HELMHOLTZ MUNICH

AIH Institute of AI for Health

Topological Machine Learning: The (W)Hole Truth

Lecture 5

Bastian Rieck (@Pseudomanifold)

Preliminaries

Do you have feedback or any questions? Write to bastian.rieck@helmholtz-muenchen.de or reach out to @Pseudomanifold on Twitter. You can find the slides and additional information with links to more literature here:

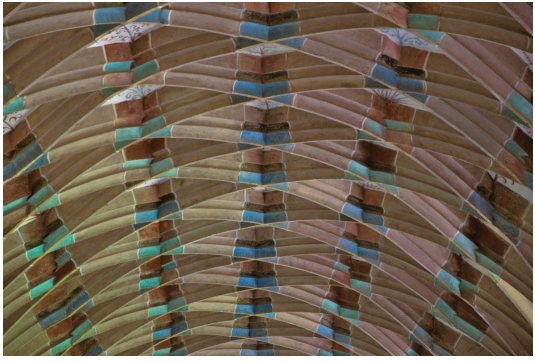
`https://heidelberg.topology.rocks`

Recap

- The *persistence diagram* is the 'basic' topological feature descriptor.
- Multiple alternatives exist, with different key properties.
- Their choice is application-dependent.

In this lecture

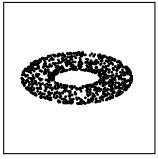
Putting everything together



How can we *build* topology-based machine learning models?

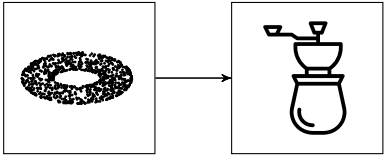
Topological machine learning

One pipeline to rule them all?

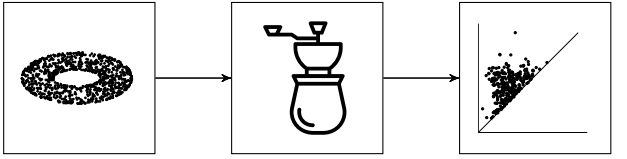


Point cloud

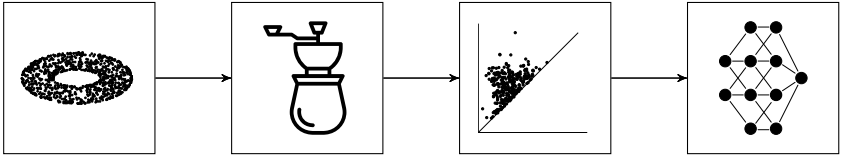
One pipeline to rule them all?



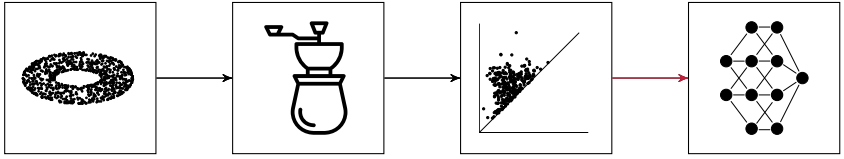
One pipeline to rule them all?



One pipeline to rule them all?



One pipeline to rule them all?



- A. Poulenard, P. Skraba and M. Ovsjanikov, 'Topological Function Optimization for Continuous Shape Matching', *Computer Graphics Forum* 37.5, 2018, pp. 13–25
- M. Moor*, M. Horn*, **B. Rieck**† and K. Borgwardt†, 'Topological Autoencoders', *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020, pp. 7045–7054, arXiv: `1906.00722 [cs.LG]`
- M. Carrière, F. Chazal, M. Glisse, Y. Ike, H. Kannan and Y. Umeda, 'Optimizing persistent homology based functions', *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021, pp. 1294–1303

Finding topology-based representations

Topological autoencoders

Topological Autoencoders

Michael Moor [†,1,2] Max Horn [†,1,2] Bastian Rieck [‡,1,2] Karsten Borgwardt [‡,1,2]

Abstract

We propose a novel approach for preserving topological structures of the input space in latent representations of autoencoders. Using *persistent homology*, a technique from topological data analysis, we calculate topological signatures of both the input and latent space to derive a topological loss term. Under weak theoretical assumptions, we construct this loss in a differentiable manner, such that the encoding learns to retain multi-scale connectivity information. We show that our approach is theoretically well-founded and that it exhibits favourable latent representations on a synthetic manifold as well as on real-world image data sets, while preserving low reconstruction errors.

1. Introduction

While topological features, in particular multi-scale features derived from persistent homology, have seen increasing use in the machine learning community (Carrière et al., 2019, Guss & Salakhutdinov, 2018, Hofer et al., 2017, 2019a,b, Ramamurthy et al., 2019, Reininghaus et al., 2015, Rieck et al., 2019a,b), employing topology *directly* as a constraint for modern deep learning methods remains a challenge. This is due to the inherently discrete nature of these computations, making backpropagation through the computation of topological signatures immensely difficult or only possible in certain special circumstances (Chen et al., 2019, Hofer et al., 2019a, Poulenard et al., 2018).

This work presents a novel approach that permits obtaining gradients during the computation of topological signatures. This makes it possible to employ topological constraints while training deep neural networks, as well as building topology-preserving autoencoders. Specifically, we make the following contributions:

1. We develop a new topological loss term for autoencoders that helps harmonise the topology of the data space with the topology of the latent space.
2. We prove that our approach is stable on the level of mini-batches, resulting in suitable approximations of the persistent homology of a data set.
3. We empirically demonstrate that our loss term aids in dimensionality reduction by preserving topological structures in data sets; in particular, the learned latent representations are useful in that the preservation of topological structures can improve interpretability.

2. Background: Persistent Homology

Persistent homology (Barannikov, 1994, Edelsbrunner & Harer, 2008) is a method from the field of computational topology, which develops tools for analysing topological features (connectivity-based features such as connected components) of data sets. We first introduce the underlying concept of simplicial homology. For a simplicial complex $\mathfrak{K}$, i.e. a generalised graph with higher-order connectivity information such as cliques, simplicial homology employs matrix reduction algorithms to assign $\mathfrak{K}$ a family of groups, the *homology groups*. The d^{th} homology group $\mathrm{H}_d(\mathfrak{K})$ of $\mathfrak{K}$ contains d-dimensional topological features, such as connected components ($d = 0$), cycles/tunnels ($d = 1$), and voids ($d = 2$). Homology groups are typically summarised by their ranks, thereby obtaining a simple invariant "signature" of a manifold. For example, a circle in $\mathbb{R}^2$ has one feature with $d = 1$ (a cycle), and one feature with $d = 0$ (a connected component).

