1-WL.

Data	$k = 1$					$k = 2$				
	Filtration									
	D	O	F	L	V	D	O	F	L	V
`cub06`	0.00	**1.00**	**1.00**	0.00	0.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
`cub08`	0.00	0.90	0.70	0.00	0.00	0.90	0.90	0.90	**1.00**	0.90
`cub10`	0.00	0.98	0.66	0.00	0.00	0.81	0.99	0.88	**1.00**	0.85
`cub12`	0.00	0.99	0.64	0.00	0.00	0.80	1.00	0.87	**1.00**	0.87
`cub14`	0.00	0.99	0.62	0.00	0.00	0.79	1.00	0.86	**1.00**	0.89

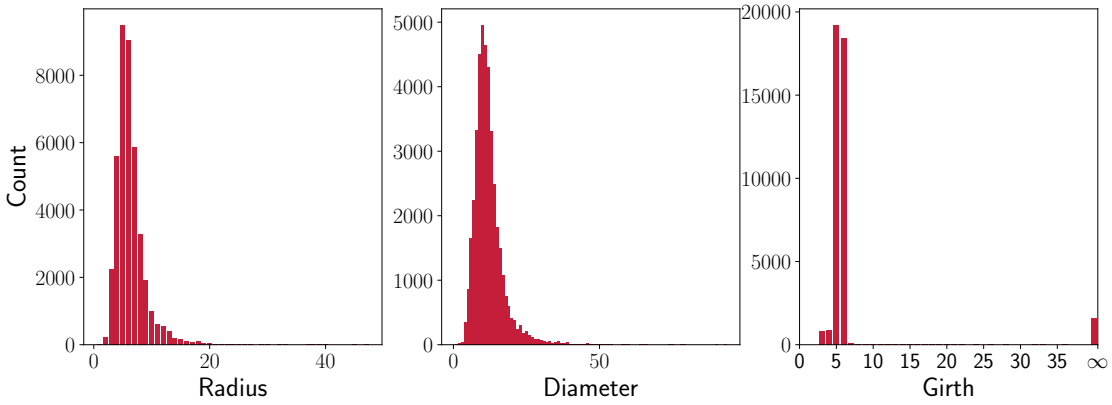


Figure B.2: Distribution of maximum radii, maximum diameters, and girths in the `ogbg-molhiv` molecular graph data set [255]. The values are concentrated on the lower end of the spectrum, with medians of 6, 11, and 5 and maximums of 47, 93, and 36, respectively. The maximum value for the girth is computed without taking into account infinite values.

B.5 Additional graph classification experiments

To assess the capacity of persistent homology in graph learning tasks, we have designed a set of graph classification experiments that use the filtrations introduced in Section 5.3 to extract persistence diagrams from graph classification data set to perform inference on them. Two fundamental differences between our approach and the one used in [222] are that, we do not learn an optimal filtration and also we do not use deep learning models to perform the classification, as we are interested

Table B.2: Success rate (↑) for distinguishing pairs of instances of the *BREC data set* when using five different filtrations at varying expansion levels of the graph (denoted by k). Due to combinatorial constraints, we did not calculate the Vietoris–Rips filtration for $k = 4$. Legend and number of graphs per category: `B` (Basic, 60), `R` (Regular, 100), `E` (Extension, 100), `C` (CFI, 100), `4, 20` (4-Vertex Condition), `D` (Distance-Regular, 20) graphs, respectively and `A` (average over full data set, 400).

	$k = 1$				
Data	*Filtration*				
	D	O	F	L	V
`Basic` (60)	0.03	0.93	0.87	**1.00**	0.00
`Regular` (100)	0.00	0.42	0.32	0.00	0.00
`Extension` (100)	0.07	0.76	0.44	0.94	0.00
`CFI` (100)	0.03	0.03	0.03	**0.06**	0.03
`4-VC` (20)	0.00	0.00	0.00	0.00	0.00
`DR` (20)	0.00	0.00	0.00	**0.05**	0.00
Average (400)	0.03	0.44	0.33	0.40	0.01
	$k = 2$				
Data	*Filtration*				
	D	O	F	L	V
`Basic` (60)	0.78	**1.00**	0.98	**1.00**	0.52
`Regular` (100)	0.39	0.54	0.50	0.48	0.39
`Extension` (100)	0.26	0.92	0.59	**1.00**	0.11
`CFI` (100)	0.03	0.03	0.03	**0.06**	0.03
`4-VC` (20)	0.00	0.00	0.00	0.00	0.00
`DR` (20)	0.00	0.00	0.00	**0.05**	0.00
Average (400)	0.29	0.52	0.43	0.54	0.21
	$k = 3$				
Data	*Filtration*				
	D	O	F	L	V
`Basic` (60)	0.83	**1.00**	0.98	**1.00**	0.58
`Regular` (100)	0.85	0.93	0.91	0.93	0.85
`Extension` (100)	0.29	0.92	0.59	**1.00**	0.16
`CFI` (100)	0.03	0.03	0.03	**0.06**	0.03
`4-VC` (20)	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
`DR` (20)	0.00	0.00	0.00	**0.05**	**0.05**
Average (400)	0.47	0.67	0.58	0.70	0.40
	$k = 4$				
Data	*Filtration*				
	D	O	F	L	V
`Basic` (60)	0.83	**1.00**	0.98	**1.00**	—
`Regular` (100)	0.89	0.97	0.95	0.97	—
`Extension` (100)	0.29	0.92	0.59	**1.00**	—
`CFI` (100)	0.03	0.03	0.03	0.06	—
`4-VC` (20)	**1.00**	**1.00**	**1.00**	**1.00**	—
`DR` (20)	0.00	0.00	0.00	0.05	—
Average (400)	0.48	0.68	0.59	0.71	—

Table B.3: Success rate (↑) for distinguishing pairs of instances of the *BREC data set* when using four different filtrations for $k = 4$. Vietoris–Rips filtration not computed for $k = 4$ due to computational constraints. The data sets `Regular`, `4-Vertex Condition`, and `Distance Regular`, from Table 5.3 are merged into the `All regular` data set. Green indicates the best performing algorithm, while orange indicates the second best.

Data	SOTA		Filtration ($k = 4$)			
	3-WL	N_2	D	O	F	L
`Basic` (60)	1.000	1.000	0.83	1.000	0.983	1.000
`All regular` (140)	0.36	0.986	0.78	0.84	0.82	0.843
`Extension` (100)	1.000	1.000	0.29	0.920	0.59	1.000
`CFI` (100)	0.600	0.00	0.03	0.03	0.03	0.060
Average (400)	0.68	0.745	0.48	0.68	0.59	0.710

Table B.4: Graph classification average accuracy (↑) and standard deviation in test for the experiments with data set `IMDB-Binary` (`I`), `PROTEINS` (`P`), `MUTAG` (`M`), and `NCI1` (`N`), and ROC-AUC (↑) for the `ogbg-molhiv` (`O`) test data set. The results are averaged over three runs using a stratified 3-means for all data set except for `ogbg-molhiv`, where only one run is performed. The best results are highlighted in bold. Experiments with the Vietoris–Rips filtration for the `PROTEINS` and `molhiv` are not reported for $k = 2$ due to out-of-resources errors during execution. S_1 and S_2 refer to accuracy of the most effective methods reported in [238] and [222], respectively. The abbreviation DS stands for data set.

$k = 1$							
DS	SOTA		Filtration				
	S_1	S_2	D	O	F	L	V
`P`	**0.75 ± 0.05**	0.74 ± 0.03	0.70 ± 0.01	0.74 ± 0.01	0.72 ± 0.02	0.72 ± 0.02	0.70 ± 0.00
`I`	0.74 ± 0.04	**0.75 ± 0.05**	**0.75 ± 0.02**	0.71 ± 0.01	0.69 ± 0.03	0.65 ± 0.01	0.68 ± 0.03
`M`	0.87 ± 0.08	-	0.86 ± 0.01	0.87 ± 0.04	**0.90 ± 0.04**	0.79 ± 0.06	0.87 ± 0.01
`N`	-	0.71 ± 0.02	0.68 ± 0.00	0.73 ± 0.00	0.69 ± 0.01	0.67 ± 0.00	0.66 ± 0.01
`O`	-	-	**0.50**	0.50	**0.50**	0.50	0.50
$k = 2$							
DS	SOTA		Filtration				
	S_1	S_2	D	O	F	L	V
`P`	0.75 ± 0.05	0.74 ± 0.03	0.70 ± 0.02	0.73 ± 0.02	0.70 ± 0.01	0.68 ± 0.01	-
`I`	0.74 ± 0.04	0.75 ± 0.05	0.73 ± 0.03	0.71 ± 0.02	0.72 ± 0.03	0.68 ± 0.03	0.66 ± 0.02
`M`	0.87 ± 0.08	-	0.86 ± 0.01	0.89 ± 0.04	0.89 ± 0.03	0.81 ± 0.04	0.87 ± 0.03
`N`	-	0.71 ± 0.02	0.68 ± 0.00	**0.74 ± 0.01**	0.70 ± 0.00	0.68 ± 0.00	0.69 ± 0.01
`O`	-	-	**0.50**	**0.50**	0.50	0.50	-

into the capacity of persistent homology to perform classification, and not in its combination with neural networks. For this endeavour, we train a Random Forest classifier on the persistent images [76] computed from the persistence diagrams extracted from the input graphs using the previous filtrations up to dimension $k = 2$. We test our approach in the `MUTAG` [346], `IMDB-Binary` [347], `PROTEINS` [348], `NC1` [349], and `ogbg-molhiv` [255] data set. Except for `ogb-molhiv`, we perform a Stratified 3-Fold experiment and provide the average and standard deviation of the multiple runs. For `ogbg-molhiv`, we perform only one experiment instance with the official train and test splits. The results are shown in Table B.4. We observe that, although we discard the data set features when using persistent homology, we are able to achieve suitably good results on some of the data sets, such as `MUTAG` and `IMDB-Binary` [267], even surpassing state-of-the-art results reported in Hofer et al. [222] and O'Bray, Rieck, and Borgwardt [238] for `IMDB-Binary`, `MUTAG`, and `NCI1`. For the `ogbg-molhiv`, however, we obtain consistent ROC-AUC values of 0.5, which is the same as random guessing. Our hypothesis is that molecular graphs rely on the annotated data set features to perform well, and the isolated topological information is not enough to perform the classification. The success of the experiments suggests that persistent homology is capable of capturing essential structural information about the graphs to classify and thus, suggests that persistent homology can be expressive in practice. We leave a more structured and comprehensive benchmark for persistent homology-like methods for graph-learning tasks to future work.

B.6 Additional results on alpha complexes

As the last set of experiments, we repeat the experiments from Section 5.3 except for the the property prediction tasks on random graphs, but using an alpha complex filtration. *Alpha complex filtrations* [53, Section III.4] are *geometric* filtrations, meaning that they are computed from point clouds in the Euclidean space, in our case, coming from embeddings of graph vertices. Concretely, we embed the vertices of our graphs in $\mathbb{R}^3$ using *Laplacian eigenmaps* [350]. Note that the embedding map is an arbitrary choice, and we leave the exploration of other embeddings

for future work, making the use of alpha complexes in this context a proof of concept. Tables B.5 to B.10 contain the BREC, classification, connected cubic, minimal Cayley, strongly-regular, and property prediction experiments, respectively, using the alpha complex filtration.

For BREC, we observe that the alpha complex filtration obtains almost perfect success rate in distinguishing the non-isomorphic graphs, failing only on the CFI category, where the alpha complex filtration still shows an strong success rate, outperforming the other filtrations and non-topological methods by a large margin. For the other isomorphism datasets (connected cubic, minimal Cayley, strongly-regular), alpha complexes obtain perfect accuracy in all cases.

For the classification tasks, though, we observe that the alpha complex filtration performs worse than the best filtrations at each task worse than the other state-of-the-art methods. Due to the good results in the graph isomorphism tasks, we hypothesize that the alpha complex is highly expressive (and sensitive) and that the Random Forest with the persistence images overfits the data without further regularization.

Finally, for the property prediction experiments, we observe that the alpha complex filtration, as in the classification tasks, performs worse than the other filtrations except for the girth prediction, where it obtains better results than curvature-based and degree filtrations.

Overall, our preliminary results suggest that the alpha complex filtration is highly expressive and capable of capturing the graph structure, but it is also sensitive to noise and overfitting, at least for Laplacian eigenmaps embeddings.

Table B.5: Success rate (↑) for distinguishing pairs of instances of the *BREC data set* when using the alpha complex filtration described at Section B.6 and persistence diagrams up to varying dimensions (denoted by k). Legend and number of graphs per category: `B` (Basic, 60), `R` (Regular, 100), `E` (Extension, 100), `C` (CFI, 100), `4, 20` (4-Vertex Condition), `D` (Distance-Regular, 20) graphs, respectively and `A` (average over full data set, 400). For the results using the other filtrations, see Table B.2.

Data	$k=1$	$k=2$	$k=3$
Basic (60)	**1.00**	**1.00**	**1.00**
Regular (100)	**1.00**	**1.00**	**1.00**
Extension (100)	**1.00**	**1.00**	**1.00**
CFI (100)	0.63	0.80	**0.87**
4-Vertex_Condition (20)	**1.00**	**1.00**	**1.00**
Distance_Regular (20)	**1.00**	**1.00**	**1.00**
Average (400)	0.91	0.95	**0.97**

Table B.6: Graph classification average accuracy (↑) and standard deviation in test for the experiments with data set `IMDB-Binary` (`I`), `PROTEINS` (`P`), `MUTAG` (`M`), and `NCI1` (`N`), and ROC-AUC (↑) for the `ogbg-molhiv` (`O`) test data set when using the alpha complex filtration described at Section B.6 and persistence diagrams up to varying dimensions (denoted by k). The results are averaged over three runs using a stratified 3-means for all data set except for `ogbg-molhiv`, where only one run is performed. The best results are highlighted in bold. For the results using the other filtrations, see Table B.4.

Data set	$k=1$	$k=2$
`P`	$\mathbf{0.66 \pm 0.02}$	0.66 ± 0.01
`I`	$\mathbf{0.59 \pm 0.04}$	0.57 ± 0.04
`M`	0.79 ± 0.03	$\mathbf{0.79 \pm 0.03}$
`N`	$\mathbf{0.62 \pm 0.01}$	0.60 ± 0.02
`O`	0.48 ± 0.00	$\mathbf{0.50 \pm 0.00}$

Table B.7: Success rate (↑) of distinguishing pairs of *connected cubic graphs* when using the alpha complex filtration described at Section B.6 and persistence diagrams up to varying dimensions (denoted by k). These graphs can be distinguished by 2-FWL but not by 1-WL. For the results using the other filtrations, see Table B.1.

Data	$k=1$	$k=2$
`cub06`	**1.00**	**1.00**
`cub08`	**1.00**	**1.00**
`cub10`	**1.00**	**1.00**
`cub12`	**1.00**	**1.00**
`cub14`	**1.00**	**1.00**

Table B.8: Success rate (↑) for distinguishing pairs of *minimal Cayley graphs* when using the alpha complex filtration described at Section B.6 and persistence diagrams up to varying dimensions (denoted by k). For the results using the other filtrations, see Table 5.2.

Data	$k = 1$	$k = 2$
`cay12`	**1.00**	**1.00**
`cay16`	**1.00**	**1.00**
`cay20`	**1.00**	**1.00**
`cay24`	**1.00**	**1.00**
`cay32`	**1.00**	**1.00**
`cay36`	**1.00**	**1.00**
`cay60`	**1.00**	**1.00**
`cay63`	**1.00**	**1.00**

Table B.9: Success rate (↑) for distinguishing pairs of *strongly-regular graphs* when using the alpha complex filtration described at Section B.6 and persistence diagrams up to varying dimensions (denoted by k). 2-FWL cannot distinguish between any of these pairs. For the results using the other filtrations, see Table 5.1.

Data	$k = 1$	$k = 2$	$k = 3$
`16622`	**1.00**	**1.00**	**1.00**
`251256`	**1.00**	**1.00**	**1.00**
`261034`	**1.00**	**1.00**	**1.00**
`281264`	**1.00**	**1.00**	**1.00**
`291467`	**1.00**	**1.00**	**1.00**
`351668`	**1.00**	**1.00**	**1.00**
`351899`	**1.00**	**1.00**	**1.00**
`361446`	**1.00**	**1.00**	**1.00**
`401224`	**1.00**	**1.00**	**1.00**

Table B.10: Accuracy (↑) when predicting the properties of graphs in the `ogbg-molhiv` molecular graph data set [255] using the alpha complex filtration described at Section B.6 and persistence diagrams up to varying dimensions (denoted by k). For the results using the other filtrations, see Table 5.4.

Data	$k = 1$	$k = 2$
Diameter	0.00	**0.01**
Girth	**0.41**	0.40
Radius	0.00	**0.01**

C. Supplemental material for Chapter 6

C.1 Distribution of labels

Tables C.1, C.2, C.3, and C.4 contain statistical information about the distribution of labels in the dataset.

Table C.1: Distribution of Betti numbers β_i for triangulations of manifolds. Percentages are rounded to the nearest integer and are computed for each pair of manifold dimension (2 or 3) and Betti number. Columns represent values of Betti numbers and contain the number of manifolds with each value.

	$\mathcal{M}$	0	1	2	3	4	5	6
β_0	2-$\mathcal{M}$	-	43,138 (100%)	-	-	-	-	-
	3-$\mathcal{M}$	-	250,359 (100%)	-	-	-	-	-
β_1	2-$\mathcal{M}$	1,670 (4%)	4,655 (11%)	14,146 (33%)	13,694 (32%)	7,917 (18%)	1,022 (2%)	34 (0%)
	3-$\mathcal{M}$	249,225 (100%)	1,134 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
β_2	2-$\mathcal{M}$	39,718 (92%)	3,420 (8%)	-	-	-	-	-
	3-$\mathcal{M}$	249,841 (100%)	518 (0%)	-	-	-	-	-
β_3	2-$\mathcal{M}$	-	-	-	-	-	-	-
	3-$\mathcal{M}$	616 (0%)	249,743 (100%)	-	-	-	-	-

Table C.2: Distribution of torsion subgroups for triangulations of manifolds. Percentages are rounded to the nearest integer and are computed for each pair of manifold dimension and homological degree.

	H_0	H_1		H_2		H_3
$\mathcal{M}$	0	$\mathbb{Z}_2$	0	$\mathbb{Z}_2$	0	0
2-$\mathcal{M}$	43,138 (100%)	39,718 (92%)	3,420 (8%)	0 (0%)	43,138 (100%)	-
3-$\mathcal{M}$	250,359 (100%)	0 (0%)	250,359 (100%)	616 (0%)	249,743 (100%)	250,359 (100%)

Table C.3: Distribution of genus for triangulations of surfaces. Percentages are rounded to the nearest integer.

$\mathcal{M}$	0	1	2	3	4	5	6	7
2-$\mathcal{M}$	306 (1%)	3,593 (8%)	5,520 (13%)	11,937 (28%)	13,694 (32%)	7,052 (16%)	1,022 (2%)	14 (0%)

Table C.4: Distribution of homeomorphism types for triangulations of manifolds. Percentages are rounded to the nearest integer and are computed for each manifold dimension. Surfaces classified as "Other" do not have any explicit homeomorphism type assigned.

$\mathcal{M}$	S^2	$\mathbb{R}P^2$	T^2	K	S^3	$S^2 \times S^1$	$S^2 \tilde{\times} S^1$	Other
2-$\mathcal{M}$	306 (1%)	1,364 (3%)	2,229 (5%)	4,655 (11%)	-	-	-	34,584 (80%)
3-$\mathcal{M}$	-	-	-	-	249,225 (100%)	518 (0%)	616 (0%)	0 (0%)

C.2 Dataset details

This section provides additional details about the MANTRA dataset and the design choices involved in its creation.

