Topological Neural Networks go Persistent, Equivariant and Continuous

Yogesh Verma [1] Amauri H. Souza [1] Vikas Garg [1 2]

Abstract

Topological Neural Networks (TNNs) incorporate higher-order relational information beyond pairwise interactions, enabling richer representations than Graph Neural Networks (GNNs). Concurrently, topological descriptors based on persistent homology (PH) are being increasingly employed to augment the GNNs. We investigate the benefits of integrating these two paradigms. Specifically, we introduce *TopNets* as a broad framework that subsumes and unifies various methods in the intersection of GNNs/TNNs and PH such as (generalizations of) RePHINE and TOGL. TopNets can also be readily adapted to handle (symmetries in) geometric complexes, extending the scope of TNNs and PH to spatial settings. Theoretically, we show that PH descriptors can provably enhance the expressivity of simplicial message-passing networks. Empirically, (continuous and $E(n)$-equivariant extensions of) TopNets achieve strong performance across diverse tasks, including antibody design, and drug property prediction.

1. Introduction

Relational data in diverse settings such as social networks (Freeman, 2004), and proteins (Jha et al., 2022) can be effectively abstracted via graphs. GNNs have enabled considerable success in representing such data (Bronstein et al., 2021). However, their limitations such as inability to distinguish non-isomorphic graphs and compute graph properties (Xu et al., 2019; Weisfeiler and Leman, 1968; Garg et al., 2020) have spurred research efforts toward designing more powerful models that can leverage higher-order interactions, e.g., hierarchical *part-whole* relations.

Topological deep learning (TDL) (Papillon et al., 2023) views graphs as 1-dimensional simplicial complexes, and employs general abstractions to process data with higher-order relational structures. TNNs, a broad class of topological neural architectures, have yielded state-of-the-art performance on various machine learning tasks (Dong et al., 2020; Chen et al., 2019; Barbarossa and Sardellitti, 2020), showcasing high potential for numerous applications.

Simultaneously, descriptors based on PH (Horn et al., 2021; Carrière et al., 2020; Immonen et al., 2023), a workhorse from topological data analysis (TDA), capture important topological information such as the number of components and independent loops. Augmenting GNNs with persistent features affords powerful representations. However, the merits of integrating persistence in TNNs remain unexplored. In particular, numerous real-world tasks involving topological objects exhibit symmetries under the Euclidean group $\mathrm{E}(n)$, such as translations, rotations, and reflections. Examples range from predicting molecular properties (Ramakrishnan et al., 2014), 3D atomic systems (Duval et al., 2023), to generative design and beyond. While various approaches use these symmetries effectively, including Tensor Field Networks (Thomas et al., 2018), $\mathrm{SE}(3)$ Transformers (Fuchs et al., 2020), EGNNs (Satorras et al., 2021), and EMPSNs (Eijkelboom et al., 2023), their expressivity remains limited as they fail to capture certain topological structures (Joshi et al., 2023) in geometrical simplicial complexes.

We strive to bridge this gap with a general recipe to leverage the best of both worlds. Specifically, we propose TopNets (**To**pological **P**ersistent Neural **Net**works) as a comprehensive framework unifying TNNs and PH. Our approach allows us to seamlessly accommodate additional contextual cues; e.g., TopNets can process spatial information via geometric color filtrations. We analyze TopNets from both theoretical and practical perspectives, illuminating their promise across diverse tasks.

We reinforce the versatility of TopNets by designing their continuous counterparts, defining associated Neural ODEs over simplicial complexes and elucidating error bounds between the discrete and continuous systems. We thus build on the remarkable success of Neural ODEs (Chen et al., 2018; Kim et al., 2023; Marion, 2023) across various domains, including spatiotemporal forecasting (Yildiz et al., 2019; Li et al., 2021; Lu et al., 2021; Kochkov et al., 2021; Brandstetter et al., 2023; Verma et al., 2024), generative model-

[1]Department of Computer Science, Aalto University, Finland [2]YaiYai Ltd. Correspondence to: Yogesh Verma <yogesh.verma@aalto.fi>.

ICML 2024 AI for Science workshop. Copyright 2024 by the author(s).

Table 1: **Overview of recent methods for relational data and summary of our contributions**. E: Equivariant, P: Persistent, C: Continuous, and HO: higher order.

Recent methods for relational data

Method	E	P	C	HO
TOGL (Horn et al., 2021)	✗	✓	✗	✗
PersLay (Carrière et al., 2020)	✗	✓	✗	✗
RePHINE (Immonen et al., 2023)	✗	✓	✗	✗
MPSN (Bodnar et al., 2021b)	✗	✗	✗	✓
CWN (Bodnar et al., 2021a)	✗	✗	✗	✓
CAN (Giusti et al., 2023)	✗	✗	✗	✓
IMPSN (Eijkelboom et al., 2023)	✓	✗	✗	✓
EGNN (Satorras et al., 2021)	✓	✗	✗	✗
E3NN (Geiger and Smidt, 2022)	✓	✗	✗	✗
GATr (Brehmer et al., 2023)	✓	✗	✗	✗
GRAND (Chamberlain et al., 2021)	✗	✗	✓	✗
GREAD (Choi et al., 2022)	✗	✗	✓	✗
GRAND++ (Thorpe et al., 2022)	✗	✗	✓	✗
TopNets (ours)	✓	✓	✓	✓

Main contributions of this work

Section 3	
Unified Framework: TopNets	
TNNs + PH $\succ$ TNNs	Prop. 1
Section 4	
$E(n)$-Equivariant TopNets (E-TopNets)	
Invariant PH embedding	Prop. 2
Section 5	
Continuous (Equivariant) TopNets	
Discretization error (TOGL)	Prop. 3
Discretization error (RePHINE)	Prop. 4
Section 6	
Experiments: graph classification, drug property prediction, and generative design	

ing (Grathwohl et al., 2018; Lipman et al., 2023; Verma et al., 2022; 2023), and graph representation learning (Poli et al., 2019; Iakovlev et al., 2020; Chamberlain et al., 2021; Thorpe et al., 2022; Choi et al., 2022).

We summarize our main contributions below:

1. (**Methodology**) we propose TopNets, a general unifying framework that combines TNN with PH and leverages persistent homology to boost the expressivity of (equivariant) message-passing simplicial networks;
2. (**Theory**) we derive a set of associated Neural-ODEs for various TNNs and PH over simplicial complexes and compute the associated discretization error bound between discrete and continuous systems;
3. (**Empirical**) TopNets achieve strong performance across diverse real-world tasks such as graph classification, drug property prediction, and generative design.[1]

We compare TopNets with several other recent methods for modeling relational data in Table 1.

2. Background

We begin with notions from topological ML, persistent homology, equivariance, and Graph ODEs that we use.

Simplicial complexes. An *abstract simplicial complex* (ASC) over a vertex set V is a set K of subsets of V (called *simplices*) such that, for every $\sigma \in K$ and every non-empty $\tau \subset \sigma$, we have that $\tau \in K$. Let σ be a simplex, then its non-empty subsets $\tau \subset \sigma$ are called *faces*, and σ is a *coface* of τ. The dimension of a simplex is equal to its cardinality minus 1, and the dimension of a simplicial complex is the maximal dimension of its simplices. We denote by $K_{[i]}$ the subset of i-dim simplices of K. Here, we represent simplices using square brackets. For instance, $K = \{[0], [1], [0, 1]\}$ denotes a 1-dim simplicial complex over $V = \{0, 1\}$, and the 0-dim simplices $[0]$ and $[1]$ are the faces of the simplex $[0, 1]$.

We also consider simplicial complexes with features. In particular, a *geometric simplicial complex* is a tuple (K, x, z) where $x : K \to \mathbb{R}^{d_x}$ and $z : K_{[0]} \to \mathbb{R}^{d_z}$ are functions that assign to a simplex σ an attribute (or color) $x(\sigma)$ and a geometric feature $z(\sigma)$, respectively. For convenience, hereafter, we denote the feature vectors of σ by x_σ and z_σ.

Graph neural networks (GNNs). Let $G = (V, E)$ be an undirected graph with vertex set V and edge set $E \subseteq V \times V$ — note that graphs are 1-dim ASCs. To obtain meaningful graph representations, message-passing GNNs (Gilmer et al., 2017; Xu et al., 2019; Velicković et al., 2017) employ a sequence of message-passing steps, where each node v aggregates messages from its neighbors $\mathcal{N}(v) = \{u : (v, u) \in E\}$ and use the resulting vector to update its own embedding. In particular, starting from $x_v^0 = x_v \; \forall v \in V$, GNNs recursively apply the update rule

$$x_v^{\ell+1} = \mathrm{Upd}_\ell \left(x_v^\ell, \mathrm{Agg}_\ell(\{\!\{ x_u^\ell : u \in \mathcal{N}(v) \}\!\}) \right),$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset, Agg_ℓ is an order-invariant function and Upd_ℓ is an arbitrary update function.