In practice, the underlying manifold $\mathbb{M}$ is unknown and we are working with a point cloud $X := \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ and a metric $\mathrm{dist}\colon X \times X \to \mathbb{R}$ such as the Euclidean distance. Persistent homology extends simplicial homology to this setting: instead of approximating $\mathbb{M}$ by means of a *single* simplicial complex, which would be an unstable procedure due to the discrete nature of X, persistent homology tracks changes in the homology groups over *multiple* scales of the metric. This is achieved by constructing a special simplicial complex, the Vietoris–Rips complex (Vietoris, 1927). For $0 \leq \epsilon < \infty$, the Vietoris–Rips complex of X at scale ϵ, denoted by $\mathfrak{R}_\epsilon(X)$, contains all

[†]Equal contribution. [‡]These authors jointly directed this work. [1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland [2]SIB Swiss Institute of Bioinformatics, Switzerland. Correspondence to: Karsten Borgwardt <karsten.borgwardt@bsse.ethz.ch>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

Michael Moor

Michael_D_Moor

Max Horn

ExpectationMax

Karsten Borgwardt

kmborgwardt

M. Moor*, M. Horn*, **B. Rieck**† and K. Borgwardt†, 'Topological Autoencoders', *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020, pp. 7045–7054, arXiv: `1906.00722 [cs.LG]`

Topological autoencoders

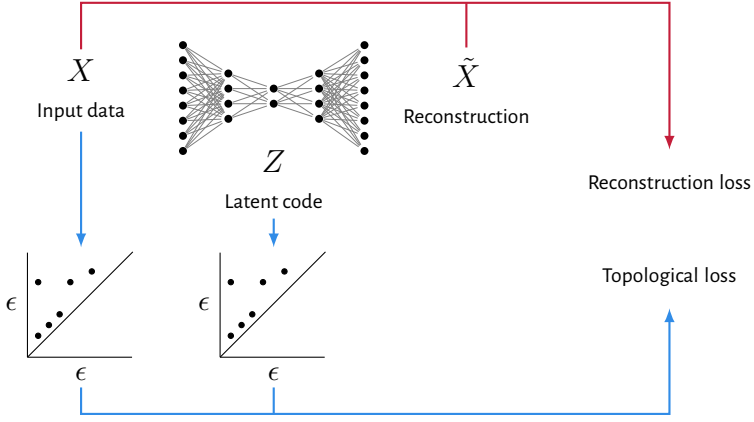
Motivation

Topological autoencoders

Motivation, continued

Topological autoencoders

Overview



Topological autoencoders

Main intuition

Align persistence diagrams of an *input batch* and of a *latent batch* using a loss function!

Why this works in theory

Let X be a point cloud of cardinality n and $X^{(m)}$ be one subsample of X of cardinality m, i.e. $X^{(m)} \subseteq X$, sampled without replacement. We can bound the probability of the persistence diagrams of $X^{(m)}$ exceeding a threshold in terms of the bottleneck distance as

$$\mathbb{P}\left(\mathsf{W}_\infty\left(\mathcal{D}^X, \mathcal{D}^{X^{(m)}}\right) > \epsilon\right) \leq \mathbb{P}\left(\mathrm{dist}_\mathsf{H}\left(X, X^{(m)}\right) > 2\epsilon\right),$$

where dist_H denotes the Hausdorff distance. In other words: *mini-batches are topologically similar if the subsampling is not too coarse*.

Topological autoencoders

Gradient calculation intuition

Distance matrix **A**

$$\begin{bmatrix} 0 & 1 & 9 & 10 \\ 1 & 0 & 7 & 8 \\ 9 & 7 & 0 & 3 \\ 10 & 8 & 3 & 0 \end{bmatrix}$$

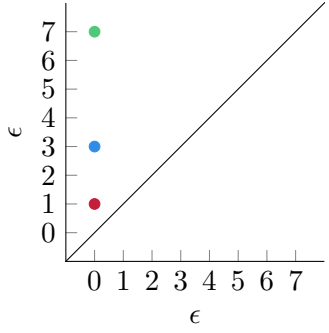
Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Gradient calculation intuition

Distance matrix **A**

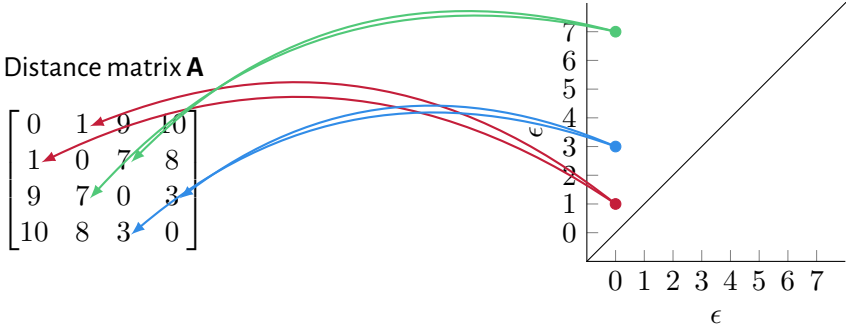
$$\begin{bmatrix} 0 & 1 & 9 & 10 \\ 1 & 0 & 7 & 8 \\ 9 & 7 & 0 & 3 \\ 10 & 8 & 3 & 0 \end{bmatrix}$$



Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Gradient calculation intuition



Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Loss term

$$\mathcal{L}_t := \mathcal{L}_{\mathcal{X}\to\mathcal{Z}} + \mathcal{L}_{\mathcal{Z}\to\mathcal{X}}$$

$$\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} := \tfrac{1}{2}\left\|\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right\|^2 \qquad \mathcal{L}_{\mathcal{Z}\to\mathcal{X}} := \tfrac{1}{2}\left\|\mathbf{A}^Z\left[\pi^Z\right] - \mathbf{A}^X\left[\pi^Z\right]\right\|^2$$

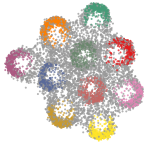
- $\mathcal{X}$: input space
- $\mathcal{Z}$: latent space
- $\mathbf{A}^X$: distances in input mini-batch
- $\mathbf{A}^Z$: distances in latent mini-batch
- π^X: persistence pairing of input mini-batch
- π^Z: persistence pairing of latent mini-batch

The loss is *bi-directional*!

