WEISFEILER AND LEHMAN GO TOPOLOGICAL: MESSAGE PASSING SIMPLICIAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The pairwise interaction paradigm of graph machine learning has predominantly governed the modelling of relational systems. However, graphs alone cannot capture the multi-level interactions present in many complex systems and the expressive power of such schemes was proven to be limited. To overcome these limitations, we propose Message Passing Simplicial Networks (MPSNs), a class of models that perform message passing on simplicial complexes (SCs) – topological objects generalising graphs to higher dimensions. To theoretically analyse the expressivity of our model we introduce a Simplicial Weisfeiler-Lehman (SWL) colouring procedure for distinguishing non-isomorphic SCs. We relate the power of SWL to the problem of distinguishing non-isomorphic graphs and show that SWL and MPSNs are strictly more powerful than the WL test and not less powerful than the 3-WL test. We deepen the analysis by comparing our model with traditional graph neural networks with ReLU activations in terms of the number of linear regions of the functions they can represent. We empirically support our theoretical claims by showing that MPSNs can distinguish challenging strongly regular graphs for which GNNs fail and, when equipped with orientation equivariant layers, they can improve classification accuracy in oriented SCs compared to a GNN baseline. Additionally, we implement a library for neural message passing on simplicial complexes that we envision to release in due course.

1 INTRODUCTION

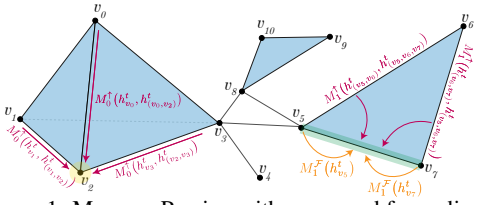

Figure 1: Message Passing with upper and face adjacencies illustrated for vertex v_2 and edge (v_5, v_7).

Graph neural networks (GNNs) have been well developed for learning features of graphs, which are one of the most common abstractions for complex systems (Bronstein et al., 2017). However, GNNs are limited in their capability of capturing higher-order interactions such as triangles or cliques (Chen et al., 2020) that are usually prominent in many natural graphs such as organic molecules (Bouritsas et al., 2020) (e.g. aromatic rings) or social networks (Milo et al., 2002). This paper tries to enhance GNN expressivity by considering local higher-order interactions. Among many modelling frameworks that have been proposed to describe complex systems with higher-order relations (Battiston et al., 2020), we specifically focus on simplicial complexes, a convenient middle ground between graphs (which are a particular case of a simplicial complex) and more general hypergraphs. Importantly, they offer strong mathematical connections to algebraic and differential topology and geometry. The simplicial Hodge Laplacian (Barbarossa & Sardellitti, 2020; Schaub et al., 2020), a discrete counterpart of the Laplacian operator in Hodge–de Rham theory (Rosenberg, 1997), provides a connection with the theory of spectral analysis and signal processing on these higher-dimensional domains.

We start by deriving from first principles a Simplicial Weisfeiler-Lehman (SWL) test for distinguishing non-isomorphic simplicial complexes. Motivated by this theoretical construction, we propose Message Passing Simplicial Networks (MPSNs), a message passing neural architecture for simplicial complexes that extends previous approaches such as GNNs and spectral simplicial convolutions (Bunch et al., 2020; Ebli et al., 2020). We then show that the proposed MPSN is as powerful as SWL. Strictly better than the conventional WL test (Weisfeiler & Leman, 1968), MPSN can be used to distinguish non-isomorphic graphs. We also show that the SWL test and MPSN are at least as powerful as the 3-WL test (Cai et al., 1992; Morris et al., 1992). Moreover, we explore the expressive

power of GNNs and MPSNs in terms of the number of linear regions of the functions they can represent (Pascanu et al., 2013; Montúfar et al., 2014). We obtain bounds for the maximal number of linear regions of MPSNs and show a higher functional complexity than GNNs and simplicial convolutional neural networks (SCNNs) (Ebli et al., 2020), for which we provide optimal bounds that might be of independent interest. Proofs and the required background are presented in the Appendix.

We will also be releasing a library based on PyTorch Geometric (Fey & Lenssen, 2019) for neural message passing on simplicial complexes. The library can handle SC datasets, simplicial message passing networks, oriented SCs, (higher-order) batching, clique complexes and many other features.

2 Message Passing Simplicial Networks

Simplicial WL. We develop a simplicial version of the WL test with the ultimate goal of deriving a message passing procedure that can retain the expressive power of the test. We call this simplicial colouring algorithm *Simplicial WL* (SWL). Given a complex $\mathcal{K}$, all the simplices $\sigma \in \mathcal{K}$ are initialised with the same colour. Given the colour c^t_σ of simplex σ at iteration t, we compute the colour of simplex σ at the next iteration c^{t+1}_σ, by (bijectively) hashing the multi-sets of colours belonging to the adjacent simplices of σ. The algorithm stops after a finite number of steps or when the colours are no longer updated. Two simplicial complexes are considered non-isomorphic if the colour histograms at any level of the complex are different.

A crucial choice has to be made about what simplices are considered to be adjacent. Unlike graphs, simplicial complexes contain many types of adjacencies: lower adjacencies (i.e. sharing of an immediate face), upper adjacencies (i.e. sharing of an immediate coface), boundary relationships (e.g. a triangle that is the face of a tetrahedron) and the opposite co-boundary relationships. To circumvent making arbitrary decisions, we start with an SWL test that includes all these possible choices. We use the following notation: $c^t_{\mathcal{F}}(\sigma) = \{\!\{c^t_\omega | \omega \in \mathcal{F}(\sigma)\}\!\}$ for the colours of the faces of σ, $c^t_{\mathcal{C}}(\sigma) = \{\!\{c^t_\omega | \omega \in \mathcal{C}(\sigma)\}\!\}$ for the colour of the cofaces, $c^t_\downarrow(\sigma) = \{\!\{(c^t_\omega, c^t_{\sigma\cap\omega}) | \omega \in \mathcal{N}_\downarrow(\sigma)\}\!\}$ for the colour of its down adjacent neighbours and the face $\sigma \cap \omega$ they share, and finally, $c^t_\uparrow(\sigma) = \{\!\{(c^t_\omega, c^t_{\sigma\cup\omega}) | \omega \in \mathcal{N}_\uparrow(\sigma)\}\!\}$ for the colours of the upper adjacent neighbours and the coface $\sigma \cup \omega$ they share. We then obtain the following update rule, which contains the complete set of adjacencies: $c^{t+1}_\sigma = \mathrm{HASH}\{c^t_\sigma, c^t_{\mathcal{F}}(\sigma), c^t_{\mathcal{C}}(\sigma), c^t_\downarrow(\sigma), c^t_\uparrow(\sigma)\}$. Starting from this complete set, we will now show that certain adjacencies can be removed without sacrificing the expressive power of the test (in terms of simplicial complexes that can be distinguished).

Theorem 1. *SWL with* $c^{t+1}_v = \mathrm{HASH}\big(c^t_v, c^t_{\mathcal{F}}(v), c^t_\uparrow(v)\big)$ *is as powerful as SWL with the generalised update rule* $\mathrm{HASH}\big(c^t_v, c^t_{\mathcal{F}}(v), c^t_{\mathcal{C}}(v), c^t_\downarrow(v), c^t_\uparrow(v)\big)$.

We note that other possible combinations of adjacencies might also fully preserve the expressive power of the general SWL test.

The SWL procedure can be used not only to distinguish simplicial complexes, but also graphs. To test the isomorphism of two graphs, SWL can test the isomorphism of the corresponding *clique complexes* of the graphs (i.e. every $(k+1)$-clique in the graph becomes a k-simplex in the complex). By taking this pre-processing step, we can relate SWL to WL:

Theorem 2. *SWL is strictly more powerful than WL.*

We present in Figure 5 (Appendix) a pair of graphs that cannot be distinguished by the WL test, but whose clique complexes can be distinguished by SWL.

Theorem 3. *SWL is not less powerful than 3-WL.*

MPSN. We propose the following message passing operations based on the four types of messages discussed in the previous section: faces, cofaces, lower adjacencies and upper adjacencies.

$$m^{t+1}_{\mathcal{F}}(v) = \mathrm{AGG}_{w\in\mathcal{F}(v)}\Big(M_{\mathcal{F}}\big(h^t_v, h^t_w\big)\Big) \qquad m^{t+1}_{\mathcal{C}}(v) = \mathrm{AGG}_{w\in\mathcal{C}(v)}\Big(M_{\mathcal{C}}\big(h^t_v, h^t_w\big)\Big) \tag{1}$$

$$m^{t+1}_\downarrow(v) = \mathrm{AGG}_{w\in\mathcal{N}_\downarrow(v)}\Big(M_\downarrow\big(h^t_v, h^t_w, h^t_{v\cap w}\big)\Big) \quad m^{t+1}_\uparrow(v) = \mathrm{AGG}_{w\in\mathcal{N}_\uparrow(v)}\Big(M_\uparrow\big(h^t_v, h^t_w, h^t_{v\cup w}\big)\Big).$$

If the underlying complex also has a particular orientation, this extra information can be easily included in the message functions defined above. Then, the update operation takes into account these four types of incoming messages and the previous colour of the simplex $h^{t+1}_v =$

$U\Big(h_v^t, m_{\mathcal{F}}^t(v), m_{\mathcal{C}}^t(v), m_{\downarrow}^{t+1}(v), m_{\uparrow}^{t+1}(v)\Big)$. To obtain a global embedding for a p-simplicial complex $\mathcal{K}$ from an MPSN with L layers, the readout function takes as input the multi-sets of colours corresponding to all the dimensions of the complex $h_G = \text{READOUT}(\{\!\{h_v^L\}\!\}_{v\in\mathcal{K}_0}, \ldots, \{\!\{h_v^L\}\!\}_{v\in\mathcal{K}_p})$.