[1] Code is available here: https://github.com/Aalto-QuML/TopNets

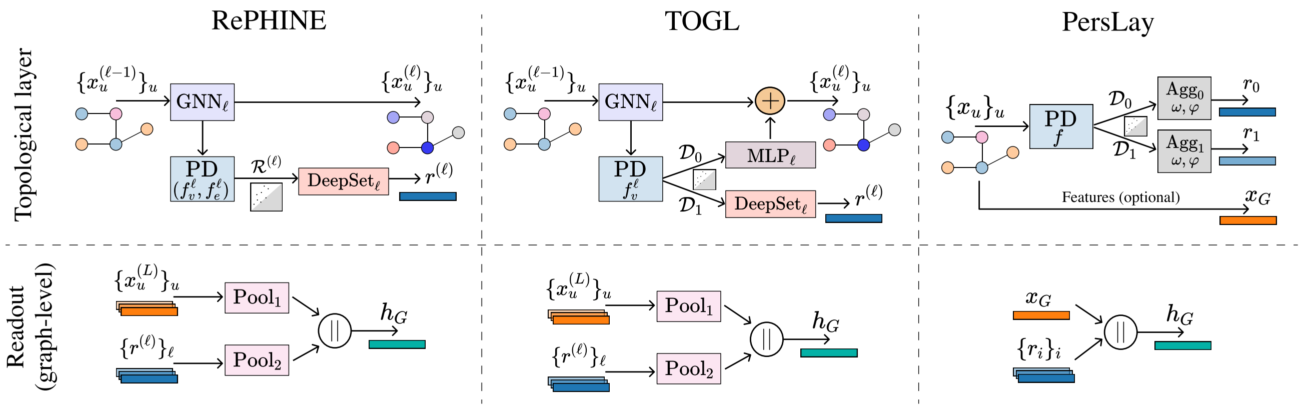

Figure 1: **Comparison of representative PH-based architectures for graph learning.**

Topological neural networks (TNNs, e.g., Bodnar et al., 2021a; Hensel et al., 2021; Hofer et al., 2017) consist of neural models for processing data with high-order relational structure. Papillon et al. (2023) provide a unified framework to describe message-passing TNNs — here we focus on models for simplicial complexes. After specifying *neighborhood structures*, which define how simplices (possibly of different dimensions) can locally interact, TNNs recursively update the simplices' embeddings via message passing. This general message-passing procedure comprises: i) message computation, ii) within-neighborhood aggregation, iii) between-neighborhood aggregation, and iv) update. More specifically, let $\mathcal{N}$ define a neighborhood structure. For each simplex $\sigma \in K^\ell$ at layer ℓ, we compute the messages $m^{\ell,\mathcal{N}}_{\sigma' \to \sigma} = \mathrm{Msg}_{\ell,\mathcal{N}}(x^\ell_\sigma, x^\ell_{\sigma'})$ from all $\sigma' \in \mathcal{N}(\sigma)$, where $\mathrm{Msg}_{\ell,\mathcal{N}}$ is an arbitrary function. Then, the messages to simplex σ are aggregated, that is,

$$m^{\ell,\mathcal{N}}_\sigma = \mathrm{WithinAgg}_\ell(\{m^{\ell,\mathcal{N}}_{\sigma' \to \sigma} : \sigma' \in \mathcal{N}(\sigma)\}), \quad (1)$$
$$m^\ell_\sigma = \mathrm{BetweenAgg}_\ell(\{m^{\ell,\mathcal{N}}_\sigma : \mathcal{N} \in \mathcal{N}_{\text{all}}\}), \quad (2)$$

where $\mathcal{N}_{\text{all}}$ is a set of neighborhoods comprising, e.g., coboundary, boundary, lower-, and upper- adjacencies (Bodnar et al., 2021b). Finally, we apply a function Update_ℓ to obtain the refined feature vector at layer $\ell + 1$ as

$$x^{\ell+1}_\sigma = \mathrm{Update}_\ell(m^\ell_\sigma, x^\ell_\sigma). \quad (3)$$

Notably, TNNs subsume a large class of models, including message-passing GNNs.

Persistent homology. A *filtration* of a simplicial complex K is a finite nested sequence of subcomplexes of K, i.e., $\emptyset = K_0 \subset K_1 \subset ... \subset K$. To obtain a valid filtration, it suffices to ensure that all the faces of a simplex σ do not appear later than σ in the filtration. To achieve that, a typical choice consists of defining a filtering (or filtration) function f on the vertices of the simplicial complex, and use it to rank each simplex $\sigma \in K$ as $o_f(\sigma) = \max_{v \in \sigma} f(v)$. Let $\alpha_1 < \cdots < \alpha_n$ be an increasing sequence of vertex filtered values, i.e., $\alpha_i \in \{f(v) : v \in K_{[0]}\}$; then, we index the filtration steps using real numbers and define the filtration of K induced by f as $K_{\alpha_i} = \{\sigma \in K : o_f(\sigma) \leq \alpha_i\}$ for $i = 1, \ldots, n$. Another common strategy adopts filtering functions on vertex features x_v and redefine $o_f(\sigma; x) = \max_{v \in \sigma} f(x_v)$. Filtrations induced by functions on vertex features (or colors) are called *vertex-color filtrations*.

The idea of persistent homology (PH) is to keep track of the appearance and disappearance of topological features (e.g., connected components, loops, voids) in a filtration. If a topological feature first appears in K_{α_i} and disappears in K_{α_j}, then we encode its persistence as a pair (α_i, α_j); if a feature does not disappear, then its persistence is (α_i, ∞). The collection of all pairs forms a multiset that we call *persistence diagram*. We use $\mathcal{D}_i$ to denote the persistence diagram for i-dim topological features. We provide more details in Appendix A.

Persistence diagrams are usually vectorized before being combined with ML models. In this regard, Carrière et al. (2020) proposed a general framework, called PersLay, that computes a vector representation for a given diagram $\mathcal{D}$ as

$$\mathrm{Agg}\left(\{\omega(p)\varphi(p) : p \in \mathcal{D}\}\right),$$

where Agg is a permutation invariant operation (e.g., mean, maximum, sum), $\omega : \mathbb{R}^2 \mapsto \mathbb{R}$ is an arbitrary function that assigns a weight to each persistence pair, and $\varphi : \mathbb{R}^2 \mapsto \mathbb{R}^q$ maps each pair to a higher dimensional space. Notably, PersLay introduces choices for φ that generalize many vectorization methods in the literature (e.g., Zaheer et al., 2017; Bubenik, 2015; Adams et al., 2016; Kusano et al., 2016).

Combining PH and GNNs. Recently, PH has been used to boost the expressive power of GNNs. Horn et al. (2021) introduce TOGL — a general approach for incorporating topological features from PH into GNN layers. In particular, TOGL leverages node embeddings at each layer of a GNN to obtain vertex-color filtrations. The 0-dim individual persistence tuples are vectorized using MLPs and added to the corresponding node features at each layer. For 1-dim tuples, TOGL applies DeepSets to get a graph-level vector that runs through the final fully-connected layers of the GNN.

Immonen et al. (2023) use independent vertex-color and edge-color filtering functions to obtain more expressive persistent diagrams called RePHINE. More specifically, RePHINE first computes persistence diagrams from a filtration induced by edge colors. Each tuple of the diagram is then augmented based on the vertex colors and the local edge-color information around each vertex. RePHINE diagrams are vectorized using DeepSets and combined with graph-level GNN embeddings in the final classifier. Figure 1 depicts the architectures of RePHINE, TOGL, and PersLay.

$\mathbf{E(n)}$-Equivariant networks. Let $\mathfrak{G}$ be a group acting on two sets $\mathcal{X}$ and $\mathcal{Y}$. We say a function $f : \mathcal{X} \to \mathcal{Y}$ is $\mathfrak{G}$-equivariant if it commutes with the group actions, i.e., for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}$, we have that $f(\mathfrak{g} \cdot x) = \mathfrak{g} \cdot f(x)$. Here, we are interested in models on geometric simplicial complexes that are equivariant to the Euclidean group $E(n)$, which comprises all translations, rotations, and reflections of the n-dim Euclidean space. Eijkelboom et al. (2023) introduce Equivariant Message-Passing Simplicial Networks (EMPSNs), which extends the $E(n)$-equivariant GNNs (Satorras et al., 2021) to geometric simplicial complexes. For each simplex $\sigma \in K^\ell$ at layer ℓ, EMPSNs compute the messages $m^{\ell,\mathcal{N}}_{\sigma' \to \sigma} = \mathrm{Msg}_{\ell,\mathcal{N}}(x^\ell_\sigma, x^\ell_{\sigma'}, \mathrm{Inv}(\sigma, \sigma'; z^\ell))$ from all $\sigma' \in \mathcal{N}(\sigma)$, where $\mathrm{Inv}(\sigma, \sigma'; z^\ell)$ denotes invariant features (e.g., volumes, angles, distances) computed using coordinates from z^ℓ. Then, the messages to simplex σ are aggregated using $\mathrm{WithinAgg}_\ell$ and $\mathrm{BetweenAgg}_\ell$ the same way as in TNNs to obtain an aggregated message m^ℓ_σ. Finally, we recursively update the features and coordinates as

$$x^{\ell+1}_\sigma = \mathrm{Update}_\ell(m^\ell_\sigma, x^\ell_\sigma) \tag{4}$$

$$z^{\ell+1}_\sigma = C \sum_{\sigma' \in \mathcal{N}_\uparrow(\sigma)} (z^\ell_\sigma - z^\ell_{\sigma'}) \phi^\ell_z(m^{\ell,\mathcal{N}_\uparrow}_{\sigma' \to \sigma}) \;\; \forall \sigma \in K_{[0]} \tag{5}$$

where $\mathcal{N}_\uparrow$ denotes the upper-adjancency, C is a normalization constant, and ϕ^ℓ_z is an arbitrary function.

Graph ODEs. Neural Ordinary Differential Equations (ODEs) represent a class of implicit deep learning models characterized by an ODE, where the vector field is parameterized by a neural network (Weinan, 2017; Dupont et al., 2019; Chen et al., 2018; Lu et al., 2018). Graph ODEs (Poli et al., 2019) generalize Neural ODEs to garphs. For instance, we can track the evolution of signals defined over the vertices of a graph as a differential equation

$$\dot{x}_v = \frac{dx_v}{dt} = f(t, x_v, \{x_u\}_{u \in \mathcal{N}(v)}). \tag{6}$$

Here, the vector field f is parameterized by a neural network. A notable feature is that, under a mild assumption on f, employing an Euler scheme for N time-steps converges to an N-layer Graph ResNet (Sander et al., 2022). This convergence implies that Graph ODEs inherently inherit the capability to incorporate relational inductive biases seen in GNNs while maintaining the dynamic system perspective of continuous-depth models. The versatility of Graph ODEs has paved the way for the design of novel graph neural networks, such as GRAND (Chamberlain et al., 2021), GREAD (Choi et al., 2022), and AbODE (Verma et al., 2023).

