Topological Graph Neural Networks

Max Horn[1, 2, *] **Edward De Brouwer**[3, *] **Michael Moor**[1, 2] **Yves Moreau**[3]
Bastian Rieck[1, 2, 4, †] **Karsten Borgwardt**[1, 2, †]

[1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland
[2]SIB Swiss Institute of Bioinformatics, Switzerland
[3]ESAT-STADIUS, KU LEUVEN, 3001 Leuven, Belgium
[4]B.R. is now with the Institute of AI for Health, Helmholtz Zentrum München, Neuherberg, Germany
[*]These authors contributed equally.
[†]These authors jointly supervised this work.

Abstract

Graph neural networks (GNNs) are a powerful architecture for tackling graph learning tasks, yet have been shown to be oblivious to eminent substructures such as cycles. We present TOGL, a novel *layer* that incorporates global topological information of a graph using persistent homology. TOGL can be easily integrated into *any type* of GNN and is strictly more expressive in terms of the Weisfeiler–Lehman graph isomorphism test. Augmenting GNNs with our layer leads to improved predictive performance for graph and node classification tasks, both on synthetic data sets (which can be classified by humans using their topology but not by ordinary GNNs) and on real-world data.

1 Introduction

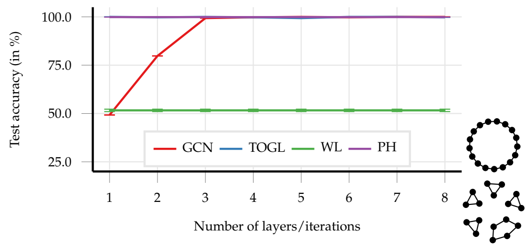
Graphs are a natural description of structured data sets in many domains, including bioinformatics, image processing, and social network analysis. Numerous methods address the two dominant graph learning tasks of graph classification or node classification. In particular, *graph neural networks* (GNNs) describe a flexible set of architectures for such tasks and have seen many successful applications over recent years [15]. At their core, many GNNs are based on iterative message passing schemes. Since these schemes are collating information over the neighbours of every node, GNNs cannot necessarily capture certain topological structures in graphs, such as cycles [1]. These structures, however, are relevant for certain applications, such as the analysis of molecular graphs, whose classification necessitates knowledge about connectivity information [7, 12].

In this paper, we address this issue by proposing a Topological Graph Layer (TOGL) that can be easily integrated into any GNN to make it 'topology-aware.' Our method is rooted in the nascent field of topological data analysis (TDA), which focuses on describing coarse structures that can be used to describe the shape of complex structured and unstructured data sets. We thus obtain a generic way to augment existing GNNs and increase their expressivity in graph learning tasks. Figure 1 provides a motivational example that showcases the potential benefits of using topological information: (i) high predictive performance is reached *earlier* for a *smaller* number of layers, and (ii) learnable topological representations outperform fixed ones if more complex topological structures are present in a data set.

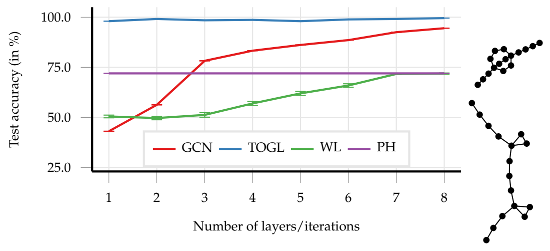
2 Computational Topology

We consider undirected graphs of the form $G = (V, E)$ with a set of vertices V and a set of edges $E \subseteq V \times V$. The basic topological features of such a graph G are the number of connected components β_0 and the number of cycles β_1. These counts are also known as the 0-dimensional and 1-dimensional *Betti numbers*,

This is a shortened version of our work 'Topological Graph Neural Networks' (arXiv:2102.07835), which is currently under review at ICLR 2022. This version has been significantly condensed.