Qualitative evaluation

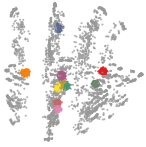
'Spheres' data set

PCA

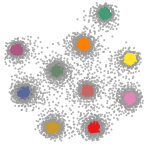


UMAP

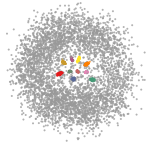
Autoencoder



Isomap



t-SNE



Topological autoencoder

Quantitative evaluation

Data set	Method	$KL_{0.01}$	$KL_{0.1}$	KL_1	ℓ-MRRE	ℓ-Cont	ℓ-Trust	ℓ-RMSE	MSE (data)
'Spheres'	Isomap	0.181	**0.420**	**0.00881**	**0.246**	**0.790**	**0.676**	10.4	
	PCA	0.332	0.651	0.01530	0.294	0.747	0.626	11.8	0.9610
	t-SNE	**0.152**	0.527	0.01271	**0.217**	0.773	**0.679**	**8.1**	
	UMAP	0.157	0.613	0.01658	0.250	0.752	0.635	**9.3**	
	AE	0.566	0.746	0.01664	0.349	0.607	0.588	13.3	**0.8155**
	TopoAE	**0.085**	**0.326**	**0.00694**	0.272	**0.822**	0.658	13.5	**0.8681**
'Fashion-MNIST'	PCA	**0.356**	**0.052**	**0.00069**	0.057	0.968	0.917	**9.1**	0.1844
	t-SNE	0.405	0.071	0.00198	**0.020**	0.967	**0.974**	41.3	
	UMAP	0.424	0.065	0.00163	0.029	**0.981**	0.959	**13.7**	
	AE	0.478	0.068	0.00125	**0.026**	0.968	**0.974**	20.7	**0.1020**
	TopoAE	**0.392**	**0.054**	**0.00100**	0.032	**0.980**	0.956	20.5	**0.1207**
'MNIST'	PCA	0.389	0.163	0.00160	0.166	0.901	0.745	**13.2**	0.2227
	t-SNE	**0.277**	**0.133**	0.00214	**0.040**	0.921	**0.946**	22.9	
	UMAP	**0.321**	0.146	0.00234	**0.051**	**0.940**	**0.938**	**14.6**	
	AE	0.620	0.155	**0.00156**	0.058	0.913	0.937	18.2	**0.1373**
	TopoAE	0.341	**0.110**	**0.00114**	0.056	**0.932**	0.928	19.6	**0.1388**

Topology-driven graph learning

Classifying *unlabelled* graphs

Using 'classical' machine learning models

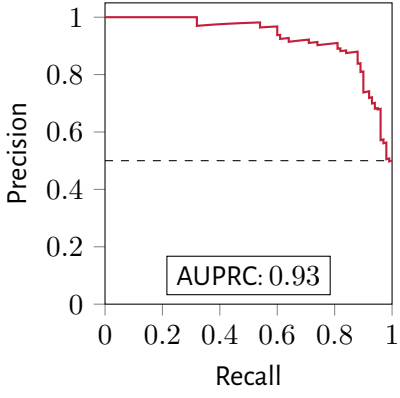
1. Calculate degree filtration (or another descriptor)
2. Repeat the analysis pipeline described above
3. Learn weights for topological descriptors to improve predictive power[1]

[1] Q. Zhao and Y. Wang, 'Learning metrics for persistence-based summaries and applications for graph classification', *Advances in Neural Information Processing Systems*, ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, vol. 32, Curran Associates, Inc., 2019, pp. 9855–9866

Betti curves

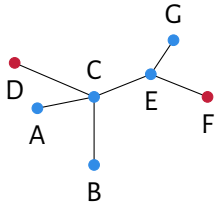
Classification scenario example

- ☆ Use `REDDIT-BINARY` data set (co-occurrence graphs)
- ☆ Calculate filtration based on *vertex degree*
- ☆ Calculate persistence diagrams for $d = 1$ (cycles)
- ☆ Given $p = 1$, use a kernel SVM for classification



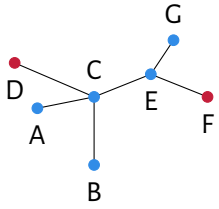
Classifying *labelled* graphs

Weisfeiler–Lehman iteration & subtree feature vector



Classifying *labelled* graphs

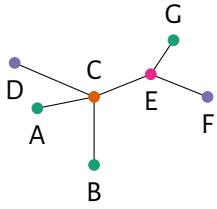
Weisfeiler–Lehman iteration & subtree feature vector



Node	Own label	Adjacent labels
A	●	●
B	●	●
C	●	● ● ● ●
D	●	●
E	●	● ● ●
F	●	●
G	●	●

Classifying *labelled* graphs

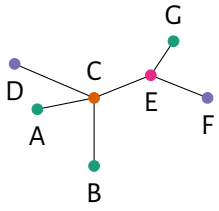
Weisfeiler–Lehman iteration & subtree feature vector



Node	Own label	Adjacent labels	Hashed label
A	●	●	●
B	●	●	●
C	●	● ● ● ●	●
D	●	●	●
E	●	● ● ●	●
F	●	●	●
G	●	●	●

Classifying *labelled* graphs

Weisfeiler–Lehman iteration & subtree feature vector



Label	●	●	●	●
Count	3	1	2	1

$$\Phi(\mathcal{G}) := (3, 1, 2, 1)$$

Compare $\mathcal{G}$ and $\mathcal{G}'$ by evaluating a kernel between $\Phi(\mathcal{G})$ and $\Phi(\mathcal{G}')$ (linear, RBF, ...).