3. A unified framework: Topological persistent neural networks (TopNets)

We now introduce a general framework that combines TNNs and PH for expressive learning on topological objects. We call this framework *topological persistent neural networks* or TopNets, in short. Notably, we show that TopNets subsume several methods at the intersection of PH and GNNs.

To motivate our framework, we show that persistent homology features bring in additional expressive power to TNNs. Bodnar et al. (2021b) introduce a Simplicial Weisfeiler-Leman (SWL) test to characterize the expressivity of simplicial message-passing networks (SMPNs) — a general TNN for simplicial complexes. They show that SWL (with clique complex lifting) is strictly more powerful than 1-WL. Our next result (Proposition 1) implies that the combination of SWL and PH is strictly more expressive than the SWL test.

Proposition 1 (SWL + PH $\succ$ SWL)**.** *There are pairs of non-isomorphic clique complexes that SWL cannot distinguish but persistence diagrams from color-based filtrations can.*

Prior works (Horn et al., 2021; Rieck, 2023; Immonen et al., 2023) have demonstrated that PH can be used to increase the power of GNNs. Proposition 1 shows that this also applies to TNNs on simplicial complexes.

Given an input simplicial complex, each layer in a TopNet first applies a general message-passing (MP) procedure to obtain a refined attributed complex, as in TNNs. Then, we compute persistence diagrams followed by a vectorization scheme that assigns each simplex a topological embedding. Next, TopNets obtain two complex-level representations: the first consists of a joint MP-PH vector derived from a combination of the features of the complex and the topological embeddings; and the second one is obtained by merging the PH-based descriptor associated with each simplex via an order-invariant function. Finally, we combine the representation of the simplices at each layer and dimension, and apply two readout layers. The first aims to combine information from different layers (but same dimension) while the second readout function further processes the resulting representations across dimensions. In the following, we formalize these steps.

Steps of a TopNet layer

1. General Message Passing (MP): Let (K^ℓ, x^ℓ) denote an attributed simplicial complex at layer ℓ. TopNets refine the attributed complex using a general TNN layer as

$$K^\ell, \tilde{x}^\ell = \text{TNNLayer}_\ell(K^{\ell-1}, x^{\ell-1}). \quad (7)$$

2. PH Vectorization: Next, we compute a persistence diagram induced by a filtering function f^ℓ followed by a vectorization procedure ψ. As a result, we obtain a topological vector representation r^ℓ_σ for each simplex σ in K^ℓ:

$$r^\ell_\sigma = \psi(\text{PD}(\sigma; f^\ell, \tilde{x}^\ell, K^\ell)) \quad \forall \sigma \in K^\ell. \quad (8)$$

We note that the map PD computes persistence diagrams for all dimensions $i = 0, 1, \ldots, \dim(K^\ell)$.

3. Topological aggregation: We combine the PH and MP embeddings of each simplex $\sigma \in K^\ell$ by applying a so-called topological aggregation function $\text{TopAgg}_{\dim(\sigma)}$ — note that the choice of topological aggregation depends on the dimension of the input simplex. We also group the topological vectors using a dimension-wise $\text{Agg}_{i,\ell}$ operation, i.e.,

$$x^\ell_\sigma = \text{TopAgg}_{\dim(\sigma)}(\tilde{x}^\ell_\sigma, r^\ell_\sigma) \quad \forall \sigma \in K^\ell \quad (9)$$

$$m^{\ell,i} = \text{Agg}_{i,\ell}(\{r^\ell_\sigma\}_{\sigma \in K^\ell_{[i]}}) \quad (10)$$

4. Readout: We then merge the features x^ℓ_σ and the topological embeddings $m^{\ell,i}$ across layers and, subsequently, across dimensions using interleaved readout functions:

$$h^{\ell,i} = \text{Pool}(\{x^\ell_\sigma\}_{\sigma \in K^\ell_{[i]}}) \quad (11)$$

$$h^i = \text{Readout}_{\text{layer}}(\{h^{\ell,i}\}_\ell, \{m^{\ell,i}\}_\ell) \quad (12)$$

$$h = \text{Readout}_{\text{dim}}(\{h^i\}_i). \quad (13)$$

The final representation h in Equation 13 is typically fed through multi-layer perceptrons (MLP) to obtain a complex-level prediction. Importantly, the formalism of TopNets includes PH-based (graph) neural networks such as TOGL (Horn et al., 2021), PersLay (Carrière et al., 2020), and RePHINE (Immonen et al., 2023) as particular cases:

a) TOGL: Here, the TNNLayer_ℓ functions correspond to GNN layers, while the computation of persistence diagrams (PD) involves vertex-color filtrations, with vectorization achieved via MLPs ψ. The topological aggregation $\text{TopAgg}^{\text{TOGL}}$ (defined in Appendix C) is specifically applied to persistence tuples of dimension $i = 0$, whose vector representations are added to the initial node features. Tuples of dimension $i = 1$ are pooled and then concatenated with the final GNN embedding for use in the subsequent readout phase.

b) PersLay: The TNNLayer_ℓ serves as an identity transformation, and the computation of the persistence diagram (PD) involves (0-dim and 1-dim) ordinary and extended persistence pairs. Moreover, $\text{TopAgg}^{\text{PersLay}}$ (defined in Appendix C) simply concatenates node features with graph-level topological vectors.

c) RePHINE: Again, GNN is the choice of TNN. However, the computation of persistence diagrams (PD) involves vertex and edge filtrations specific to *RePHINE*. The results are aggregated using a DeepSet function $\text{Agg}_{i,\ell}$ to yield a topological embedding $m^{\ell,i}$ for each layer ℓ and dimension i. The topological aggregation function $\text{TopAgg}^{\text{RePHINE}}$ (defined in Appendix C)outputs the simplex features $\tilde{x}^\ell_\sigma$. Finally, in conjunction with the simplex features from the final layer, the topological embeddings are concatenated and pooled for subsequent use in the downstream readout phase.

More details about deductions can be found in Appendix C.

4. E(n) Equivariant TopNets

In this section, we extend TopNets to deal with topological objects that are symmetric to rotation, reflections and translations — i.e., to actions of the Euclidean group $E(n)$. In particular, we consider geometric SCs, and build upon EMPSNs (Eijkelboom et al., 2023) and invariant filtering functions to propose Equivariant TopNets (E-TopNets). Compared to regular TopNets, E-TopNets employ modified general message passing and PH vectorization steps (Eqs. 7 and 8) — the other steps remain untouched.

Starting from an input geometric (attributed) SC (K^0, x^0, z^0); at each layer ℓ, E-TopNets recursively obtain a refined SC via an EMPSN layer as

$$K^\ell, \tilde{x}^\ell, z^\ell = \text{EMPSNLayer}_\ell(K^{\ell-1}, x^{\ell-1}, z^{\ell-1}).$$

To achieve an equivariant variant of TopNets, one could disregard the vertex coordinates z^ℓ when computing persistence diagrams. For instance, this can be obtained from an *i-simplex-color filtration* (Definition 1). This generalizes the notion of vertex-color filtrations to higher dimensions. Thus, 0-simplex-color filtrations are vertex-color ones, 1-simplex-color filtrations correspond to edge-color filtrations, and so on.

Definition 1 (i-simplex-color filtrations)**.** *Let* (K, x) *be an* *attributed simplicial complex and $f : \mathbb{R}^{d_x} \to \mathbb{R}^+$ a filtering function. Also, let $\alpha_1 < \cdots < \alpha_n$ with $\alpha_j \in \{f(x_w) : w \in K_{[i]}\}$. An i-simplex-color filtration induced by f is a sequence of complexes $K_{\alpha_j} = \{\sigma \in K : o_f(\sigma; x) \leq \alpha_j\}$ for $j = 1, \ldots, n$, where*

$$o_f(\sigma; x) = \begin{cases} \max\limits_{\tau \subset \sigma : \dim(\tau) = i} f(x_\tau) & \text{, if } \dim(\sigma) \geq i \\ 0 & \text{, otherwise.} \end{cases}$$

Obtaining persistence diagrams from i-simplex-color filtrations incurs losing (possibly) relevant geometric information. Thus, here, we are interested in filtering functions that leverage both attributes and coordinates, as in geometric color-based filtrations (Definition 2).

Definition 2 (Geometric i-simplex-color filtrations)**.** *Let (K, x, z) be a geometric simplicial complex and f a filtering function. Also, let $\alpha_1 < \cdots < \alpha_n$ with $\alpha_j \in \{f(x_w, \cdot) : w \in K_{[i]}\}$. A geometric i-simplex-color filtration induced by f is a sequence $K_{\alpha_j} = \{\sigma \in K : o_f(\sigma; x, z) \leq \alpha_j\}$ for $j = 1, \ldots, n$, where*

$$o_f(\sigma; x, z) = \begin{cases} \max\limits_{\substack{\tau \subset \sigma : \\ \dim(\tau) = i}} f(x_\tau, \mathrm{Inv}(\{z_v\}_{v \in \tau})) & , \dim(\sigma) \geq i \\ 0 & \text{, otherwise} \end{cases}$$

and $\mathrm{Inv}(\cdot)$ is any $E(n)$- and S_n-invariant function.

For many tasks, e.g., in graph learning, colors are only given to 0-dim simplices. In such cases, we can obtain colors to higher-order simplices σ via a learnable permutation invariant function on the colors of the vertices in σ. Thus, we can rewrite the filtering functions in Definition 2 as $f(\phi(\{x_v\}_{v \in \tau}), \mathrm{Inv}(\{z_v\}_{v \in \tau}))$. As usual, we parameterize f using multilayer perceptrons and ϕ using DeepSets.