Lemma 4. *MPSNs are at most as powerful as SWL in distinguishing non-isomorphic simplicial complexes.*

Theorem 5. *MPSNs with a sufficient number of layers and injective message, aggregate and update functions are as powerful as SWL in distinguishing non-isomorphic simplicial complexes.*

This theorem, combined with Theorem 2, provides an important corollary that MPSNs are not only beneficial for statistical tasks on higher-dimensional simplicial complexes, but they can also improve over standard GNNs on graph machine learning tasks.

Corollary 6. *There exists an MPSN that is more powerful than WL at distinguishing non-isomorphic graphs when using a clique-complex lifting.*

Theorem 7. *MPSNs generalise certain spectral convolution operators (Ebli et al., 2020; Bunch et al., 2020) defined over simplicial complexes.*

Next we show how MPSNs handle the symmetries present in simplicial complexes. We defer the treatment of orientation equivariance and invariance in oriented SCs for Appendix E.

Theorem 8 (Informal)**.** *An MPSN layer is (simplex) permutation equivariant, while an MPSN network ending with a readout layer is (simplex) permutation invariant.*

Linear regions analysis. While the WL test has been used for studying the expressive power of GNNs, other tools have been used to study the expressive power of conventional neural networks, like fully connected and convolutional. One such tool is based on the number of linear regions of networks using piece-wise linear activations (Pascanu et al., 2013; Montúfar et al., 2014). We show how this tool can also be used to approximate the number of linear regions of the functions represented by GNNs, SCNNs (Ebli et al., 2020) and MPSNs. We focus on the case where the message function is a linear layer and AGG is sum followed by ReLU. We obtain new results in all cases, showing superior capacity of MPSNs. The details of the notation and proofs of Theorems 9, 10, and 11 are relegated to Appendix C.

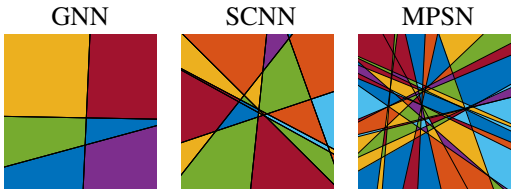

Figure 2: A 2D slice of the input feature spaces of GNN, SCNN, MPSN layers with $S_0 = S_1 = 3$, $S_2 = 1$ (the complex is a triangle), $d_0 = d_1 = d_2 = 1$, $m = 3$, colored by linear regions of the represented functions, for a random choice of the weights.

Theorem 9 (Number of linear regions of a GNN layer)**.** *Consider a graph G with S_0 nodes, node input features of dimension $d \geq 1$, and node output features of dimension m. Suppose the aggregation function $\mathcal{H}$ as function of X is linear and invertible. Then, the number of linear regions of the functions represented by a ReLU GNN layer (9) has the optimal upper bound $R_{\text{GNN}} = \left(2\sum_{i=0}^{d-1}\binom{m-1}{i}\right)^{S_0}$. This applies to aggregation functions with no trainable parameters including GCN convolution (Kipf & Welling, 2017), spectral GNN (Defferrard et al., 2016; Bruna et al., 2014), and traditional message passing (Gilmer et al., 2017).*

The above result should be compared with the optimal upper bound for a standard dense ReLU layer without biases, which for d inputs and m outputs is $2\sum_{i=0}^{d-1}\binom{m-1}{i}$.

Theorem 10 (Number of linear regions for an SCNN layer)**.** *Consider a p-dimensional simplicial complex with S_n n-simplicies for $n = 0, 1, \ldots, p$. Suppose that each M_n is invertible. Then the number of linear regions of the functions represented by a ReLU SCNN layer (10) has the optimal upper bound $R_{\text{SCNN}} = \prod_{n=0}^{p}\left(2\sum_{i=0}^{d_n-1}\binom{m_n-1}{i}\right)^{S_n}$, where, for each of the n-simplices, d_n is the input feature dimension and m_n is the number of output features.*

Theorem 11 (Number of linear regions of an MPSN layer)**.** *With the above settings, the maximum number of linear regions of the functions represented by a ReLU MPSN layer is upper bounded by $R_{\text{MPSN}} \leq \prod_{n=0}^{p}\left(2\sum_{i=0}^{d_{n-1}+d_n+d_{n+1}-1}\binom{m-1}{i}\right)^{S_n}$, where we set $d_{-1} = d_{p+1} = 0$.*

We also note the 'trivial' upper bound, with $N := \sum_{n=0}^{p} S_n d_n$ *and* $M := \sum_{n=0}^{p} S_n m$, $R_{\text{MPSN}} \leq 2\sum_{j=0}^{N-1}\binom{M-1}{j}$. *Moreover, if* $m \leq d_n$ *and* $\text{rank}(O_n) = S_n$ *for all* n, *writing* $S = \max\{S_0, \ldots, S_p\}$, *we have the lower bound* $R_{\text{MPSN}} \geq 2\sum_{j=0}^{N-(mS)-1}\binom{M-1}{j}$.

We note that the MPSN lower bound (26) surpasses the SCNN upper bound (22), with order $\prod_n (\sum_{n'} mS_{n'})^{d_n S_n}$ and $\prod_n (mS_n)^{d_n S_n}$, respectively, when $d_n \geq m$. The GNN bound (21) is a special case of the SCNN bound (22) with $p = 0$, and has asymptotic order $m^{d_0 S_0}$.

3 Experiments

Verification on SR Graphs We experimentally validate our theoretical result on the expressive power of our proposed architecture on the task of distinguishing hard pairs of isomorphic graphs. In particular, we benchmark our architecture on 9 synthetic datasets comprising families of Strongly Regular graphs . Strongly Regular (SR) graphs represent 'hard' instances of graph isomorphism, as pairs thereof cannot provably be distinguished by the 3-WL test (we refer readers to Section B.1 for a formal proof).

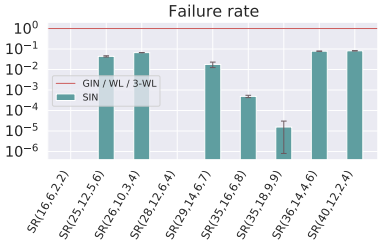

Figure 3: Failure rate on the task of distinguishing SR graphs; log-scale, the smaller the better. Each SR family is denoted as SR(v, k, λ, μ), where the four parameters indicate the number of nodes, degree, number of common neighbors between adjacent vertices and non-adjacent ones. GIN fails to distinguish all graph pairs in all families.

Results are illustrated in Figure 3, where we show performance on our isomorphism problem in terms of failure rate, that is the fraction of non-distinguished pairs. As it can be observed, our architecture is able to distinguish the majority of graph pairs in all families, since, in contrast to standard graph neural networks, it is able to access information related to the presence and number of cliques therein. We additionally run a comparable GIN (Xu et al., 2019b) architecture and empirically verify its inability to distinguish any pair, theoretically justified by its expressive power being upper-bounded by 1-WL. Additional experimental details are in Section G.1. We also include graph classification results on real-world graphs in Appendix G.3.

Trajectory Classification In this section, we turn our attention to an application involving oriented simplicial complexes. Inspired by Schaub et al. (2020), we generate a synthetic dataset of trajectories on the simplicial complex from Figure 4. All trajectories pass either through the bottom left corner or the top-right corner, thus giving rise to two different classes that we aim to distinguish. Due to the two holes present in the complex, the trajectories of the two classes correspond to approximately orthogonal directions in the space of harmonic functions of the L_1 Hodge-Laplacian (Schaub et al., 2020). Therefore, we hypothesise, that an orientation invariant MPSN network with orientation equivariant layers should easily distinguish the two classes. We describe our model and other task details in Appendix G.2. As a baseline, we consider a similar message passing GNN operating in the line graph of the 1-skeleton of the simplicial complex, without being (explicitly) aware of the orientation and the triangles present in the graph. As expected, the MPSN model obtains 96.5% accuracy compared to 71.5% for the GNN baseline.

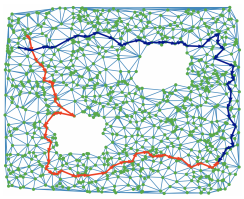

Figure 4: Two sample trajectories from the two classes.

4 Conclusion

We introduce a provably powerful message passing procedure for simplicial complexes relying on local higher-order interactions. We motivate our message passing framework by the introduction of SWL, a colouring algorithm for simplicial complex isomorphism testing, generalising the WL test for graphs. We prove that when graphs are lifted in the simplicial complex space via their clique complex, SWL and MPSNs are more expressive than the WL test. At the same time, we produce an estimate for the number of linear regions of GNNs, SCNNs and MPSNs, which also reveals the superior expressive power of our model. We empirically confirm these results on a dataset of challenging strongly regular graphs, on a problem of trajectory classification, and by computing 2D slices through the linear regions of the models.