Static topological features for graph classification

A Persistent Weisfeiler–Lehman Procedure for Graph Classification

Bastian Rieck [*1] Christian Bock [*1] Karsten Borgwardt [1]

Abstract

The Weisfeiler–Lehman graph kernel exhibits competitive performance in many graph classification tasks. However, its subtree features are not able to capture connected components and cycles, topological features known for characterising graphs. To extract such features, we leverage propagated node label information and transform unweighted graphs into metric ones. This permits us to augment the subtree features with topological information obtained using *persistent homology*, a concept from topological data analysis. Our method, which we formalise as a generalisation of Weisfeiler–Lehman subtree features, exhibits favourable classification accuracy and its improvements in predictive performance are mainly driven by including cycle information.

1. Introduction

Graph-structured data sets are ubiquitous in a variety of different application domains, each of them posing a separate challenge while also requiring different tasks to be solved. A common task involves *graph classification*, for which a variety of methods exists. These methods comprise convolutional neural networks (Duvenaud et al. 2015), recurrent neural networks (Lei et al. 2017), or Hilbert space methods (Vishwanathan et al. 2010), the latter also being referred to as *graph kernels*. While several approaches for defining graph kernels exist, the most common one uses the $\mathcal{R}$-convolution framework (Haussler 1999), which makes it possible to define the similarity between two graphs as a function of the similarity of their substructures.

Substructures that have been used for graph classification range from graphlets (Shervashidze et al. 2009), i.e. small non-isomorphic graphs of fixed size, over shortest paths (Borgwardt & Kriegel 2005), to random walks (Gärtner et al. 2003, Kashima et al. 2003, Sugiyama & Borgwardt 2015). One of the most powerful substructures is the set of *subtree patterns* (Ramon & Gärtner 2003), i.e. patterns based on rooted subgraphs of a graph. Their computational complexity made their applicability somewhat limited, until the Weisfeiler–Lehman (WL) graph kernel framework (Shervashidze & Borgwardt 2009, Shervashidze et al. 2011) was developed. Properly trained, it still constitutes the state-of-the-art method for many graph classification tasks. The framework is based on the idea of iteratively propagating (node) label information through a graph, leading to a feature vector representation that can be used to assess the dissimilarity of two graphs.

One of the disadvantages of this framework is that its relabelling step, i.e. the step in which subtree patterns are being *compressed*, is somewhat "brittle": labels are only compared with a Dirac kernel, making their dissimilarity a coarse function. Moreover, the subtree feature vector only contains counts of compressed labels and can neither account for their relevance with respect to the topology of the graph nor capture connected components and cycles, both of which are important and interpretable features for characterising graphs (Rieck et al. 2018, Sizemore et al. 2017). We thus propose an enhancement of the original WL stabilisation procedure that uses recent advances in topological data analysis (Munch 2017) to alleviate these issues. Our contributions are as follows:

- We measure the relevance of topological features (connected components and cycles) in graphs and use them to define a novel set of WL subtree features, which we show to be a generalised version of the original ones.
- We develop a topology-based kernel that uses an iterative variant of the WL stabilisation procedure to classify non-attributed graphs.
- We demonstrate that our proposed features perform favourably on a range of graph classification benchmark data sets. In particular, we empirically show that the inclusion of *cycle* information yields classification accuracy improvements over state-of-the-art methods.

[*]Equal contribution [1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland. Correspondence to: Bastian Rieck <bastian.rieck@bsse.ethz.ch>, Karsten Borgwardt <karsten.borgwardt@bsse.ethz.ch>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Christian Bock
chrs_bock

Karsten Borgwardt
kmborgwardt

- The Weisfeiler–Lehman algorithm vectorises labelled graphs
- Persistent homology captures relevant topological features
- We can *combine* them to obtain a *generalised* formulation
- This requires a distance between multisets

B. Rieck*, C. Bock* and K. Borgwardt, 'A Persistent Weisfeiler–Lehman Procedure for Graph Classification', *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ed. by K. Chaudhuri and R. Salakhutdinov, Proceedings of Machine Learning Research 97, PMLR, 2019, pp. 5448–5458

A distance between label multisets

Let $A = \{l_1^{a_1}, l_2^{a_2}, \dots\}$ and $B = \{l_1^{b_1}, l_2^{b_2}, \dots\}$ be two multisets that are defined over the same label alphabet $\Sigma = \{l_1, l_2, \dots\}$.

Transform the sets into count vectors, i.e. $\vec{x} := [a_1, a_2, \dots]$ and $\vec{y} := [b_1, b_2, \dots]$.

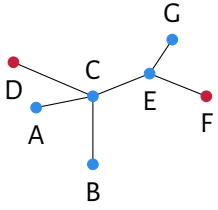
Calculate their *multiset distance* as

$$\text{dist}(\vec{x}, \vec{y}) := \left(\sum_i |a_i - b_i|^p \right)^{\frac{1}{p}},$$

i.e. the p^{th} Minkowski distance, for $p \in \mathbb{R}$. Since nodes and their multisets are in one-to-one correspondence, we now have a metric on the graph!

Multiset distance

Example for $p = 1$



$$\begin{aligned}\mathrm{dist}(C, E) &= \mathrm{dist}\big(\{\bullet^3, \bullet^1\}, \{\bullet^2, \bullet^1\}\big)\\ &= \mathrm{dist}([3,1],[2,1])\\ &= 1\end{aligned}$$

$$\begin{aligned}\mathrm{dist}(C, A) &= \mathrm{dist}\big(\{\bullet^3, \bullet^1\}, \{\bullet^1\}\big)\\ &= \mathrm{dist}([3,1],[1,0])\\ &= 3\end{aligned}$$

Extending the multiset distance to a distance between vertices

Use vertex label from *previous* Weisfeiler–Lehman iteration, i.e. $\mathsf{l}_{v_i}^{(h-1)}$, as well as $\mathsf{l}_{v_i}^{(h)}$, the one from the *current* iteration:

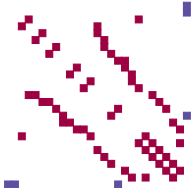
$$\mathsf{dist}(v_i, v_j) := \left[\mathsf{l}_{v_i}^{(h-1)} \neq \mathsf{l}_{v_j}^{(h-1)}\right] + \mathsf{dist}\left(\mathsf{l}_{v_i}^{(h)}, \mathsf{l}_{v_j}^{(h)}\right) + \tau$$

$\tau \in \mathbb{R}_{>0}$ is required to make this into a proper metric. This turns *any* labelled graph into a weighted graph whose persistent homology we can calculate!