As a remark, persistence diagrams extracted from geometric 0-simplex-color filtrations are not more expressive than their non-geometric counterparts — i.e., vertex-color (VC) filtrations. The reason is that the only $E(n)$-invariant function of a single element is a constant function, i.e., the condition $f(z) = f(\mathfrak{g} \cdot z)$ for all $\mathfrak{g} \in E(n)$ implies that f is a constant function. Thus, we refer to their non-geometric variant whenever we mention VC filtrations.

We also note that, to achieve a geometric extension of RePHINE diagrams, we can simply replace its edge-color filtration with a geometric 1-simplex-color filtration and then use an independent vertex-color function as in the original formulation. This highlights that the vertex coordinates are only used to define filtrations, and any persistence descriptor and vectorization procedure can be applied — having no impact on the equivariance of E-TopNets. Our next result (Proposition 2) establishes the invariance of persistence diagrams from geometric i-simplex-color filtrations.

Proposition 2 (Invariant persistence diagrams)**.** *For any $i \geq 0$, persistence diagrams for any dimension obtained from geometric i-simplex-color filtrations are $E(n)$-invariant.*

We can rewrite the PH vectorization step of E-TopNets as

$$r_\sigma^\ell = \psi(\mathrm{PD}(\sigma; f_{\text{inv}}^\ell, \tilde{x}^\ell, z^\ell, K^\ell)) \quad \forall \sigma \in K^\ell$$

where f_{inv}^ℓ denotes one or more $E(n)$-invariant filtering functions used to induce a geometric i-simplex-color filtration for some i in $\{0, 1, \ldots, \dim(K^\ell)\}$.

5. Continuous (Equivariant) TopNets

In this section, we expand the general framework of (Equivariant) TopNets to encompass continuous systems. Unlike conventional E-TopNets, Continuous E-TopNets use a continuous message-passing scheme based on EMPSNs. For each simplex $\sigma \in K^t$ at time-step t, we compute the messages $m_{\sigma' \to \sigma}^{t, \mathcal{N}} = \mathrm{Msg}_{t, \mathcal{N}}(x_\sigma^t, x_{\sigma'}^t, \mathrm{Inv}(\sigma, \sigma'; z^t))$ from all $\sigma' \in \mathcal{N}(\sigma)$. Then, the messages to simplex σ are aggregated using $\mathrm{WithinAgg}_t$ and $\mathrm{BetweenAgg}_t$ the same way as in TNNs to obtain an aggregated message m_σ^t. Finally, we apply the following functions to obtain the refined feature vectors as

$$\dot{x}_\sigma = \mathrm{Update}(m_\sigma^t, x_\sigma^t) \tag{14}$$

$$\dot{z}_\sigma = C \sum_{\sigma' \in \mathcal{N}_\uparrow(\sigma)} (z_\sigma^t - z_{\sigma'}^t) \phi_z(m_{\sigma' \to \sigma}^{t, \mathcal{N}_\uparrow}) \; \forall \; \sigma \; \in \; K_{[0]} \tag{15}$$

where C is a constant and ϕ_z is an arbitrary non-linear mapping. The forward solution of x and z can be accurately approximated with numerical solvers such as RK4 (Runge, 1895) with low computational cost. The geometrical filtrations and topological embeddings are computed in the same way as described in the previous section.

Interestingly, one can define a set of associated Neural ODEs for a given PH-based (graph) neural network such as TOGL and RePHINE. We derive the set of neural ODEs and utilize it to derive discretization error bounds between discrete and continuous trajectories.

5.1. Discretization Error Bound

We compute discretization error bounds between the trajectories for discrete and continuous versions of RePHINE and TOGL. All the proofs can be found in Appendix E.

Proposition 3 (Discretization error for TOGL)**.** *The discretization error $e_v(\ell) = x_v^{\ell/N} - x_v^\ell$ for node v at layer ℓ between the node features of N-layer (with time-step size h) continuous and discrete TOGL networks is bounded as*

$$\|e_v(\ell)\|_1 \leq R_1(h) \frac{N(\exp(L_m + L_\beta) - 1)}{L_m + L_\beta} \tag{16}$$

Table 2: **Predictive performance on graph classification.**

TNN	Topological Agg	Diagram	Method	NCI109 ↑	IMDB-B ↑	NCI1 ↑	MOLHIV ↑	PROTEINS ↑
GCN	TopAgg$^{\text{RePHINE}}$	VC	Discrete	77.92 ±1.03	64.80 ±1.30	79.08 ±1.06	73.64 ±1.29	69.46 ±1.83
			Continuous	80.37 ±2.21	73.40 ±3.40	81.75 ±2.93	72.41 ±3.29	72.89±2.10
		RePHINE	Discrete	79.18 ±1.97	69.40 ±3.78	80.44 ±0.94	75.98 ±1.80	71.25 ±1.60
			Continuous	80.63 ±1.56	76.00 ±2.10	82.15 ±1.75	74.90 ±2.78	73.79 ±1.30
GIN	TopAgg$^{\text{RePHINE}}$	VC	Discrete	78.35 ±0.68	69.80 ±0.84	79.12 ±1.23	73.37 ±4.36	69.46 ±2.48
			Continuous	80.39±1.13	74.00 ±3.25	82.18 ±1.56	71.90 ±5.20	72.89 ±2.15
		RePHINE	Discrete	79.23 ±1.67	72.80 ±2.95	80.92 ±1.92	73.71 ±0.91	72.32 ±1.89
			Continuous	81.60 ±0.95	76.00 ±1.60	84.16 ±1.89	72.10 ±4.27	73.79 ±1.45
MPSN	TopAgg$^{\text{RePHINE}}$	VC	Discrete	79.40 ±2.74	66.50 ±3.65	77.10 ±1.37	72.40 ±3.90	70.50 ±1.75
			Continuous	80.10 ±3.45	73.00 ±1.80	81.10 ±4.64	72.70 ±4.65	71.20 ±3.20
		RePHINE	Discrete	79.43 ±1.65	67.20 ±2.85	81.22 ±1.48	71.20 ±4.78	71.70 ±2.56
			Continuous	80.40 ±3.55	74.00 ±2.65	83.20 ±3.24	71.50 ±4.54	72.10 ±2.35

where L_m and L_β are Lipshitz constants, and R_1 is a remainder term associated with the Taylor expansion of continuous TOGL.

Proposition 4 (Discretization error for RePHINE). *Let $x_v^{\ell/N}$ and $r^{\ell/N}$ be the node and topological embeddings of an N-time-step continuous RePHINE model at time-step ℓ, respectively. Similarly, let x_v^ℓ and r^ℓ be the node and topological embeddings of a discrete N-layer RePHINE at layer ℓ. Then, we can bound the discretization errors $e_v(\ell) = x_v^{\ell/N} - x_v^\ell$ and $e_r(\ell) = r^{\ell/N} - r^\ell$ as follows:*

$$\|e_v(\ell)\|_1 \leq R_1(h)\frac{N(\exp(L_m)-1)}{L_m} \tag{17}$$

$$\|e_r(\ell)\|_1 \leq L_\beta^\ell \|e_v(\ell-1)\|_1 + \frac{L_\beta^\ell L_m}{N}\|e_v(\ell-1)\|_1 + R_1(h) - R_1(m^{\ell-1}) \tag{18}$$

where L_m, L_β^ℓ are Lipshitz constants, and R_1 are the remainder terms associated with the Taylor expansion of continuous RePHINE.

Table 3: **Comparison with TOGL**. We used TopAgg$^{\text{TOGL}}$ for aggregating the PH embeddings.

Model	Diagram	Enzymes ↑	DD ↑	Proteins ↑
GCN	-	65.8 ±4.6	72.8 ±4.1	76.1 ±2.4
TOGL	VC	53.0 ±9.2	73.2 ±4.7	76.0 ±3.9
Cont. TopNets		69.7 ±3.2	73.1 ±1.9	78.7 ±2.7
GIN	-	50.0 ±12.3	70.8 ±3.8	72.3 ±3.3
TOGL	VC	43.8 ±7.9	75.2 ±4.2	73.6 ±4.8
Cont. TopNets		58.3 ±8.2	77.3 ±4.5	79.5 ±3.9

Implication. The bound indicates that the proximity to the ODE solution cannot be assured since it is uncertain whether $R_1(h)N \to 0$. This suggests the necessity of incorporating additional regulatory assumptions over the network to obtain the Neural ODE in the large depth limit. This observation resonates closely with the analysis conducted by Sander et al. (2022) in characterizing Neural ODEs with ResNets.

6. Experiments

Tasks We assess the performance of TopNets on diverse tasks: (i) we evaluate our method performance on real-world graph classification data between discrete and continuous counterparts across various GNNs and TNNs in section 6.1, (ii) we benchmark its efficacy in property prediction using QM9 molecular data, highlighting the effectiveness of its equivariant method in Section 6.2, and (iii) we demonstrate TopNets utility by co-designing antibody sequence and structure using the SAbDab database in Section 6.3.

Baselines On graph classification tasks, we evaluate TopNets using standard vertex-color (VC) and RePHINE (Immonen et al., 2023) persistence diagrams. We adopt different GNN/TNN architectures like GCN (Kipf and Welling, 2016), GIN (Xu et al., 2019), TOGL (Horn et al., 2021), and MPSN (Bodnar et al., 2021a) and process the persistence diagrams exactly the same way using DeepSets. We also compare the performance between each method's continuous and discrete counterparts. On QM9 property prediction tasks we compare to several equivariant methods like NMP (Gilmer et al., 2017), TFN (Thomas et al., 2018), SE(3)-Tr (Fuchs et al., 2020), DimeNet++(Gasteiger et al., 2020a), SphereNet (Liu et al., 2021), MPSN(Bodnar et al., 2021a), EGNN (Satorras et al., 2021) and IMPSN (Eijkelboom et al., 2023). Lastly, on CDR-H3 Antibody design, we compare to recent SOTA like RefineGNN (Jin et al., 2022), MEAN (Kong et al., 2023) and AbODE (Verma et al., 2023).