REFERENCES

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *NIPS*, pp. 1993–2001, 2016.

K. H. Baik and L. L. Miller. Topological approach for testing equivalence in heterogeneous relational databases. *The Computer Journal*, 33, 1990.

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

Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and dynamics. *Physics Reports*, 2020.

Jean-Daniel Boissonnat and C. Maria. The simplex tree: An efficient data structure for general simplicial complexes. *Algorithmica*, 70:406–427, 2014.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv:2006.09252*, 2020.

Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, September 1973.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

Eric Bunch, Qian You, Glenn Fung, and Vikas Singh. Simplicial 2-complex convolutional neural networks. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1):564–568, 2008.

Zachary Chase. The maximum number of triangles in a graph of given maximum degree. *Advances in Combinatorics*, 2020.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *NeurIPS*, 2020.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*, 2016.

Pim de Haan, Taco Cohen, and Max Welling. Natural graph networks. In *NeurIPS*, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

Massimo Ferri, Dott Mattia G Bergomi, and Lorenzo Zu. Simplicial complexes from graphs towards graph persistence. *arXiv:1805.10716*, 2018.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Wenying Gan, P. Loh, and B. Sudakov. Maximizing the number of independent sets of a fixed size. *Combinatorics, Probability and Computing*, 24:521–527, 2015.

M. R. Garey and R. L. Graham. On cubical graphs. *Journal of Combinatorial Theory (B)*, 18:84–95, 1975.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272, 2017.

Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. Cell complex neural networks. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

Allen Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

T. Kaczynski, K. Mischaikow, and M. Mrozek. *Computational Homology*. Springer-Verlag New York, 2004.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, 407(1):458 – 473, 2008.

Ming Li, Zheng Ma, Yu Guang Wang, and Xiaosheng Zhuang. Fast Haar transforms for graph neural networks. *Neural Networks*, pp. 188–198, 2020.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *ICLR*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *NeurIPS*, 2019.

R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Guido F Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *NeurIPS*, 2020.

Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

P. Orlik and H. Terao. *Arrangements of Hyperplanes*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1992.

Razvan Pascanu, Guido Montúfar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. In *ICLR*, 2013.

Steven Rosenberg. *The Laplacian on a Riemannian Manifold: An Introduction to Analysis on Manifolds*. London Mathematical Society Student Texts. Cambridge University Press, 1997.

W. E. Roth. On direct product matrices. *Bulletin of the American Mathematical Society*, 40(6): 461–468, 06 1934.

Michael T Schaub, Austin R Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie. Random walks on simplicial complexes and the normalized Hodge 1-Laplacian. *SIAM Review*, 62(2):353–391, 2020.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 12(Sep):2539–2561, 2011.

Richard P. Stanley. An introduction to hyperplane arrangements. In *Lecture notes, IAS/Park City Mathematics Institute*, 2004.

Matus Telgarsky. Benefits of depth in neural networks. In *COLT*, 2016.

The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.4.1 edition, 2021. URL `https://gudhi.inria.fr/doc/3.4.1/`.

Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.

H. Wagner, C. Chen, and E. Vu¸cini. Efficient computation of persistent homology for cubical data. In *Topological Methods in Data Analysis and Visualization II. Mathematics and Visualization*. Springer, Berlin, Heidelberg, 2011.

Yu Guang Wang, Ming Li, Zheng Ma, Guido Montúfar, Xiaosheng Zhuang, and Yanan Fan. Haar graph pooling. In *ICML*, 2020.

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.

Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *ICML*, 2020.

Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. In *ICLR*, 2019a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. In *ICML*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019b.

T. Zaslavsky. *Facing up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes: Face-count Formulas for Partitions of Space by Hyperplanes*. Memoirs of the American Mathematical Society. American Mathematical Society, 1975.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Xuebin Zheng, Bingxin Zhou, Ming Li, Yu Guang Wang, and Junbin Gao. MathNet: Haar-like wavelet multiresolution-analysis for graph representation and learning. *arXiv:2007.11202*, 2020a.

Xuebin Zheng, Bingxin Zhou, Yu Guang Wang, and Xiaosheng Zhuang. Decimated framelet system on graphs and fast G-framelet transforms. *arXiv:2012.06922*, 2020b.

Xuebin Zheng, Bingxin Zhou, Junbin Gao, Yu Guang Wang, Pietro Liò, Ming Li, and Guido Montúfar. How framelets enhance graph neural networks. *arXiv preprint arXiv:2102.06986*, 2021.

A BACKGROUND

Simplicial complexes Combinatorially, a simplicial complex $\mathcal{K}$ is a family of sets closed under taking subsets (Hatcher, 2000). A member $\sigma \in \mathcal{K}$ with cardinality $k+1$ is called a *k-dimensional simplex* or simply a k-simplex. One can interpret simplices as geometric objects: 0-simplices as *vertices*, 1-simplices as *edges*, 2-simplices as *triangles*, and so on. A simplicial complex is intuitively a collection of such simplices that intersect 'nicely' (see Figure 1). Each k-simplex has $k+1$ *faces* of dimension $k-1$ (e.g. the three edges of a triangle). If σ_1 is a face of σ_2, then σ_2 is a *coface* of σ_1. Two k-simplicies are *lower adjacent* if they have a common face (e.g. two triangles sharing an edge) and *upper adjacent* if they have a common coface (e.g. two vertices connected by an edge). We denote by $\mathcal{F}(\sigma), \mathcal{C}(\sigma), \mathcal{N}_{\downarrow}(\sigma), \mathcal{N}_{\uparrow}(\sigma)$ the sets of faces, cofaces, lower and upper neighbours of simplex σ. We use $\mathcal{K}_d$ to refer to all the simplicies of dimension d in the complex. An *oriented simplicial complex* is a simplicial complex together with a total order on its set of 0-simplices. The orientation of a simplicial complex is needed for defining simplicial homology (Hatcher, 2000) – the theory of counting d-dimensional 'holes' in the complex (e.g. connected components, cycles or voids). However, orientations do not encode any topological information, as simplicial homology is independent of the choice of order.

Graph neural networks GNNs are neural models for processing signals on 1-dimensional simplicial complexes (graphs). Most GNN models follow the *message passing* scheme of Gilmer et al. (2017) or its simplified variants. Under this framework, the tth layer of a GNN performs the following message and updates computations:

$$m_v^{t+1} = \text{AGG}_{w \in \mathcal{N}(v)} \Big(M \big(h_v^t, h_w^t, h_{e(v,w)}^t \big) \Big) \tag{2}$$

$$h_v^{t+1} = U \Big(h_v^t, m_v^{t+1} \Big), \tag{3}$$

where h_v^t denotes the features of node v at layer t, $h_{e(v,w)}^t$ represents the features of the edge $e(v,w)$ and $\mathcal{N}(v)$ is the set of neighbours of node v. Some architectures do not use any edge features. AGG is a permutation-invariant aggregation operator such as sum, mean, or max, and the message and update functions M and U are typically learnable.

For node/edge regression and classification tasks, the final representations h_v^L and $h_{e(v,w)}^L$ are used to make the prediction. For graph-level tasks, such as graph classification, all the final individual representations of the nodes are combined into a single vector by a readout function:

$$h_G = \text{READOUT}\big(\{h_v^L | v \in G\}\big). \tag{4}$$

Most GNN architectures differ only in the design of aggregate (AGG), message (M) and update (U).

The Weisfeiler-Lehman (WL) test The WL (or 1-WL) graph-isomorphism test is a fast heuristic providing a necessary condition for two graphs to be isomorphic (Weisfeiler & Leman, 1968). It is an iterative color refinement procedure for the graph nodes. Given a graph G, all the nodes $v \in \mathcal{V}_G$ are initialised with the same colour c_v^0; they are then iteratively refined as: $c_v^{t+1} = \text{HASH}\big(c_v^t, \{\!\{c_u^t\}\!\}_{u \in \mathcal{N}(v)}\big)$, that is by applying an injective map on the multiset of neighboring colours. The algorithm stops when the graph colouring is stable (colours are no longer updated), yielding a colour histogram. If two graphs are associated with two different colour histograms they are deemed non-isomorphic. However, if the histograms are identical, then the WL test is non-conclusive: the two graphs are *possibly*, but *not necessarily*, isomorphic. The analogy between color refinement and message passing has been analyzed in Xu et al. (2019b); Morris et al. (2019), where the expressive power of standard graph neural networks is at most as the WL test.

The k-WL test is a higher-order extension of the WL algorithm that works on k-tuples instead of individual nodes. With the exception of 1-WL and 2-WL tests (that are equivalent), $(k+1)$-WL is strictly stronger than k-WL, for any $k \geq 2$, that is, there exist graphs on which k-WL fails and $(k+1)$-WL succeeds, but not vice versa. k-WL is thus a hierarchy or increasingly more powerful graph isomorphism tests, sometimes referred to as the *Weisfeiler-Lehman hierarchy*. Maron et al. (2019) proposed a form of graph neural networks equivalent to 3-WL that however lack locality and suffer from high memory and computational complexity.

B PROOFS OF SWL THEORY RESULTS

We first introduce the required notion and notation. We note that even though these results mainly refer to simplicial complexes, they also apply to graphs because any graph is also a simplicial complex. The first definition formalises the notion of 'colouring' for the simplices of a simplicial complex.