(a) CYCLES



(b) NECKLACES

Figure 1: As a motivating example, we introduce two topology-based data sets whose graphs can be easily distinguished by humans; the left data set can be trivially classified by all topology-based methods (since access to β_0 or β_1 is sufficient for classification), while the right data set necessitates *learnable* topological features. We show the performance of using a GCN *and* TOGL, our method, as compared to a 'pure' GCNs, the Weisfeiler–Lehman (WL) graph kernel, and a static topological method (PH) based on a degree filtration of a graph. When there is more than one layer, we use TOGL as the second layer, while all remaining layers are regular GCN layers (we still denote this hybrid model by 'TOGL' in the legend).

respectively, and can be computed efficiently. Betti numbers are invariant under graph isomorphism [5, pp. 103–133]. The expressivity of Betti numbers can be increased by assuming the existence of a *graph filtration*, i.e. a sequence of nested subgraphs of G such that $\emptyset = G^{(0)} \subseteq G^{(1)} \subseteq G^{(2)} \subseteq \cdots \subseteq G^{(n-1)} \subseteq G^{(n)} = G$. A filtration makes it possible to obtain more insights into the graph by 'monitoring' topological features of *each* $G^{(i)}$ and calculating their topological relevance, also referred to as their *persistence*. If a topological feature appears for the first time in $G^{(i)}$ and disappears in $G^{(j)}$, we assign this feature a persistence of $j - i$. Equivalently, we can represent the feature as a tuple (i, j), which we collect in a *persistence diagram* $\mathcal{D}$. This process was formalised and extended to a wider class of structured data sets, namely simplicial complexes, and is known under the name of *persistent homology*. One of its core concepts is the use of a filtration function $f\colon V \to \mathbb{R}^d$, with $d = 1$ typically, to accentuate certain structural features of a graph. This replaces the aforementioned tuples of the form (i, j) by tuples based on the values of f, i.e. (f_i, f_j). Persistent homology has shown excellent promise in different areas of machine learning research (see Hensel et al. [6] for a recent survey); choosing or learning an appropriate filtration function f is crucial for high predictive performance [7, 18].

Notation. We denote the calculation of persistence diagrams of a graph G under some filtration f by $\mathbf{ph}(G, f)$. This will result in two persistence diagrams $\mathcal{D}_0, \mathcal{D}_1$, containing information about topological features in dimension 0 (connected components) and dimension 1 (cycles). The cardinality of $\mathcal{D}_0$ is equal to the number of nodes n in the graphs and each tuple in the 0-dimensional diagram is associated with the *vertex* that created it. The cardinality of $\mathcal{D}_1$ is the number of cycles.

3 TOGL: A Topological Graph Layer

TOGL is a new type of graph neural network layer that is capable of utilising multi-scale topological information of input graphs. In this section, we give a brief overview of the components of this layer before discussing algorithmic details, theoretical expressivity, computational complexity, and limitations. The layer takes as input a graph $G = (V, E)$ equipped with a set of n vertices V and a set of edges E, along with a set of d-dimensional node attribute vectors $x^{(v)} \in \mathbb{R}^d$ for $v \in V$. These node attributes can either be node features of a data set or hidden representations learnt by some GNN. We employ a family of k vertex filtration functions of the form $f_i\colon \mathbb{R}^d \to \mathbb{R}$ for $i = 1, \ldots, k$. Each filtration function f_i can focus on different properties of the graph. The image of f_i is finite and results in a set of node values $a_i^{(1)} < \cdots < a_i^{(n)}$ such that the graph G is filtered according to $\emptyset = G_i^{(0)} \subseteq G_i^{(1)} \subseteq \cdots \subseteq G_i^{(n)} = G$, where $G_i^{(j)} = \left(V_i^{(j)}, E_i^{(j)}\right)$, with