Vertex distance, multi-scale properties

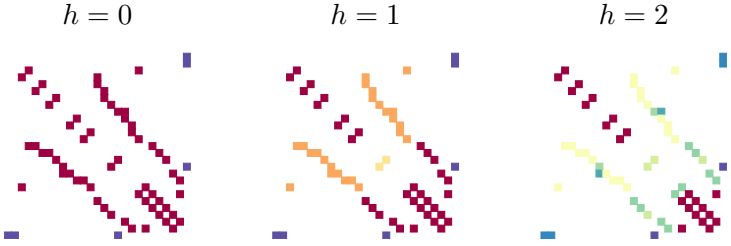
Example

$h = 0$



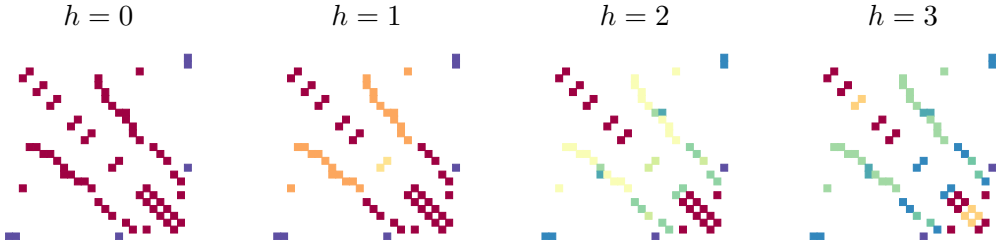
Vertex distance, multi-scale properties

Example



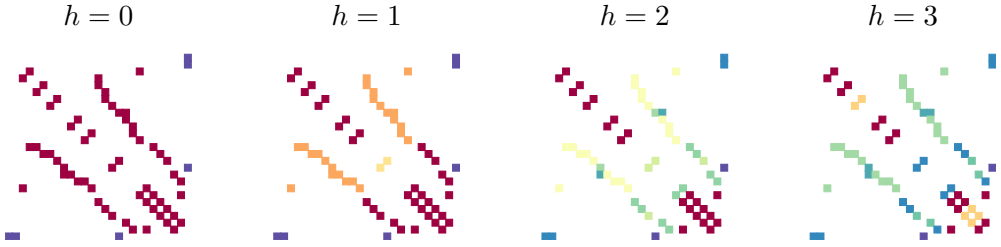
Vertex distance, multi-scale properties

Example



Vertex distance, multi-scale properties

Example



Persistence-based Weisfeiler–Lehman feature vectors

Connected components

$$\Phi^{(h)}_{\text{P-WL}} := \left[\mathfrak{p}^{(h)}(l_0), \mathfrak{p}^{(h)}(l_1), \dots\right]$$
$$\mathfrak{p}^{(h)}(l_i) := \sum_{\mathsf{l}(v)=l_i} \mathsf{pers}(v)^p,$$

Cycles

$$\Phi^{(h)}_{\text{P-WL-C}} := \left[\mathfrak{z}^{(h)}(l_0), \mathfrak{z}^{(h)}(l_1), \dots\right]$$
$$\mathfrak{z}^{(h)}(l_i) := \sum_{l_i \in \mathsf{l}(u,v)} \mathsf{pers}(u,v)^p,$$

Persistence-based Weisfeiler–Lehman feature vectors

Connected components

$$\Phi^{(h)}_{\text{P-WL}} := \left[\mathfrak{p}^{(h)}(l_0), \mathfrak{p}^{(h)}(l_1), \dots\right]$$

$$\mathfrak{p}^{(h)}(l_i) := \sum_{\mathsf{l}(v)=l_i} \mathsf{pers}(v)^p,$$

Cycles

$$\Phi^{(h)}_{\text{P-WL-C}} := \left[\mathfrak{z}^{(h)}(l_0), \mathfrak{z}^{(h)}(l_1), \dots\right]$$

$$\mathfrak{z}^{(h)}(l_i) := \sum_{l_i \in \mathsf{l}(u,v)} \mathsf{pers}(u,v)^p,$$

Bonus

We can re-define the vertex distance to obtain the original Weisfeiler–Lehman subtree features (plus information about cycles):

$$\mathsf{dist}(v_i, v_j) := \begin{cases} 1 & \text{if } v_i \neq v_j \\ 0 & \text{otherwise} \end{cases}$$

Classification results

	D & D	MUTAG	NCI1	NCI109	PROTEINS	PTC-MR	PTC-FR	PTC-MM	PTC-FM
V-Hist	78.32 ± 0.35	85.96 ± 0.27	64.40 ± 0.07	63.25 ± 0.12	72.33 ± 0.32	58.31 ± 0.27	68.13 ± 0.23	66.96 ± 0.51	57.91 ± 0.83
E-Hist	72.90 ± 0.48	85.69 ± 0.46	63.66 ± 0.11	63.27 ± 0.07	72.14 ± 0.39	55.82	65.53	61.61	59.03
RetGK*	81.60 ± 0.30	90.30 ± 1.10	84.50 ± 0.20		75.80 ± 0.60	62.15 ± 1.60	67.80 ± 1.10	67.90 ± 1.40	63.90 ± 1.30
WL	79.45 ± 0.38	87.26 ± 1.42	85.58 ± 0.15	84.85 ± 0.19	76.11 ± 0.64	63.12 ± 1.44	67.64 ± 0.74	67.28 ± 0.97	64.80 ± 0.85
Deep-WL*		82.94 ± 2.68	80.31 ± 0.46	80.32 ± 0.33	75.68 ± 0.54	60.08 ± 2.55			
P-WL	79.34 ± 0.46	86.10 ± 1.37	85.34 ± 0.14	84.78 ± 0.15	75.31 ± 0.73	63.07 ± 1.68	67.30 ± 1.50	68.40 ± 1.17	64.47 ± 1.84
P-WL-C	78.66 ± 0.32	90.51 ± 1.34	85.46 ± 0.16	84.96 ± 0.34	75.27 ± 0.38	64.02 ± 0.82	67.15 ± 1.09	68.57 ± 1.76	65.78 ± 1.22
P-WL-UC	78.50 ± 0.41	85.17 ± 0.29	85.62 ± 0.27	85.11 ± 0.30	75.86 ± 0.78	63.46 ± 1.58	67.02 ± 1.29	68.01 ± 1.04	65.44 ± 1.18

Graph representations

Fundamental properties

- Two graphs $\mathcal{G}$ and $\mathcal{G}'$ can have a *different* number of vertices.
- Hence, we require a *vectorised representation* $f : \mathcal{G} \to \mathbb{R}^d$ of graphs.
- Such a representation f needs to be *permutation-invariant*.