Implementation TopNets is implemented in PyTorch (Paszke et al., 2019). Details regarding hyperparameters training are in Appendix B.

6.1. Graph Classification

The results presented in Table 2 and Table 3 demonstrate the performance of TopNets on graph classification. These results offer a detailed assessment of different GNN/TNN architectures, PH vectorization methods, and their contin-

Table 4: **Test Mean absolute error (MAE) on QM9 dataset**. The △ denotes the methods trained with different train-test splits, and ** denotes the reproduced results. Benchmarks are from Eijkelboom et al. (2023). We denote the best-performing methods in **bold** and the second-best ones in blue. We used $\text{TopAgg}_{\text{RePHINE}}$ for aggregating the PH embeddings.

Architecture	**Diagram**	**Method**	α bohr^3	$\Delta\epsilon$ meV	ϵ_{HOMO} meV	ϵ_{LUMO} meV	μ D	C_v cal/mol K	R^2 bohr^3	ZPVE meV
DimeNet++△	-	-	0.044	33	25	20	0.030	0.023	0.331	1.21
SphereNet△	-	-	0.046	32	23	18	0.026	0.021	0.292	1.21
NMP	-	-	0.092	69	43	38	0.030	0.040	0.180	1.50
SE(3)-Tr	-	-	0.142	53	35	33	0.051	0.054	-	-
TFN	-	-	0.223	58	40	38	0.064	0.101	-	-
MPSN	-	-	0.266	153	89	77	0.101	0.122	0.887	3.02
EGNN	-	-	0.071	48	**29**	25	**0.028**	0.031	**0.106**	1.55
IMPSN**	-	-	**0.066**	51	32	25	0.031	**0.027**	0.114	1.44
IMPSN	VC	Disc. E-TopNets	0.083	**47**	37	**24**	0.035	0.032	0.125	1.45
		Cont. E-TopNets	0.075	49	36	27	0.030	0.035	0.129	1.43
	RePHINE	Disc. E-TopNets	0.072	57	33	28	0.029	0.028	0.132	1.39
		Cont. E-TopNets	0.070	50	35	25	0.032	0.030	0.118	**1.37**

uous counterparts. The reported results include the mean and standard deviation of predictive metrics — AUROC for MOLHIV and accuracy for the remaining datasets. This comprehensive analysis provides valuable insights into TopNets performance. Notably, incorporating the continuous component consistently improves downstream performance across all datasets, TNNs, and TopAgg schemes.

6.2. Molecular data - QM9

The QM9 dataset, introduced by Ramakrishnan et al. (2014), comprises small molecules with a maximum of 29 atoms in 3D space. Each atom is characterized by a 3D position and a five-dimensional one-hot node embedding representing the atom type, denoted as $(\mathrm{H}, \mathrm{C}, \mathrm{N}, \mathrm{O}, \mathrm{F})$. The dataset's primary objective is to predict various chemical properties of the molecules, which remain invariant to translations, rotations, and reflections on the atom positions. Following the data preparation strategy of Eijkelboom et al. (2023); Satorras et al. (2021), we partition the dataset into training, validation, and test sets. The mean absolute error between predictions and ground truth for test set is reported in Table 4, revealing the competitive performance of TopNets compared to baselines. Notably, on many targets, TopNets achieve results nearly on par with SOTA approaches, surpassing in predicting ZPVE, $\Delta\epsilon$ and ϵ_{LUMO}. This achievement is intriguing as our architecture, not specifically tailored for molecular tasks, lacks many molecule-specific intricacies, like Bessel function embeddings (Gasteiger et al., 2020b).

6.3. CDR-H3 Antibody Design

We took the antigen-antibody complexes dataset from Structural Antibody Database (Dunbar et al., 2014) and removed invalid data points. We followed a strategy similar to Verma et al. (2023) for data preparation and splitting and employ Amino Acid Recovery (AAR) and RMSD for quantitative evaluation. AAR is defined as the overlapping rate between the predicted 1D sequences and the ground truth. RMSD is calculated via the Kabsch algorithm (Kabsch, 1976) based on C_α spatial features of the CDR residues. Table 5 showcases the performance of TopNets compared to the baseline methods over CDR-H3 design. TopNets outperform other methods in terms of sequence prediction, thus improving over the SOTA and demonstrating the benefit of persistent homology in generative design.

Table 5: Results on CDR-H3 design benchmark. We report AAR and RMSD metrics. TopNets significantly outperform baselines on AAR while being competitive on RMSD. We used $\text{TopAgg}_{\text{RePHINE}}$ for aggregating the PH embeddings.

Method	Diagram	AAR % (↑)	RMSD (↓)
LSTM	-	15.69 $\pm$ 0.91	(N/A)
C-LSTM	-	15.48 $\pm$ 1.17	(N/A)
RefineGNN	-	21.13 $\pm$ 1.59	6.00 $\pm$ 0.55
C-RefineGNN	-	18.88 $\pm$ 1.37	6.22 $\pm$ 0.59
MEAN	-	36.38 $\pm$ 3.08	2.21 $\pm$ 0.16
AbODE	-	39.8 $\pm$ 1.17	1.73 $\pm$ 0.11
TopNets	VC	43.00 $\pm$ 1.34	1.73 $\pm$ 0.21
	RePHINE	**44.80** $\pm$ 1.57	1.75 $\pm$ 0.17

7. Conclusion and Limitations

We introduce TopNets to illustrate the theoretical and practical benefits of including persistent features in topological networks, and their geometric and continuous-time extensions. TopNets incur considerable computational expense due to costs involved in computing PH embeddings as well as higher-order message-passing. Additionally, our research is confined to simplicial complexes, and exploring combinatorial complexes is an interesting avenue for future work.

Acknowledgements

This work has been supported by the Research Council of Finland under the *HEALED* project (grant 13342077), Jane and Aatos Erkko Foundation project (grant 7001703) on "Biodesign: Use of artificial intelligence in enzyme design for synthetic biology", and Finnish Center for Artificial Intelligence FCAI (Flagship programme). We acknowledge CSC – IT Center for Science, Finland, for providing generous computational resources.

References

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence Images: A Stable Vector Representation of Persistent Homology. *Journal of Machine Learning Research*, 18(8):1–35, 2016.

Sergio Barbarossa and Stefania Sardellitti. Topological signal processing over simplicial complexes. *IEEE Transactions on Signal Processing*, 68:2992–3007, 2020.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning (ICML)*, 2021b.

Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh Gupta. Clifford neural layers for PDE modeling. In *ICLR*, 2023.

Johann Brehmer, Pim De Haan, Sönke Behrends, and Taco Cohen. Geometric algebra transformers. *arXiv preprint arXiv:2305.18415*, 2023.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velicković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

P. Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102, 2015.

Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures. In *Artificial Intelligence and Statistics (AISTATS)*, 2020.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.

Binghui Chen, Weihong Deng, and Jiani Hu. Mixed high-order attention network for person re-identification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 371–381, 2019.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. Gread: Graph neural reaction-diffusion equations. *arXiv preprint arXiv:2211.14208*, 2022.

Jean-Pierre Demailly. *Analyse numérique et équations différentielles*. EDP sciences Les Ulis, 2006.

Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*, 2020.

James Dunbar, Konrad Krawczyk, Jinwoo Leem, Terry Baker, Angelika Fuchs, Guy Georges, Jiye Shi, and Charlotte M Deane. Sabdab: the structural antibody database. *Nucleic acids research*, 42(D1):D1140–D1146, 2014.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

Alexandre Duval, Simon V. Mathis, Chaitanya K. Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D. Malliaros, Taco Cohen, Pietro Lio, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems, 2023.

H. Edelsbrunner and J. Harer. *Computational Topology - an Introduction.* American Mathematical Society, 2010.

Floor Eijkelboom, Rob Hesselink, and Erik Bekkers. E(n) equivariant message passing simplicial networks. *arXiv preprint arXiv:2305.07100*, 2023.

Linton Freeman. The development of social network analysis. *A Study in the Sociology of Science*, 1(687):159–167, 2004.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

Vikas K. Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, 2020.

Johannes Gasteiger, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020a.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020b.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

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

Lorenzo Giusti, Claudio Battiloro, Lucia Testa, Paolo Di Lorenzo, Stefania Sardellitti, and Sergio Barbarossa. Cell attention networks. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2023.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021.

C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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

Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020.

Johanna Immonen, Amauri H. Souza, and Vikas Garg. Going beyond persistent homology using persistent homology. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Kanchan Jha, Sriparna Saha, and Hiteshi Singh. Prediction of protein–protein interaction using graph neural networks. *Scientific Reports*, 12(1):8360, 2022.

Wengong Jin, Jeremy Wohlwend, Regina Barzilay, and Tommi Jaakkola. Iterative refinement graph neural network for antibody sequence-structure co-design, 2022.

Chaitanya K. Joshi, Cristian Bodnar, Simon V. Mathis, Taco Cohen, and Pietro Liò. On the expressive power of geometric graph neural networks, 2023.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

Timothy Doyeon Kim, Tankut Can, and Kamesh Krishnamurthy. Trainability, expressivity and interpretability in gated neural odes. *arXiv preprint arXiv:2307.06398*, 2023.

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

Dmitrii Kochkov, Jamie Smith, Ayya Alieva, Qing Wang, Michael Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

Xiangzhe Kong, Wenbing Huang, and Yang Liu. Conditional antibody design as 3d equivariant graph translation, 2023.

Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted Gaussian kernel for topological data analysis. In *International Conference on Machine Learning (ICML)*, pages 2004–2013, 2016.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

Yi Liu, Limei Wang, Meng Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d graph networks. *arXiv preprint arXiv:2102.05013*, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3276–3285. PMLR, 2018.