$V_i^{(j)} := \left\{v \in V \mid f_i\left(x^{(v)}\right) \leq a_i^{(j)}\right\}$, and $E_i^{(j)} := \left\{v, w \in E \mid \max\left\{f_i\left(x^{(v)}\right), f_i\left(x^{(w)}\right)\right\} \leq a_i^{(j)}\right\}$. Given this filtration, we calculate a set of persistence diagrams, i.e. $\mathbf{ph}(G, f_i) = \left\{\mathcal{D}_i^{(0)}, \ldots, \mathcal{D}_i^{(l)}\right\}$. We fix $l = 1$ (i.e. we are capturing connected components and cycles) to simplify our current implementation, but our layer may in principle be extended to arbitrary values of l. In order to benefit from representations that are trainable end-to-end, we use an *embedding function* $\Psi^{(l)}\colon \left\{\mathcal{D}_1^{(l)}, \ldots, \mathcal{D}_k^{(l)}\right\} \to \mathbb{R}^{n' \times d}$ for embedding persistence diagrams into a high-dimensional space that will be used to obtain the vertex representations, where n' is the number of *vertices* n if $l = 0$ and the number of *edges* if $l = 1$. This step is crucial as it enables us to use the resulting topological features as node features, making TOGL a layer that can be integrated into arbitrary GNNs. We experimented with different embedding functions Ψ; the results described in this work are based on a `DeepSets` approach [17], making it possible to take interactions between different points in the persistence diagram into account. We compute our family of k vertex-based filtrations using $\Phi\colon \mathbb{R}^d \to \mathbb{R}^k$, an MLP with a single hidden layer, such that $f_i := \pi_i \circ \Phi$, i.e. the projection of Φ to the ith dimension. We apply Φ to the hidden representations $x^{(v)}$ of all vertices in the graph. Moreover, we treat the resulting persistence diagrams in dimension 0 and 1 differently. For dimension 0, we have a bijective mapping of tuples in the persistence diagram to the vertices of the graph, which was previously exploited in topological representation learning [10]. Therefore, we aggregate $\Psi^{(0)}$ with the original node attribute vector $x^{(v)}$ of the graph in a residual fashion, i.e. $\tilde{x}^{(v)} = x^{(v)} + \Psi^{(0)}\left(\mathcal{D}_1^{(0)}, \ldots, \mathcal{D}_k^{(0)}\right)[v]$, where $\Psi^{(0)}[v]$ denotes taking vth row of $\Psi^{(0)}$ (i.e the topological embedding of vertex v). The output of our layer for dimension 0 therefore results in a new representation $\tilde{x}^{(v)} \in \mathbb{R}^d$ for each vertex v, making it compatible with any subsequent (GNN) layers. By contrast, $\Psi^{(1)}$ is pooled into a graph-level representation, to be used in the final classification layer of a GNN. This is necessary because there is no bijective mapping to the vertices, but rather to edges.

Expressive Power. The expressive power of graph neural networks is well-studied [3, 16] and typically assessed via the iterative Weisfeiler–Lehman label refinement scheme, denoted as WL[1]. Given a graph with an initial set of vertex labels, WL[1] collects the labels of neighbouring vertices for each vertex in a multiset and 'hashes' them into a new label, using a perfect hashing scheme so that vertices/neighbourhoods with the same labels are hashed to the same value. This procedure is repeated and stops either when a maximum number of iterations has been reached or no more label updates happen. The result of each iteration h of the algorithm for a graph G is a feature vector $\phi_G^{(h)}$ that contains individual label counts. Originally conceived as a test for graph isomorphism [14], WL[1] has been successfully used for graph classification [11]. Surprisingly, Xu et al. [16] showed that standard graph neural networks based on message passing are *no more powerful* than WL[1]. It turns out that persistent homology (and TOGL by transitivity) extend the expressivity of WL[1]. We first state a 'lower-bound' type of result.

Theorem 1. *Persistent homology is* at least *as expressive as WL[1], i.e. if the WL[1] label sequences for two graphs G and G' diverge, there exists an injective filtration f such that the corresponding* 0*-dimensional persistence diagrams $\mathcal{D}_0$ and $\mathcal{D}_0'$ are not equal.*

Proof sketch. We first assume the existence of a sequence of WL[1] labels and show how to construct a filtration function f from this. While f will result in persistence diagrams that are different, thus serving to distinguish G and G', it does not necessarily satisfy injectivity. We therefore show that there is an injective function $\tilde{f}$ that is arbitrarily close to f and whose corresponding persistence diagrams $\widetilde{\mathcal{D}_0}$, $\widetilde{\mathcal{D}_0'}$ do *not* coincide. Please refer to the extended version of this preprint[1] for more details. □

To prove that persistent homology and TOGL are more expressive than a GCN, we show that there are pairs of graphs G, G' that cannot be distinguished by WL[1] but that *can* be distinguished by $\mathbf{ph}(\cdot)$ and by TOGL, respectively: let G be a graph consisting of the disjoint union of two triangles, i.e. , and let G' be a graph consisting of a hexagon, i.e. . WL[1] will be unable to distinguish these two graphs because all multisets in every iteration will be the same. Persistent homology, by contrast, can distinguish G from G'

[1] `https://arxiv.org/abs/2102.07835`

Table 1: Predictive performance of our method. We depict the test accuracy obtained on various benchmark data sets when only considering structural information (i.e. the network has access to *uninformative* node features).