C.2.1 Data format

As a complement to Section 6.1 in the main text, we provide an extended description of dataset attributes. Each dataset consists of a list of triangulations, with each triangulation having the following attributes:

- `id` (required, `str`): This attribute refers to the original ID of the triangulation following [284]. This facilitates comparisons to the original dataset if necessary and simplifies future contributions by other authors.
- `triangulation` (required, `list` of `list` of `int`): A doubly-nested list of the top-level simplices of the triangulation.
- `n_vertices` (required, `int`): The number of vertices in the triangulation. This is **not** the number of simplices.
- `name` (required, `str`): A canonical name of the triangulation, such as `S^2` for the two-dimensional sphere. If no canonical name exists, we store an empty string.
- `betti_numbers` (required, `list` of `int`): A list of the Betti numbers of the triangulation, computed using $\mathbb{Z}$ coefficients. This implies that torsion coefficients are stored in another attribute.
- `torsion_coefficients` (required, `list` of `str`): A list of the torsion coefficients of the triangulation. An empty string `''` indicates that no torsion coefficients are available in that dimension. Otherwise, the original spelling of torsion coefficients is retained, so a valid entry might be `'Z_2'`.
- `genus` (optional, `int`): For 2-manifolds, contains the genus of the triangulation.

- `orientable` (optional, `bool`): Specifies whether the triangulation is orientable or not.

We picked JSON as our underlying data format, since it facilitates exchanging information, extending the dataset, and can be easily processed in all major programming languages. By only ever storing the top-level simplices of a simplicial complex, the dataset can be easily compressed. Moreover, the textual format facilitates tracking changes over different versions of the dataset. The subsequent listing depicts a simple example of the data format; see below for additional design choices.


```
[
  {
    "id": "manifold_2_4_1",
    "triangulation": [
      [1,2,3],
      [1,2,4],
      [1,3,4],
      [2,3,4]
    ],
    "dimension": 2,
    "n_vertices": 4,
    "betti_numbers": [
      1,
      0,
      1
    ],
    "torsion_coefficients": [
      "",
      "",
      ""
    ],
    "name": "S^2",
    "genus": 0,
    "orientable": true
  },
  {
    "id": "manifold_2_5_1",
    "triangulation": [
      [1,2,3],
      [1,2,4],
      [1,3,5],
```

```
        [1,4,5],
        [2,3,4],
        [3,4,5]
      ],
      "dimension": 2,
      "n_vertices": 5,
      "betti_numbers": [
        1,
        0,
        1
      ],
      "torsion_coefficients": [
        "",
        "",
        ""
      ],
      "name": "S^2",
      "genus": 0,
      "orientable": true
    }
  ]
```

C.2.2 Design choices

The datasets are converted from their original (mixed) lexicographical format [351].

A triangulation in lexicographical format could look like this:

```
manifold_lex_d2_n6_#1=[[1,2,3],[1,2,4],[1,3,4],[2,3,5],
  [2,4,5],[3,4,6],
  [3,5,6],[4,5,6]]
```

A triangulation in *mixed* lexicographical format could look like this:

```
manifold_2_6_1=[[1,2,3],[1,2,4],[1,3,5],[1,4,6],
  [1,5,6],[2,3,4],[3,4,5],[4,5,6]]
```

This format is hard to parse and error-prone. Moreover, any *additional* information about the triangulations, including information about homology groups or orientability, for instance, requires additional files. We thus decided to use a format that permits us to keep everything in one place, including any additional attributes for a specific triangulation. A desirable data format needs to satisfy the following properties:

(1) It should be easy to parse and modify, ideally in a number of programming languages.

(2) It should be human-readable and `diff`-able in order to permit simplified comparisons.

(3) It should scale reasonably well to larger triangulations.

After some considerations, we decided to opt for `gzip`-compressed JSON files. JSON is well-specified and supported in virtually all major programming languages out of the box. While the compressed file is *not* human-readable on its own, the uncompressed version can easily be used for additional data analysis tasks. This also greatly simplifies maintenance operations on the dataset. While it can be argued that there are formats that scale even better, they are not well-applicable to our use case since each triangulation typically consists of different numbers of top-level simplices. This rules out column-based formats like Parquet.

As for the *storage* of the data as such, we decided to keep only the top-level simplices (as is done in the original format) since this substantially saves disk space. The drawback is that the client has to supply the remainder of the triangulation. Given that the triangulations in our dataset are not too large, we deem this to be an acceptable compromise. Moreover, data structures such as simplex trees can be used to further improve scalability if necessary. Finally, our data format includes, whenever possible and available, additional information about a triangulation, including the Betti numbers and a *name*, i.e., a canonical description, of the topological space described by the triangulation. We opted to minimize any inconvenience that would arise from having to perform additional parsing operations.

Overall, this data format remains extensible—permitting additional information about a triangulation or attributes like coordinates—while still benefiting from easy accessibility. We make our code and data publicly available and use Zenodo for long-term archival with DOIs. The most recent version of our dataset is accessible via:

`https://doi.org/10.5281/zenodo.14103582`

Old versions are archived and can be accessed using our data loader. We hope that this system, while not perfect, may serve as a suitable starting point for other benchmark datasets.

C.3 Model details

We provide a brief description of the models used in the experiments.

Message passing neural networks. GCN [91], GAT [92], and UniMP [291] are examples of message-passing networks. In the case of GCN and GAT, adjacency with self-loops is used as neighborhood sets for nodes, whereas UniMP uses concatenated adjacencies up to a order k, meaning that we consider as neighbors of a vertex all the other vertices of the graph at a distance of at most k from the vertex. In the case of GAT, the fundamental difference lie in the message computation, where the message from a simplex τ to a simplex σ depends on a concept of attention, which is computed using the features of τ and σ and a learnable parameter Θ.

In the case of simplicial complexes, SAN [93] and SCN [94] use (upper and lower) higher-order Laplacians to define neighborhoods, SCCN [95] uses (co)adjacency and incidence structures, and SCCNN [96] uses all together.

Non-message-passing neural networks. We select graph and cellular transformers and multi-layer perceptrons (MLP) for comparison. Graph and cellular transformers are based on the original transformer's decoder architecture [31]. Original transformer architectures are permutation-invariant networks that use positional encoding to break the symmetry of the input data by means of localizing the position of each element in the input sequence.

C.4 Hyperparameter details

More information on the meaning of specific hyperparameters can be found in the PyTorch geometric, TopoModelX, CellMP, CT and DECT original implementations.

C.5 Additional experimental details

Table C.5 reports the mean and standard deviation of training iterations processed per second for each model and dataset. Sections C.5.1 to C.5.3 report the full set of experimental results.

Feature vector initialization analysis. We observe different behaviours for the two families of models, graph-based and simplicial complex-based. For the graph models, random initialization works slightly better or equal than the degree features. On the other hand, for the simplicial complex models, upper- and lower-connectivity index initializations consistently outperform their random counterparts on average. Degrees and upper-connectivity indices for vertices coincide for both families of models, suggesting that higher-order connectivity indices contain more useful information than their dimension zero counterpart to predict topological properties, supporting the need for models that leverage higher-order information of the input. Having signal contained in features can make sense if the task in question requires additional information. For example, molecules are more than just combinatorial or topological objects: the types of atoms and the nature of bonds are important for predicting their properties. However, in purely topological tasks, such as predicting topological invariants, the need to enforce topological information into features raises the question: do MP-based models correctly capture topological properties in the first place? Still, standard deviations in the aggregated data for simplicial complex-based models is large, and better ablation is needed to fully understand the differences in initializations and the expressivity of higher-order indices in the context of topological prediction tasks.

Graph models & DECT	
GAT	
Hidden neurons	64
Hidden layers	4
Readout	Mean
Dropout last linear layer	0.5
Activation last layer	Identity
GCN	
Hidden neurons	64
Hidden layers	4
Readout	Mean
Dropout last linear layer	0.5
Activation last layer	Identity
MLP	
Hidden neurons	64
Hidden layers	4
Readout	Mean
Dropout last linear layer	0.0
Activation last layer	Identity
TAG	
Hidden channels	64
Hidden layers	4
Readout	Mean
Dropout last linear layer	0.5
Activation last layer	Identity
UniMP	
Hidden channels	64
Hidden layers	4
Readout	Mean
Dropout last linear layer	0.5
Activation last layer	Identity
DECT	
Hidden channels	64
Hidden layers	3
Number of θ	32
Bump steps	32
r	1.1
Normalized	True

Topological models	
SAN	
Hidden channels	64
Hidden layers	1
n-filters	2
Order harmonic	5
Epsilon harmonic	1e-1
Readout	Sum of sums per dimension
SCCN	
Hidden channels	64
Hidden layers	2
Maximum rank	2
Aggregation activation function	Sigmoid
Readout	Sum of sums per dimension
SCCNN	
Hidden channels	64
Hidden layers	2
Order of convolutions	1
Order of simplicial complexes	1
Readout	Sum of sums per dimension
SCN	
Hidden channels per dimension	Same as input
Hidden layers	2
Readout	Sum of sums per dimension
CellMP	
Hidden channels	64
Hidden layers	10
Dropout	0.5
Hidden dimension multiplier for final linear layer	2
Readout	Sum of sums per dimension
CT	
Hidden channels	64
Positional encoding type	Hodge Laplacian Eigenvectors
Positional encoding lengths	8
Hidden layers	2
Number of heads	8
Dropout	0
Hidden layers in the final MLP	2
Attention tensor diagram	Adjacent dimensions
Mask type	Sum
Readout	Average of dimension zero features

Table C.5: Mean and standard deviation of training iterations processed per second (↑), as measured by PyTorch Lightning [352], across all experiments for each model and dataset. The measurements are subject to variations caused by external server usage fluctuations.

Model (Class)	2-$\mathcal{M}^0$	2-$\mathcal{M}^0_H$	3-$\mathcal{M}^0$
MLP ($\mathcal{G}$)	9.72 ± 4.54	12.39 ± 13.22	13.29 ± 8.36
GAT ($\mathcal{G}$)	9.41 ± 4.02	11.33 ± 10.60	11.01 ± 6.52
UniMP ($\mathcal{G}$)	9.31 ± 3.79	11.71 ± 10.72	12.42 ± 6.80
TAG ($\mathcal{G}$)	9.41 ± 3.53	11.46 ± 10.66	9.76 ± 6.55
GCN ($\mathcal{G}$)	9.45 ± 3.79	12.10 ± 11.43	12.72 ± 7.59
SAN ($\mathcal{T}$)	0.65 ± 0.97	1.21 ± 2.93	0.53 ± 0.31
SCN ($\mathcal{T}$)	0.83 ± 2.63	1.72 ± 6.99	0.83 ± 0.64
SCCN ($\mathcal{T}$)	0.85 ± 3.06	1.89 ± 7.90	0.80 ± 0.53
SCCNN ($\mathcal{T}$)	0.73 ± 1.93	1.67 ± 5.79	0.79 ± 0.53
CellMP ($\mathcal{T}$)	2.31 ± 2.38	2.32 ± 2.43	0.25 ± 0.19
CT ($\mathcal{T}$)	1.06 ± 2.19	1.16 ± 2.77	0.59 ± 0.28
DECT ($\mathcal{T}$)	6.59 ± 3.63	6.61 ± 3.65	12.78 ± 8.03

C.5.1 Betti number prediction

Table C.6: Full results for the Betti number prediction task on all datasets with mean and standard deviation reported over 5 runs. In this table, we report AUROC as performance metric. Transforms are abbreviated as DT (Degree Transform), DTO (Degree Transform Onehot) and RNF (Random Node Features).

		AUROC								
		β_1			β_2			β_3		
Dataset	Model (Class)	DT	DTO	RNF	DT	DTO	RNF	DT	DTO	RNF
$2\text{-}\mathcal{M}^0$	GAT ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	GCN ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	MLP ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	TAG ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	UniMP ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	CellMP ($\mathcal{G}$)	0.62 ± 0.07		$\mathbf{0.84 \pm 0.00}$	0.49 ± 0.06		0.52 ± 0.02			
	CT ($\mathcal{T}$)	$\mathbf{0.93 \pm 0.01}$		0.66 ± 0.02	0.55 ± 0.00		$\mathbf{0.53 \pm 0.01}$			
	DECT ($\mathcal{T}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	SAN ($\mathcal{T}$)	0.55 ± 0.05		0.69 ± 0.06	0.52 ± 0.21		$\mathbf{0.53 \pm 0.01}$			
	SCCN ($\mathcal{T}$)	$\mathbf{0.93 \pm 0.04}$		0.78 ± 0.04	0.55 ± 0.00		$\mathbf{0.53 \pm 0.01}$			
	SCCNN ($\mathcal{T}$)	0.50 ± 0.01		0.50 ± 0.02	0.50 ± 0.19		0.52 ± 0.04			
	SCN ($\mathcal{T}$)	0.56 ± 0.13		0.51 ± 0.03	$\mathbf{0.63 \pm 0.17}$		0.48 ± 0.07			
$3\text{-}\mathcal{M}^0$	GAT ($\mathcal{G}$)	$\mathbf{0.23 \pm 0.00}$	$\mathbf{0.23 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	0.14 ± 0.00
	GCN ($\mathcal{G}$)	$\mathbf{0.23 \pm 0.00}$	$\mathbf{0.23 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	0.14 ± 0.00
	MLP ($\mathcal{G}$)	$\mathbf{0.23 \pm 0.00}$	$\mathbf{0.23 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	0.14 ± 0.00
	TAG ($\mathcal{G}$)	$\mathbf{0.23 \pm 0.00}$	$\mathbf{0.23 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	0.14 ± 0.00
	UniMP ($\mathcal{G}$)	$\mathbf{0.23 \pm 0.00}$	$\mathbf{0.23 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	0.14 ± 0.00
	CellMP ($\mathcal{G}$)	$\mathbf{0.23 \pm 0.00}$		0.23 ± 0.00	0.12 ± 0.00		$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00		0.14 ± 0.00
	CT ($\mathcal{T}$)	$\mathbf{0.23 \pm 0.00}$		0.23 ± 0.00	0.12 ± 0.00		$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00		0.14 ± 0.00
	DECT ($\mathcal{T}$)	$\mathbf{0.23 \pm 0.00}$	$\mathbf{0.23 \pm 0.00}$	0.23 ± 0.00	0.12 ± 0.00	$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	0.14 ± 0.00
	SAN ($\mathcal{T}$)	0.17 ± 0.09		$\mathbf{0.24 \pm 0.01}$	0.12 ± 0.05		$\mathbf{0.12 \pm 0.00}$	$\mathbf{0.19 \pm 0.04}$		$\mathbf{0.15 \pm 0.01}$
	SCCN ($\mathcal{T}$)	$\mathbf{0.23 \pm 0.00}$		0.23 ± 0.00	0.12 ± 0.00		$\mathbf{0.12 \pm 0.00}$	0.14 ± 0.00		0.14 ± 0.00
	SCCNN ($\mathcal{T}$)	0.21 ± 0.11		0.20 ± 0.05	0.12 ± 0.04		0.11 ± 0.01	0.11 ± 0.05		0.13 ± 0.02
	SCN ($\mathcal{T}$)	0.20 ± 0.04		0.23 ± 0.00	$\mathbf{0.15 \pm 0.04}$		$\mathbf{0.12 \pm 0.00}$	0.11 ± 0.07		0.14 ± 0.02
$2\text{-}\mathcal{M}^0_H$	GAT ($\mathcal{G}$)	0.21 ± 0.00	$\mathbf{0.21 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	GCN ($\mathcal{G}$)	0.21 ± 0.00	$\mathbf{0.21 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	MLP ($\mathcal{G}$)	0.21 ± 0.00	$\mathbf{0.21 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	TAG ($\mathcal{G}$)	0.21 ± 0.00	$\mathbf{0.21 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	UniMP ($\mathcal{G}$)	0.21 ± 0.00	$\mathbf{0.21 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	CellMP ($\mathcal{G}$)	0.23 ± 0.01		$\mathbf{0.29 \pm 0.01}$	$\mathbf{0.52 \pm 0.04}$		$\mathbf{0.51 \pm 0.02}$			
	CT ($\mathcal{T}$)	0.27 ± 0.01		0.21 ± 0.00	$\mathbf{0.52 \pm 0.03}$		0.50 ± 0.00			
	DECT ($\mathcal{T}$)	0.21 ± 0.00	$\mathbf{0.21 \pm 0.00}$	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	SAN ($\mathcal{T}$)	0.25 ± 0.01		0.22 ± 0.02	0.48 ± 0.04		0.50 ± 0.02			
	SCCN ($\mathcal{T}$)	$\mathbf{0.29 \pm 0.01}$		0.23 ± 0.01	$\mathbf{0.52 \pm 0.01}$		0.50 ± 0.02			
	SCCNN ($\mathcal{T}$)	0.20 ± 0.05		0.23 ± 0.03	0.49 ± 0.03		$\mathbf{0.51 \pm 0.02}$			
	SCN ($\mathcal{T}$)	0.22 ± 0.00		0.21 ± 0.00	0.49 ± 0.02		0.50 ± 0.01			
$2\text{-}\mathcal{M}^1_H$	GAT ($\mathcal{G}$)	0.22 ± 0.00	0.21 ± 0.00	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	GCN ($\mathcal{G}$)	0.22 ± 0.00	0.21 ± 0.00	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	MLP ($\mathcal{G}$)	0.21 ± 0.01	0.21 ± 0.00	0.21 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	TAG ($\mathcal{G}$)	0.22 ± 0.00	$\mathbf{0.22 \pm 0.00}$	0.22 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	UniMP ($\mathcal{G}$)	0.22 ± 0.00	$\mathbf{0.22 \pm 0.00}$	0.22 ± 0.00	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00			
	CellMP ($\mathcal{T}$)	0.23 ± 0.03		$\mathbf{0.26 \pm 0.00}$	0.49 ± 0.01		0.49 ± 0.01			
	CT ($\mathcal{T}$)	0.23 ± 0.02		0.21 ± 0.00	0.50 ± 0.00		0.50 ± 0.00			
	SAN ($\mathcal{T}$)	0.24 ± 0.01		0.22 ± 0.01	0.50 ± 0.00		0.49 ± 0.01			
	SCCN ($\mathcal{T}$)	$\mathbf{0.27 \pm 0.01}$		0.22 ± 0.01	$\mathbf{0.52 \pm 0.02}$		$\mathbf{0.51 \pm 0.01}$			
	SCCNN ($\mathcal{T}$)	0.21 ± 0.03		0.22 ± 0.02	0.50 ± 0.01		0.50 ± 0.01			
	SCN ($\mathcal{T}$)	0.21 ± 0.00		0.21 ± 0.00	0.49 ± 0.02		$\mathbf{0.51 \pm 0.01}$			

Table C.7: Full results for the Betti number prediction task on all datasets with mean and standard deviation reported over 5 runs. In this table, we report accuracy as performance metric. Transforms are abbreviated as DT (Degree Transform), DTO (Degree Transform Onehot) and RNF (Random Node Features).