Definition 12 (Colouring function)**.** *A colouring function is a function $c : K \to \Sigma$ that takes as input a simplex from a simplicial complex K and outputs a colouring of that simplex from a countable alphabet Σ. We denote the space of all colouring functions for a complex K as $\mathbb{C}_K$. We abuse the notation slightly and use c_v interchangeably with $c(v)$.*

Definition 13 (Histogram)**.** *The histogram of a multiset $\{\!\{\sigma_i\}\!\}$ with $a_i \in \Sigma$ is a function $H : \Sigma \to \mathbb{N}$, where $H(\sigma)$ specifies the number of occurences of σ in the multiset.*

The next definition generalised the concept of a colouring algorithm such as WL or SWL.

Definition 14 (Colouring algorithm)**.** *A colouring algorithm $\mathcal{A}$ is a computable function that takes as input a complex K and outputs a colouring from $\mathbb{C}_K$ with the property that for any two K_1 and K_2, the colour histograms of $\mathcal{A}(K_1)$ and $\mathcal{A}(K_2)$ over the simplicies of a certain dimension are different only if K_1 and K_2 are not isomorphic.*

In other words, if the colour histograms at any level of the complex (vertices, edges, triangles, etc.) are different, it implies that the simplicial complexes are different. However, this does not mean any two non-isomorphic simplicial complexes *must* have different colourings. Finally, we introduce the notion of colour-refinement as a way of comparing two colourings.

Definition 15 (Simplicial complex colour refinement)**.** *Let c_1 and c_2 be two colouring functions for a complex K. We say that $c_1 \sqsubseteq c_2$ (i.e. c_1 color-refines c_2) if for any two simplicies $v, w \in K$ we have that $c_1(v) = c_1(w)$ implies that $c_2(v) = c_2(w)$.*

We now prove a lemma that will be used repeatedly in our proofs in this section. It shows how color refinement can be used to compare the ability of certain coloring algorithms.

Lemma 16. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two colouring algorithms. If $\mathcal{A}_1(K) \sqsubseteq \mathcal{A}_2(K)$ for any complex K, then any two non-isomorphic simplicial complexes K_1 and K_2 that can be distinguished by $\mathcal{A}_2$ can also be distinguished by $\mathcal{A}_1$. In other words, $\mathcal{A}_1$ is at least as powerful as $\mathcal{A}_2$.*

Proof. Let K_1 and K_2 be two non-isomorphic simplicial complexes that can be distinguished by $\mathcal{A}_2$. Because they can be distinguished, this means that there exists a certain dimension where the colour histograms produced by $\mathcal{A}_2(K_1)$ and $\mathcal{A}_2(K_2)$ are different. We denote these histograms by sets of tuples $\{(c_i, f_i)\}_i$ that contain a unique colour c_i and its frequency f_i among the simplicies of the given dimension. The two sets can be different in three (potentially concurrent) ways:

1. $\mathcal{A}_2(K_1)$ contains a colour not present in $\mathcal{A}_2(K_2)$ (among the simplicies of the considered dimension).
2. $\mathcal{A}_2(K_2)$ contains a colour not present in $\mathcal{A}_2(K_1)$ (among the simplicies of the considered dimension).
3. The frequencies of a certain colour present in both histograms are different.

By $\mathcal{A}_1(K) \sqsubseteq \mathcal{A}_2(K)$ for any K, any colour histogram under $\mathcal{A}_1$ will 'refine' the colour histogram under $\mathcal{A}_2$. That is, it splits each $H_2(c_i) = f_i$ in the $\mathcal{A}_2$ histogram (H_2) into multiple sub-colours with their own frequency $H_1(c_{i,j}) = f_{i,j}$ where j is an index over the sub-colours, H_2 is the histogram produced by the colouring of $\mathcal{A}_1$, and $\sum_j H_1(c_{i,j}) = H_2(c_i)$. It then implies that the histograms of the two graphs will also be different under the colouring of $\mathcal{A}_1$ for any of the three cases listed above. □

Let us now proceed by proving Theorem 1. We will first introduce and prove the following:

Lemma 17. *SWL with* $\mathrm{HASH}\big(c_v^t, c_{\mathcal{F}}^t(v), c_{\downarrow}^t(v), c_{\uparrow}^t(v)\big)$ *is as powerful as SWL with the generalised update rule* $\mathrm{HASH}\big(c_v^t, c_{\mathcal{F}}^t(v), c_{\mathcal{C}}^t(v), c_{\downarrow}^t(v), c_{\uparrow}^t(v)\big)$.

Proof. Let a denote the colouring of the general version and b the colour of the restricted version. Then, a $\sqsubseteq$ b because of considering the additional cofaces in the colouring procedure. We will now prove the converse. Suppose $b_v^{t+1} = b_w^{t+1}$. Then, we have that the arguments of the hash function are equal. Thus, $b_v^t = b_w^t, b_\downarrow^t(v) = b_\downarrow^t(w), b_\uparrow^t(v) = b_\uparrow^t(w)$, and $b_\mathcal{F}^t(v) = b_\mathcal{F}^t(w)$. Because the multi-sets of the upper colours are equal, the multi-sets of the second entries of the tuples inside them (i.e. the co-face colours) are also equal to $\{\!\{b_{v\cup v'}^t | v' \in \mathcal{N}_\uparrow(v)\}\!\} = \{\!\{b_{w\cup w'}^t | w' \in \mathcal{N}_\uparrow(w)\}\!\}$. As v and w have the same dimension and their multi-sets have the same size, they have the same number of cofaces. Due to that each of these co-faces shows up the same number of times in the multi-sets, the colours of the cofaces are thus the same: $\{\!\{b_x^t | x \in \mathcal{C}(v)\}\!\} = \{\!\{b_y^t | y \in \mathcal{C}(w)\}\!\}$. By the induction, the arguments of the hash function used by a are the same for v and w. Thus, $a_v^{t+1} = a_w^{t+1}$. □

Proof of Theorem 1. Let b denote the colouring based on the first rule and a the colouring as used in the proof of Lemma 17. As before, it is trivial to show $a^t \sqsubseteq b^t$ by the additional argument (the colours of the down adjacencies). We now prove that $b^{2t} \sqsubseteq a^t$ by induction.

The base case holds by definition. We assume the statement is true for t and prove it for $t+1$, as follows. Suppose $b_v^{2t+2} = b_w^{2t+2}$. By expanding the hash function two steps back in time, we obtain $b_v^{2t} = b_w^{2t}, b_\mathcal{F}^{2t}(v) = b_\mathcal{F}^{2t}(w), b_\uparrow^{2t}(v) = b_\uparrow^{2t}(w)$.

Suppose for the sake of contradiction that $b_\downarrow^{2t}(v) \neq b_\downarrow^{2t}(w)$. This means that there exists a pair of colours $(l_0 = b_x^{2t}, l_1 = b_{x\cap v}^{2t})$ with $x \in \mathcal{N}_\downarrow(v)$ that shows up (without loss of generality) more times in $b_\downarrow^{2t}(v)$ than in $b_\downarrow^{2t}(w)$. Let us partition these apparitions in these two multi-sets by the faces $f_1 \in \mathcal{F}(v)$ and $f_2 \in \mathcal{F}(w)$ having the colour $b_{f1}^{2t} = b_{f2}^{2t} = l_1$. By $b_\mathcal{F}^{2t}(v) = b_\mathcal{F}^{2t}(w)$, we have the same number of such faces / partitions for v and w.

Note that for each such face f of v, the size of the set $A_v(f) = \{x | b_x^{2t} = l_0, x \in \mathcal{N}_\downarrow(v), f = x \cap v\}$ (containing the lower adjacent simplices of v with colour l_0 and sharing face f) determines the colour of f at step $t+1$ since the colours $\{\!\{b_x^{2t} | x \in A_v(f)\}\!\}$ show up in the tuples of $b_\uparrow^{2t}(f)$, which is an input of the hash function colouring f. The same applies for the faces of w. Because there are more colors $l = (l_0, l_1)$ in $b_\downarrow^{2t}(v)$ than in $b_\downarrow^{2t}(w)$ (by the assumption in the paragraph above), the histogram of the sizes of these partitions are different between v and w. This means that in the next time step, $\{\!\{b_{f_1}^{2t+1}\}\!\} \neq \{\!\{b_{f_2}^{2t+1}\}\!\}$. Since these colours are also different from the colours of the other faces of v and w at step $2t+1$, we have that $b_\mathcal{F}^{2t+1}(v) \neq b_\mathcal{F}^{2t+1}(w)$. Finally, this implies that $b_v^{2t+2} \neq c_w^{2t+2}$, which is a contradiction. Therefore, $b_\downarrow^{2t}(v) = b_\downarrow^{2t}(w)$.

Now, we apply the induction hypothesis to obtain that $a_v^t = a_w^t, a_\mathcal{F}^t(v) = a_\mathcal{F}^t(w), a_\uparrow^t(v) = a_\uparrow^t(w)$ and $a_\downarrow^t(v) = a_\downarrow^t(w)$. Then, $b^{2t} \sqsubseteq a^t$. □

We show a slightly weaker version of Theorem 2.