	Graph classification				*Node classification*	
METHOD	DD	ENZYMES	MNIST	PROTEINS	CLUSTER	PATTERN
GAT-4	63.3±3.7	21.7± 2.9	63.2±10.4	67.5± 2.6	16.7± 0.0	58.3±8.8
GIN-4	**75.6**±**2.8**	21.3± 6.5	83.4± 0.9	**74.6**± **3.1**	16.4± 0.1	84.8±0.0
GCN-4 (*baseline*)	68.0±3.6	22.0± 3.3	76.2± 0.5	68.8± 2.8	16.7± 0.0	85.6±0.0
GCN-3-TOGL-1	75.1±2.1	**30.3**± **6.5**	**84.8**± **0.4**	73.8± 4.3	**16.8**± **0.0**	**86.7**±**0.0**
GCN-3-TOGL-1 (static)	68.0±2.4	23.7± 5.4	82.9± 0.0	71.2± 5.1	**16.8**± **0.0**	85.8±0.0

using their Betti numbers. We have $\beta_0(G) = \beta_1(G) = 2$, because G consists of two connected components and two cycles, whereas $\beta_0(G') = \beta_1(G') = 1$ as G' only consists of one connected component and one cycle. Together with Theorem 1, this example implies that persistent homology is *strictly* more powerful than WL[1].

4 Experiments

Next to the synthetic data sets shown in Figure 1, we also applied our method to standard graph/node classification benchmarking data sets, albeit with a slight 'twist:' we remove all node attribute features and replace them by random features. This ensures that classification performance is driven *only* by topological information and nothing else. We compare a hybrid model, consisting of a GCN with three layers and a single TOGL layer as the *second* layer,[2] with several well-known graph neural network architectures [2, 8, 13]. Table 1 depicts the results for graph and node classification tasks on such graphs; we observe that the performance of GCN-3-TOGL-1 *always* outperforms the GCN-4 baseline. For MNIST, the gains are substantial, with an increase of more than 8%, but other data sets exhibit similar improvements. This demonstrates the utility of TOGL in making additional structural information available to improve classification performance. While this was not the primary goal of this experiment, we also see that we perform favourably compared to other graph neural networks (in the case of ENZYMES, we even outperform them by a margin of 8%).

5 Conclusion

We presented TOGL, a generically-applicable layer that incorporates topological information into any GNN architecture. We proved that TOGL, due to its filtration functions (i.e. input functions) being learnable, is more expressive than WL[1], the Weisfeiler–Lehman test for graph isomorphism. On data sets with pronounced topological structures, we found that our method helps a GCN achieve high predictive performance. For future work, we are most interested in the investigation of additional regularisation strategies for improving the training process. Furthermore, we hypothesise that the use of different filtration types [9], together with improved persistent homology algorithms [4], will prove beneficial.

[2]This hybrid model has approximately the same number of parameters as its comparison partner, a GCN with four layers, thus ensuring that the comparison is fair.

References

[1] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint 2006.09252*, 2021.

[2] X. Bresson and T. Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[3] Z. Chen, L. Chen, S. Villar, and J. Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

[4] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

[5] A. Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[6] F. Hensel, M. Moor, and B. Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021. ISSN 2624-8212. doi: 10.3389/frai.2021.681108.

[7] C. D. Hofer, F. Graf, B. Rieck, M. Niethammer, and R. Kwitt. Graph filtration learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 4314—4323. PMLR, 2020.

[8] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[9] N. Milosavljević, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG)*, pages 216–225, 2011.

[10] M. Moor, M. Horn, B. Rieck, and K. Borgwardt. Topological autoencoders. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7045–7054. PMLR, 2020.

[11] N. Shervashidze and K. Borgwardt. Fast subtree kernels on graphs. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22, pages 1660–1668. Curran Associates Inc., 2009.

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

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[14] B. Weisfeiler and A. A. Lehman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno–Technicheskaja Informatsia*, 9:12–16, 1968.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

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

[17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, pages 3391–3401. Curran Associates, Inc., 2017.

[18] Q. Zhao and Y. Wang. Learning metrics for persistence-based summaries and applications for graph classification. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 9855–9866. Curran Associates, Inc., 2019.