		Accuracy											
		β_0			β_1			β_2			β_3		
Dataset	Model (Class)	DT	DTO	RNF	DT	DTO	RNF	DT	DTO	RNF	DT	DTO	RNF
2-$\mathcal{M}^0$	GAT ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.31 ± 0.00	0.31 ± 0.00	0.31 ± 0.00	0.92 ± 0.00	**0.92 ± 0.00**	**0.92 ± 0.00**			
	GCN ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.31 ± 0.00	0.31 ± 0.00	0.31 ± 0.00	0.92 ± 0.00	**0.92 ± 0.00**	**0.92 ± 0.00**			
	MLP ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.31 ± 0.00	0.31 ± 0.00	0.31 ± 0.00	0.92 ± 0.00	**0.92 ± 0.00**	**0.92 ± 0.00**			
	TAG ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.32 ± 0.01	**0.33 ± 0.01**	0.32 ± 0.00	0.92 ± 0.00	**0.92 ± 0.00**	**0.92 ± 0.00**			
	UniMP ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.33 ± 0.00	0.32 ± 0.01	0.32 ± 0.01	0.92 ± 0.00	**0.92 ± 0.00**	**0.92 ± 0.00**			
	CellMP ($\mathcal{G}$)	0.46 ± 0.50		**1.00 ± 0.00**	0.39 ± 0.35		**0.90 ± 0.01**	0.46 ± 0.44		**0.92 ± 0.00**			
	CT ($\mathcal{T}$)	**1.00 ± 0.00**		**1.00 ± 0.00**	**0.93 ± 0.00**		0.87 ± 0.00	**0.93 ± 0.00**		**0.92 ± 0.00**			
	DECT ($\mathcal{T}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.32 ± 0.00	0.32 ± 0.00	0.32 ± 0.00	0.92 ± 0.00	**0.92 ± 0.00**	**0.92 ± 0.00**			
	SAN ($\mathcal{T}$)	0.09 ± 0.04		0.57 ± 0.18	0.12 ± 0.10		0.54 ± 0.11	0.52 ± 0.14		0.73 ± 0.08			
	SCCN ($\mathcal{T}$)	**1.00 ± 0.00**		0.71 ± 0.06	**0.93 ± 0.00**		0.67 ± 0.05	**0.93 ± 0.00**		0.79 ± 0.04			
	SCCNN ($\mathcal{T}$)	0.00 ± 0.00		0.01 ± 0.00	0.03 ± 0.02		0.03 ± 0.01	0.33 ± 0.37		0.49 ± 0.12			
	SCN ($\mathcal{T}$)	0.33 ± 0.38		0.29 ± 0.07	0.21 ± 0.26		0.25 ± 0.10	0.62 ± 0.36		0.65 ± 0.08			
3-$\mathcal{M}^0$	GAT ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
	GCN ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
	MLP ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
	TAG ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
	UniMP ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
	CellMP ($\mathcal{G}$)	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**
	CT ($\mathcal{T}$)	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**
	DECT ($\mathcal{T}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
	SAN ($\mathcal{T}$)	0.01 ± 0.00		0.51 ± 0.12	0.49 ± 0.13		0.71 ± 0.11	0.51 ± 0.22		0.78 ± 0.10	0.01 ± 0.01		0.52 ± 0.07
	SCCN ($\mathcal{T}$)	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**	**1.00 ± 0.00**		**1.00 ± 0.00**
	SCCNN ($\mathcal{T}$)	0.00 ± 0.00		0.00 ± 0.00	0.48 ± 0.14		0.48 ± 0.05	0.60 ± 0.08		0.49 ± 0.12	0.00 ± 0.00		0.00 ± 0.00
	SCN ($\mathcal{T}$)	0.95 ± 0.06		0.95 ± 0.08	0.85 ± 0.19		0.99 ± 0.01	0.80 ± 0.16		0.99 ± 0.00	0.58 ± 0.31		0.92 ± 0.08
2-$\mathcal{M}^0_H$	GAT ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.54 ± 0.00	**0.54 ± 0.00**	0.54 ± 0.00	0.70 ± 0.00	**0.70 ± 0.00**	**0.70 ± 0.00**			
	GCN ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.54 ± 0.00	**0.54 ± 0.00**	0.54 ± 0.00	0.70 ± 0.00	**0.70 ± 0.00**	**0.70 ± 0.00**			
	MLP ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.54 ± 0.00	**0.54 ± 0.00**	0.54 ± 0.00	0.70 ± 0.00	**0.70 ± 0.00**	**0.70 ± 0.00**			
	TAG ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.54 ± 0.00	**0.54 ± 0.00**	0.54 ± 0.00	0.70 ± 0.00	**0.70 ± 0.00**	**0.70 ± 0.00**			
	UniMP ($\mathcal{G}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.54 ± 0.00	**0.54 ± 0.00**	0.54 ± 0.00	0.70 ± 0.00	**0.70 ± 0.00**	**0.70 ± 0.00**			
	CellMP ($\mathcal{G}$)	0.05 ± 0.09		0.98 ± 0.00	0.18 ± 0.24		**0.65 ± 0.01**	0.14 ± 0.31		0.69 ± 0.01			
	CT ($\mathcal{T}$)	**1.00 ± 0.00**		**1.00 ± 0.00**	0.36 ± 0.06		0.54 ± 0.00	0.64 ± 0.17		**0.70 ± 0.01**			
	DECT ($\mathcal{T}$)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.54 ± 0.00	**0.54 ± 0.00**	0.54 ± 0.00	0.70 ± 0.00	**0.70 ± 0.00**	**0.70 ± 0.00**			
	SAN ($\mathcal{T}$)	0.07 ± 0.06		0.26 ± 0.16	0.26 ± 0.05		0.26 ± 0.11	0.43 ± 0.09		0.43 ± 0.06			
	SCCN ($\mathcal{T}$)	**1.00 ± 0.00**		0.48 ± 0.03	**0.69 ± 0.03**		0.40 ± 0.03	**0.71 ± 0.01**		0.52 ± 0.01			
	SCCNN ($\mathcal{T}$)	0.00 ± 0.00		0.01 ± 0.00	0.08 ± 0.10		0.12 ± 0.08	0.27 ± 0.29		0.35 ± 0.08			
	SCN ($\mathcal{T}$)	0.01 ± 0.02		0.13 ± 0.03	0.20 ± 0.01		0.19 ± 0.03	0.25 ± 0.35		0.43 ± 0.03			
2-$\mathcal{M}^1_H$	GAT ($\mathcal{G}$)	0.00 ± 0.00	0.64 ± 0.50	**1.00 ± 0.00**	0.20 ± 0.00	0.43 ± 0.16	**0.54 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**			
	GCN ($\mathcal{G}$)	0.00 ± 0.00	0.68 ± 0.46	**1.00 ± 0.00**	0.20 ± 0.00	0.47 ± 0.15	**0.54 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**			
	MLP ($\mathcal{G}$)	0.53 ± 0.49	**1.00 ± 0.00**	**1.00 ± 0.00**	0.34 ± 0.14	**0.54 ± 0.00**	**0.54 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**			
	TAG ($\mathcal{G}$)	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.20 ± 0.00	0.20 ± 0.00	0.20 ± 0.01	**0.70 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**			
	UniMP ($\mathcal{G}$)	0.00 ± 0.00	0.01 ± 0.01	0.02 ± 0.01	0.20 ± 0.00	0.20 ± 0.00	0.20 ± 0.01	**0.70 ± 0.00**	**0.70 ± 0.00**	**0.70 ± 0.00**			
	CellMP ($\mathcal{T}$)	0.00 ± 0.00		0.00 ± 0.00	0.03 ± 0.06		0.04 ± 0.02	0.09 ± 0.21		0.30 ± 0.00			
	CT ($\mathcal{T}$)	**1.00 ± 0.00**		**1.00 ± 0.00**	**0.54 ± 0.00**		**0.54 ± 0.00**	0.62 ± 0.18		**0.70 ± 0.00**			
	SAN ($\mathcal{T}$)	0.00 ± 0.00		0.01 ± 0.01	0.00 ± 0.00		0.09 ± 0.06	0.56 ± 0.31		0.41 ± 0.31			
	SCCN ($\mathcal{T}$)	0.07 ± 0.15		0.03 ± 0.03	0.05 ± 0.12		0.02 ± 0.01	0.10 ± 0.13		0.25 ± 0.10			
	SCCNN ($\mathcal{T}$)	0.00 ± 0.00		0.00 ± 0.00	0.12 ± 0.11		0.07 ± 0.09	0.48 ± 0.32		0.24 ± 0.33			
	SCN ($\mathcal{T}$)	0.00 ± 0.00		0.03 ± 0.01	0.16 ± 0.09		0.13 ± 0.05	0.28 ± 0.39		0.35 ± 0.16			

C.5.2 Orientability prediction

Table C.8: Full results for the orientability prediction task on all datasets with AUROC and accuracy reported. Mean and standard deviation are taken over 5 runs. Transforms are abbreviated as DT (Degree Transform), DTO (Degree Transform Onehot) and RNF (Random Node Features).

		AUROC			*Accuracy*		
Dataset	Model (Class)	DT	DTO	RNF	DT	DTO	RNF
$2\text{-}\mathcal{M}_0$	GAT ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.92 ± 0.00	$\mathbf{0.92 \pm 0.00}$	0.92 ± 0.00
	GCN ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.92 ± 0.00	$\mathbf{0.92 \pm 0.00}$	0.92 ± 0.00
	MLP ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.92 ± 0.00	$\mathbf{0.92 \pm 0.00}$	0.92 ± 0.00
	TAG ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.92 ± 0.00	$\mathbf{0.92 \pm 0.00}$	0.92 ± 0.00
	UniMP ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.92 ± 0.00	$\mathbf{0.92 \pm 0.00}$	0.92 ± 0.00
	CellMP ($\mathcal{T}$)	$\mathbf{0.65 \pm 0.07}$		$\mathbf{0.55 \pm 0.00}$	0.64 ± 0.26		$\mathbf{0.93 \pm 0.00}$
	CT ($\mathcal{T}$)	0.55 ± 0.00		0.50 ± 0.00	$\mathbf{0.93 \pm 0.00}$		0.92 ± 0.00
	DECT ($\mathcal{T}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.92 ± 0.00	$\mathbf{0.92 \pm 0.00}$	0.92 ± 0.00
	SAN ($\mathcal{T}$)	0.52 ± 0.03		0.51 ± 0.02	0.92 ± 0.00		0.92 ± 0.01
	SCCN ($\mathcal{T}$)	0.55 ± 0.01		0.54 ± 0.01	$\mathbf{0.93 \pm 0.00}$		$\mathbf{0.93 \pm 0.00}$
	SCCNN ($\mathcal{T}$)	0.55 ± 0.09		0.53 ± 0.02	0.87 ± 0.11		0.91 ± 0.01
	SCN ($\mathcal{T}$)	0.53 ± 0.04		0.50 ± 0.01	0.91 ± 0.02		0.92 ± 0.01
$3\text{-}\mathcal{M}_0$	GAT ($\mathcal{G}$)	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
	GCN ($\mathcal{G}$)	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
	MLP ($\mathcal{G}$)	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
	TAG ($\mathcal{G}$)	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
	UniMP ($\mathcal{G}$)	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
	CellMP ($\mathcal{T}$)	$\mathbf{0.18 \pm 0.04}$		$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		$\mathbf{1.00 \pm 0.00}$
	CT ($\mathcal{T}$)	0.14 ± 0.00		$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		$\mathbf{1.00 \pm 0.00}$
	DECT ($\mathcal{T}$)	0.14 ± 0.00	$\mathbf{0.14 \pm 0.00}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
	SAN ($\mathcal{T}$)	0.14 ± 0.00		$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		$\mathbf{1.00 \pm 0.00}$
	SCCN ($\mathcal{T}$)	0.14 ± 0.00		$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		$\mathbf{1.00 \pm 0.00}$
	SCCNN ($\mathcal{T}$)	0.14 ± 0.00		$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		$\mathbf{1.00 \pm 0.00}$
	SCN ($\mathcal{T}$)	0.14 ± 0.00		$\mathbf{0.14 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		$\mathbf{1.00 \pm 0.00}$
$2\text{-}\mathcal{M}_H^0$	GAT ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.70 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	GCN ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.70 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	MLP ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.70 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	TAG ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.70 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	UniMP ($\mathcal{G}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.70 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	CellMP ($\mathcal{T}$)	0.51 ± 0.01		0.50 ± 0.01	0.31 ± 0.02		$\mathbf{0.70 \pm 0.01}$
	CT ($\mathcal{T}$)	0.52 ± 0.03		0.50 ± 0.00	0.72 ± 0.02		$\mathbf{0.70 \pm 0.00}$
	DECT ($\mathcal{T}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.70 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	SAN ($\mathcal{T}$)	0.50 ± 0.02		$\mathbf{0.51 \pm 0.02}$	0.59 ± 0.08		0.60 ± 0.03
	SCCN ($\mathcal{T}$)	$\mathbf{0.54 \pm 0.01}$		0.50 ± 0.01	$\mathbf{0.73 \pm 0.00}$		0.65 ± 0.02
	SCCNN ($\mathcal{T}$)	0.50 ± 0.01		0.50 ± 0.01	0.54 ± 0.23		0.59 ± 0.10
	SCN ($\mathcal{T}$)	0.51 ± 0.02		$\mathbf{0.51 \pm 0.01}$	0.55 ± 0.23		0.61 ± 0.01
$2\text{-}\mathcal{M}_H^1$	GAT ($\mathcal{G}$)	$\mathbf{0.50 \pm 0.00}$	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	GCN ($\mathcal{G}$)	$\mathbf{0.50 \pm 0.00}$	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	MLP ($\mathcal{G}$)	$\mathbf{0.50 \pm 0.00}$	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	TAG ($\mathcal{G}$)	$\mathbf{0.50 \pm 0.00}$	$\mathbf{0.50 \pm 0.01}$	0.50 ± 0.00	0.64 ± 0.15	0.65 ± 0.10	$\mathbf{0.70 \pm 0.00}$
	UniMP ($\mathcal{G}$)	$\mathbf{0.50 \pm 0.01}$	$\mathbf{0.50 \pm 0.00}$	0.50 ± 0.00	0.60 ± 0.15	$\mathbf{0.70 \pm 0.00}$	$\mathbf{0.70 \pm 0.00}$
	CellMP ($\mathcal{T}$)	$\mathbf{0.50 \pm 0.01}$		0.50 ± 0.00	0.38 ± 0.19		$\mathbf{0.70 \pm 0.00}$
	CT ($\mathcal{T}$)	$\mathbf{0.50 \pm 0.00}$		0.50 ± 0.00	$\mathbf{0.70 \pm 0.00}$		$\mathbf{0.70 \pm 0.00}$
	SAN ($\mathcal{T}$)	$\mathbf{0.50 \pm 0.00}$		0.50 ± 0.01	0.54 ± 0.22		0.49 ± 0.18
	SCCN ($\mathcal{T}$)	$\mathbf{0.50 \pm 0.00}$		$\mathbf{0.51 \pm 0.01}$	$\mathbf{0.70 \pm 0.00}$		0.69 ± 0.02
	SCCNN ($\mathcal{T}$)	$\mathbf{0.50 \pm 0.00}$		0.50 ± 0.01	0.46 ± 0.22		0.55 ± 0.17
	SCN ($\mathcal{T}$)	$\mathbf{0.50 \pm 0.00}$		0.50 ± 0.00	0.54 ± 0.22		0.68 ± 0.04

C.5.3 Homeomorphism prediction

Table C.9: Full results for the homeomorphism type prediction task on the full set of surfaces. Performances are reported with mean and standard deviation over five runs with different seeds.

		AUROC			*Accuracy*		
Dataset	Model (Class)	DT	DTO	RNF	DT	DTO	RNF
2-$\mathcal{M}_0$	GAT ($\mathcal{G}$)	0.46 ± 0.00	$\mathbf{0.46 \pm 0.00}$	0.47 ± 0.01	0.80 ± 0.00	$\mathbf{0.80 \pm 0.00}$	0.80 ± 0.00
	GCN ($\mathcal{G}$)	0.46 ± 0.00	$\mathbf{0.46 \pm 0.00}$	0.47 ± 0.01	0.80 ± 0.00	$\mathbf{0.80 \pm 0.00}$	0.80 ± 0.00
	MLP ($\mathcal{G}$)	0.46 ± 0.00	$\mathbf{0.46 \pm 0.00}$	0.46 ± 0.01	0.80 ± 0.00	$\mathbf{0.80 \pm 0.00}$	0.80 ± 0.00
	TAG ($\mathcal{G}$)	0.46 ± 0.00	$\mathbf{0.46 \pm 0.00}$	0.46 ± 0.01	0.80 ± 0.00	$\mathbf{0.80 \pm 0.00}$	0.80 ± 0.00
	UniMP ($\mathcal{G}$)	0.46 ± 0.00	$\mathbf{0.46 \pm 0.00}$	0.46 ± 0.01	0.80 ± 0.00	$\mathbf{0.80 \pm 0.00}$	0.80 ± 0.00
	CellMP ($\mathcal{T}$)	0.85 ± 0.11		$\mathbf{0.89 \pm 0.01}$	0.80 ± 0.34		$\mathbf{0.94 \pm 0.01}$
	CT ($\mathcal{T}$)	$\mathbf{0.91 \pm 0.01}$		0.69 ± 0.15	$\mathbf{0.94 \pm 0.00}$		0.92 ± 0.01
	DECT ($\mathcal{T}$)	0.45 ± 0.00	0.45 ± 0.00	0.45 ± 0.00	0.80 ± 0.00	$\mathbf{0.80 \pm 0.00}$	0.80 ± 0.00
	SAN ($\mathcal{T}$)	0.54 ± 0.10		0.67 ± 0.16	0.35 ± 0.36		0.72 ± 0.26
	SCCN ($\mathcal{T}$)	0.85 ± 0.08		0.66 ± 0.03	0.78 ± 0.35		0.77 ± 0.30
	SCCNN ($\mathcal{T}$)	0.54 ± 0.10		0.61 ± 0.02	0.11 ± 0.00		0.26 ± 0.07
	SCN ($\mathcal{T}$)	0.37 ± 0.12		0.50 ± 0.04	0.50 ± 0.41		0.73 ± 0.09
2-$\mathcal{M}_H^0$	GAT ($\mathcal{G}$)	0.48 ± 0.00	0.49 ± 0.00	0.48 ± 0.00	0.54 ± 0.00	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	GCN ($\mathcal{G}$)	0.49 ± 0.00	0.48 ± 0.01	0.50 ± 0.02	0.54 ± 0.00	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	MLP ($\mathcal{G}$)	0.49 ± 0.00	0.49 ± 0.00	0.48 ± 0.01	0.54 ± 0.00	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	TAG ($\mathcal{G}$)	0.49 ± 0.00	0.49 ± 0.00	0.49 ± 0.01	0.54 ± 0.00	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	UniMP ($\mathcal{G}$)	0.49 ± 0.00	0.49 ± 0.00	0.49 ± 0.01	0.54 ± 0.00	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	CellMP ($\mathcal{T}$)	0.63 ± 0.14		$\mathbf{0.82 \pm 0.00}$	0.19 ± 0.03		$\mathbf{0.73 \pm 0.00}$
	CT ($\mathcal{T}$)	$\mathbf{0.83 \pm 0.01}$		0.50 ± 0.02	$\mathbf{0.74 \pm 0.00}$		0.54 ± 0.00
	DECT ($\mathcal{T}$)	0.50 ± 0.00	$\mathbf{0.50 \pm 0.00}$	0.48 ± 0.01	0.54 ± 0.00	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	SAN ($\mathcal{T}$)	0.49 ± 0.10		0.59 ± 0.10	0.54 ± 0.03		0.61 ± 0.03
	SCCN ($\mathcal{T}$)	0.80 ± 0.00		0.65 ± 0.05	0.73 ± 0.00		0.57 ± 0.03
	SCCNN ($\mathcal{T}$)	0.59 ± 0.10		0.52 ± 0.02	0.51 ± 0.12		0.55 ± 0.01
	SCN ($\mathcal{T}$)	0.53 ± 0.11		0.49 ± 0.06	0.30 ± 0.14		0.43 ± 0.03
2-$\mathcal{M}_H^1$	GAT ($\mathcal{G}$)	0.41 ± 0.03	$\mathbf{0.53 \pm 0.02}$	0.50 ± 0.01	$\mathbf{0.54 \pm 0.00}$	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	GCN ($\mathcal{G}$)	0.42 ± 0.04	0.51 ± 0.04	0.50 ± 0.01	$\mathbf{0.54 \pm 0.00}$	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	MLP ($\mathcal{G}$)	0.43 ± 0.04	0.49 ± 0.06	0.50 ± 0.01	$\mathbf{0.54 \pm 0.00}$	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	TAG ($\mathcal{G}$)	0.42 ± 0.04	0.50 ± 0.03	0.43 ± 0.01	0.51 ± 0.09	0.33 ± 0.15	0.54 ± 0.00
	UniMP ($\mathcal{G}$)	0.45 ± 0.03	0.42 ± 0.03	0.41 ± 0.01	0.45 ± 0.13	$\mathbf{0.54 \pm 0.00}$	0.54 ± 0.00
	CellMP ($\mathcal{T}$)	0.57 ± 0.06		$\mathbf{0.62 \pm 0.02}$	0.47 ± 0.17		$\mathbf{0.55 \pm 0.01}$
	CT ($\mathcal{T}$)	$\mathbf{0.72 \pm 0.13}$		0.49 ± 0.01	0.51 ± 0.20		0.54 ± 0.00
	SAN ($\mathcal{T}$)	0.49 ± 0.02		0.53 ± 0.04	0.48 ± 0.18		0.54 ± 0.00
	SCCN ($\mathcal{T}$)	0.67 ± 0.04		0.53 ± 0.04	$\mathbf{0.54 \pm 0.00}$		0.54 ± 0.00
	SCCNN ($\mathcal{T}$)	0.51 ± 0.01		0.51 ± 0.01	$\mathbf{0.54 \pm 0.00}$		0.54 ± 0.00
	SCN ($\mathcal{T}$)	0.51 ± 0.07		0.49 ± 0.04	0.35 ± 0.18		0.53 ± 0.03