Pierre Marion. Generalization bounds for neural ordinary differential equations and deep residual networks. *arXiv preprint arXiv:2305.06648*, 2023.

Mathilde Papillon, Sophia Sanborn, Mustafa Hajij, and Nina Miolane. Architectures of topological deep learning: A survey on topological neural networks. *ArXiv e-prints*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

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

B. Rieck. On the expressivity of persistent homology in graph learning. *arXiv: 2302.09826*, 2023.

Carl Runge. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

Michael Sander, Pierre Ablin, and Gabriel Peyré. Do residual neural networks discretize neural ordinary differential equations? *Advances in Neural Information Processing Systems*, 35:36520–36532, 2022.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Matthew Thorpe, Tan Minh Nguyen, Heidi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representation (ICLR)*, 2022.

Petar Velicković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Yogesh Verma, Samuel Kaski, Markus Heinonen, and Vikas Garg. Modular flows: Differential molecular generation. *arXiv preprint arXiv:2210.06032*, 2022.

Yogesh Verma, Markus Heinonen, and Vikas Garg. AbODE: Ab initio antibody design using conjoined ODEs. In *Proceedings of the 40th International Conference on Machine Learning*, pages 35037–35050. PMLR, 2023.

Yogesh Verma, Markus Heinonen, and Vikas Garg. ClimODE: Climate and weather forecasting with physics-informed neural ODEs. In *The Twelfth International Conference on Learning Representations*, 2024.

Ee Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11, 2017.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks. *NeurIPS*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

A. Persistent homology

Persistent homology (PH) stands as a cornerstone in topological data analysis (TDA). At its core, PH seeks to capture multiresolution topological features (e.g., connected components, loops, voids, etc.) from data. Here, we offer a short overview of PH and direct readers to (Hensel et al., 2021) and (Edelsbrunner and Harer, 2010) for an exhaustive treatment.

In the following, we consider topological spaces given by simplicial complexes. In particular, consider a simplicial complex denoted by K. The p-chains are formal sums $c = \sum a_i \sigma_i$, where $a_i \in \mathbb{Z}/2\mathbb{Z}$ and σ_i represent p-dimensional simplices in K. By equipping p-chains with addition, we obtain the group $C_p(K)$. Another important notion is that of boundary of a simplex. Consider a p-simplex $\sigma = [v_0, ..., v_p] \in K$. The boundary of σ corresponds to the sum of its $(p-1)$-dimensional faces, i.e.,

$$\partial_p \sigma = \sum_{j=0}^{p} [v_0, ..., v_{j-1}, v_{j+1}, \dots, v_p].$$

Importantly, we can extend this definition to define the boundary homomorphism $\partial_p : C_p(K) \to C_{p-1}(K)$, where $\partial_p \sum a_i \sigma_i = \sum a_i \partial_p \sigma_i$. Then, we can define a sequence of groups, also called a chain complex, as:

$$...C_{p+1}(K) \xrightarrow{\partial_{p+1}} C_p(K) \xrightarrow{\partial_p} C_{p-1}(K)...$$

where groups are connected via boundary homomorphisms. The p-th homology group comprises p-chains with empty boundaries (i.e., $\partial_p \sigma = 0$), whereby each of these specific p-chains (cycles) represents a boundary of a distinct simplex in $C_{p+1}(K)$. Hence, we define the p-th homology group H_p as the quotient space:

$$H_p = \text{ker}\partial_p / \text{Im}\partial_{(p+1)}.$$

The p-th Betti number of K, denoted by β_p, is equal to the rank of H_p.

In persistent homology, we keep track of the evolution of Betti numbers across a sequence of chain complexes. The sequence of complexes arise from a filtration — a nested sequence of simplicial subcomplexes $\emptyset \subset K_{\alpha_1} \subset \ldots \subset K_{\alpha_n} = K$, indexed by timestamps α_i (with $\alpha_{i+1} > \alpha_i$ for all i). By computing the homology groups for each of these simplicial complexes, we obtain detailed topological information from K. In practice, this is done by associating a pair of timestamps (α_i, α_j) for every element of the homology groups (or topological features), indicating the filtration timestamp at which it emerged and disappeared. The persistence of a point (α_i, α_j) denotes the duration for which the corresponding feature persisted. We set $\alpha_j = \infty$ if the topological feature persists until the final filtration timestamp. Formally, let $Z_p(K_{\alpha_i}) = \text{ker}\partial_p^{\alpha_i}$ and $B_p(K_{\alpha_i}) = \text{Im}\partial_p^{\alpha_i}$ be the standard p-cycle and p-boundary groups for the complex K_{α_i}. Then, the pth persistent homology groups are

$$H_p^{i,j} = Z_p(K_{\alpha_i}) / (B_{p+1}(K_{\alpha_j}) \cap Z_p(K_{\alpha_i}))$$

for all $1 \leq i \leq j \leq n$. Again, the p-th persistent Betti number $\beta_p^{i,j}$ corresponds to the rank of $H_p^{i,j}$. Finally, a persistence diagram comprising the persistence pairs (α_i, α_j) with corresponding multiplicities given by $\mu_p^{i,j} = (\beta_p^{i,j-1} - \beta_p^{i,j}) - (\beta_p^{i-1,j-1} - \beta_p^{i-1,j})$ encodes the persistent homology groups.

B. Implementation Details

Below are the implementation details.

B.1. Graph Classification

We followed the following hyperparameters and training setup in Table 6 to conduct our experiments on real-world graph classification.

B.2. Molecular Data QM9

For the discrete case, we followed the data-preparation strategies, training setup, and hyperparameters as outlined by Eijkelboom et al. (2023). We enhanced each layer with an Equivariant RePHINE layer, inspired by the original RePHINE (Immonen et al., 2023), incorporating Euclidean distance as an invariant feature in the filtration function. The Vertex Cloud (VC) retained its absence of 3D positional information, consistent with (Immonen et al., 2023). For the continuous case,

Table 6: Default hyperparameters for TopNets for Graph Classification Benchmark

Hyperparameter	Meaning	Value
Solver	ODE-Solver	`adaptive-heun,euler`
GNN	GNN Architecture	{GCN,GIN,MPSN}
PH	Type of PH	{VC,TOGL,RePHINE}
Steps	Number of steps for ODE solver	{20,15,10,5}
Node Hidden Dim	Latent dimension of node features	128
PH embed dim	Latent dimension of PH features	64
Num Filt	Number of filtrations	8
Hiden Filtration	Hidden dimension of filtration functions	16
Batch Size	Size of batches	64
LR	Learning Rate	0.001
Scheduler	Learning Rate scheduler	`Cosine-Annealing-LR`
Epochs	Number of epochs	300

we employed a single layer of EMPSN to parameterize the ODE dynamics, leveraging the odeint package to solve these dynamics. Additionally, an Equivariant RePHINE layer was applied per time step. Solver options included `euler` and `adaptive-heun`, with the number of time steps ranging from 5 to 20. Filtration parameters remained consistent with those described in Table 6, alongside identical training hyperparameters and setup as in the original EPMSN paper.

B.3. CDR-H3 Antibody Design

We followed the following hyperparameters to conduct our experiments on CDR-H3 Antibody Design.

Table 7: Default hyperparameters for TopNets for CDR-H3 Antibody Design

Hyperparameter	Meaning	Value
GNN	GNN Architecture	TransformerConv (Shi et al., 2020)
PH	Type of PH	{VC,RePHINE}
Layers	Number of layers	4
Node Hidden Dim	Latent dimension of node features	[128,256,128,64]
PH embed dim	Latent dimension of PH features	64
Num Filt	Number of filtrations	8
Hiden Filtration	Hidden dimension of filtration functions	16
Batch Size	Size of batches	32
LR	Learning Rate	0.001
Scheduler	Learning Rate scheduler	`Cosine-Annealing-LR`
Epochs	Number of epochs	1000

C. Deduction from TopNets

In our study we restrict ourselves to 1-dim simplicial complexes (otherwise mentioned) and here we showcases the deductions of various methods from TopNets.

$$\mathrm{TopAgg}^{\mathrm{TOGL}} = \begin{cases} \mathrm{TopAgg}_0(\tilde{x}_\sigma^\ell, r_\sigma^\ell) = \tilde{x}_\sigma + r_\sigma^\ell, \\ \mathrm{TopAgg}_1(\tilde{x}_\sigma^\ell, r_\sigma^\ell) = \tilde{x}_\sigma^\ell \\ \mathrm{Agg}_{\ell,0}(\{r_\sigma^\ell\}) : \mathrm{NA} \\ \mathrm{Agg}_{\ell,1}(\{r_\sigma^\ell\}) = \mathrm{DeepSet}_\ell(\{r_\sigma^\ell\}) \end{cases}$$

The readout layers for TOGL concatenate the aggregated topological embeddings (1-dim) with the last layer pooled 0-dim simplex features and using it for downstream tasks such as classification.

In case of PersLay, they does not use any TNN layers over the node features, thus $\mathrm{TopAgg}_{0,1}$ are N/A, and the other aggregation is performed as,

$$\mathrm{TopAgg}^{\text{PersLay}} = \begin{cases} \mathrm{Agg}_0(\{r_\sigma\}) = \mathrm{DeepSet}_0(\{r_\sigma\}), \\ \mathrm{Agg}_1(\{r_\sigma\}) = \mathrm{DeepSet}_1(\{r_\sigma\}) \end{cases}$$

However PersLay, utilizes an additional option to use the pooled 0-dim simplex features via concatenating it with the aggregated topological embeddings (0-dim and 1-dim) and using it for downstream tasks such as classification.

$$\mathrm{TopAgg}^{\text{RePHINE}} = \begin{cases} \mathrm{TopAgg}_0(\tilde{x}_\sigma^\ell, r_\sigma^\ell) = \tilde{x}_\sigma, \\ \mathrm{TopAgg}_1(\tilde{x}_\sigma^\ell, r_\sigma^\ell) = \tilde{x}_\sigma^\ell, \\ \mathrm{Agg}_{\ell,0}(\{r_\sigma^\ell\}) = \mathrm{DeepSet}_\ell(\{r_\sigma^\ell\}) \\ \mathrm{Agg}_{\ell,1}(\{r_\sigma^\ell\}) = \mathrm{DeepSet}_\ell(\{r_\sigma^\ell\}) \end{cases}$$

The readout layers for RePHINE concatenate the aggregated topological embeddings (0-dim and 1-dim) with the last layer pooled 0-dim simplex features and using it for downstream tasks such as classification.