Lemma 18. *SWL is at least as powerful as WL.*

Proof. Let G be a graph with a clique complex K. Let c^t be the colouring of the nodes of G at iteration t of WL and sc^t the colouring of the (same) vertices in K at iteration t of SWL. To prove the lemma, we will show by induction that sc^t is a refinement of c^t. Combined with Lemma 16, this proves the result.

For the base case, the implication holds at initialisation since all nodes are assigned the same colour. That is: $sc^0(v) = sc^0(w) \implies c^0(v) = c^0(w)$. For the induction step, suppose $sc^{t+1}(v) = sc^{t+1}(w)$, for two 0-simplices v and w in K. In this case, the arguments of the HASH function must be equal. Concretely, as vertices are only upper adjacent and have no faces, $sc^t(v) = sc^t(w)$ and $sc_\uparrow^t(v) = sc_\uparrow^t(w)$. In turn, the sets formed of the first entry of the tuples in these two multi-sets are also equal: $SC_v = \{\!\{sc_x^t\}\!\} = \{\!\{sc_y^t\}\!\} = SC_w$, for all $x \in \mathcal{N}_\uparrow(v)$ and $y \in \mathcal{N}_\uparrow(w)$. By the induction hypothesis, the equalities $sc^t(v) = sc^t(w)$ and $SC_v = SC_w$ imply $c^t(v) = c^t(w)$ and $C_v = \{\!\{c_x^t\}\!\} = \{\!\{c_y^t\}\!\} = C_w$. Since these are the arguments the WL hash function uses to compute the colours of v and w at the next step, we obtain $c^{t+1}(v) = c^{t+1}(w)$. □

Informally, this proof shows that the information coming from the colouring of the upper layers of the complex will refine the colouring of the vertices. This means that SWL will be able to distinguish

just through its vertex-level colour histogram at least the same set of graphs that WL can distinguish. However, this proof disregards the histograms of the higher-levels and these can indeed be used to show that SWL is strictly more powerful than WL, which is done in Theorem 2.

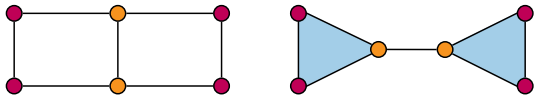

Figure 5: Two graphs that cannot be distinguished by 1-WL, but have distinct clique complexes (the second contains triangles).

Proof of Theorem 2. Based on Lemma 18, it is sufficient to present a pair of graphs that cannot be distinguished by WL, but whose clique complexes can be distinguished by SWL. Such a pair is included in Figure 5. While WL produces the same colouring for both graphs, one clique complex contains two triangles, while the other has no triangles. □

Proof of Lemma 4. Let sc^t_v and h^t_v be the colouring of the simplex v at iteration t of SWL and the t-th layer of an MPSN, respectively. We will show by induction that sc^t refines the colouring of h^t. For this proof, it is convenient to use the most general version of SWL, containing the complete set of adjacencies.

As above, the base case holds since both methods start with the same initial colour. For the induction step, suppose we have two simplicies v and w such that $sc^{t+1}(v) = sc^{t+1}(w)$. Because the SWL colouring is an injective mapping, the arguments to the HASH function are the same. This means that $sc^t(v) = sc^t(w)$ and the multi-sets of colours formed by their neighbours are identical: $sc^t_\downarrow(v) = sc^t_\downarrow(w), sc^t_\uparrow(v) = sc^t_\uparrow(w), sc^t_\mathcal{F}(v) = sc^t_\mathcal{F}(w), sc^t_\mathcal{C}(v) = sc^t_\mathcal{C}(w)$. By the induction hypothesis, these multi-sets will be equal under the colouring h^t. Enumerating all, $h^t(v) = h^t(w), h^t_\downarrow(v) = h^t_\downarrow(w), h^t_\uparrow(v) = h^t_\uparrow(w), h^t_\mathcal{F}(v) = h^t_\mathcal{F}(w), h^t_\mathcal{C}(v) = h^t_\mathcal{C}(w)$. Because the exact same multi-sets are supplied as input to the MESSAGE, AGGREGATE and UPDATE functions, their output are also the same for v and v. Then, $h^{t+1}_v = h^{t+1}_w$. Applying Lemma 16 thus proves the lemma. □

Proof of Theorem 5. By Lemma 4, we only need to show that for an MPSN model satisfying the conditions in the theorem, we have that $h^t \sqsubseteq sc^t$.

The base case can be proved by definition. For the step case, given that the UPDATE, AGGREGATE and MESSAGE functions are injective, for any two simplicies $h^{t+1}_v = h^{t+1}_w$, the inputs to these functions at the previous iteration were also the same. As in our previous proofs, by applying the induction hypothesis, the inputs to the SWL HASH function at iteration t for v and w are also equal and $sc^{t+1}_v = sc^{t+1}_w$. It follows $h^t \sqsubseteq sc^t$, $sc^t \sqsubseteq h^t$ (Lemma 4) and, consequently, $sc^t = h^t$. □

B.1 Higher-order WL and Strongly Regular Graphs

Higher-order variants of the standards WL procedure operate on node tuples rather than single nodes and iteratively apply color refinement steps thereon.

k-WL The k-WL is one such higher-order variants. It specifically operates on node k-tuples by refining their colors based on the generalized notion of j-neighborhood. The j-neighborhood ($j \in \{1, \dots, k\}$) for node k-tuple $\mathbf{v} = (v_1, v_2, \dots, v_k)$ is defined as $\mathcal{N}_j(\mathbf{v}) = \{(v_1, \dots, v_{j-1}, w, v_{j+1}, \dots, v_k) | w \in \mathcal{V}_G\}$. The algorithm first initialises node tuples based on their isomorphism type: two k-tuples $\mathbf{v}^a = (v^a_1, v^a_2, \dots, v^a_k)$, $\mathbf{v}^b = (v^b_1, v^b_2, \dots, v^b_k)$ have the same isomorphism type (and are thus assigned the same initial colour $c_{\mathbf{v}^a} = c_{\mathbf{v}^b}$) iff (i) $\forall i, j \in \{1, \dots, k\}, v^a_i = v^a_j \Leftrightarrow v^b_i = v^b_j$, (ii) $\forall i, j \in \{1, \dots, k\}, v^a_i \sim v^a_j \Leftrightarrow v^b_i \sim v^b_j$, where $\sim$ indicates adjacency. Given this initial colouring, the procedure iteratively applies the following color refinement step

$$c^{t+1}_\mathbf{v} = \text{HASH}\Big(c^t_\mathbf{v}, M^t(\mathbf{v})\Big), \tag{5}$$

$$M^t(\mathbf{v}) = \big(\{\!\{c^t_\mathbf{u} | \mathbf{u} \in \mathcal{N}_j(\mathbf{v})\}\!\}\big| j = 1, 2, \dots, k\big) \tag{6}$$

until the colouring does not change further. The k-WL procedure can be employed to *test* the isomorphism between graphs in the same way as the standard WL one is. For any $k \geq 2$, it is known that $(k+1)$-WL test is strictly stronger than k-WL one, i.e. there exist exemplary pairs of non-isomorphic graphs that k-WL cannot distinguish while $(k+1)$-WL can, but not vice-versa. Local variants of k-WL have recently been introduced in Morris et al. (2020).

k-FWL The k-Folklore WL procedure (k-FWL) is another higher-order variant of the standard WL. Similarly to k-WL, it operates by refining the colors of node k-tuples, initialised based on their isomorphism type. However, it employs a different notion of neighborhood and refinement step. The Folklore j-neighborhood for node k-tuple $\mathbf{v}$ is defined as $\mathcal{N}_j^F(\mathbf{v}) = \big((j, v_2, \ldots, v_k), (v_1, j, \ldots, v_k), \ldots, (v_1, \ldots, v_{k-1}, j)\big)$, with $j \in \mathcal{V}_G$. The algorithm iteratively applies the steps

$$c_{\mathbf{v}}^{t+1} = \mathrm{HASH}\Big(c_{\mathbf{v}}^t, M^{F,t}(\mathbf{v})\Big), \tag{7}$$

$$M^{F,t}(\mathbf{v}) = \{\!\{\big(c_{\mathbf{u}}^t | \mathbf{u} \in \mathcal{N}_j^F(\mathbf{v})\big) \big| j \in \mathcal{V}_G\}\!\} \tag{8}$$

until the colouring does not change. It is known that k-FWL is equivalent to $(k+1)$-WL for $k \geq 2$.

Strongly Regular Graphs A Strongly Regular graph in the family *SR(n,d,λ,μ)* is a regular graph with n nodes and degree d, for which every two adjacent nodes always have λ mutual neighbours and every two non-adjacent nodes always have μ mutual neighbours. This class of graphs is of particular interest due to the following lemma.

Lemma 19. *No pair of Strongly Regular graphs in family* SR(n,d,λ,μ) *can be distinguished by the* 2*-FWL test.*

Proof. Let us denote by $\mathcal{V}_G^2$ the set of all node 2-tuples in graph G. We note that three isomorphism types are induced by considering node 2-tuples:

(1) *node type*: $\mathbf{v} = (v_1, v_1)$

(2) *edge type*: $\mathbf{v} = (v_1, v_2)$ with $v_1 \sim v_2$ (the two nodes are adjacent in the original graph)

(3) *non-edge type*: $\mathbf{v} = (v_1, v_2)$ with $v_1 \not\sim v_2$ (the two nodes are *not* adjacent in the original graph).