D. Supplemental material for Chapter 7

D.1 Details on LapPE and HodgeLapPE

A popular positional encoding for graph transformers is graph Laplacian eigenvectors (LapPE) [301]. Let $0 \leq \tilde{\lambda}_1 \leq \cdots \leq \tilde{\lambda}_{\tilde{p}} \leq 2$ be the distinct eigenvalues of the normalized graph Laplacian $\tilde{L}_0$ for a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$. As $\tilde{L}_0$ is a real symmetric matrix,

$$\tilde{\lambda}_i = \min_{\substack{\|x\|=1 \\ x \in \langle \tilde{V}_{i-1} \rangle^\perp}} x^T \tilde{L}_0 x = \min_{\substack{\|x\|=1 \\ x \in \langle \tilde{V}_{i-1} \rangle^\perp}} \|\tilde{B}_1^T (D^+)^{1/2} x\|^2 = \min_{\substack{\|x\|=1 \\ x \in \langle \tilde{V}_{i-1} \rangle^\perp}} \sum_{(v_i, v_j) \in E} \left(\frac{x_i}{\sqrt{d(v_i)}} - \frac{x_j}{\sqrt{d(v_j)}} \right)^2, \tag{D.1}$$

where $d(v)$ is the degree of v and $\tilde{V}_{i-1}$ is a set of linearly independent eigenvectors of $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{i-1}$. The minimum attained for any unit-norm eigenvector of $\tilde{\lambda}_i$ is unique up to a sign for non-degenerate eigenvalues. Eigenvectors corresponding to small eigenvalues give a *gradient* on the graph. The *close* vertices (with respect to the adjacency) have close eigenvector coordinates, thus, represent a *relative* position encoding of the vertices of the graph. Thus, LapPE assigns to each vertex v_i a vector $\text{LapPE}(v_i) = (e_i^1, \ldots, e_i^k)$, where $\{e_i^j \mid j = 1, \ldots, k\}$ are eigenvectors of the k smallest eigenvalues counting multiplicities, where k is a hyperparameter. To avoid the sign ambiguity for normalized eigenvectors of non-degenerate eigenvalues, positional encodings are multiplied by random signs at each iteration during training, with the objective of making the neural network invariant to this ambiguity.

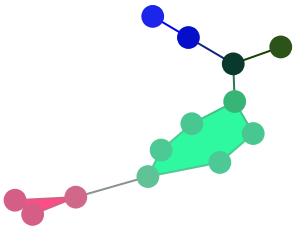


Figure D.1: BSPe positional encoding of length three for a cellular complex with two 2-cells. To generate a colour from the positional encoding, we normalize each coordinate of the positional encodings to the $[0, 1]$ range, generating normalized RGB colours. Note that close cells are assigned similar colours.

HodgeLapPE is the extension of LapPE to cells of arbitrary rank using the unnormalized Hodge Laplacian. However, without normalizing the Hodge Laplacian, we can obtain eigenvalues that are not small enough to make the previous Rayleigh quotient from Equation (D.1) meaningful even in the case of simple simplicial complexes. With the unnormalized version we also do not obtain straightforward formulae that require close cells to have close coordinates in the eigenvector. The following examples illustrate the previous drawbacks.

Example D.1.1 (Large eigenvalues extending LapPE to simplicial complexes using the unnormalized Hodge Laplacian)**.** Take a triangle graph G with vertices v_1, v_2, v_3. The eigenvalues of the unnormalized Hodge Laplacian L_1 given the orientation induced by the ordering of the vertices are 0 and 3. For $\lambda = 3$, an eigenvector of unit length is $(1/\sqrt{2})\,(1, 0, 1)$, that assigns the same coordinates to two of the edges and a different coordinate to the other one, although the three edges are clearly neighborhoods and must have similar representations, making eigenvectors useless in this case.

Example D.1.2 (Rayleigh quotient of the Hodge Laplacian does not produce a gradient of arbitrary dimensional cells)**.** If we want to produce positional encodings for k-cells using the k-th Hodge Laplacian, we obtain

$$\begin{aligned}\lambda_i = \min_{\substack{\|x\|=1 \\ x\in\langle S_{i-1}\rangle^\perp}} x^T \mathbf{L_k} x = \min_{\substack{\|x\|=1 \\ x\in\langle S_{i-1}\rangle^\perp}} \|\widetilde{B}_k x\|^2 + \|\widetilde{B}_{k+1}^T x\|^2 \\ = \min_{\substack{\|x\|=1 \\ x\in\langle S_{i-1}\rangle^\perp}} (1 - \mathbb{1}(k=0)) \sum_{\gamma\in S_{i-1}} \left(\sum_{\substack{\sigma_j\in S_i \\ \gamma<\sigma_j}} (s(\gamma,\sigma_j)x_j) \right)^2 \\ + (1 - \mathbb{1}(d = \dim(K))) \sum_{\gamma\in S_{i+1}} \left(\sum_{\substack{\sigma_j\in S_i \\ \sigma_j<\gamma}} (s(\sigma_j,\gamma)x_j) \right)^2,\end{aligned} \tag{D.2}$$

where $\gamma < \sigma$ means that γ is a proper face of σ and $s(\gamma, \sigma)$ is the value of γ in the boundary of σ. In this case, taking the previous triangle graph G with three vertices, the eigenvalue $\lambda = 0$ has a unit eigenvector $(1/\sqrt{3})\,(-1, -1, 1)$, that assigns to two of the edges the same coordinates and to the third edge the opposite coordinate, though being the three of them neighbors since they are all adjacent. We refer

to [353] for more information about Equation (D.2) in the context of Topological Signal Processing.

D.2 Positional encoding details

In this section, we extend the details about the positional encodings built in section 7.1.3.

Random walks on cellular complexes. Let K be a regular cellular complex. We describe a random walk on the set of k-cells of K. To this end, we first recall that the number of upper and lower adjacent k-cells of a given cell $\sigma \in K_k$ are named respectively the $(0, k+1)$-upper and $(0, k-1)$-lower degree of σ [354],

$$\deg_U^{0,k+1}(\sigma) = \#\{\sigma' \in K_k \colon \sigma \sim_U \sigma'\}; \qquad \deg_L^{0,k-1}(\sigma) = \#\{\sigma' \in K_k \colon \sigma \sim_L \sigma'\},$$

where we write $\sigma \sim_U \sigma'$ and $\sigma \sim_L \sigma'$ for simplices σ, σ' of the same dimension whenever σ and σ' are incident either to a common simplex of dimension $\dim \sigma + 1$ or to a common simplex of dimension $\dim \sigma - 1$, respectively. On the one hand, for each $k \geq 0$, we define a random upper k-walk based on upper adjacencies of the k-cells of K. At each step, we move from a k-cell σ_i to any upper adjacent k-cell σ_j with probability proportional to the number of $(k+1)$-cells in common. To describe this process, we consider a weighted undirected graph G_k^{up}, whose vertices are the k-cells of K and the weight of each edge (σ_i, σ_j) is the number of $(k+1)$-cells whose closure contains both cells (if a k-cell is not upper adjacent to any k-cell, we draw a loop on the corresponding vertex with weight equal to 1). Thus, the upper random k-walk is described by the left stochastic matrix $RW_k^{\text{up}} = wA_k^{\text{up}}(D_k^{\text{up}})^{-1}$, where wA_k^{up} and D_k^{up} denote the weighted adjacency and diagonal weighted degree matrices of the graph G_k^{up}.

On the other hand, for each $k > 0$, we define a random lower k-walk through lower adjacencies of the k-cells of K. In this case, we move from a k-cell σ_i to any lower adjacent k-cell σ_j with probability proportional to the number of $(k-1)$-faces in common. As in the previous case, the random lower walk can be described as a

random walk on a weighted graph G_k^{down}, whose vertices are the k-cells of K and the weight of an edge (σ_i, σ_j) is set as the number of $(k-1)$-cells that both cells have in common (as before, if a k-cell is not lower adjacent to any other k-cell, then we draw a loop on it with weight equal to 1). The lower random k-walk is described by the left stochastic matrix $\mathbf{RW}_k^{\text{down}} = wA_k^{\text{down}}(D_k^{\text{down}})^{-1}$, where wA_k^{down} and D_k^{down} denote the corresponding weighted adjacency and diagonal weighted degree matrices of the graph G_k^{down}. The matrices wA_k^{up} and wA_k^{down} correspond respectively to the upper and lower adjacency matrices A_k^{up} and A_k^{down} with the diagonal entries in null rows replaced with 1.

We can combine both processes to obtain a random walk in which information flows through upper and lower adjacencies, in line with [303]. The idea is as follows: if we are in a k-cell σ with upper and lower adjacent k-cells, we take a step with equal probability via either upper or lower connections. If σ has upper adjacent k-cells but not lower ones, we move following the random upper k-walk process, and vice versa. Lastly, if σ has neither upper nor lower connections, then we do not move.

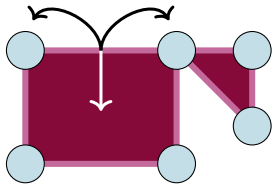
The left stochastic matrix that describes the random k-walk is defined for $\sigma_i, \sigma_j \in K_k$ by

$$(RW_k)_{\sigma_i\sigma_j} = \begin{cases} \frac{1}{2}(RW_k^{\text{up}})_{\sigma_i\sigma_j} + \frac{1}{2}(RW_k^{\text{down}})_{\sigma_i\sigma_j} & \text{if } \deg_U^{0,k+1}(\sigma_j) \neq 0 \text{ and } \deg_L^{0,k-1}(\sigma_j) \neq 0 \\ (RW_k^{\text{up}})_{\sigma_i\sigma_j} & \text{if } \deg_U^{0,k+1}(\sigma_j) \neq 0 \text{ and } \deg_L^{0,k-1}(\sigma_j) = 0 \\ (RW_k^{\text{down}})_{\sigma_i\sigma_j} & \text{if } \deg_U^{0,k+1}(\sigma_j) = 0 \text{ and } \deg_L^{0,k-1}(\sigma_j) \neq 0 \\ \mathbb{1}(i=j) & \text{if } \deg_U^{0,k+1}(\sigma_j) = \deg_L^{(0,k-1)}(\sigma_j) = 0. \end{cases}$$

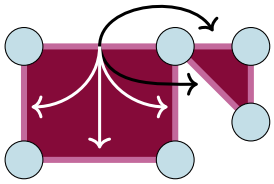
An example of the differences between transitions from an edge in the random walks described in this section and the barycentric subdivision random walks of RWBSPe are described in fig. D.2.

Topological Slepians. Let K be an oriented regular two-dimensional cellular complex, with Hodge Laplacians L_1 and L_2. Hodge Laplacians admit a Hodge decomposition [355], such that $\mathbb{R}^{K_k}$ space can be decomposed as

$$\mathbb{R}^{K_k} = \text{im}\Big(B_k^T\Big) \oplus \text{im}\Big(B_{k+1}\Big) \oplus \ker\Big(L_k\Big), \tag{D.3}$$



(a) RWBSPe random walk possible transitions from the upper-left edge.



(b) RWPe random walk possible transitions from the upper-left edge.

Figure D.2: Differences between RWBSPe and RWPe random walks. RWBSPe random walks can jump from a cell to all its incident and coincident cells, while RWPe random walks can jump from a cell to all its upper and lower adjacent cells.

where $\oplus$ denotes direct sum of vector spaces, and $\ker(-)$ and $\mathrm{im}(-)$ are the kernel and image spaces of a matrix, respectively. The k-feature maps can be represented by means of eigenvector bases of the corresponding Hodge Laplacian. Using the decomposition $L_k = U_k \Lambda_k U_k^T$, the k-th Cellular Fourier Transform (k-CWFT) of a scalar k-feature map F_k in $\mathbb{R}^{K_k}$ is the projection of F_k onto the eigenvectors of L_k [306]:

$$\widehat{F}_k = U_k^T F_k. \tag{D.4}$$

We refer to the eigenvalue set $\mathcal{B}_k$ of the k-CWFT as the frequency domain. An immediate consequence of the Hodge decomposition in (Equation (D.3)) is that the eigenvectors belonging to $\mathrm{im}(L_1^d)$ are orthogonal to those belonging to $\mathrm{im}(L_1^u)$. Therefore, the eigenvectors of L_1 are given by the union of the eigenvectors of L_1^u, the eigenvectors of L_1^d, and the kernel of L_1. We now introduce two localization operators acting onto a k-cell concentration set (thus, onto the topological domain), say $\mathcal{S}_k \subset K^k$, and onto a spectral concentration set (thus, onto the frequency domain), say $\mathcal{F}_k \subset \mathcal{B}_k$, respectively. In particular, we define a cell-limiting operator onto the k-cell set $\mathcal{S}_k$ as

$$C_{\mathcal{S}_k} = \mathrm{diag}(1_{\mathcal{S}}) \in \mathbb{R}^{|K_k| \times |K_k|}, \tag{D.5}$$

where $1_{\mathcal{S}_k} \in \mathbb{R}^{|K_k|}$ is a vector having ones in the index positions specified in $\mathcal{S}_k$, and zero otherwise; and $\mathrm{diag}(z)$ denotes a diagonal matrix having z on the diagonal.

A scalar k-feature map F_k is perfectly localized onto the set $\mathcal{S}_k$ if $C_{\mathcal{S}_k} F_k = F_k$. Similarly, the frequency limiting operator is defined as

$$B_{\mathcal{F}_k} = U_k \operatorname{diag}(1_{\mathcal{F}_k})\, U_k^T \in \mathbb{R}^{|K_k| \times |K_k|}, \tag{D.6}$$

that can be interpreted as a band-pass filter over the frequency set $\mathcal{F}_k$. A k-feature map is perfectly localized over the bandwidth $\mathcal{F}_k$ if $B_{\mathcal{F}_k} F_k = F_k$. The matrices in Equations (D.5) and (D.6) are proper projection operators.

At this point, *k-topological Slepians* are defined as orthonormal vectors that are maximally concentrated over the k-cell set $\mathcal{S}_k$, and perfectly localized onto the bandwidth $\mathcal{F}_k$:

$$\begin{aligned} s_k^i = & \arg\max_{s_k^i} \|C_{\mathcal{S}_k} s_k^i\|_2^2 \\ & \text{subject to } \|s_k^i\| = 1, \quad B_{F_k} s_k^i = s_k^i, \\ & \qquad \langle s_k^i, s_k^j \rangle = 0, \quad j = 1, \ldots, i-1, \text{ if } i > 1, \end{aligned} \tag{D.7}$$

for $i = 1, \ldots, |K_k|$. As shown in [304], the solution of Problem (D.7) is given by the eigenvectors of the matrix operator $B_F C_F B_{F_k}$, i.e.,

$$B_{\mathcal{F}_k} C_{\mathcal{S}_k} B_{\mathcal{F}_k} s_k^i = \lambda_k^i s_k^i. \tag{D.8}$$

It is then clear that the maximum number of k-topological Slepians per each pair of concentration sets $\{\mathcal{S}_k, \mathcal{F}_k\}$ is given by $\operatorname{rank}\{B_{\mathcal{F}_k} C_{\mathcal{S}_k} B_{\mathcal{F}_k}\}$. For this reason and to have a more exhaustive representation of structural and topological properties of the complex [304], we choose a sequence of M concentration sets $\{\mathcal{S}_{k,i}, \mathcal{F}_{k,i}\}_{i=1}^M$ and concatenate the corresponding Slepians. From Equation (D.3), the frequency domain can be partitioned into two separate sets: (i) the set of eigenvalues of L_k^{down}, say $\mathcal{F}_k^d$, and (ii) the set of eigenvalues of L_k^{up}, say $\mathcal{F}_k^u$. For the same reason, we can define two distinct sequences of (not necessarily disjoint) k-cell concentration sets: (i) $\mathcal{K}_k^d$ sets based on lower adjacency (encoded by L_k^{down}), that we refer to as *lower sets*; (ii) $\mathcal{K}_k^u$ sets based on upper adjacency (encoded by L_k^{up}), that we refer to as *upper sets*. It is then natural to associate the frequency concentration set $\mathcal{F}_k^u$ to each upper k-cell concentration set, and the frequency concentration set

$\mathcal{F}_k^d$ to each lower k-cell concentration set. Finally, suppose that the kernel of the Laplacian L_k is not empty. In that case, topological Slepians can be combined with the harmonic eigenvectors of L_k, i.e. the eigenvectors associated with the zero eigenvalues, resulting in a mixed positional encoding strategy. The number of topological Slepians can then be controlled either by tuning $\mathcal{K}_k^d$ and $\mathcal{K}_k^u$, or by taking just the top Slepians per each pair of concentration sets. In this work, we choose the upper and lower k-cell concentration sets as the adjacency and coadjacency of each k-cell including the k-cell itself, respectively, obtaining $|\mathcal{K}_k|$ pairs of concentration sets for each rank k. In this way, we are also sure that the obtained set of topological Slepians comes with theoretical guarantees, i.e., it is an (A, B)-frame [304, 356] In general, the choice of the localization sets is an interesting directions, that could be easily combined with prior knowledge about the complex and the data at hand.

D.3 Architectural details and evaluations on graph benchmarks

The set of results of our experiments for all the combinations of attention mechanisms and positional encodings is reported in Table D.2.

D.3.1 Architecture details

In this section, we describe the two different topological transformer layers proposed in the main text in detail. Following the prenorm [300] design, at the end of the transformer block, composed of several transformer layers, a final layer norm is applied, either for each rank in the case of the pairwise attention transformer layer, and for the whole set of cells in the case of the general attention transformer layer.