Note that wherever we utilise $\mathrm{TopAgg}^{\mathrm{TOGL/RePHINE/PersLay}}$ as the topological aggregation method we utilise their specific readout layers as well. The Table 8 summarizes the deduction further from TopNets for various methods.

Table 8: Deduction of PH-based methods from TopNets

Module	Meaning	TOGL	PersLay	RePHINE
TNNLayer	TNN/GNN Architecture	{GCN,GIN}	-	{GCN,GIN}
PD	Type of PH-diagrams used	VC	VC, Point transformations	RePHINE
f^ℓ	Filtration functions	f_v	f_v	(f_v^ℓ, f_e^ℓ)
ψ	Diagram combining functions	DeepSets	DeepSets	DeepSets
TopAgg	Topological Aggregation	$\mathrm{TopAgg}^{\text{TOGL}}$	$\mathrm{TopAgg}^{\text{PersLay}}$	$\mathrm{TopAgg}^{\text{RePHINE}}$

D. Proofs

D.1. Proof of Proposition 1

Let us first introduce two important notions of neighborhood for simplicial complexes: the boundary-adjacency and the upper-adjacency neighborhoods. Let σ be a simplex. Then, the boundary neighborhood of σ is given by $\mathcal{B}(\sigma) = \{\tau \subset \sigma : \dim(\tau) = \dim(\sigma) - 1\}$ — the set of σ's faces of dimension $\dim(\sigma) - 1$. The upper-adjacency neighborhood of σ is $\mathcal{N}_\uparrow(\sigma) = \{\sigma' : \exists \delta \text{ such that } \sigma \subset \delta, \sigma' \subset \delta \text{ and } \dim(\delta) - 1 = \dim(\sigma') = \dim(\sigma)\}$ — i.e., there exists a simplex δ that is co-face of both σ and σ' with dimension equal to $\dim(\sigma') + 1$.

Consider simplices of a graph (1-dim complex). If σ is a vertex, it has no boundary neighborhood and its upper-adjacency neighborhood are the vertices directly connected to σ. On the other hand, if σ is an edge, it has no upper-adjacency neighborhood and its boundary one is given by the vertices that σ is incident to.

The simplicial Weisfeiler-Leman test (Bodnar et al., 2021b) resembles the original 1-WL test but takes into account the colors of the simplices of both boundary adjacency and upper adjacency in the hash (aggregating) function. Every simplex has an associated color. For a proper definition, we refer to Bodnar et al. (2021b).

To prove Proposition 1, it suffices to i) show a pair of clique complexes that SWL cannot distinguish, ii) and derive a color-based filtration that produces different persistence diagrams. Consider the clique complexes K and K' in Figure 2.

We know that the multisets of colors of 0-simplices (vertices) from K and K' are identical at any iteration of the WL algorithm. This stems from the fact that these graphs are known to be indistiguashable by 1-WL and that the only valid

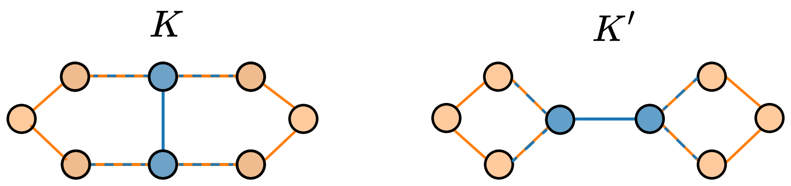

Figure 2: Two non-isomorphic simplicial complexes.

neighborhood structure for vertices is the classic one (adjacent vertices) — upper-adjacency neighborhood. In other words, for each vertex in $v \in K$ with computation tree T_v, there is a corresponding vertex $v' \in K'$ such that T_v is isomorphic to $T_{v'}$ for any depth. We also note that, in SWL, the color-refinement procedure for a vertex v from upper-adjacency includes the color of the edge that v is incident to. However, in our example, the color of each edge is fully defined by the history of colors of its incident vertices. Thus, we can disregard the colors of 0-simplices.

Similarly, if $\sigma = [u, v]$ is an edge, its only neighbors are u and v (boundary adjacency). If we consider edges of the same colors in K and K', their neighbors have isomorphic computation trees. As a result, at every iteration of the test, the colors used to update these edges are exactly the same. Therefore, SWL cannot distinguish these complexes. As noted by Bodnar et al. (2021b), when SWL is applied to 1-simplicial complexes, i.e. graphs, it corresponds to the 1-WL test.

To prove that there exists a color-based filtration that distinguishes these graphs. We can directly leverage Theorem 2 in (Immonen et al., 2023) to show that there is a color-disconnecting set to these graphs $Q = \{\text{blue}\}$. If we remove the blue edges from K and K', they end up with different numbers of connected components. This concludes the proof.

D.2. Proof of Proposition 2

Consider a geometric simplicial complex (K, x, z) and geometric i-simplex-color filtrations induced by a function f. Let $R \in \mathrm{E}(n)$ be a group element that acts on the 0-simplex positional features. Recall that geometric i-simplex-color filtrations leverage a function $\mathrm{Inv}(\cdot)$, which is invariant to $E(n)$ group actions. Thus, for any simplex τ, we have that $\mathrm{Inv}(\{z_v\}_{v \in \tau}) = \mathrm{Inv}(R \cdot \{z_v\}_{v \in \tau})$. The diagrams of any dimension are fully determined by the filtrations, which in turn are obtained from the simplex rank function $o_f(\sigma)$ as

$$o_f(\sigma) = \begin{cases} \max\limits_{\tau \subset \sigma : \dim(\tau) = i} f(x_\tau, \mathrm{Inv}(\{z_v\}_{v \in \tau})) & \text{if } \dim(\sigma) \geq i \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

Note that group actions only affect geometric i-simplex-color filtrations via the input of the Inv function. Thus, we can write

$$R \cdot o_f(\sigma) = \begin{cases} \max\limits_{\tau \subset \sigma : \dim(\tau) = i} f(x_\tau, \mathrm{Inv}(R \cdot \{z_v\}_{v \in \tau})) & \text{if } \dim(\sigma) \geq i \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

Recalling that invariant features remain intact via the transformation, this would imply $o_f(\sigma) = R \cdot o_f(\sigma)$, which would lead to identical filtrations and, consequently, the same persistence diagrams (of any dimension) and topological embeddings. This holds for any $i \geq 0$.

E. Approximation Error Bounds

Below are the bounds for the TOGL and RePHINE cases. Note that we assume a fixed simplicial complex for deriving the bound.

E.1. TOGL

E.1.1. Continuous Counterpart

The dynamics of the TOGL-GNN for a node v can be described as,

$$x_v^\ell = \mathrm{TNNLayer}_\ell(x_v^{\ell-1}, K) + \psi(\mathrm{PD}(\sigma; f_\theta, x_v^{\ell-1}, K)) \tag{21}$$

For clarity of exposition, let $\mathrm{TNNLayer}_\ell(x^{\ell-1}, K) = x_v^{\ell-1} + m_v^\ell$, where m_v^ℓ is the aggregated message as described in Section 5. The continuous depth counterpart can be written as a graph ODE, parametrized by the following differential equation,

$$\dot{x}_v^t = \psi(\mathrm{PD}(\sigma; f_\theta, x^t, K)) + m_v^t \tag{22}$$

E.1.2. Error Bound

We consider N-layered TOGL GNN and assume an Euler discretization scheme for the ODE system consisting of N steps to be consistent. We define $s_\ell = \ell/N = \ell h$, where $h = 1/N$ is the step size and, s_ℓ represents a time at ℓ^{th} step. We utilize the Taylor expansion as,

$$x_v^{s_\ell + h} = x_v^{s_\ell} + h\dot{x}_v^{s_\ell} + R_1(h) \tag{23}$$

We consider a simple modification of the discrete TOGL GNN network for N-depth by letting the mapping explicitly depend on the depth of the network as,

$$x_v^\ell = x_v^{\ell-1} + \frac{1}{N}\left(\psi(\mathrm{PD}(\sigma; f_\theta, x^{\ell-1}, K)) + m_v^{\ell-1}\right) \tag{24}$$

We consider the error $e_v(\ell) = x_v^{s_\ell} - x_v^\ell$, where x_v^ℓ is the node v embeddings after l TOGL-GNN layers,

$$\begin{aligned}
e_v(\ell+1) - e_v(\ell) &= x_v^{s_{\ell+1}} - x_v^{s_\ell} + x_v^\ell - x_v^{\ell+1} && (25)\\
&= h\dot{x}_v^{s_\ell} + R_1(h) - \frac{1}{N}\left(\psi(\mathrm{PD}(\sigma; f_\theta, x^\ell, K)) + m_v^\ell\right) && (26)\\
&= R_1(h) + h\left(m_v^{s_\ell} - m_v^\ell\right) && (27)\\
&\quad + h\left(\psi(\mathrm{PD}(\sigma; f_\theta, x^{s_\ell}, K)) - \psi(\mathrm{PD}(\sigma; f_\theta, x^\ell, K))\right) && (28)
\end{aligned}$$