These three isomorphism types partition the tuple set $\mathcal{V}_G^2$ into the three subsets $\mathcal{V}_{G(1)}^2, \mathcal{V}_{G(2)}^2, \mathcal{V}_{G(3)}^2$ such that any tuple $\mathbf{v} \in \mathcal{V}_{G(i)}^2$ is of isomorphism type i. We write $\mathbf{v}^{(i)}$ to indicate $\mathbf{v} \in \mathcal{V}_{G(i)}^2$ for simplicity.

At initialisation, the 2-FWL algorithm assigns a colour to each tuple $\mathbf{v} \in \mathcal{V}_G^2$ based on its isomorphism type, that is $\forall \mathbf{v} \in \mathcal{V}_{G(i)}^2, c_{\mathbf{v}} = c_i^0$. The colouring is therefore constant within partitions. Then, we notice that the colouring is kept constant within partitions through the application of the refinement steps described by Equation 7. In other words, the 2-FWL procedure cannot produce a colour partitioning of the set of node 2-tuples that is finer than the one at initialisation. This is shown by induction on the step t of color refinement.

The base case evidently holds for $t = 0$ since all tuples in the same partition are assigned the same colour by the 2-FWL initialisation procedure. For the induction step we assume that the colouring is constant within each partition at t and show that it maintains constant at $t+1$, that is, after the application of one colour refinement step. This is proved by showing that all node tuples within the same partition have their colour refined identically. We will show this for each of the three partitions separately, leveraging on the induction hypothesis and the properties of Strongly Regular graphs.

A node tuple $\mathbf{v} = (v_1, v_1) \in \mathcal{V}_{G(1)}^2$ has $\mathcal{N}_j^F(\mathbf{v}) = \big((j, v_1), (v_1, j)\big), j \in \mathcal{V}_G$. Therefore, *any* $\mathbf{v} \in \mathcal{V}_{G(1)}^2$ has exactly:

- $(j = v_1)$ 1 neighborhood of the form $\big(\mathbf{w}^{(1)}, \mathbf{w}^{(1)}\big)$, associated with color tuple (c_1^t, c_1^t);

- $(j \sim v_1)$ d neighborhoods of the form $\left(\mathbf{w}^{(2)}, \mathbf{u}^{(2)}\right)$, associated with color tuple (c_2^t, c_2^t);
- $(j \not\sim v_1)$ $n-d-1$ neighborhoods of the form $\left(\mathbf{w}^{(3)}, \mathbf{u}^{(3)}\right)$, associated with color tuple (c_3^t, c_3^t).

For *any* $\mathbf{v} \in \mathcal{V}^2_{G(1)}$ we then have

$$c_{\mathbf{v}}^{t+1} = \text{HASH}\Big(c_1^t, M^{F,t}(\mathbf{v})\Big)$$
$$M^{F,t}(\mathbf{v}) = \{\!\{\underbrace{(c_1^t, c_1^t)}_{1 \text{ time}}, \underbrace{(c_2^t, c_2^t)}_{d \text{ times}}, \underbrace{(c_3^t, c_3^t)}_{n-d-1 \text{ times}}\}\!\}.$$

A node tuple $\mathbf{v} = (v_1, v_2) \in \mathcal{V}^2_{G(2)}$ has $\mathcal{N}_j^F(\mathbf{v}) = \big((j, v_2), (v_1, j)\big), j \in \mathcal{V}_G$. Therefore, *any* $\mathbf{v} \in \mathcal{V}^2_{G(2)}$ has exactly:

- $(j = v_2)$ 1 neighborhood of the form $\left(\mathbf{w}^{(1)}, \mathbf{u}^{(2)}\right)$, associated with color tuple (c_1^t, c_2^t);
- $(j = v_1)$ 1 neighborhood of the form $\left(\mathbf{w}^{(2)}, \mathbf{u}^{(1)}\right)$, associated with color tuple (c_2^t, c_1^t);
- $(j \sim v_2, j \sim v_1)$ λ neighborhoods of the form $\left(\mathbf{w}^{(2)}, \mathbf{u}^{(2)}\right)$, associated with color tuple (c_2^t, c_2^t);
- $(j \sim v_2, j \not\sim v_1)$ $d-\lambda$ neighborhoods of the form $\left(\mathbf{w}^{(2)}, \mathbf{u}^{(3)}\right)$, associated with color tuple (c_2^t, c_3^t);
- $(j \not\sim v_2, j \sim v_1)$ $d-\lambda$ neighborhoods of the form $\left(\mathbf{w}^{(3)}, \mathbf{u}^{(2)}\right)$, associated with color tuple (c_3^t, c_2^t);
- $(j \not\sim v_2, j \not\sim v_1)$ $k = n-2-2d+\lambda$ neighborhoods of the form $\left(\mathbf{w}^{(3)}, \mathbf{u}^{(3)}\right)$, associated with color tuple (c_3^t, c_3^t).

For *any* $\mathbf{v} \in \mathcal{V}^2_{G(2)}$ we have

$$c_{\mathbf{v}}^{t+1} = \text{HASH}\Big(c_2^t, M^{F,t}(\mathbf{v})\Big)$$
$$M^{F,t}(\mathbf{v}) = \{\!\{\underbrace{(c_1^t, c_2^t)}_{1 \text{ time}}, \underbrace{(c_2^t, c_1^t)}_{1 \text{ time}}, \underbrace{(c_2^t, c_2^t)}_{\lambda \text{ times}}, \underbrace{(c_2^t, c_3^t)}_{d-\lambda \text{ times}}, \underbrace{(c_3^t, c_2^t)}_{d-\lambda \text{ times}}, \underbrace{(c_3^t, c_3^t))}_{k \text{ times}}\}\!\}.$$

A node tuple $\mathbf{v} = (v_1, v_2) \in \mathcal{V}^2_{G(3)}$ has $\mathcal{N}_j^F(\mathbf{v}) = \big((j, v_2), (v_1, j)\big), j \in \mathcal{V}_G$. Therefore, *any* $\mathbf{v} \in \mathcal{V}^2_{G(3)}$ has exactly:

- $(j = v_2)$ 1 neighborhood of the form $\left(\mathbf{w}^{(1)}, \mathbf{u}^{(3)}\right)$, associated with color tuple (c_1^t, c_3^t);
- $(j = v_1)$ 1 neighborhood of the form $\left(\mathbf{w}^{(3)}, \mathbf{u}^{(1)}\right)$, associated with color tuple (c_3^t, c_1^t);
- $(j \sim v_2, j \sim v_1)$ μ neighborhoods of the form $\left(\mathbf{w}^{(2)}, \mathbf{u}^{(2)}\right)$, associated with color tuple (c_2^t, c_2^t);
- $(j \sim v_2, j \not\sim v_1)$ $d-\mu$ neighborhoods of the form $\left(\mathbf{w}^{(2)}, \mathbf{u}^{(3)}\right)$, associated with color tuple (c_2^t, c_3^t);
- $(j \not\sim v_2, j \sim v_1)$ $d-\mu$ neighborhoods of the form $\left(\mathbf{w}^{(3)}, \mathbf{u}^{(2)}\right)$, associated with color tuple (c_3^t, c_2^t);
- $(j \not\sim v_2, j \not\sim v_1)$ $k = n-2-2d+\mu$ neighborhoods of the form $\left(\mathbf{w}^{(3)}, \mathbf{u}^{(3)}\right)$, associated with color tuple (c_3^t, c_3^t).

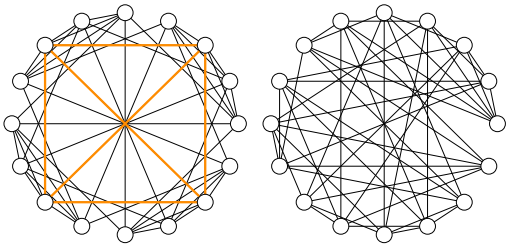

Figure 6: Rooks 4x4 graph and the Shrikhande graph: Strongly Regular non-isomorphic graphs with parameters SR(16,6,2,2). Our approach can distinguish them: only Rook's graph (left) possesses 4-cliques (orange) and thus the two graphs are associated with distinct clique complexes. The 3-WL test fails to distinguish them.

For *any* $\mathbf{v} \in \mathcal{V}^2_{G(3)}$ we then have

$$c^{t+1}_{\mathbf{v}} = \text{HASH}\Big(c^t_3, M^{F,t}(\mathbf{v})\Big)$$
$$M^{F,t}(\mathbf{v}) = \{\{\underbrace{(c^t_1, c^t_3)}_{1 \text{ time}}, \underbrace{(c^t_3, c^t_1)}_{1 \text{ time}}, \underbrace{(c^t_2, c^t_2)}_{\mu \text{ times}}, \underbrace{(c^t_2, c^t_3)}_{d-\mu \text{ times}}, \underbrace{(c^t_3, c^t_2)}_{d-\mu \text{ times}}, \underbrace{(c^t_3, c^t_3))}_{k \text{ times}}\}\}.$$

This proves the induction and confirms that all tuples in the same partition have the same colour at any colour refinement time step t.