Pairwise attention transformer layer. Following the usual prenorm design [300], the output of the cellular transformer layer for a specific rank k_t is denoted $\mathbf{X}_{k_t,l+1}$

and computed in six steps, as follows:

$$\mathbf{X}^1_{k_t,l} = \text{LayerNorm}_{k_t}(\mathbf{X}_{k_t,l}),$$

$$\mathbf{X}^1_{k_s,l} = \text{LayerNorm}_{k_s}(\mathbf{X}_{k_s,l}) \text{ for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}^2_{k_s\to k_t,l} = \mathcal{A}^\bullet_{k_s\to k_t}(\mathbf{X}^1_{k_t,l}, \mathbf{X}^1_{k_s,l}) \text{ for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}^3_{k_s\to k_t,l} = \text{Dropout}(\mathbf{X}^2_{k_s\to k_t,l}) \text{ for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}^4_{k_s\to k_t,l} = \mathbf{X}_{k_t,l} + \mathbf{X}^3_{k_s\to k_t,l} \text{ for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}^5_{k_s\to k_t,l} = \text{LayerNorm}(\mathbf{X}^4_{k_s\to k_t,l}) \text{ for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}^6_{k_s\to k_t,l} = \text{Dropout}(\text{FFN}_2(\text{Dropout}(\text{ReLU}(\text{FFN}_1(\mathbf{X}^5_{k_s\to k_t,l})))))$$
$$\text{for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}_{k_s\to k_t,l+1} = \mathbf{X}^4_{k_s\to k_t,l} + \mathbf{X}^6_{k_s\to k_t,l} \text{ for each } k_s \text{ in the tensor diagram},$$

$$\mathbf{X}_{k_t,l+1} = \sum_{k_s} \mathbf{X}_{k_s\to k_t,l+1}.$$

The LayerNorm is unique for each rank k_t and each layer l.

General attention transformer layer. Similarly to the pairwise attention transformer layer, the general attention transformer layer performs the following steps:

$$\mathbf{X}^1_l = \text{LayerNorm}(\mathbf{X}_l),$$

$$\mathbf{X}^2_l = \mathcal{A}^\bullet_g(\mathbf{X}^1_l),$$

$$\mathbf{X}^3_l = \text{Dropout}(\mathbf{X}^2_l),$$

$$\mathbf{X}^4_l = \mathbf{X}_l + \mathbf{X}^3_l,$$

$$\mathbf{X}^5_l = \text{LayerNorm}(\mathbf{X}^4_l),$$

$$\mathbf{X}^6_l = \text{Dropout}(\text{FFN}_2(\text{Dropout}(\text{ReLU}(\text{FFN}_1(\mathbf{X}^5_l))))),$$

$$\mathbf{X}_{l+1} = \mathbf{X}^4_l + \mathbf{X}^6_l.$$

Training details. All experiments use a `Cosine Annealing` scheduler with linear warmup, an `AdamW` optimizer with $\epsilon = 1^{-8}$, $(\mu_1, \mu_2) = (0.9, 0.999)$ and variable peak learning rate, and a gradient clipping norm of 5. All our transformer architectures, after the transformer layers, use a fully-connected readout whose dropout and number of hidden layers is fixed for each set of experiments, followed by a global

add pool layer over all the vertex signals to perform prediction or regression. The fully-connected block begins with a number of neurons equivalent to the hidden dimension of the transformers and concludes with a number of neurons corresponding to the network's output number. Throughout the block, each hidden layer has half as many neurons as its predecessor.

Features on cells. All graphs in the three datasets contain at least discrete features for the vertices. For the `GCB` dataset, we associate to each edge a feature corresponding to concatenating the signals of its endpoints. For the `ogbg-molhiv` and `ZINC` datasets, the edges contain features, so we do not change them. As a first step in the transformer architecture, we learn an embedding for the discrete features. For the edge features in `GCB`, each vertex feature is embedded individually. For the three datasets, features on the 2-cells are given by sum of the embedded signals of their vertices.

`GCB` architectures. The first six models in Table 7.1 assesses all graph neural networks with different graph pooling layers and common architecture composition given by MP(32)-Pool-MP(32)-Pool-MP(32)-GlobalPool-Dense(Softmax), where MP(32) is a Chebyshev convolutional layer [357] with 32 hidden units, Pool is a pooling message passing layer, GlobalPool is a global pool layer used as readout, and Dense(Softmax) is a dense layer with softmax activation. Skip connections were used. The other state-of-the-art models consist of models proposed in [317, 318].

D.3.2 Dataset statistics

We use the following molecular graph datasets, primarily obtained from MoleculeNet [309]. Unless otherwise specified, we follow the scaffold splitting protocol (80/10/10) provided by OGB [358]. Below, we list the fine-tuning datasets:

(1) **BBBP:** Contains 2039 molecules with binary labels for blood-brain barrier penetration.

(2) **Tox21:** Comprises 7831 molecules with binary labels indicating toxicity for 12 different targets.

(3) **ClinTox:** Includes 1478 drugs with two binary annotations: (1) toxicity in clinical trials, and (2) FDA approval status.

(4) **HIV:** A dataset of 41k molecules annotated with binary labels for their ability to inhibit HIV virus replication.

(5) **BACE:** Consists of 1513 molecules with binary labels indicating binding results for inhibitors of human β-secretase 1.

(6) **SIDER:** Consists of 1427 approved drugs, each annotated with 27 side-effect groups. The prediction task is to determine whether a drug belongs to each side-effect group.

(7) **MUV:** Consists of 93k molecules curated from PubChem bioassays to remove screening artifacts [359]. It provides 17 challenging tasks typically used to assess virtual screening performance.

(8) **FreeSolv:** Contains 642 molecules with hydration free energy data in water.

(9) **ESOL:** Contains 1128 common organic small molecules with water solubility data (log solubility in mols per liter).

(10) **Lipo:** Consists of 4200 molecules with experimental data for the octanol/water distribution coefficient.

Dataset	**#Graphs**	**Avg. #Atoms**	**Avg. #Bonds**	**Split**	**#Classes/Task**
BBBP	2,039	24.1	26.0	Scaffold	1 (Classification)
Tox21	7,831	18.6	19.3	Scaffold	12 (Classification)
ClinTox	1,478	26.2	27.9	Scaffold	2 (Classification)
HIV	41,127	25.5	25.5	Scaffold	1 (Classification)
BACE	1,513	34.1	36.9	Scaffold	1 (Classification)
SIDER	1,427	33.6	35.4	Scaffold	27 (Classification)
MUV	93,087	24.2	26.3	Scaffold	17 (Classification)
Freesolv	642	8.7	8.4	Scaffold	Regression
ESOL	1,427	13.3	13.7	Scaffold	Regression
LIPO	93,087	27.0	29.5	Scaffold	Regression

Table D.1: Statistics of the used datasets.

Table D.2: Ablation studies for the attention mechanism and positional encoding combinations on `GCB`. The reported score is test accuracy (↑). The best results are shown in bold, and the second-best results are underlined.

Attention Type	Positional Encoding	Accuracy (↑)
$\mathcal{A}^s_{k_s \to k_t}$	RWBSPe	**0.7544 ± 0.0169**
$\mathcal{A}^s_g$	HodgeLapPE	<u>0.7509 ± 0.0061</u>
$\mathcal{A}^s_{k_s \to k_t}$	HodgeLapPE	0.7456 ± 0.0132
$\mathcal{A}^s_g$	BSPe	0.7456 ± 0.0061
$\mathcal{A}^s_{k_s \to k_t}$	BSPe	0.7456 ± 0.0110
$\mathcal{A}^s_{k_s \to k_t}$	RWPe	0.7404 ± 0.0161
$\mathcal{A}^s_g$	RWBSPe	0.7333 ± 0.0110
$\mathcal{A}^s_g$	RWPe	0.7281 ± 0.0132
$\mathcal{A}^s_g$	TopoSlepiansPE	0.7228 ± 0.0030
$\mathcal{A}^d_{k_s \to k_t}$	HodgeLapPE	0.7123 ± 0.0249
$\mathcal{A}^d_{k_s \to k_t}$	RWPe	0.7123 ± 0.0122
$\mathcal{A}^d_{k_s \to k_t}$	BSPe	0.7070 ± 0.0199
$\mathcal{A}^s_{k_s \to k_t}$	TopoSlepiansPE	0.7035 ± 0.0322
$\mathcal{A}^d_{k_s \to k_t}$	RWBSPe	0.6982 ± 0.0213
$\mathcal{A}^d_g$	RWPe	0.6825 ± 0.0080
$\mathcal{A}^d_g$	TopoSlepiansPE	0.6807 ± 0.0290
$\mathcal{A}^d_{k_s \to k_t}$	TopoSlepiansPE	0.6807 ± 0.0110
$\mathcal{A}^d_g$	RWBSPe	0.6754 ± 0.0169
$\mathcal{A}^d_g$	BSPe	0.6754 ± 0.0219
$\mathcal{A}^d_g$	HodgeLapPE	0.6737 ± 0.0418

D.4 Additional details and evaluations on MoleculeNet

D.4.1 Experimental details

On exclusion of QM7, QM8, and QM9 datasets. We exclude the QM7, QM8, and QM9 datasets from the MoleculeNet suite due to their inherent dependency on 3D atomic coordinates. Our work aims to extend the capabilities of transformers to capture topological notions and our work can always be combined with a geometric one. The inclusion of this geometric information creates an incompatibility with our evaluation framework and the message we strive to convey. Concretely, these datasets derive labels (e.g., atomization energy, electronic spectra, and dipole moments) from quantum mechanical simulations that explicitly require precise interatomic distances

and angles—information not captured by our AMCC representation, which relies only on combinatorial information from the molecular structure.

D.4.2 Hyperparameters

We report the used hyperparameters both for the graph dataset GCB and for the MoleculeNet in Table D.3.

Table D.3: Cellular transformer hyperparameters for the GCB (left) and MoleculeNet (right) experiments. The attention type and positional encodings vary with configuration. PE stands for positional encodings.

Parameter	GCB	BBBP	Tox21	ClinTox	HIV	BACE	SIDER	MUV	FreeSolv	ESOL	Lipo
#Layers	12	8	4	4	8	3	4	8	12	12	12
Hidden dimension (d^h)	80	80	64	40	80	30	32	80	8	8	80
# Attention heads (m)	8	8	4	4	8	3	4	8	8	8	8
Hidden dimension of each head	10	10	16	10	10	10	8	10	1	1	10
Attention dropout	0.0	0.25	0.1	0.0	0.0	0.25	0.1	0.0	0.1	0.0	0.1
Embedding dropout	0.0	0.25	0.0	0.0	0.0	0.25	0.0	0.0	0.1	0.0	0.1
Readout MLP dropout	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.25
Max epochs	100	200	500	100	500	200	500	100	200	500	1000
Peak learning rate	1×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-3}
Batch size	32	64	256	64	128	64	256	512	64	256	128
Warmup epochs	10	0	0	0	0	0	0	0	0	0	0
Weight decay	1×10^{-5}	1×10^{-2}	1×10^{-5}	1×10^{-1}	1×10^{-5}	1×10^{-2}	1×10^{-5}	1×10^{-5}	1×10^{-4}	1×10^{-4}	1×10^{-2}
# hidden layers readout MLP	0	5	5	3	5	5	5	3	3	3	5
PE preprocessing	Conc.	Conc.	Conc.	Conc.	Conc.	Conc.	Conc.	Conc.	Conc.	Conc.	Conc.
Mol. descriptors used	None	MACCS	None	None	None	MACCS	MACCS	None	None	None	None

D.4.3 Higher-order molecular representation

Molecular representation learning is typically conducted within the graph domain, where atoms are represented as nodes and bonds as edges. However, molecules exhibit higher-order structural features, such as rings, functional groups, and three-dimensional cavities, that cannot be effectively captured using simple graph-based representations. These higher-order relationships are crucial in determining molecular properties. For example, the aromaticity of a benzene ring is a property of its cyclic structure, rather than any individual atom or bond. Similarly, functional groups, such as carboxyl or hydroxyl, involve multi-atom interactions that define their reactivity and properties. Furthermore, three-dimensional cavities and pockets in proteins are critical for their binding properties and interactions with ligands.

Traditional methods, which treat each atom and bond equivalently, often overlook these subtle yet significant structural features. To address this limitation, we introduce *augmented molecular cellular complexes (AMCCs)*, a novel framework

that explicitly incorporates these higher-order motifs. In this approach, atoms are treated as 0-cells (nodes), bonds as 1-cells (edges), and rings or functional groups as 2-cells (faces). The augmented nature of AMCCs comes from the enhanced features assigned to these cells, allowing for a more comprehensive representation of the molecular structure compared to prior works. By explicitly incorporating these higher-order structural motifs into the molecular representation, AMCCs capture structural information that are often crucial for predicting molecular properties, as demonstrated in our experiments.

Mathematically, an augmented molecular cellular complex can be defined as $K = (K_0, K_1, K_2, f_{K_0}, f_{K_1}, f_{K_2})$. Here, K_0 denotes the set of 0-cells (atoms), K_1 represents the set of 1-cells (bonds), and K_2 denotes the set of 2-cells (rings, functional groups, or other higher-order features). The corresponding feature maps f_{K_0}, f_{K_1}, and f_{K_2} represent the attributes of atoms, bonds, and higher-order features, respectively.

D.4.4 Feature representation

(1) *Atom-level (0-cell) Features.* Feature vector of an *atom* includes:

- *Atomic Number*: The atomic number of the atom.
- *Total Valence*: The total number of bonds to an atom.
- *Degree*: The degree of the atom, i.e., the number of neighbors it has.
- *Implicit Valence*: The number of implicit hydrogens the atom has.
- *Aromaticity*: A binary flag indicating whether the atom is part of an aromatic ring (1 if True, else 0).
- *Chiral Tag*: An integer representation of the atom's chirality.
- *Formal Charge (offset)*: The formal charge of the atom offset by a constant value (e.g., +3).
- *Hybridization*: An integer representation of the atom's hybridization state (e.g., sp2, sp3).

(2) *Bond-level (1-cell) Features.* Feature vector of a *bond* includes:

- *Bond Type*: An integer representation of the bond type (single, double, triple, or aromatic).
- *Conjugation*: A binary flag indicating whether the bond is conjugated (1 if True, 0 otherwise).
- *Ring Membership*: A binary flag indicating whether the bond is part of a ring (1 if True, 0 otherwise).
- *Stereo Configuration*: An integer representation of the bond's stereo configuration:
  - *No stereochemistry*
  - *Unspecified stereochemistry (cis)*
  - *Z configuration*
  - *E configuration (trans)*

  While single and triple bonds do not exhibit cis/trans or E/Z stereochemistry like double bonds, they can be involved in other types of stereochemical phenomena.
- *Rotatability*: A binary flag indicating whether the bond is rotatable (1 if rotatable, 0 otherwise).
- *Smallest Ring Size*: The smallest ring size that the bond is a part of, or 0 if not in any ring.
- *Electronegativity Difference*: The difference in electronegativity between the atoms of the bond, indicating bond polarity.
- *Hydrogen Bond Flag*: A binary flag indicating whether the bond is a hydrogen bond (1 if True, 0 otherwise).

(3) *Ring-level (2-cell) Features.* Feature vector of a *ring* includes:

- *Ring Size*: Number of atoms in the ring.
- *Aromaticity Flag*: A binary flag indicating whether the ring is aromatic (1 if True, 0 otherwise).
- *Heteroatom Count*: The number of non-carbon atoms in the ring.

- *Saturatedness*: A binary flag indicating whether the ring is saturated (only single bonds, 1 if True, 0 otherwise).
- *Has Fusion*: A binary flag indicating if the ring shares any atom with another ring (1 if True, 0 otherwise).
- *Average Electronegativity*: Average electronegativity of atoms in the ring.

D.4.5 Extended evaluation of topological positional encodings

We provide an extended assessment of different positional encodings on our cellular transformers in table D.4. In particular we extend table 7.5 with the product attention asks.

Table D.4: PE ablations for our CT on the FreeSolv subset.

Attention	PE	RMSE
$\mathcal{A}^s_{k_s \to k_t}$	BSPe	0.732 ± 0.087
$\mathcal{A}^s_{k_s \to k_t}$	HodgeLapEig	0.768 ± 0.139
$\mathcal{A}^s_{k_s \to k_t}$	RWBSPe	0.800 ± 0.161
$\mathcal{A}^s_{k_s \to k_t}$	RWPe	0.707 ± 0.145
$\mathcal{A}^s_{k_s \to k_t}$	TopoSlepiansPE	0.757 ± 0.163
$\mathcal{A}^d_{k_s \to k_t}$	BSPe	0.753 ± 0.024
$\mathcal{A}^d_{k_s \to k_t}$	HodgeLapEig	0.717 ± 0.042
$\mathcal{A}^d_{k_s \to k_t}$	RWBSPe	$\underline{0.702} \pm \underline{0.032}$
$\mathcal{A}^d_{k_s \to k_t}$	RWPe	$\mathbf{0.662 \pm 0.045}$
$\mathcal{A}^d_{k_s \to k_t}$	TopoSlepiansPE	0.810 ± 0.183

References

[1] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. URL: https://doi.org/10.1007/BF02478259.

[2] Marvin Minsky. *A Neural-Analogue Calculator Based upon a Probability Model of Reinforcement*. Tech. rep. Cambridge, MA: Harvard University Psychological Laboratories, Jan. 1952.

[3] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. URL: http://dx.doi.org/10.1037/h0042519.

[4] John C. Hay, B. Lynch, and David Russell Bedford Smith. "Mark I Perceptron Operators' Manual". In: 1960.

[5] Luciano Floridi. "AI and Its New Winter: from Myths to Realities". In: *Philosophy & Technology* 33.1 (Mar. 2020), pp. 1–3. URL: https://doi.org/10.1007/s13347-020-00396-6.

[6] Abrar Al-Heeti. *Waymo Is Going Fully Autonomous in 5 New Cities. Everything to Know About the Robotaxi*. 2025. URL: https://www.cnet.com/roadshow/news/waymo-is-going-fully-autonomous-in-five-new-cities-everything-to-know-about-the-robotaxi/ (visited on 11/20/2025).

[7] Mustafa Baniodeh et al. *Scaling Laws of Motion Forecasting and Planning – Technical Report*. 2025. arXiv: 2506.08228 [cs.LG]. URL: https://arxiv.org/abs/2506.08228.

[8] Bernardino Romera-Paredes et al. "Mathematical discoveries from program search with large language models". In: *Nature* 625.7995 (Jan. 2024), pp. 468–475. URL: https://doi.org/10.1038/s41586-023-06924-6.

[9] Bogdan Georgiev et al. *Mathematical exploration and discovery at scale*. 2025. arXiv: 2511.02864 [cs.NE]. URL: https://arxiv.org/abs/2511.02864.

[10] DeepMind. *Accelerating fusion science through learned plasma control*. 2025. URL: https://deepmind.google/blog/accelerating-fusion-science-through-learned-plasma-control/ (visited on 11/20/2025).

[11] Chiyuan Zhang et al. "Understanding deep learning requires rethinking generalization". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Sy8gdB9xx.

[12] Chiyuan Zhang et al. "Understanding deep learning (still) requires rethinking generalization". In: *Commun. ACM* 64.3 (Feb. 2021), pp. 107–115. URL: https://doi.org/10.1145/3446776.

[13] Theodore Papamarkou et al. "Position: Topological Deep Learning is the New Frontier for Relational Learning". In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 39529–39555. URL: %5Curl%7Bhttps://proceedings.mlr.press/v235/papamarkou24a.html%7D.

[14] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG].

[15] Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological Data Analysis with Applications*. Cambridge University Press, 2021.

[16] Sergio Barbarossa and Stefania Sardellitti. "Topological Signal Processing: Making Sense of Data Building on Multiway Relations". In: *IEEE Signal Processing Magazine* 37.6 (2020), pp. 174–183.

[17] Claudio Battiloro. "Signal Processing and Learning over Topological Spaces". PhD thesis. Sapienza University of Rome, Mar. 2024.

[18] Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. "Why Deep Learning Works: A Manifold Disentanglement Perspective". In: *IEEE Transactions on Neural Networks and Learning Systems* 27.10 (2016), pp. 1997–2008.