We assume m_v,ψ to be L_m, L_β-Lipschitz ($L_\beta = L_\psi L_\theta$, due to the composition of ψ and f_θ), giving us, (note that the parametrization of m_v^ℓ and $m_v^{s_\ell}$ is the same, and only differs in inputs.)

$$\begin{aligned}
\|e_v(\ell+1) - e_v(\ell)\| &\leq R_1(h) + hL_m\|e_v(\ell)\| + hL_\beta\|e_v(\ell)\| && (29)\\
\|e_v(\ell+1)\| &\leq R_1(h) + \left(1 + \frac{L_m + L_\beta}{N}\right)\|e_v(\ell)\| && (30)
\end{aligned}$$

Using the discrete Gronwall lemma (Sander et al., 2022; Demailly, 2006), we get the following relation, where $e_v(0) = 0$,

$$\begin{aligned}
\|e_v(\ell)\| &\leq 0 + R_1(h) \sum_{0 \leq j \leq n-1} \exp\left(\frac{L_m + L_\beta}{N}(N-1-j)\right) && (31)\\
&\leq R_1(h)\frac{\exp(\frac{L_m + L_\psi}{N}N) - 1}{\exp(\frac{L_m+L_\beta}{N}) - 1} && (32)
\end{aligned}$$

But, $\exp(\frac{L_m+L_\beta}{N}) - 1 \geq \frac{L_m+L_\beta}{N}$, using that we get,

$$\|e_v(\ell)\| \leq R_1(h)\frac{N(\exp(L_m + L_\beta) - 1)}{L_m + L_\beta} \tag{33}$$

E.2. RePHINE

E.2.1. Continuous Counterpart

The dynamics of RePHINE-GNN for node v can be expressed as,

$$\begin{aligned}
x_v^\ell &= \mathrm{TNNLayer}_\ell(x_v^{\ell-1}, K) && (34)\\
r^\ell &= \psi^\ell(\mathrm{PD}(\sigma; f_\theta^\ell, x^{\ell-1}, K)) && (35)
\end{aligned}$$

Let $\text{TNNLayer}_\ell(x_v^{\ell-1}, K) = x_v^{\ell-1} + m_v^\ell$, where m_v^ℓ is the aggregate as described in Section 5 and $x^\ell = \{x_u^\ell\}_u$. Moreover, collecting all node updates, the recursive update can be expressed as $x^\ell = x^{\ell-1} + m^\ell$, where m^ℓ are the message updates for the all node embeddings. RePHINE parameterizes each layer filtration function f_θ^ℓ and DeepSet function ψ^ℓ distinctively. The continuous depth counterpart can be written as a coupled latent graph ODE, parametrized by the following set of differential equations as,

$$\dot{x}_v^t = m_\sigma^t \tag{36}$$

$$r^t = \psi^t(\text{PD}(\sigma; f_\theta^t, x^t, K)) \tag{37}$$

E.2.2. Error Bound

We consider N layered RePHINE GNN, and assume an Euler discretization scheme for the ODE system consisting of N steps to be consistent. We define $s_\ell = \ell/N = \ell h$, where $h = \frac{1}{N}$ is the step size and, s_ℓ represents a time at ℓ^{th} step. We derive the error bounds both for the node features and topological embeddings as follows.

Node Embeddings We utilize the Taylor expansion, as,

$$x_v^{s_\ell + h} = x_v^{s_\ell} + h\dot{x}_v^{s_\ell} + R_1(h) \tag{38}$$

We consider a simple modification of the discrete RePHINE GNN network for N-depth by letting the mapping explicitly depend on the depth of the network as,

$$x_v^\ell = x_v^{\ell-1} + \frac{1}{N} m_v^\ell \tag{39}$$

We consider the node-embedding error, $e_v(\ell) = x_v^{s_\ell} - x_v^\ell$,

$$e_v(\ell+1) - e_v(\ell) = x_v^{s_{\ell+1}} - x_v^{s_\ell} + x_v^\ell - x_v^{\ell+1} \tag{40}$$

$$= h\dot{x}_v^{s_\ell} + R_1(h) - \frac{1}{N} m_v^\ell \tag{41}$$

$$= R_1(h) + h\left(m_v^{s_\ell} - m_v^\ell\right) \tag{42}$$

Assuming m_σ to be L_m-Lipschitz, gives us (note that the parametrization of $m_\sigma^{s_\ell}$ and m_σ^ℓ is the same, and only differs in inputs.)

$$\|e_v(\ell+1) - e_v(\ell)\| \leq R_1(h) + hL_m\|e_v(\ell)\| \tag{43}$$

$$\|e_v(\ell+1)\| \leq R_1(h) + \left(1 + \frac{L_m}{N}\right)\|e_v(\ell)\| \tag{44}$$

Using the discrete Gronwall lemma, we get the following relation, where $\mathbf{e}_x(0) = 0$,

$$\|e_v(\ell)\| \leq 0 + R_1(h) \sum_{0 \leq j \leq n-1} \exp(\frac{L_m}{N}(N-1-j)) \tag{45}$$

$$\leq R_1(h) \frac{\exp(\frac{L_m}{N}N) - 1}{\exp(\frac{L_m}{N}) - 1} \tag{46}$$

But, $\exp(\frac{L_m}{N}) - 1 \geq \frac{L_m}{N}$, using that we get,

$$\|e_v(\ell)\| \leq R_1(h) \frac{N(\exp(L_m) - 1)}{L_m} \tag{47}$$

Topological Embeddings We consider the bound on topological-embedding in this section. Let r^ℓ is the topological-embedding after l RePHINE-GNN layers, the error bound can be computed as,

$$e_r(\ell) = r^{s_\ell}(x^{s_\ell}, K) - r^\ell(x^\ell, K) \tag{48}$$

$$= r^{s_\ell}(x^{s_{\ell-1}}) + h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}} m^{s_{\ell-1}} + R_1(h) - r^\ell(x^{\ell-1} + \frac{1}{N} m^{\ell-1}) \tag{49}$$

Now, using the Taylor expansion to expand the second term, we can write ($h = 1/N$)

$$r^{\ell}(x^{\ell-1} + \frac{1}{N}m^{\ell-1}) = r^{\ell}(x^{\ell-1}) + h\frac{dr^{\ell}(x^{\ell-1})}{dx^{\ell-1}}m^{\ell-1} + R_1(m^{\ell-1}) \tag{50}$$

Putting into the original equation, we get,

$$e_r(\ell) = \underbrace{r^{s_\ell}(x^{s_{\ell-1}}) - r^{\ell}(x^{\ell-1})}_{\text{First Term}} + \underbrace{h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}m^{s_{\ell-1}} - h\frac{dr^{\ell}(x^{\ell-1})}{dx^{\ell-1}}m^{\ell-1}}_{\text{Second Term}} \tag{51}$$

$$+ R_1(h) - R_1(m^{\ell-1}) \tag{52}$$

We simplify each term as follows,

First Term: The first term denotes the difference between the topological embeddings, and we assume that $\psi^{s_\ell} \equiv \psi^{\ell}$, as both functions are evaluated at the ℓ layer (step), and let it be L^{ℓ}_{β}-Lipschitz ($L^{\ell}_{\beta} = L^{\ell}_{\psi}L^{\ell}_{\theta}$, due to the composition of ψ^{ℓ} and f^{ℓ}_{θ}) at the time-step, giving us

$$\|r^{s_\ell}(x^{s_{\ell-1}}) - r^{\ell}(x^{\ell-1})\| = \|\psi^{s_\ell}(\mathrm{PD}(\sigma; f^{s_\ell}_{\theta}, x^{s_\ell - 1}, K)) - \psi^{\ell}(\mathrm{PD}(\sigma; f^{\ell}_{\theta}, x^{\ell-1}, K)\| \tag{53}$$

$$\leq L^{\ell}_{\beta}\|e_v(\ell-1)\| \tag{54}$$

Second Term: We simplify the second term as follows, by adding and subtracting a term as,

$$= h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}m^{s_{\ell-1}} - h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}m^{\ell-1} + h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}m^{\ell-1} - h\frac{dr^{\ell}(x^{\ell-1})}{dx^{\ell-1}}m^{\ell-1} \tag{55}$$

$$= h\left(\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}\left(m^{s_{\ell-1}} - m^{\ell-1}\right) + m^{\ell-1}\left(\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}} - \frac{dr^{\ell}(x^{\ell-1})}{dx^{\ell-1}}\right)\right) \tag{56}$$

where the parts of the second term can be simplified as,

$$\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}} = \frac{|r^{s_\ell}(x^{s_{\ell-1+h}}) - r^{s_\ell}(x^{s_{\ell-1}})|}{|x^{s_{\ell-1+h}} - x^{s_{\ell-1}}|} \leq L^{\ell}_{\beta} \tag{57}$$

Similarly, the other term,

$$\frac{dr^{\ell}(x^{\ell-1})}{dx^{\ell-1}} = \frac{|r^{\ell}(x^{\ell-1+h}) - r^{\ell}(x^{\ell-1})|}{|x^{\ell-1+h} - x^{\ell-1}|} \leq L^{\ell}_{\beta} \tag{58}$$

leading to $\leq (L^{\ell}_{\beta} - L^{\ell}_{\beta}) = 0$. So, the equation will become,

$$h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}\left(m^{s_{\ell-1}} - m^{\ell-1}\right) \leq h\frac{dr^{s_\ell}(x^{s_{\ell-1}})}{dx^{s_{\ell-1}}}L_m\|e_v(\ell-1)\| \tag{59}$$

$$\leq \frac{L^{\ell}_{\beta}L_m}{N}\|e_v(\ell-1)\| \tag{60}$$

Collecting all the terms, it will account for,

$$\|e_r(\ell)\| \leq L^{\ell}_{\beta}\|e_v(\ell-1)\| + \frac{L^{\ell}_{\beta}L_m}{N}\|e_v(\ell-1)\| + R_1(h) - R_1(m^{\ell-1}) \tag{61}$$