If the colouring is constant within partitions at any 2-FWL step, then the colour histogram associated with a graph at step t purely depends on the cardinality of each of the three. We notice that, for *any* $G \in$ *SR(n,d,λ,μ)*, they are completely determined by the first two parameters:

$$|\mathcal{V}^2_{G(1)}| = n, \quad |\mathcal{V}^2_{G(2)}| = nd, \quad |\mathcal{V}^2_{G(3)}| = |\mathcal{V}^2_G| - (n + nd).$$

Given the above, any two graphs $G_1, G_2 \in$ *SR(n,d,λ,μ)* are associated with the same colour histograms at any step of the 2-FWL procedure and, therefore, cannot possibly deemed non-isomorphic by the last. □

We leverage on Lemma 19 to prove Theorem 3.

Proof of Theorem 3. In virtue of Lemma 19 and the fact that 2-FWL is as powerful as 3-WL, Theorem 3 is proved by exhibiting a pair of Strongly Regular graphs in the same family that are distinguished by the SWL test. This pair is given by the two graphs in Figure 6: Rook's 4x4 graph (G_1) and the Shrikhande graph (G_2), (the only) members of the *SR(16,6,2,2)* family of Strongly Regular graphs. The SWL test which considers their clique 3-complexes distinguish them due to the fact that, differently from G_1, G_2 possesses no 4-cliques, thus its associated complex has no 3-simplices. □

C Linear Regions of GNN, SCNN and MPSN

While the WL test has been used for studying the expressive power of GNNs, other tools have been used to study the expressive power of conventional neural networks, like fully connected and convolutional. One such tool is based on the number of linear regions of networks using piece-wise linear activations (Pascanu et al., 2013; Montúfar et al., 2014). This number has been used to draw distinctions between the expressive power of shallow and deep network architectures (Pascanu et al., 2013; Montúfar et al., 2014). It can also be related to the approximation properties of the networks (Telgarsky, 2016) and it has also been considered to shed light into the representational power of (standard) convolutional networks (Xiong et al., 2020). We show how this tool can also be used to approximate the number of linear regions of the functions represented by GNNs and message passing simplicial networks. We focus on the case where the message function is a linear layer and AGG is sum followed by ReLU. We obtain new results in all cases, showing superior capacity of MPSNs. A network with piece-wise linear gates computes functions that are piece-wise linear over the input space. A linear region is a maximal connected subset of the input over which the function is affine.

Background on hyperplane arrangements A function $f\colon \mathbb{R}^N \to \mathbb{R}^M$ is a *piecewise linear function* if its graph $\{(x, f(x))\colon x \in \mathbb{R}^N\} \subseteq \mathbb{R}^N \times \mathbb{R}^M$ consists of a finite number of polyhedral pieces. Projecting these polyhedra back onto $\mathbb{R}^N$ by $(x, y) \mapsto x$ defines a polyhedral subdivision of $\mathbb{R}^N$. The *linear regions* of the function are the N dimensional pieces in this subdivision. These are the (inclusion maximal) connected regions of the input space where the function is affine linear.

Let $\psi\colon \mathbb{R} \to \mathbb{R}; s \mapsto \max\{0, s\}$ be the linear rectification. A ReLU with N inputs defines a function $y\colon x \mapsto \psi(w^\top x)$, which for any fixed value of the weight vector $w \in \mathbb{R}^N$, $w \neq 0$, has gradient with respect to the input vector $x \in \mathbb{R}^N$ equal to 0 on the open halfspace $\{x\colon w^\top x < 0\}$ and equal to w on the open halfspace $\{x\colon w^\top x > 0\}$. Hence a ReLU defines a piecewise linear function with two linear regions. A layer of ReLUs $\psi(w_i^\top x)$, $i = 1, \ldots, M$ has linear regions given by the intersection of linear regions of the individual ReLUs. The number of linear regions of the function represented by the layer is equal to the number of connected components that are left behind once we remove $\cup_{i=1}^M A_i$ from $\mathbb{R}^N$, where $A_i = \{x \in \mathbb{R}^N\colon w_i^\top x = 0\}$ is the hyperplane dividing the two linear regions of the ith ReLU. Hence the linear regions of a layer of ReLUs can be described in terms of a *hyperplane arrangement*, i.e. a collection $\mathcal{A} = \{A_i\colon i = 1, \ldots, M\}$ of hyperplanes.

An arrangement of hyperplanes in $\mathbb{R}^N$ is *in general position* if the intersection of any k hyperplanes in the arrangement has the expected dimension. We will focus on *central* arrangements, where each hyperplane contains the origin. A central arrangement is in general position when the normal vectors $w_{i_1}, \ldots, w_{i_k}$ of any $k \leq N$ of the hyperplanes are linearly independent. The following well-known result from the theory of hyperplane arrangements will be particularly important in our discussion.

Theorem 20. *Let $\mathcal{A}$ be a central arrangement of M hyperplanes in $\mathbb{R}^N$ in general position. Then the number of regions of the arrangement, denoted $r(\mathcal{A})$, is equal to $2\sum_{j=0}^{N-1}\binom{M-1}{j}$. This is also the maximum number of regions of any central arrangement of M hyperplanes in $\mathbb{R}^N$.*

This result can be derived from Zaslavsky's theorem (Zaslavsky, 1975), which expresses the number of regions of an arbitrary arrangement, not necessarily in general position, in terms of properties of a partially ordered set, namely the collection of intersections of the hyperplanes in the arrangement partially ordered by reverse inclusion.

We will focus on central arrangements, but we point out that similar results to Theorem 20 can be derived for the case of non-central hyperplane arrangements. An non-central arrangement of (affine) hyperplanes in $\mathbb{R}^N$ is in general position when any $k \leq N$ of the hyperplanes intersect in a set of dimension $N - k$, and any $k > N$ of the hyperplanes have an empty intersection. For such an arrangement, the number of regions is $\sum_{j=0}^{N}\binom{M}{j}$.

The main challenges in computing the number of regions defined hyperplane arrangements happen when the hyperplanes satisfy some type of constraints and are not in general position. The type of layers that we discuss in the following correspond to central arrangements subject to certain constraints, namely that the normal vectors are the rows of a matrix with a particular block Kronecker product structure.

GNN The result in Theorem 11 contrasts with the following complexity of the plain GNNs. We start with the simple case of Graph Neural Networks (GNNs). A graph $G = (V, E, \omega)$ is a set of triplets with vertices $V = \{v_i\}_{i=1}^{S_0}$, edges $E \subseteq V \times V$, and edge weight function $\omega\colon E \to \mathbb{R}$. The graph has an adjacency matrix A with the (i, j)th entry $a_{ij} = \omega(v_i, v_j)$. Each node has a d-dimensional feature, and we collect the feature vectors into a matrix $X \in \mathbb{R}^{S_0 \times d}$. We consider a GNN convolutional layer of the form

$$X^{\text{out}} = \psi\left(\mathcal{H}(A, X^{\text{in}})W_0\right), \tag{9}$$

where ψ is the entry-wise ReLU, $\mathcal{H}(A, X)$ is an aggregation mapping, and $W_0 \in \mathbb{R}^{d \times m}$ are the trainable weights. We restate in the following theorem for Theorem 9 and give at full proof.

Theorem 21 (Number of linear regions of a GNN layer)**.** *Consider a graph G with S_0 nodes, node input features of dimension $d \geq 1$, and node output features of dimension m. Suppose the aggregation function $\mathcal{H}$ as function of X is linear and invertible. Then, the number of linear regions of the functions represented by a ReLU GNN layer (9) has the optimal upper bound*

$$R_{\text{GNN}} = \left(2\sum_{i=0}^{d-1}\binom{m-1}{i}\right)^{S_0}.$$

This applies to aggregation functions with no trainable parameters including GCN convolution (Kipf & Welling, 2017), spectral GNN (Defferrard et al., 2016; Bruna et al., 2014), and traditional message passing (Gilmer et al., 2017).

Proof. For simplicity, we write $Y = \mathcal{H}(A, X)^T \in \mathbb{R}^{d \times S_0}$ and $V = W^T \in \mathbb{R}^{m \times d}$. We denote $Y_{:j}$ the $j$the column of Y, and $V_{i:}$ the ith row of V. The GNN layer defines hyperplanes, for $i = 1, \ldots, m,\ j = 1, \ldots, S_0$,

$$A_{ij} := \left\{ Y \in \mathbb{R}^{d \times S_0} : V_{i:} Y_{:j} = 0 \right\}.$$

The arrangement $\mathcal{A} = \{A_{ij} \colon i = 1, \ldots, m, j = 1, \ldots, S_0\}$ is a direct sum of the arrangements $\mathcal{A}_j = \{A_{ij} \colon i = 1, \ldots, m\}$, $j = 1, \ldots, S_0$. It can be shown (see Zaslavsky, 1975) that this implies $r(\mathcal{A}) = \prod_{j=1}^{S_0} r(\mathcal{A}_j)$.

Each $\mathcal{A}_j$ is an arrangement of m hyperplanes in $\mathbb{R}^N$, $N = dS_0$, whose normals span a subspace of dimension at most d, irrespective of S_0. Counting the number of regions defined by $\mathcal{A}_j$ is equivalent to counting the number of regions defined by its *essentialization* $\mathrm{ess}(\mathcal{A}_j)$, which is the arrangement that the hyperplanes define on the span of their normal vectors. We can regard $\mathrm{ess}(\mathcal{A}_j)$ as a (central) arrangement of m hyperplanes in $\mathbb{R}^d$ with normals $V_{i:} \in \mathbb{R}^d$, $i = 1, \ldots, m$. For generic choices of the weight matrix $W^\top = V$, this is a central arrangement in general position. Hence, by Theorem 20, $r(\mathcal{A}_j) = r(\mathrm{ess}(\mathcal{A}_j)) = 2 \sum_{i=0}^{d-1} \binom{m-1}{i}$.