[19] Hao Li et al. "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: %5Curl%7Bhttps://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf%7D.

[20] Nelson Elhage et al. "Toy Models of Superposition". In: *Transformer Circuits Thread* (2022). URL: https://transformer-circuits.pub/2022/toy_model/index.html.

[21] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[22] Jose A. Perea. *A Brief History of Persistence*. 2018. arXiv: 1809.03624 [math.AT]. URL: https://arxiv.org/abs/1809.03624.

[23] Herbert Edelsbrunner and Dmitriy Morozov. "Persistent homology: theory and practice". In: *Proceedings of the European congress of mathematics*. Vol. 2012. 2012.

[24] Patrizio Frosini. "Measuring shapes by size functions". In: *Intelligent Robots and Computer Vision X: Algorithms and Techniques*. Ed. by David P. Casasent. Vol. 1607. International Society for Optics and Photonics. SPIE, 1992, pp. 122–133. URL: https://doi.org/10.1117/12.57059.

[25] Vanessa Robins. "Towards computing homology from approximations". In: *Topology Proceedings* 24 (Jan. 1999).

[26] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. "Topological Persistence and Simplification". In: *Discrete & Computational Geometry* 28.4 (Nov. 2002), pp. 511–533. URL: https://doi.org/10.1007/s00454-002-2885-2.

[27] Afra Zomorodian and Gunnar Carlsson. "Computing Persistent Homology". In: *Discrete & Computational Geometry* 33.2 (Feb. 2005), pp. 249–274. URL: https://doi.org/10.1007/s00454-004-1146-y.

[28] Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. "A Framework for Differential Calculus on Persistence Barcodes". In: *Foundations of Computational Mathematics* 22.4 (Aug. 2022), pp. 1069–1131. URL: https://doi.org/10.1007/s10208-021-09522-y.

[29] Mathieu Carriere et al. "Optimizing persistent homology based functions". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 1294–1303. URL: https://proceedings.mlr.press/v139/carriere21a.html.

[30] Mustafa Hajij et al. *Topological Deep Learning: Going Beyond Graph Data*. 2022. arXiv: 2206.00606 [cs.LG].

[31] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[32] Luis Müller et al. "Attending to Graph Transformers". In: *Transactions on Machine Learning Research* (2024). URL: https://openreview.net/forum?id=HhbqHBBrfZ.

[33] Ladislav Rampášek and Guy Wolf. "Hierarchical Graph Neural Nets can Capture Long-Range Interactions". In: *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*. 2021, pp. 1–6.

[34] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

[35] Cristian Bodnar et al. "Weisfeiler and Lehman Go Cellular: CW Networks". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=uVPZCMVtsSG.

[36] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival". en. In: *Proc Natl Acad Sci U S A* 108.17 (Apr. 2011), pp. 7265–7270.

[37] Zixuan Cang et al. "A topological approach for protein classification". In: *Computational and Mathematical Biophysics* 3.1 (2015). URL: https://doi.org/10.1515/mlbmb-2015-0009.

[38] Chad Giusti et al. "Clique topology reveals intrinsic geometric structure in neural correlations". In: *Proceedings of the National Academy of Sciences* 112.44 (2015), pp. 13455–13460. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1506407112. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1506407112.

[39] Yongjin Lee et al. "Quantifying similarity of pore-geometry in nanoporous materials". In: *Nature Communications* 8.1 (May 2017), p. 15396. URL: https://doi.org/10.1038/ncomms15396.

[40] Pratyush Pranav et al. “The topology of the cosmic web in terms of persistent Betti numbers”. In: *Monthly Notices of the Royal Astronomical Society* 465.4 (Nov. 2016), pp. 4281–4310. eprint: https://academic.oup.com/mnras/article-pdf/465/4/4281/10254227/stw2862.pdf. URL: https://doi.org/10.1093/mnras/stw2862.

[41] Rubén Ballester, Carles Casacuberta, and Sergio Escalera. *Topological Data Analysis for Neural Networks*. Springer Briefs in Computer Science. Springer Cham, 2025. URL: https://link.springer.com/book/9783032082824.

[42] Ciprian A. Corneanu et al. “What Does It Mean to Learn in Deep Networks? And, How Does One Detect Adversarial Attacks?” In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4752–4761.

[43] Ciprian A. Corneanu, Sergio Escalera, and Aleix M. Martinez. “Computing the Testing Error Without a Testing Set”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2674–2682.

[44] Rubén Ballester et al. “Predicting the generalization gap in neural networks using topological data analysis”. In: *Neurocomputing* 596 (2024), p. 127787. URL: https://www.sciencedirect.com/science/article/pii/S0925231224005587.

[45] Rubén Ballester, Carles Casacuberta, and Sergio Escalera. “Decorrelating neurons using persistence”. In: *Proceedings of the 2nd NeurIPS Workshop on Symmetry and Geometry in Neural Representations*. Ed. by Sophia Sanborn et al. Vol. 228. Proceedings of Machine Learning Research. PMLR, 16 Dec 2024, pp. 164–182. URL: https://proceedings.mlr.press/v228/ballester24a.html.

[46] Rubén Ballester and Bastian Rieck. “On the Expressivity of Persistent Homology in Graph Learning”. In: *Proceedings of the Third Learning on Graphs Conference*. Ed. by Guy Wolf and Smita Krishnaswamy. Vol. 269. Proceedings of Machine Learning Research. PMLR, 26–29 Nov 2025, 42:1–42:31. URL: https://proceedings.mlr.press/v269/ballester25a.html.

[47] Rubén Ballester et al. “MANTRA: The Manifold Triangulations Assemblage”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=X6y5CC44HM.

[48] Melih Barsbey et al. “Higher-Order Molecular Learning: The Cellular Transformer”. In: *ICLR 2025 Workshop on Generative and Experimental Perspectives for Biomolecular Design*. 2025. URL: https://openreview.net/forum?id=GW3h79mxrF.

[49] Rubén Ballester et al. *Attending to Topological Spaces: The Cellular Transformer*. 2024. arXiv: 2405.14094 [cs.LG]. URL: https://arxiv.org/abs/2405.14094.

[50] Mathilde Papillon et al. “ICML 2023 Topological Deep Learning Challenge: Design and Results”. In: *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*. Ed. by Timothy Doster et al. Vol. 221. Proceedings of Machine Learning Research. PMLR, 28 Jul 2023, pp. 3–8.

[51] Mustafa Hajij et al. “TopoX: A Suite of Python Packages for Machine Learning on Topological Domains”. In: *Journal of Machine Learning Research* 25.374 (2024), pp. 1–8. URL: http://jmlr.org/papers/v25/24-0110.html.

[52] James R. Munkres. *Topology*. eng. 2nd. Upper Saddle River, NJ: Prentice Hall, Inc., 2000.

[53] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010, pp. I–XII, 1–241.

[54] Vidit Nanda. *Computational Algebraic Topology Lecture Notes*. https://people.maths.ox.ac.uk/nanda/cat/TDANotes.pdf. 2022.

[55] Michael Aschbacher. "Combinatorial cell complexes". In: *Progress in Algebraic Combinatorics*. Vol. 24. Mathematical Society of Japan, 1996, pp. 1–81.

[56] Mustafa Hajij et al. "Combinatorial complexes: bridging the gap between cell complexes and hypergraphs". In: *2023 57th Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2023, pp. 799–803.

[57] Pavel S Aleksandrov. *Combinatorial topology*. Vol. 1. Courier Corporation, 1998.

[58] V. A. Kovalevsky. "Finite topology as applied to image analysis". In: *Comput. Vision Graph. Image Process.* 46.2 (May 1989), pp. 141–161. URL: https://doi.org/10.1016/0734-189X(89)90165-5.

[59] Leo J Grady and Jonathan R Polimeni. *Discrete calculus: Applied analysis on graphs for computational science*. Vol. 3. Springer, 2010.

[60] Timothy E Goldberg. *Combinatorial Laplacians of simplicial complexes*. 2002. URL: https://pi.math.cornell.edu/~goldberg/Papers/CombinatorialLaplacians.pdf.

[61] James R. Munkres. *Elements of Algebraic Topology*. Menlo Park, California, USA: Addison–Wesley Publishing Company, 1984.

[62] Tibor Radó. "Über den Begriff der Riemannschen Fläche". In: *Acta Litt. Sci. Szeged* 2 (1925), pp. 101–121.

[63] Edwin E. Moise. "Affine Structures in 3-Manifolds: V. The Triangulation Theorem and Hauptvermutung". In: *Annals of Mathematics* 56.1 (1952), pp. 96–114. (Visited on 09/18/2024).

[64] Michael Hartley Freedman. "The topology of four-dimensional manifolds". In: *Journal of Differential Geometry* 17.3 (1982), pp. 357–453. URL: https://doi.org/10.4310/jdg/1214437136.

[65] Ciprian Manolescu. *Lectures on the triangulation conjecture*. 2024. arXiv: 1607.08163 [math.GT]. URL: https://arxiv.org/abs/1607.08163.

[66] Allen Hatcher. *Algebraic Topology*. Cambridge, UK: Cambridge University Press, 2002.

[67] Serge Lawrencenko and Seiya Negami. "Constructing the Graphs That Triangulate Both the Torus and the Klein Bottle". In: *Journal of Combinatorial Theory, Series B* 77 (1999), pp. 211–218.

[68] John W. Morgan and Gang Tian. *Ricci flow and the Poincaré conjecture*. Vol. 3. Clay Mathematics Monographs. American Mathematical Society and Clay Mathematics Institute, 2007.

[69] Frédéric Chazal et al. *The Structure and Stability of Persistence Modules*. Vol. 10. SpringerBriefs in Mathematics. Springer, 2016.

[70] Gunnar Carlsson and Vin de Silva. "Zigzag Persistence". In: *Foundations of Computational Mathematics* 10.4 (Aug. 2010), pp. 367–405. URL: https://doi.org/10.1007/s10208-010-9066-0.

[71] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. "Stability of Persistence Diagrams". In: *Discrete & Computational Geometry* 37.1 (Jan. 2007), pp. 103–120. URL: https://doi.org/10.1007/s00454-006-1276-5.

[72] Magnus Botnan and Michael Lesnick. "Algebraic stability of zigzag persistence modules". In: *Algebraic & Geometric Topology* 18.6 (Oct. 2018), pp. 3133–3204. URL: https://doi.org/10.2140%2Fagt.2018.18.3133.

[73] Marco Guerra et al. "Homological scaffold via minimal homology bases". In: *Scientific Reports* 11.1 (2021), p. 5355. URL: https://doi.org/10.1038/s41598-021-84486-1.

[74] Frédéric Chazal, Vin de Silva, and Steve Oudot. "Persistence stability for geometric complexes". In: *Geometriae Dedicata* 173.1 (Dec. 2014), pp. 193–214. URL: https://doi.org/10.1007/s10711-013-9937-z.

[75] Peter Bubenik. "Statistical Topological Data Analysis using Persistence Landscapes". In: *Journal of Machine Learning Research* 16.3 (2015), pp. 77–102. URL: http://jmlr.org/papers/v16/bubenik15a.html.

[76] Henry Adams et al. "Persistence Images: A Stable Vector Representation of Persistent Homology". In: *Journal of Machine Learning Research* 18.8 (2017), pp. 1–35. URL: http://jmlr.org/papers/v18/16-337.html.

[77] Harish Chintakunta et al. "An entropy-based persistence barcode". In: *Pattern Recognition* 48.2 (2015), pp. 391–401. URL: https://www.sciencedirect.com/science/article/pii/S0031320314002453.

[78] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, 2009.

[79] Nieves Atienza, Rocio Gonzalez-Diaz, and Manuel Soriano-Trigueros. "On the stability of persistent entropy and new summary functions for topological data analysis". In: *Pattern Recognition* 107 (2020), p. 107509. URL: https://www.sciencedirect.com/science/article/pii/S0031320320303125.

[80] Thomas Bonis et al. "Persistence-Based Pooling for Shape Pose Recognition". In: *Computational Topology in Image Context*. Ed. by Alexandra Bac and Jean-Luc Mari. Cham: Springer International Publishing, 2016, pp. 19–29.

[81] Barbara Di Fabio and Massimo Ferri. "Comparing persistence diagrams through complex vectors". In: *V. Murino and E. Puppo (Eds.), Image Analysis and Processing – ICIAP 2015*. Vol. 9279. Lecture Notes in Computer Science. Springer, 2015.

[82] Bernadette J. Stolz et al. "Topological data analysis of task-based fMRI data from experiments on schizophrenia". In: *Journal of Physics: Complexity* 2.3 (May 2021), p. 035006. URL: https://dx.doi.org/10.1088/2632-072X/abb4c6.

[83] Christopher Michael Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Ed. by Springer Cham. 1st ed. 2023.

[84] Philipp Grohs and Gitta Kutyniok. *Mathematical Aspects of Deep Learning*. Cambridge University Press, 2022.

[85] Mathilde Papillon et al. *Architectures of Topological Deep Learning: A Survey of Message-Passing Topological Neural Networks*. 2024. arXiv: 2304.10031 [cs.LG]. URL: https://arxiv.org/abs/2304.10031.

[86] Simon J. D. Prince. *Understanding Deep Learning*. MIT Press, 2023. URL: http://udlbook.com.

[87] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. URL: https://doi.org/10.1038/nature14236.

[88] Charline Le Lan. "Understanding representation learning for deep reinforcement learning". PhD thesis. University of Oxford, 2023.

[89] Vincent François-Lavet et al. "An Introduction to Deep Reinforcement Learning". In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pp. 219–354. URL: http://dx.doi.org/10.1561/2200000071.

[90] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[91] Thomas N. Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[92] Petar Veličković et al. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[93] L. Giusti et al. *Simplicial Attention Neural Networks*. 2022. arXiv: 2203.07485 [cs.LG]. URL: https://arxiv.org/abs/2203.07485.

[94] Hanrui Wu et al. "Simplicial Complex Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.1 (2024), pp. 561–575.

[95] Ruochen Yang, Frederic Sala, and Paul Bogdan. "Efficient Representation Learning for Higher-Order Data With Simplicial Complexes". In: *Proceedings of the First Learning on Graphs Conference*. Ed. by Bastian Rieck and Razvan Pascanu. Vol. 198. Proceedings of Machine Learning Research. PMLR, Sept. 2022, 13:1–13:21. URL: https://proceedings.mlr.press/v198/yang22a.html.

[96] Maosheng Yang and Elvin Isufi. *Convolutional Learning on Simplicial Complexes*. 2023. arXiv: 2301.11163 [cs.LG]. URL: https://arxiv.org/abs/2301.11163.

[97] Marco Montagna, Simone Scardapane, and Lev Telyatnikov. *Topological Deep Learning with State-Space Models: A Mamba Approach for Simplicial Complexes*. 2024. arXiv: 2409.12033 [cs.LG]. URL: https://arxiv.org/abs/2409.12033.

[98] Max Horn et al. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=oxxUMeFwEHd.

[99] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML]. URL: https://arxiv.org/abs/1607.06450.

[100] Biao Zhang and Rico Sennrich. "Root Mean Square Layer Normalization". English. In: *Advances in Neural Information Processing Systems 32*. Vol. 32. 33rd Conference on Neural Information Processing Systems, NeurIPS 2019 ; Conference date: 08-12-2019 Through 14-12-2019. Curran Associates Inc, Dec. 2019, pp. 12360–12371. URL: https://neurips.cc/.

[101] Vijay Prakash Dwivedi and Xavier Bresson. "A Generalization of Transformer Networks to Graphs". In: *AAAI Workshop on Deep Learning on Graphs: Methods and Applications* (2021).

[102] Vijay Prakash Dwivedi et al. "Graph Neural Networks with Learnable Structural and Positional Representations". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=wTTjnvGphYj.

[103] Paul J. Kelly. "A congruence theorem for trees". In: *Pacific Journal of Mathematics* 7.1 (1957), pp. 961–968.

[104] Charles J. Colbourn. "On testing isomorphism of permutation graphs". In: *Networks* 11.1 (1981), pp. 13–21.

[105] Pan Li and Jure Leskovec. "The Expressive Power of Graph Neural Networks". In: *Graph Neural Networks: Foundations, Frontiers, and Applications*. Ed. by Lingfei Wu et al. Singapore: Springer Singapore, 2022, pp. 63–98.

[106] Christopher Morris et al. "Weisfeiler and Leman go Machine Learning: The Story so far". 2021. arXiv: 2112.09992 [cs.LG].

[107] Christopher Morris et al. "Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks". In: *AAAI Conference on Artificial Intelligence*. 2019.

[108] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

[109] Martin Grohe. "The Logic of Graph Neural Networks". In: *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2021, pp. 1–17.

[110] Jin-yi Cai, Martin Furer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identification". In: *30th Annual Symposium on Foundations of Computer Science*. 1989, pp. 612–617.

[111] Rubén Ballester, Carles Casacuberta, and Sergio Escalera. *Topological Data Analysis for Neural Network Analysis: A Comprehensive Survey*. 2024. arXiv: 2312.05840 [cs.LG].

[112] Felix Hensel, Michael Moor, and Bastian Rieck. "A Survey of Topological Machine Learning Methods". In: *Frontiers in Artificial Intelligence* 4 (2021). URL: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2021.681108.

[113] Yann Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[114] Ben Zhang and Hongwei Lin. "Functional loops: Monitoring functional organization of deep neural networks using algebraic topology". In: *Neural Networks* 174 (2024), p. 106239. URL: https://www.sciencedirect.com/science/article/pii/S0893608024001631.

[115] Songzhu Zheng et al. "Topological Detection of Trojaned Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=1r2EannVuIA.

[116] National Institute of Standards and Technology. *NIST TrojAI Competition Dataset*. https://pages.nist.gov/trojai/docs/index.html#round-1. 2019.

[117] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. "Topology of Deep Neural Networks". In: *J. Mach. Learn. Res.* 21.1 (Jan. 2020).

[118] Matthew Wheeler, Jose Bouza, and Peter Bubenik. "Activation Landscapes as a Topological Summary of Neural Network Performance". In: *2021 IEEE International Conference on Big Data (Big Data)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2021, pp. 3865–3870. URL: https://doi.ieeecomputersociety.org/10.1109/BigData52589.2021.9671368.

[119] Naoki Akai, Takatsugu Hirayama, and Hiroshi Murase. "Experimental stability analysis of neural networks in classification problems with confidence sets for persistence diagrams". In: *Neural Networks* 143 (2021), pp. 42–51. URL: https://www.sciencedirect.com/science/article/pii/S0893608021001994.

[120] Yang Zhao and Hao Zhang. "Quantitative Performance Assessment of CNN Units via Topological Entropy Calculation". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=xFOyMwWPkz.

[121] Lei Yang, Mengxue Xu, and Yunan He. "Unraveling Convolution Neural Networks: A Topological Exploration of Kernel Evolution". In: *Applied Sciences* 14.5 (2024). URL: https://www.mdpi.com/2076-3417/14/5/2197.

[122] Satoru Watanabe and Hayato Yamana. "Topological measurement of deep neural networks using persistent homology". In: *Annals of Mathematics and Artificial Intelligence* 90.1 (Jan. 2022), pp. 75–92. URL: https://doi.org/10.1007/s10472-021-09761-3.

[123] Satoru Watanabe and Hayato Yamana. "Overfitting measurement of convolutional neural networks using trained network weights". In: *International Journal of Data Science and Analytics* 14.3 (Sept. 2022), pp. 261–278. URL: https://doi.org/10.1007/s41060-022-00332-1.

[124] Davis Blalock et al. "What is the State of Neural Network Pruning?" In: *Proceedings of the 3rd MLSys Conference, Austin, TX, USA*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. Proceedings of Machine Learning and Systems. 2020, pp. 129–146. URL: https://proceedings.mlsys.org/paper_files/paper/2020/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf.