Now and then

Shallow approaches

- node2vec (encoder–decoder)
- Graph kernels (RKHS feature maps)
- Laplacian-based embeddings

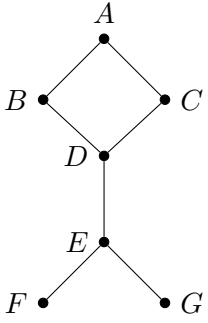
Deep approaches

- Graph convolutional networks
- Graph isomorphism networks
- Graph attention networks

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

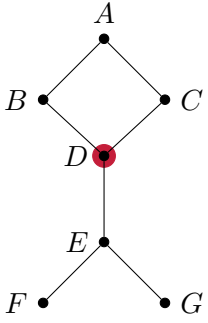


- ☆ Operations remain local.
- ☆ Only require some aggregation function.
- ☆ Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

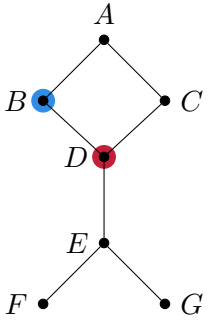


- ☆ Operations remain local.
- ☆ Only require some aggregation function.
- ☆ Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

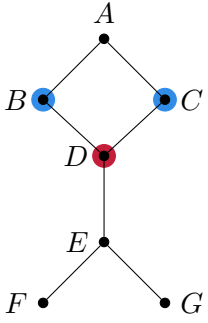


- ☆ Operations remain local.
- ☆ Only require some aggregation function.
- ☆ Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

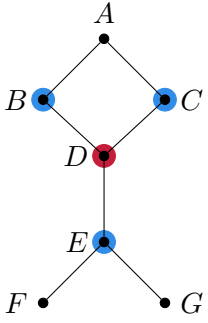


- ☆ Operations remain local.
- ☆ Only require some aggregation function.
- ☆ Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

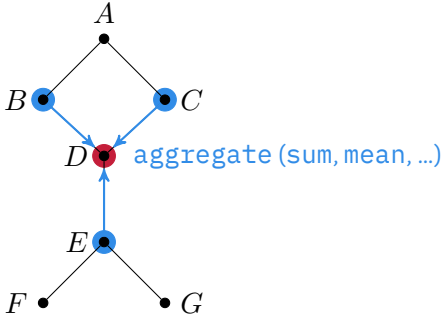


- ☆ Operations remain local.
- ☆ Only require some aggregation function.
- ☆ Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.



- Operations remain local.
- Only require some aggregation function.
- Representations can be combined.

Graph neural networks in a nutshell

$$
\begin{aligned}
a_v^{(k)} &:= \texttt{aggregate}^{(k)}\left(\left\{h_u^{(k-1)} \mid u \in \mathcal{N}_{\mathcal{G}}(v)\right\}\right) \\
h_v^{(k)} &:= \texttt{combine}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right) \\
h_{\mathcal{G}} &:= \texttt{readout}\left(\left\{h_v^{(K)} \mid v \in \mathsf{vert}_{\mathcal{G}}\right\}\right)
\end{aligned}
$$

This terminology follows K. Xu, W. Hu, J. Leskovec and S. Jegelka, 'How Powerful are Graph Neural Networks?', *ICLR*, 2019.

Example

Graph convolutional networks

For this architecture, `combine` is directly integrated into `aggregate`. In *matrix form*, we have

$$\mathbf{H}^{(\mathbf{k})} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{\mathbf{1}}{\mathbf{2}}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{\mathbf{1}}{\mathbf{2}}}\mathbf{H}^{(\mathbf{k}-\mathbf{1})}\Theta^{(k-1)}\right),$$

with $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{I}$ being the *augmented adjacency matrix*, $\tilde{\mathbf{D}}$ its degree matrix, and $\Theta^{(k-1)}$ a *learnable weight matrix* of dimensions $d^{k-1} \times d^k$.

A topological layer for graph classification

Published as a conference paper at ICLR 2022

TOPOLOGICAL GRAPH NEURAL NETWORKS

Max Horn[1,2,*] **Edward De Brouwer**[3,*] **Michael Moor**[1,2] **Yves Moreau**[3]
Bastian Rieck[1,2,4,5,†] **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]Institute of AI for Health, Helmholtz Munich, 85764 Neuherberg, Germany
[5]Technical University of Munich, 80333 Munich, 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 the Weisfeiler–Lehman graph isomorphism test) than message-passing GNNs. Augmenting GNNs with TOGL 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 representation 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 (Wu et al., 2021). At their core, many GNNs are based on iterative message passing schemes (see Shervashidze and Borgwardt (2009) for an introduction to iterative message passing in graphs and Sanchez-Lengeling et al. (2021) for an introduction to GNNs). Since these schemes are collating information over the neighbours of every node, GNNs cannot necessarily capture certain topological structures in graphs, such as cycles (Bouritsas et al., 2021). These structures are highly relevant for applications that require connectivity information, such as the analysis of molecular graphs (Hofer et al., 2020; Swenson et al., 2020).

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 emerging field of topological data analysis (TDA), which focuses on describing coarse structures that can be used to study the shape of complex structured and unstructured data sets at multiple scales. 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.

Our contributions. We propose TOGL, a novel layer based on TDA concepts that can be integrated into any GNN. Our layer is differentiable and capable of learning contrasting topological representations of a graph. We prove that TOGL enhances expressivity of a GNN since it incorporates the ability to work with multi-scale topological information in a graph. Moreover, we show that TOGL improves predictive performance of several GNN architectures when topological information is relevant for the respective task.

1

Max Horn
@ExpectationMax

Edward De Brouwer
@EdwardOnBrew

Michael Moor
@Michael_D_Moor

Yves Moreau

Karsten Borgwardt
@kmborgwardt

M. Horn*, E. De Brouwer*, M. Moor, Y. Moreau, **B. Rieck**† and K. Borgwardt†, 'Topological Graph Neural Networks', *ICLR*, 2022

Motivation

Status quo

- Graphs are topological objects.
- But GNNs are *incapable* of recognising certain topological structures!