[125] Satoru Watanabe and Hayato Yamana. "Deep Neural Network Pruning Using Persistent Homology". In: *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. 2020, pp. 153–156.

[126] Abir Barbara, Younès Bennani, and Joseph Karkazan. "On the Use of Persistent Homology to Control the Generalization Capacity of a Neural Network". In: *Neural Information Processing*. Ed. by Biao Luo et al. Singapore: Springer Nature Singapore, 2024, pp. 274–286.

[127] Méziane Yacoub and Younès Bennani. "HVS: A heuristic for variable selection in multilayer artificial neural network classifier". In: *HAL* (1997).

[128] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. *The UCI Machine Learning Repository*. https://archive.ics.uci.edu. 1987.

[129] Bastian Rieck et al. "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ByxkijC5FQ.

[130] Leander Girrbach et al. “Caveats of neural persistence in deep neural networks”. In: *40th International Conference on Machine Learning. 2nd Annual TAG in Machine Learning*. 2023.

[131] Thomas Gebhart, Paul Schrater, and Alan Hylton. “Characterizing the Shape of Activation Space in Deep Neural Networks”. In: *2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2019, pp. 1537–1542.

[132] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[133] Morgane Goibert, Elvis Dohmatob, and Thomas Ricatte. “An Adversarial Robustness Perspective on the Topology of Neural Networks”. In: *NeurIPS ML Safety Workshop*. 2022. URL: https://openreview.net/forum?id=EtGd7pF237i.

[134] Mathieu Carrière, Marco Cuturi, and Steve Oudot. “Sliced Wasserstein Kernel for Persistence Diagrams”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 664–673.

[135] Matthew Muller, Steve Kroon, and Stephan Chalup. “Topological Dynamics of Functional Neural Network Graphs During Reinforcement Learning”. In: *Neural Information Processing*. Ed. by Biao Luo et al. Singapore: Springer Nature Singapore, 2024, pp. 190–204.

[136] Daniel Lütgehetmann et al. “Computing Persistent Homology of Directed Flag Complexes”. In: *Algorithms* 13.1 (2020). URL: https://www.mdpi.com/1999-4893/13/1/19.

[137] Asim Kumar Debnath et al. “Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity”. In: *Journal of Medicinal Chemistry* 34.2 (1991), pp. 786–797. eprint: https://doi.org/10.1021/jm00106a046. URL: https://doi.org/10.1021/jm00106a046.

[138] Serguei Barannikov et al. “Representation Topology Divergence: A Method for Comparing Neural Network Representations.” In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 1607–1626. URL: https://proceedings.mlr.press/v162/barannikov22a.html.

[139] Nikita Klyuchnikov et al. *NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing*. 2020. arXiv: 2006.07116 [cs.LG]. URL: https://arxiv.org/abs/2006.07116.

[140] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. *Simplicial Neural Networks*. 2020. arXiv: 2010.03633 [cs.LG]. URL: https://arxiv.org/abs/2010.03633.

[141] Eric Bunch et al. “Simplicial 2-Complex Convolutional Neural Networks”. In: *TDA & Beyond*. 2020. URL: https://openreview.net/forum?id=TLbnsKrt6J-.

[142] A. Keros, Vidit Nanda, and Kartic Subr. "Dist2Cycle: A Simplicial Neural Network for Homology Localization". In: *AAAI Conference on Artificial Intelligence*. 2021.

[143] T. Mitchell Roddenberry, Nicholas Glaze, and Santiago Segarra. "Principled Simplicial Neural Networks for Trajectory Prediction". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 9020–9029. URL: https://proceedings.mlr.press/v139/roddenberry21a.html.

[144] Maosheng Yang, Elvin Isufi, and Geert Leus. *Simplicial Convolutional Neural Networks*. 2021. arXiv: 2110.02585 [cs.LG]. URL: https://arxiv.org/abs/2110.02585.

[145] Cristian Bodnar et al. "Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 1026–1037. URL: https://proceedings.mlr.press/v139/bodnar21a.html.

[146] Yuzhou Chen, Yulia R. Gel, and H. Vincent Poor. "BScNets: Block Simplicial Complex Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.6 (June 2022), pp. 6333–6341. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20583.

[147] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. "Simplicial Attention Networks". In: *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*. 2022. URL: https://openreview.net/forum?id=ScfRNWkpec.

[148] Claudio Battiloro et al. "Generalized Simplicial Attention Neural Networks". In: *IEEE Transactions on Signal and Information Processing over Networks* 10 (2024), pp. 833–850.

[149] See Hian Lee, Feng Ji, and Wee Peng Tay. "SGAT: Simplicial Graph Attention Network". In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. Ed. by Lud De Raedt. Main Track. International Joint Conferences on Artificial Intelligence Organization, July 2022, pp. 3192–3200. URL: https://doi.org/10.24963/ijcai.2022/443.

[150] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[151] Albert Gu and Tri Dao. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". In: *First Conference on Language Modeling*. 2024. URL: https://openreview.net/forum?id=tEYskw1VY2.

[152] Mustafa Hajij et al. "High Skip Networks: A Higher Order Generalization Of Skip Connections". In: *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*. 2022. URL: https://openreview.net/forum?id=Sc8glB-k6e9.

[153] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241.

[154] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[155] Claudio Battiloro et al. "$E(n)$ Equivariant Topological Neural Networks". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=Ax3uliEBVR.

[156] Mustafa Hajij et al. *Simplicial Complex Representation Learning*. 2022. arXiv: 2103.04046 [cs.LG]. URL: https://arxiv.org/abs/2103.04046.

[157] Kelly Maggs, Celia Hacker, and Bastian Rieck. "Simplicial Representation Learning with Neural k-forms". In: *International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=Djw0XhjHZb.

[158] Manuel Lecha et al. "Higher-Order Topological Directionality and Directed Simplicial Neural Networks". In: *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2025, pp. 1–5.

[159] Cristian Bodnar et al. "Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks". In: *International Conference on Machine Learning (ICML)*. Ed. by Marina Meila and Tong Zhang. Proceedings of Machine Learning Research 139. PMLR, 2021, pp. 1026–1037.

[160] Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. "Cell Complex Neural Networks". In: *NeurIPS Workshop TDA and Beyond* (2020).

[161] Lorenzo Giusti et al. "Cell attention networks". In: *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2023, pp. 1–8.

[162] T. Mitchell Roddenberry, Michael T. Schaub, and Mustafa Hajij. "Signal Processing On Cell Complexes". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 8852–8856.

[163] Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. "Transformers generalize deepsets and can be extended to graphs & hypergraphs". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28016–28028.

[164] Zehua Hu et al. "A Semi-supervised Framework with Efficient Feature Extraction and Network Alignment for User Identity Linkage". In: *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part II 26*. Springer. 2021, pp. 675–691.

[165] Ruochi Zhang, Yuesong Zou, and Jian Ma. "Hyper-SAGNN: a self-attention based graph neural network for hypergraphs". In: *International Conference on Learning Representations (ICLR)*. 2020.

[166] Jianling Wang et al. "Next-item recommendation with sequential hypergraphs". In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 2020, pp. 1101–1110.

[167] James Clift et al. "Logic and the 2-Simplicial Transformer". In: *International Conference on Learning Representations*. 2020.

[168] Cai Zhou, Rose Yu, and Yusu Wang. *On the Theoretical Expressive Power and the Design Space of Higher-Order Graph Transformers*. 2024. arXiv: 2404.03380 [cs.LG].

[169] Yuzhou Chen and Yulia R. Gel. "Topological Pooling on Graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.6 (2023), pp. 7096–7103.

[170] Chaolong Ying, Xinjian Zhao, and Tianshu Yu. "Boosting Graph Pooling with Persistent Homology". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=WcmqdY2AKu.

[171] Simon Zhang, Soham Mukherjee, and Tamal K. Dey. "GEFL: Extended Filtration Learning for Graph Classification". In: *Proceedings of the First Learning on Graphs Conference*. Ed. by Bastian Rieck and Razvan Pascanu. Vol. 198. Proceedings of Machine Learning Research. PMLR, 2022, 16:1–16:26.

[172] Yogesh Verma, Amauri H Souza, and Vikas Garg. "Topological Neural Networks go Persistent, Equivariant, and Continuous". In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 49388–49407.

[173] Xue Ye, Fang Sun, and Shiming Xiang. "TREPH: A Plug-In Topological Layer for Graph Neural Networks". In: *Entropy* 25.2 (2023).

[174] Johanna Immonen, Amauri Souza, and Vikas Garg. "Going beyond persistent homology using persistent homology". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 63150–63173.

[175] Andac Demir et al. "ToDD: Topological compound fingerprinting in computer-aided drug discovery". In: *Advances in Neural Information Processing Systems* 35 (2022).

[176] Yuankai Luo, Lei Shi, and Veronika Thost. "Improving self-supervised molecular representation learning using persistent homology". In: *Advances in Neural Information Processing Systems* 36 (2024).

[177] Xu Gong et al. "MIFS: An adaptive multipath information fused self-supervised framework for drug discovery". In: *Neural Networks* (2025).

[178] Nicolas Swenson et al. "PersGNN: applying topological data analysis and geometric deep learning to structure-based protein function prediction". In: *arXiv preprint arXiv:2010.16027* (2020).

[179] Zixuan Cang and Guo-Wei Wei. "TopologyNet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions". In: *PLoS computational biology* 13.7 (2017), e1005690.

[180] Aharon Azulay and Yair Weiss. "Why do deep convolutional networks generalize so poorly to small image transformations?" In: *Journal of Machine Learning Research* 20.184 (2019), pp. 1–25. URL: http://jmlr.org/papers/v20/19-519.html.

[181] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6572.

[182] Yiding Jiang et al. *NeurIPS 2020 Competition: Predicting Generalization in Deep Learning*. 2020. arXiv: 2012.07976 [cs.LG]. URL: https://arxiv.org/abs/2012.07976.

[183] Carlos Lassance et al. *Ranking Deep Learning Generalization using Label Variation in Latent Geometry Graphs*. 2020. arXiv: 2011.12737 [cs.LG].

[184] Parth Natekar and Manik Sharma. *Representation Based Complexity Measures for Predicting Generalization in Deep Learning*. 2020. arXiv: 2012.02775 [cs.LG].

[185] Sumukh K. Aithal, Dhruva Kashyap, and Natarajan Subramanyam. *Robustness to Augmentations as a Generalization metric*. 2021. arXiv: 2101.06459 [cs.LG].

[186] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *The 3rd International Conference on Learning Representations (ICLR2015)*. 2015. URL: https://arxiv.org/abs/1409.1556.

[187] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[188] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2014. arXiv: 1312.4400 [cs.NE].

[189] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 2011. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[190] Thomas G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural Comput.* 10.7 (Oct. 1998), pp. 1895–1923. URL: https://doi.org/10.1162/089976698300017197.

[191] David L. Davies and Donald W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227.

[192] Yuepeng Zhou et al. "Improving the performance of VGG through different granularity feature combinations". In: *IEEE Access* 9 (2020), pp. 26208–26220.

[193] Thomas H. Cormen et al. *Introduction to Algorithms*. 3rd. The MIT Press, 2009.

[194] Nikola Milosavljević, Dmitriy Morozov, and Primoz Škraba. "Zigzag Persistent Homology in Matrix Multiplication Time". In: *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*. SoCG'11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 216–225. URL: https://www.mrzv.org/publications/zzph-mmt/socg11/.

[195] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

[196] Ehsan Nezhadarya et al. "Adaptive Hierarchical Down-Sampling for Point Cloud Classification". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 12953–12961.

[197] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2.

[198] Dashti Ali et al. *A Survey of Vectorization Methods in Topological Data Analysis*. 2022. arXiv: 2212.09703 [math.AT].

[199] Vitor A. C. Horta et al. "Extracting knowledge from Deep Neural Networks through graph analysis". In: *Future Generation Computer Systems* 120 (2021), pp. 109–118.

[200] Marlene R. Cohen and Adam Kohn. "Measuring and interpreting neuronal correlations". In: *Nature Neuroscience* 14.7 (2011), pp. 811–819.

[201] Adam Kohn and Matthew A. Smith. "Stimulus dependence of neuronal correlation in primary visual cortex of the macaque". In: *Journal of Neuroscience* 25.14 (2005), pp. 3661–3673.

[202] Gaojie Jin et al. "How does Weight Correlation Affect Generalisation Ability of Deep Neural Networks?" In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21346–21356. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/f48c04ffab49ff0e5d1176244fdfb65c-Paper.pdf.

[203] Michael Cogswell et al. "Reducing Overfitting in Deep Networks by Decorrelating Representations". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016.

[204] Jay A. Hennig et al. "Constraints on neural redundancy". en. In: *Elife* 7 (Aug. 2018).

[205] Beatriz E. P. Mizusaki and Cian O'Donnell. "Neural circuit function redundancy in brain disorders". In: *Current Opinion in Neurobiology* 70 (2021), pp. 74–80. URL: https://www.sciencedirect.com/science/article/pii/S0959438821000787.

[206] Pavlo Molchanov et al. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=SJGCiw5gl.

[207] Bernard Chazelle. "A Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity". In: *Journal of the ACM* 47.6 (Nov. 2000), pp. 1028–1047. URL: https://doi.org/10.1145/355541.355562.

[208] Robert Endre Tarjan. "Efficiency of a Good But Not Linear Set Union Algorithm". In: *Journal of the ACM* 22.2 (Apr. 1975), pp. 215–225. URL: https://doi.org/10.1145/321879.321884.

[209] Gunnar Carlsson and Rickard Brüel Gabrielsson. "Topological Approaches to Deep Learning". In: *Topological Data Analysis*. Ed. by Nils A. Baas et al. Cham: Springer International Publishing, 2020, pp. 119–146.

[210] David Bertoin et al. “Numerical influence of ReLU’(0) on backpropagation”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 468–479. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/043ab21fc5a1607b381ac3896176dac6-Paper.pdf.

[211] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. *MNIST Handwritten Digit Database, AT & T Labs*. 2010. URL: http://yann.lecun.com/exdb/mnist.

[212] Robert Tibshirani. “Regression shrinkage and selection via the Lasso: a retrospective”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.3 (2011), pp. 273–282. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2011.00771.x.

[213] Janez Demšar. “Statistical Comparisons of Classifiers over Multiple Data Sets”. In: *Journal of Machine Learning Research* 7.1 (2006), pp. 1–30. URL: http://jmlr.org/papers/v7/demsar06a.html.

[214] Scott Rostrup, Shweta Srivastava, and Kishore Singhal. “Fast and Memory-Efficient Minimum Spanning Tree on the GPU”. In: *International Journal of Computational Science and Engineering (IJCSE)* 8.1 (Feb. 2013), pp. 21–33. URL: https://doi.org/10.1504/IJCSE.2013.052115.

[215] Peter Sanders and Matthias Schimek. *Engineering Massively Parallel MST Algorithms*. 2023. arXiv: 2302.12199 [cs.DC]. URL: https://arxiv.org/abs/2302.12199.

[216] Tolga Birdal et al. “Intrinsic Dimension, Persistent Homology and Generalization in Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 6776–6789. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/35a12c43227f217207d4e06ffefe39d3-Paper.pdf.

[217] Michael M. Bronstein et al. “Geometric Deep Learning: Going beyond Euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[218] Mathieu Carrière et al. “PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by Silvia Chiappa and Roberto Calandra. Proceedings of Machine Learning Research 108. PMLR, 2020, pp. 2786–2796.

[219] Yuzhou Chen, Baris Coskunuzer, and Yulia Gel. “Topological Relational Learning on Graphs”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 27029–27042.

[220] Christoph Hofer et al. “Deep Learning with Topological Signatures”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. 2017, pp. 1634–1644.

[221] Christoph D. Hofer, Roland Kwitt, and Marc Niethammer. “Learning Representations of Persistence Barcodes”. In: *Journal of Machine Learning Research* 20.126 (2019), pp. 1–45.

[222] Christoph D. Hofer et al. “Graph Filtration Learning”. In: *International Conference on Machine Learning (ICML)*. Ed. by Hal Daumé III and Aarti Singh. Proceedings of Machine Learning Research 119. PMLR, 2020, pp. 4314–4323. arXiv: 1905.10996 [cs.LG].

[223] Bastian Rieck, Christian Bock, and Karsten Borgwardt. “A Persistent Weisfeiler–Lehman Procedure for Graph Classification”. In: *International Conference on Machine Learning (ICML)*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Proceedings of Machine Learning Research 97. PMLR, 2019, pp. 5448–5458.

[224] Zuoyu Yan et al. “Neural Approximation of Graph Topological Features”. In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=qwjrO7Rewqy.

[225] Qi Zhao and Yusu Wang. “Learning metrics for persistence-based summaries and applications for graph classification”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. 2019, pp. 9855–9866.

[226] Qi Zhao et al. “Persistence Enhanced Graph Neural Network”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by Silvia Chiappa and Roberto Calandra. Proceedings of Machine Learning Research 108. PMLR, 2020, pp. 2896–2906.

[227] Zhengdao Chen et al. “Can Graph Neural Networks Count Substructures?” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 10383–10395.

[228] Giorgos Bouritsas et al. “Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2023), pp. 657–668.

[229] Chaitanya K. Joshi et al. “On the Expressive Power of Geometric Graph Neural Networks”. In: *International Conference on Machine Learning (ICML)*. Ed. by Andreas Krause et al. Proceedings of Machine Learning Research 202. PMLR, 2023, pp. 15330–15355.

[230] Ryoma Sato. “A Survey on The Expressive Power of Graph Neural Networks”. 2020. arXiv: 2003.04078 [cs.LG].

[231] Peter Bubenik et al. “Persistent homology detects curvature”. In: *Inverse Problems* 36.2 (2020), p. 025008.

[232] Sunhyuk Lim, Facundo Memoli, and Osman Berat Okutan. *Vietoris–Rips Persistent Homology, Injective Metric Spaces, and The Filling Radius*. 2020. arXiv: 2001.07588 [math.AT].

[233] Renata Turkeš, Guido Montúfar, and Nina Otter. “On the Effectiveness of Persistent Homology”. In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=DRjUkfExCix.

[234] Henry Adams and Baris Coskunuzer. “Geometric Approaches to Persistent Homology”. In: *SIAM Journal on Applied Algebra and Geometry* 6.4 (2022), pp. 685–710.

[235] Sandra Kiefer. “Power and limits of the Weisfeiler-Leman algorithm”. PhD thesis. Dissertation, RWTH Aachen University, 2020, 2020.

[236] Brendan McKay. *Collection of strongly-regular graphs*. URL: https://users.cecs.anu.edu.au/~bdm/data/graphs.html.

[237] Yanbo Wang and Muhan Zhang. *Towards Better Evaluation of GNN Expressiveness with BREC Dataset*. 2023. arXiv: 2304.07702 [cs.LG].

[238] Leslie O’Bray, Bastian Rieck, and Karsten Borgwardt. “Filtration Curves for Graph Representation”. In: *International Conference on Knowledge Discovery & Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1267–1275.

[239] Bastian Rieck, Filip Sadlo, and Heike Leitte. “Topological Machine Learning with Persistence Indicator Functions”. In: *Topological Methods in Data Analysis and Visualization V*. Ed. by Hamish Carr et al. Cham, Switzerland: Springer, 2020, pp. 87–101.

[240] Yann Ollivier. “Ricci curvature of metric spaces”. In: *Comptes Rendus Mathematique* 345.11 (2007), pp. 643–646.

[241] E. Levina and P. Bickel. “The Earth Mover’s distance is the Mallows distance: some insights from statistics”. In: *IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 2001, pp. 251–256.

[242] Cédric Villani. *Optimal Transport. Old and New*. Grundlehren der mathematischen Wissenschaften 338. Springer, 2009.

[243] Corinna Coupette, Sebastian Dalleiger, and Bastian Rieck. “Ollivier–Ricci Curvature for Hypergraphs: A Unified Framework”. In: *International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=sPCKNl5qDps.

[244] Joshua Southern et al. “Curvature Filtrations for Graph Generative Model Evaluation”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 63036–63061.

[245] Areejit Samal et al. “Comparative analysis of two discretizations of Ricci curvature for complex networks”. In: *Scientific Reports* 8.1 (2018), p. 8650.

[246] László Babai. “Chromatic number and subgraphs of Cayley graphs”. In: *Theory and Applications of Graphs*. Ed. by Yousef Alavi and Don R. Lick. 1978, pp. 10–22.

[247] László Babai. “Automorphism Groups, Isomorphism, Reconstruction”. In: *Handbook of Combinatorics*. MIT Press, 1996, pp. 1447–1540.

[248] Cai Heng Li. “On isomorphisms of connected Cayley graphs”. In: *Discrete Mathematics* 178.1 (1998), pp. 109–122.

[249] Cai Heng Li. “On isomorphisms of finite Cayley graphs—a survey”. In: *Discrete Mathematics* 256.1 (2002), pp. 301–334.

[250] Dave Witte Morris, Joy Morris, and Gabriel Verret. *Isomorphisms of Cayley graphs on nilpotent groups*. 2016. arXiv: 1603.01883 [math.CO].

[251] Brian Alspach. "Isomorphism and Cayley graphs on abelian groups". In: *Graph Symmetry: Algebraic Methods and Applications*. Ed. by Geňa Hahn and Gert Sabidussi. Dordrecht, Netherlands: Springer, 1997, pp. 1–22.

[252] Nino Shervashidze et al. "Weisfeiler–Lehman Graph Kernels". In: *Journal of Machine Learning Research* 12.77 (2011), pp. 2539–2561.

[253] L. Lovász. "Spectra of graphs with transitive groups". In: *Periodica Mathematica Hungarica* 6.2 (1975), pp. 191–195.

[254] Pál András Papp and Roger Wattenhofer. "A Theoretical Comparison of Graph Neural Network Extensions". In: *International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Proceedings of Machine Learning Research 162. PMLR, 2022, pp. 17323–17345.

[255] Weihua Hu et al. "Open Graph Benchmark: Datasets for Machine Learning on Graphs". In: *arXiv preprint arXiv:2005.00687* (2020).

[256] A. K. Hartmann and M. Mézard. "Distribution of diameters for Erdős–Rényi random graphs". In: *Physical Review E* 97 (3 2018), p. 032128.

[257] Ernst Röell and Bastian Rieck. "Differentiable Euler Characteristic Transforms for Shape Classification". In: *International Conference on Learning Representations*. 2024. arXiv: 2310.07630 [cs.LG]. URL: https://openreview.net/forum?id=MO632iPq3I.

[258] Elizabeth Munch. "An Invitation to the Euler Characteristic Transform". Preprint. 2023. arXiv: 2310.10395 [cs.CG].

[259] Lewis Marsh and David Beers. "Stability and Inference of the Euler Characteristic Transform". Preprint. 2023. arXiv: 2303.13200 [math.ST].

[260] Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. "Persistent homology transform for modeling shapes and surfaces". In: *Information and Inference: A Journal of the IMA* 3.4 (2014), pp. 310–344.

[261] Frédéric Chazal, Vin de Silva, and Steve Oudot. "Persistence stability for geometric complexes". In: *Geometriae Dedicata* 173.1 (2014), pp. 193–214.

[262] Frédéric Chazal et al. *The structure and stability of persistence modules*. Vol. 10. SpringerBriefs in Mathematics. Cham, Switzerland: Springer, 2016.

[263] Ulrich Bauer, Claudia Landi, and Facundo Mémoli. "The Reeb Graph Edit Distance is Universal". In: *Foundations of Computational Mathematics* 21.5 (2021), pp. 1441–1464.

[264] John Palowitch et al. "GraphWorld: Fake Graphs Bring Real Insights for GNNs". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 3691–3701.

[265] Jan Tönshoff et al. "Where Did the Gap Go? Reassessing the Long-Range Graph Benchmark". In: *The Second Learning on Graphs Conference*. 2023. URL: https://openreview.net/forum?id=rIUjwxc5lj.

[266] Renming Liu et al. "Taxonomy of Benchmarks in Graph Representation Learning". In: *Proceedings of the First Learning on Graphs Conference*. Ed. by Bastian Rieck and Razvan Pascanu. Proceedings of Machine Learning Research 198. PMLR, 2022, 6:1–6:25. arXiv: 2206.07729 [cs.LG]. URL: https://proceedings.mlr.press/v198/liu22a.html.

[267] Federico Errica et al. “A Fair Comparison of Graph Neural Networks for Graph Classification”. In: *International Conference on Learning Representations (ICLR)*. 2020. URL: https://openreview.net/forum?id=HygDF6NFPB.

[268] Maya Bechler-Speicher et al. “Graph Neural Networks Use Graphs When They Shouldn’t”. In: *Proceedings of the 41st International Conference on Machine Learning (ICML)*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 2024, pp. 3284–3304.

[269] Corinna Coupette et al. *No Metric to Rule Them All: Toward Principled Evaluations of Graph-Learning Datasets*. 2025. arXiv: 2502.02379 [cs.LG]. URL: https://arxiv.org/abs/2502.02379.

[270] Maya Bechler-Speicher et al. *Position: Graph Learning Will Lose Relevance Due To Poor Benchmarks*. 2025. arXiv: 2502.14546 [cs.LG]. URL: https://arxiv.org/abs/2502.14546.

[271] Mathieu Alain et al. “Gaussian Processes on Cellular Complexes”. In: *International Conference on Machine Learning (ICML)*. 2024.

[272] Karthikeyan Natesan Ramamurthy, Aldo Guzmán-Sáenz, and Mustafa Hajij. “TOPO-MLP: A Simplicial Network without Message Passing”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5.

[273] Maosheng Yang, Viacheslav Borovitskiy, and Elvin Isufi. “Hodge-Compositional Edge Gaussian Processes”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2024.

[274] Justin Gilmer et al. “Neural Message Passing for Quantum Chemistry”. In: *International Conference on Machine Learning (ICML)*. 2017.

[275] Austin R. Benson et al. “Simplicial closure and higher-order link prediction”. In: *Proceedings of the National Academy of Sciences* 115.48 (2018), E11221–E11230. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1800683115. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1800683115.

[276] Bosiljka Tadić, Miroslav Andjelković, and Roderick Melnik. “Functional Geometry of Human Connectomes”. In: *Scientific Reports* 9.1 (Aug. 2019), p. 12060. URL: https://doi.org/10.1038/s41598-019-48568-5.

[277] Chad Giusti, Robert Ghrist, and Danielle S. Bassett. “Two’s company, three (or more) is a simplex”. In: *Journal of Computational Neuroscience* 41.1 (Aug. 2016), pp. 1–14. URL: https://doi.org/10.1007/s10827-016-0608-6.

[278] Yam Eitan et al. “Topological Blind Spots: Understanding and Extending Topological Deep Learning Through the Lens of Expressivity”. In: *International Conference on Learning Representations (ICLR)*. 2025. URL: https://openreview.net/forum?id=EzjsoomYEb.

[279] Lev Telyatnikov et al. *TopoBenchmarkX: A Framework for Benchmarking Topological Deep Learning*. 2024. arXiv: 2406.06642 [cs.LG]. URL: https://arxiv.org/abs/2406.06642.

[280] Guillermo Bernárdez et al. *ICML Topological Deep Learning Challenge 2024: Beyond the Graph Domain*. 2024. arXiv: 2409.05211 [cs.LG]. URL: https://arxiv.org/abs/2409.05211.

[281] Jakob Jonsson. *Simplicial Complexes of Graphs*. Heidelberg, Germany: Springer, 2007.

[282] Rahul Paul and Stephan Chalup. "Estimating Betti Numbers Using Deep Learning". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–7.

[283] Simon Apers et al. "A (simple) classical algorithm for estimating Betti numbers". In: *Quantum* 7 (Dec. 2023), p. 1202. URL: http://dx.doi.org/10.22331/q-2023-12-06-1202.

[284] Frank H. Lutz. *The Manifold Page*. Accessed: September 19, 2024. 2017. URL: https://www3.math.tu-berlin.de/IfM/Nachrufe/Frank_Lutz/stellar/.

[285] Noémie Jaquier et al. "Geometry-aware Bayesian Optimization in Robotics using Riemannian Matérn Kernels". In: *Proceedings of the 5th Conference on Robot Learning*. 2022.

[286] Keenan Crane. "Discrete differential geometry: An applied introduction". In: *Notices of the AMS* (2018).

[287] Boris Bonev et al. "Spherical Fourier Neural Operators: Learning Stable Dynamics on the Sphere". In: *International Conference on Machine Learning (ICML)*. 2023.

[288] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the Manifold Hypothesis". In: *Journal of the American Mathematical Society* (2016).

[289] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[290] Tom Preston-Werner. *Semantic Versioning 2.0.0*. http://semver.org. Accessed: September 21, 2024.

[291] Yunsheng Shi et al. "Masked label prediction: Unified message passing model for semi-supervised classification". In: *arXiv preprint arXiv:2009.03509* (2020).

[292] Jian Du et al. "Topology adaptive graph convolutional networks". In: *arXiv preprint arXiv:1710.10370* (2017).

[293] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159.

[294] Qimai Li, Zhichao Han, and Xiao-Ming Wu. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning". In: AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018.

[295] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *International Conference on Learning Representations (ICLR)*. 2021. URL: https://openreview.net/forum?id=i80OPhOCVH2.

[296] Jake Topping et al. "Understanding Over-Squashing and Bottlenecks on Graphs via Curvature". In: *International Conference on Learning Representations (ICLR)*. 2022. URL: https://openreview.net/forum?id=7UmjRGzp-A.

[297] Mathieu Alain et al. "Graph Classification Gaussian Processes via Hodgelet Spectral Features". In: *Advances in Neural Information Processing Systems (NeurIPS) 2024 Workshop Workshop on Bayesian Decision-making and Uncertainty (BDU)*. 2024. URL: https://arxiv.org/pdf/2410.10546.

[298] Yi Jiang et al. “Topological representations of crystalline compounds for the machine-learning prediction of materials properties”. In: *npj computational materials* 7.1 (2021), p. 28.

[299] Ladislav Rampášek et al. “Recipe for a General, Powerful, Scalable Graph Transformer”. In: *Advances in Neural Information Processing Systems* 35 (2022).

[300] Ruibin Xiong et al. “On Layer Normalization in the Transformer Architecture”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. PMLR, 2020.

[301] Vijay Prakash Dwivedi et al. “Benchmarking Graph Neural Networks”. In: *Journal of Machine Learning Research* 24.43 (2023), pp. 1–48.

[302] Cai Zhou, Xiyuan Wang, and Muhan Zhang. “Facilitating Graph Neural Networks with Random Walk on Simplicial Complexes”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[303] Michael T. Schaub et al. “Random Walks on Simplicial Complexes and the Normalized Hodge 1-Laplacian”. In: *SIAM Review* 62.2 (2020), pp. 353–391.

[304] Claudio Battiloro, Paolo Di Lorenzo, and Sergio Barbarossa. “Topological Slepians: Maximally Localized Representations of Signals Over Simplicial Complexes”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5.

[305] Michelle L. Wachs. “Poset topology: tools and applications”. In: *arXiv preprint math/0602226* (2006). arXiv: `math/0602226 [math.CO]`.

[306] Stefania Sardellitti and Sergio Barbarossa. “Topological Signal Representation and Processing over Cell Complexes”. In: *arXiv preprint arXiv:2201.08993* (2022).

[307] Kha-Dinh Luong and Ambuj K Singh. “Fragment-based pretraining and finetuning on molecular graphs”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[308] Zaixi Zhang et al. “Motif-based graph self-supervised learning for molecular property prediction”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15870–15882.

[309] Zhenqin Wu et al. “MoleculeNet: a benchmark for molecular machine learning”. In: *Chemical science* 9.2 (2018), pp. 513–530.

[310] Filippo Maria Bianchi, Claudio Gallicchio, and Alessio Micheli. “Pyramidal Reservoir Graph Neural Network”. In: *Neurocomputing* 470 (2022), pp. 389–404.

[311] *Benchmark dataset for graph classification GitHub repository*. `https://github.com/FilippoMB/Benchmark_dataset_for_graph_classification`. Accessed: 2024-04-14. 2024.

[312] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. “Weighted Graph Cuts without Eigenvectors A Multilevel Approach”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.11 (2007), pp. 1944–1957.

[313] Filippo Maria Bianchi et al. “Hierarchical Representation Learning in Graph Neural Networks With Node Decimation Pooling”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.5 (2022), pp. 2195–2207.

[314] Zhitao Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[315] Hongyang Gao and Shuiwang Ji. "Graph U-Nets". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. 2019.

[316] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. "Spectral clustering with graph neural networks for graph pooling". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 874–883.

[317] Alessio Martino, Alessandro Giuliani, and Antonello Rizzi. "(Hyper)Graph Embedding and Classification via Simplicial Complexes". In: *Algorithms* 12.11 (2019).

[318] Alessio Martino and Antonello Rizzi. "(Hyper)graph Kernels over Simplicial Complexes". In: *Entropy* 22.10 (2020).

[319] Yuyang Wang et al. "Molecular contrastive learning of representations via graph neural networks". In: *Nature Machine Intelligence* 4.3 (2022).

[320] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2017.

[321] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2018.

[322] Kristof T. Schütt et al. "Quantum-chemical insights from deep tensor neural networks". In: *Nature communications* 8.1 (2017).

[323] Yiqun Wang et al. "Mgcn: descriptor learning using multiscale gcns". In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 122–1.

[324] Kevin Yang et al. "Analyzing learned molecular representations for property prediction". In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3370–3388.

[325] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[326] Corinna Cortes. "Support-Vector Networks". In: *Machine Learning* (1995).

[327] Florian Graf et al. *The Flood Complex: Large-Scale Persistent Homology on Millions of Points*. 2025. arXiv: `2509.22432 [cs.LG]`. URL: `https://arxiv.org/abs/2509.22432`.

[328] Musashi Ayrton Koyama et al. *Faster computation of degree-1 persistent homology using the reduced Vietoris-Rips filtration*. 2024. arXiv: `2307.16333 [math.AT]`. URL: `https://arxiv.org/abs/2307.16333`.

[329] Simon Zhang, Mengbai Xiao, and Hao Wang. *GPU-Accelerated Computation of Vietoris-Rips Persistence Barcodes*. 2020. arXiv: `2003.07989 [cs.CG]`.

[330] Bernardo Ameneyro, Vasileios Maroulas, and George Siopsis. "Quantum persistent homology". In: *Journal of Applied and Computational Topology* 8.7 (Nov. 2024), pp. 1961–1980. URL: `https://doi.org/10.1007/s41468-023-00160-7`.

[331] Brittany Terese Fasy et al. “Confidence sets for persistence diagrams”. In: *The Annals of Statistics* 42.6 (2014), pp. 2301–2339. URL: https://doi.org/10.1214/14-AOS1252.

[332] Elizabeth Munch. “An Invitation to the Euler Characteristic Transform”. In: *The American Mathematical Monthly* 132.1 (2025), pp. 15–25. eprint: https://doi.org/10.1080/00029890.2024.2409616.

[333] Ernst Röell and Bastian Rieck. “Differentiable Euler Characteristic Transforms for Shape Classification”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=MO632iPq3I.

[334] Archit Rathore et al. “TopoAct: Visually Exploring the Shape of Activations in Deep Learning”. In: *Computer Graphics Forum* 40.1 (2021), pp. 382–397. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14195.

[335] Gunnar Carlsson. *Topological Data Analysis and Mechanistic Interpretability*. https://www.lesswrong.com/posts/6oF6pRr2FgjTmiHus/topological-data-analysis-and-mechanistic-interpretability. Accessed: 2025-03-21.

[336] Thomas Bonis et al. *Random walks on simplicial complexes*. 2024. arXiv: 2404.08803 [math.PR]. URL: https://arxiv.org/abs/2404.08803.

[337] Michael T. Schaub et al. “Random Walks on Simplicial Complexes and the Normalized Hodge 1-Laplacian”. In: *SIAM Review* 62.2 (2020), pp. 353–391. eprint: https://doi.org/10.1137/18M1201019. URL: https://doi.org/10.1137/18M1201019.

[338] Alhussein Fawzi et al. “Discovering faster matrix multiplication algorithms with reinforcement learning”. In: *Nature* 610.7930 (2022), pp. 47–53.

[339] Trieu H. Trinh et al. “Solving olympiad geometry without human demonstrations”. In: *Nature* 625.7995 (Jan. 2024), pp. 476–482. URL: https://doi.org/10.1038/s41586-023-06747-5.

[340] Colin Adams. *Artificial Intelligence in knot theory*. Jan. 2022. URL: https://amathr.org/artificial-intelligence-in-knot-theory/.

[341] Joseph B. Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical Society* 7.1 (1956), pp. 48–50.

[342] Jin-Yi Cai, Martin Fürer, and Neil Immerman. “An optimal lower bound on the number of variables for graph identification”. In: *Combinatorica* 12.4 (1992), pp. 389–410.

[343] V. Arvind et al. “On the Power of Color Refinement”. In: *Fundamentals of Computation Theory*. Ed. by Adrian Kosowski and Igor Walukiewicz. Cham, Switzerland: Springer, 2015, pp. 339–350.

[344] Béla Bollobás. “Distinguishing Vertices of Random Graphs”. In: *Graph Theory*. Ed. by Béla Bollobás. North–Holland Mathematics Studies 62. Amsterdam, Netherlands: North–Holland, 1982, pp. 33–49.

[345] Marco Cuturi. “Sinkhorn Distances: Lightspeed Computation of Optimal Transport”. In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. 2013.

[346] Asim Kumar Debnath et al. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity". In: *Journal of Medicinal Chemistry* 34.2 (1991), pp. 786–797.

[347] Pinar Yanardag and SVN Vishwanathan. "Deep graph kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 1365–1374.

[348] Karsten M. Borgwardt et al. "Protein function prediction via graph kernels". In: *Bioinformatics* 21.suppl 1 (June 2005), pp. i47–i56.

[349] Nikil Wale and George Karypis. "Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification". In: *Sixth International Conference on Data Mining (ICDM'06)*. 2006, pp. 678–689.

[350] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". In: *Neural Computation* 15.6 (2003), pp. 1373–1396.

[351] Frank H. Lutz. "Enumeration and Random Realization of Triangulated Surfaces". In: *Discrete Differential Geometry*. Ed. by Alexander I. Bobenko et al. Basel, Switzerland: Birkhäuser, 2008, pp. 235–253.

[352] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. URL: `https://github.com/Lightning-AI/lightning`.

[353] Sergio Barbarossa and Stefania Sardellitti. "Topological Signal Processing Over Simplicial Complexes". In: *IEEE Transactions on Signal Processing* 68 (2020).

[354] Daniel Hernández Serrano, Juan Hernández-Serrano, and Darío Sánchez Gómez. "Simplicial degree in complex networks. Applications of topological data analysis to network science". In: *Chaos Solitons Fractals* 137 (2020), pp. 109839, 21.

[355] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

[356] O. Rioul and M. Vetterli. "Wavelets and signal processing". In: *IEEE Signal Proc. Mag.* 8.4 (1991), pp. 14–38.

[357] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[358] Weihua Hu et al. "Open graph benchmark: Datasets for machine learning on graphs". In: *Advances in neural information processing systems* 33 (2020).

[359] Sebastian G Rohrer and Knut Baumann. "Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data". In: *Journal of chemical information and modeling* 49.2 (2009), pp. 169–184.