Challenge

What can we gain when imbuing them with knowledge about the topology?

Taking stock

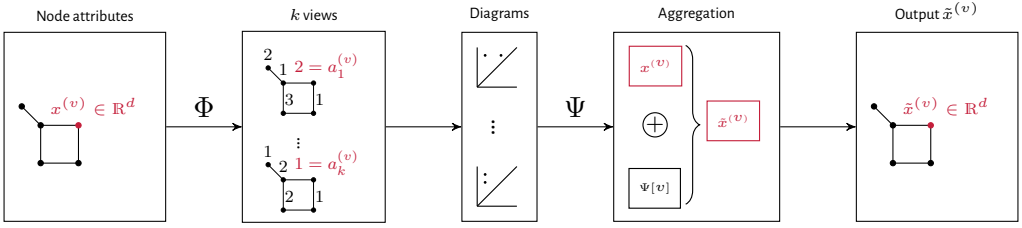
- Filtrations provide multi-scale topological features.
- Persistence diagrams serve as topological descriptors.

Questions

- How to obtain 'good' filtrations?
- How to use persistence diagrams (i.e. multi-sets) in a differentiable setting?

Topological graph neural networks

Overview



- Use a node map $\Phi\colon \mathbb{R}^d \to \mathbb{R}^k$ to create k different filtrations of the graph.
- Use a coordinatisation function Ψ to create *compatible* representations of the node attributes.

Choosing Φ and Ψ

- The node map Φ can be realised using a *neural network*.
- The coordinatisation function Ψ can be realised using *any* vectorisation of persistence diagrams (landscapes, images, …), but we found a *differentiable coordinatisation function* to be most effective.[2]

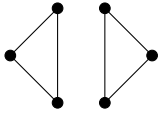
[2] C. D. Hofer, F. Graf, **B. Rieck**, M. Niethammer and R. Kwitt, 'Graph Filtration Learning', *ICML*, 2020.

Expressivity of TOGL

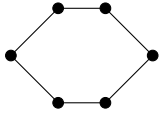
Theorem

*TOGL (and persistent homology) is **more expressive** than WL[1], i.e. (i) if the WL[1] label sequences for two graphs G and G' diverge, there exists an injective filtration f such that the corresponding persistence diagrams $\mathcal{D}_0$ and $\mathcal{D}'_0$ are not equal, and (ii) there are graphs that WL[1] cannot distinguish but TOGL can!*

Example graphs



$\mathcal{G}$



$\mathcal{G}'$

Experiments

- Take existing GNN architecture.
- Replace one layer by TOGL.
- Measure predictive performance.

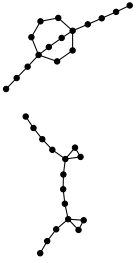
This strategy ensures that the number of parameters is approximately the same, thus facilitating a fair comparison!

Synthetic data sets

Binary classification problem; generate same number of graphs for each of the classes. Use simple topological structures that are nevertheless challenging to detect with standard GNNs.

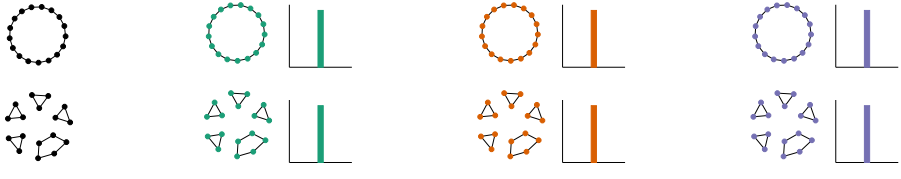
Cycles

Necklaces



`Cycles` data set

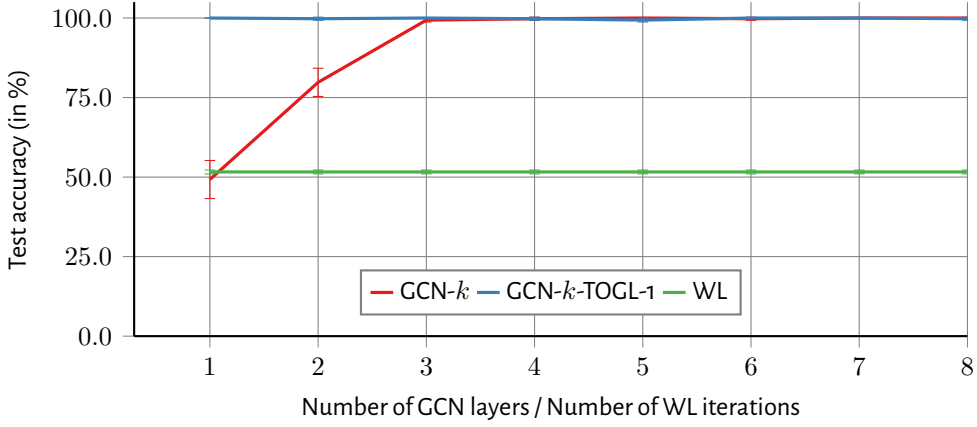
Weisfeiler–Lehman subtree features



These graphs cannot be distinguished based on their WL[1] information.

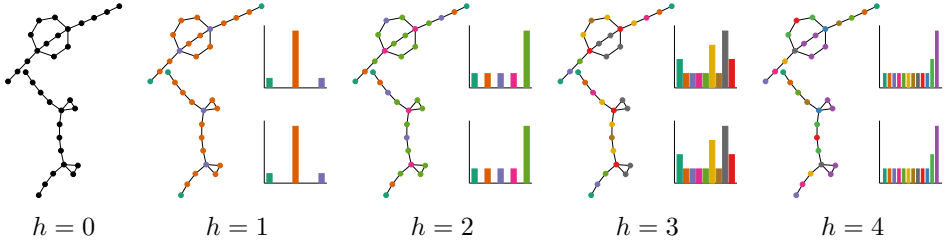
Expressivity

`Cycles` data set



Necklaces data set

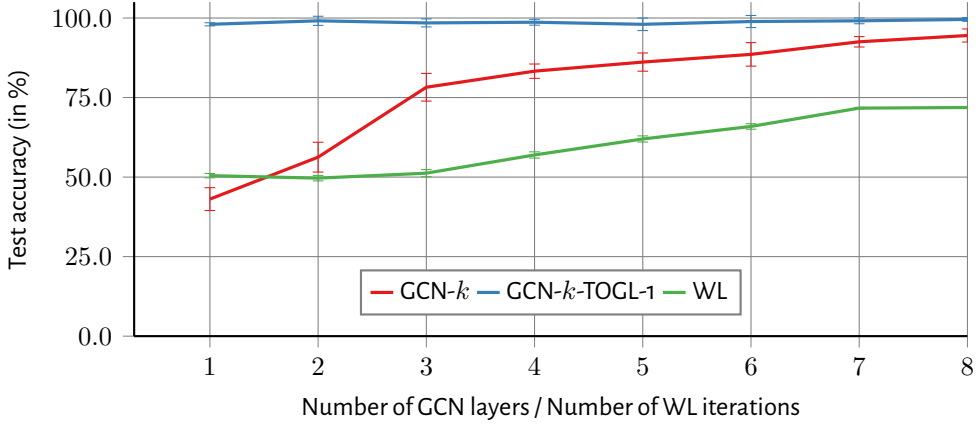
Weisfeiler–Lehman subtree features



These graphs cannot be distinguished based on their WL[1] information (some of the graphs in the data set *can* be distinguished, though).

Expressivity

`Necklaces` data set



Classifying graphs/nodes based on structural features alone

Existing data sets tend to 'leak' information into node attributes, thus decreasing the utility of topological features. Hence, we replaced all node features by random ones.

	Graph classification				*Node classification*
METHOD	DD	ENZYMES	MNIST	PROTEINS	Pattern
GCN-4	68.0 ± 3.6	22.0 ± 3.3	76.2 ± 0.5	68.8 ± 2.8	85.5 ± 0.4
GCN-3-TOGL-1	$\mathbf{75.1 \pm 2.1}$	$\mathbf{30.3 \pm 6.5}$	$\mathbf{84.8 \pm 0.4}$	$\mathbf{73.8 \pm 4.3}$	$\mathbf{86.6 \pm 0.1}$
GIN-4	75.6 ± 2.8	21.3 ± 6.5	83.4 ± 0.9	$\mathbf{74.6 \pm 3.1}$	84.8 ± 0.0
GIN-3-TOGL-1	$\mathbf{76.2 \pm 2.4}$	$\mathbf{23.7 \pm 6.9}$	$\mathbf{84.4 \pm 1.1}$	73.9 ± 4.9	$\mathbf{86.7 \pm 0.1}$
GAT-4	63.3 ± 3.7	21.7 ± 2.9	63.2 ± 10.4	67.5 ± 2.6	$\mathbf{73.1 \pm 1.9}$
GAT-3-TOGL-1	$\mathbf{75.7 \pm 2.1}$	$\mathbf{23.5 \pm 6.1}$	$\mathbf{77.2 \pm 10.5}$	$\mathbf{72.4 \pm 4.6}$	59.6 ± 3.3

Classifying benchmark data sets

While we improve baseline classification performance, the best performance is *not* driven by the availability of topological structures!

	Graph classification							*Node classification*
Method	CIFAR-10	DD	ENZYMES	MNIST	PROTEINS-full	IMDB-B	REDDIT-B	CLUSTER
GATED-GCN-4	**67.3 ± 0.3**	72.9 ± 2.1	65.7 ± 4.9	**97.3 ± 0.1**	**76.4 ± 2.9**	—	—	**60.4 ± 0.4**
WL	—	77.7 ± 2.0	54.3 ± 0.9	—	73.1 ± 0.5	71.2 ± 0.5	78.0 ± 0.6	—
WL-OA	—	**77.8 ± 1.2**	58.9 ± 0.9	—	73.5 ± 0.9	74.0 ± 0.7	87.6 ± 0.3	—
GCN-4	54.2 ± 1.5	72.8 ± 4.1	**65.8 ± 4.6**	90.0 ± 0.3	*76.1 ± 2.4*	68.6 ± 4.9	**92.8 ± 1.7**	57.0 ± 0.9
GCN-3-TOGL-1	*61.7 ± 1.0*	*73.2 ± 4.7*	53.0 ± 9.2	95.5 ± 0.2	76.0 ± 3.9	*72.0 ± 2.3*	89.4 ± 2.2	*60.4 ± 0.2*
	7.5	0.4	−12.8	5.5	−0.1	3.4	−3.4	3.4
GIN-4	54.8 ± 1.4	70.8 ± 3.8	*50.0 ± 12.3*	*96.1 ± 0.3*	72.3 ± 3.3	72.8 ± 2.5	81.7 ± 6.9	58.5 ± 0.1
GIN-3-TOGL-1	*61.3 ± 0.4*	*75.2 ± 4.2*	43.8 ± 7.9	96.1 ± 0.1	*73.6 ± 4.8*	**74.2 ± 4.2**	89.7 ± 2.5	*60.4 ± 0.2*
	6.5	4.4	−6.2	0.0	1.3	1.4	8.0	1.9
GAT-4	57.4 ± 0.6	71.1 ± 3.1	26.8 ± 4.1	94.1 ± 0.3	71.3 ± 5.4	*73.2 ± 4.1*	44.2 ± 6.6	56.6 ± 0.4
GAT-3-TOGL-1	*63.9 ± 1.2*	*73.7 ± 2.9*	*51.5 ± 7.3*	*95.9 ± 0.3*	*75.2 ± 3.9*	70.8 ± 8.0	*89.5 ± 8.7*	*58.4 ± 3.7*
	6.5	2.6	24.7	1.8	3.9	−2.4	45.3	1.8

Conclusion

- ‘If all you have is nails, everything looks like a hammer.’[3] Our data sets may actually *stymie* progress in GNN research because their classification does not necessarily require structural information.
- Nevertheless, higher-order structures (such as cliques) can be crucial in discerning between different graphs or data sets.
- Can we also learn sparse filtrations?

Acknowledgements

My co-authors Edward, Karsten, Max, Michael, and Yves.

Software

`https://github.com/aidos-lab/pytorch-topological`
Looking for additional contributors!

[3] Credit: Mikael Vejdemo-Johannson