For the number of regions of the entire arrangement $\mathcal{A}$, corresponding to the number of linear regions of the function expressed by the layer, we obtain $R_{\mathrm{GNN}} = r(\mathcal{A}) = \prod_{j=1}^{S_0} r(\mathcal{A}_j) = \left(2 \sum_{i=0}^{d-1} \binom{m-1}{i}\right)^{S_0}$. This concludes the proof. □

The above result should be compared with the optimal upper bound for a standard dense ReLU layer without biases, $\psi(Wx)$, which for d inputs and m outputs is $2 \sum_{i=0}^{d-1} \binom{m-1}{i}$.

The invertibility condition for the aggregation function $\mathcal{H}$ can be relaxed, but is satisfied by many commonly used graph convolutions: i) For an undirected graph, the normalised adjacency matrix has non-negative eigenvalues. If the eigenvalues are all positive, the aggregation function is invertible. ii) The Fourier transform is the square matrix of eigenvectors, as used in spectral GNN (Bruna et al., 2014). When the graph Laplacian is non-singular, Fourier transform matrix, that is aggregation mapping is invertible. iii) For the transform Φ by graph wavelet basis, Haar wavelet basis or graph framelets, the Φ are all invertible (Xu et al., 2019a; Li et al., 2020; Zheng et al., 2020a;b; 2021; Wang et al., 2020). So the bound in Theorem 21 applies to them.

SCNN Simplicial Complex Neural Networks (SCNNs) were proposed by Ebli et al. (2020). An SCNN layer for features on a p-simplicial complex is defined in terms of matrices M_n (e.g. simplicial Laplacian) by

$$H_n^{\mathrm{out}} = \psi\big(M_n H_n^{\mathrm{in}} W_n\big), \quad n = 0, \ldots, p. \tag{10}$$

In this type of layer, the features on simplices of different dimensions $n = 0, 1 \ldots, p$ are computed in parallel. The following theorem is a restatement of Theorem 10, for which we now give full proof and explanation.

Theorem 22 (Number of linear regions for an SCNN layer)**.** *Consider a p-dimensional simplicial complex with S_n n-simplicies for $n = 0, 1, \ldots, p$. Suppose that each simplicial Laplacian in dimension n L_n is invertible. Then the maximum number of linear regions of the functions represented by a ReLU SCNN layer $\mathcal{N}$ of the form (10) is*

$$R_{\mathcal{N}} = \prod_{n=0}^{p} \left(2 \sum_{i=0}^{d_n - 1} \binom{m_n}{i} \right)^{S_n}, \tag{11}$$

where, for each of the n-simplices, d_n is the input feature dimension and m_n is the number of output features. Consider a p-dimensional simplicial complex with S_n n-simplicies for $n = 0, 1, \ldots, p$. Suppose that each simplicial Laplacian in dimension n L_n M_n is invertible. Then the number of linear

regions of the functions represented by a ReLU SCNN layer (10) of the form $H_n^{\text{out}} = \psi(L_n H_n^{\text{in}} W_n)$, $n = 1, \ldots, p$, has the optimal upper bound

$$R_{\text{SCNN}} = \prod_{n=0}^{p} \left(2 \sum_{i=0}^{d_n - 1} \binom{m_n - 1}{i} \right)^{S_n},$$

where, for each n-simplex, d_n is the input feature dimension and m_n is the number of output features.

The product over n in (22) reflects the fact that the features over simplices of different dimensions do not interact. The GNN bound in Theorem 21 is recovered as the special case of the SCNN bound with $p = 0$.

It is instructive to compare the SCNN bound in Theorem 22 with the optimal bound for a dense layer. By Roth's lemma (Roth, 1934), $\text{vec}(M_n H_n^{\text{in}} W_n) = (W_n^\top \otimes M_n) \cdot \text{vec}(H_n^{\text{in}})$, where vec denotes column-by-column vectorization and $\otimes$ the Kronecker product. Hence, for each n, we can regard the SCNN layer as a standard layer $\psi(Ux)$ with weight matrix $U = (W_n^\top \otimes M_n) \in \mathbb{R}^{(m_n S_n) \times (S_n d_n)}$ and input vector $x = \text{vec}(H_n^{\text{in}}) \in \mathbb{R}^{S_n d_n}$. Notice that for the SCNN layer, the weight matrix has a specific structure. A standard dense layer with $S_n d_n$ inputs and $m_n S_n$ ReLUs with generic weights and no biases computes functions with $2 \sum_{i=0}^{S_n d_n - 1} \binom{m_n S_n - 1}{i}$ regions.

Proof of Theorem 22. By the definition of the SCNN layer, for each dimension n, each of the S_n n-simplices in the simplicial complex has d_n input features. Similar to the proof of Theorem 21, the arrangement for the n-dimensional simplices corresponds to a direct sum of S_n arrangements, each of m_n hyperplanes in $\mathbb{R}^{d_n}$. Hence the number of linear regions for this part of the complex is

$$\left(2 \sum_{i=0}^{d_n - 1} \binom{m_n - 1}{i} \right)^{S_n}. \tag{12}$$

Now for the entire layer, the arrangements for the different n are also combined as a direct sum, so that their number of regions multiply. We have n ranging from dimension 0 to p, so that

$$\prod_{n=0}^{p} \left(2 \sum_{i=0}^{d_n - 1} \binom{m_n - 1}{i} \right)^{S_n}. \tag{13}$$

This concludes the proof. □

MPSN In our Message Passing Simplicial Network (MPSN), the features on simplices of different dimensions are allowed to interact. For a p-dimensional complex, denote the set of n-simplices by $\mathcal{S}_n$ with $S_n = |\mathcal{S}_n|$. Denote the n-simplex input feature dimension by d_n, and the output feature dimension by $m_n = m$, $n = 0, \ldots, p$. We consider an MPSN layer with linear message functions, sum aggregation for all messages and an update function taking the sum of the messages followed by a ReLU activation. For each $v \in \mathcal{S}_n$, the output feature vector takes the form:

$$h_{n,v}^{\text{out}} = \psi\Big(\sum_{w \in \mathcal{S}_n} M_{n,v,w} h_{n,w}^{\text{in}} W_n + \sum_{w \in \mathcal{S}_{n-1}} U_{n,v,w} h_{n-1,w}^{\text{in}} W_{n-1} + \sum_{w \in \mathcal{S}_{n+1}} O_{n,v,w} h_{n+1,w}^{\text{in}} W_{n+1} \Big), \tag{14}$$

where ψ is an entry-wise activation ($s \mapsto \max\{0, s\}$ for ReLU), $W_n \in \mathbb{R}^{d_n \times m_n}$ are trainable weight matrices, $\mathcal{F}(v) \subseteq \mathcal{S}_{n-1}$ denotes the set of $(n-1)$-faces and $\mathcal{C}(v) \subseteq \mathcal{S}_{n+1}$ the set of $(n+1)$-co-faces (containing faces) of $v \in \mathcal{S}_n$ in the simplicial complex. Further, $U_n \in \mathbb{R}^{S_n \times S_{n-1}}$, $M_n \in \mathbb{R}^{S_n \times S_n}$ and $O_n \in \mathbb{R}^{S_n \times S_{n+1}}$ are some choice of adjacency matrices for the simplicial complex. These could be, for instance, the Hodge Laplacian matrix L_n and the corresponding boundary matrices B_n, $B_{n+1}^\top$ that relate simplicies to their faces and cofaces (Barbarossa & Sardellitti, 2020).

We can rewrite (14) more generality and more concisely as follows. For each n the output features on $\mathcal{S}_n$ can be written as

$$\begin{aligned} H_n^{\text{out}} &= \psi(M_n H_n + U_n H_{n-1} + O_n H_{n+1}) \\ &= \psi\left([U_n H_{n-1} | L_n H_n | O_n H_{n+1}] \begin{bmatrix} W_{n-1} \\ W_n \\ W_{n+1} \end{bmatrix} \right), \end{aligned} \tag{15}$$

for some fixed matrices $U_n \in \mathbb{R}^{S_n \times S_{n-1}}$, $M_n \in \mathbb{R}^{S_n \times S_n}$ and $O_n \in \mathbb{R}^{S_n \times S_{n+1}}$ depending only on the simplicial complex. To avoid clutter, we omit the superscript "in" of the input feature matrices H_n. Concatenating (15) for all n, we can write the entire MPSN layer as

$$\begin{bmatrix} H_0^{\text{out}} \\ H_1^{\text{out}} \\ H_2^{\text{out}} \\ \vdots \\ H_p^{\text{out}} \end{bmatrix} = \psi\left(\begin{bmatrix} M_0 H_0 & O_0 H_1 & & & \\ U_1 H_0 & M_1 H_1 & O_1 H_2 & & \\ & U_2 H_1 & M_2 H_2 & O_2 H_3 & \\ & & & \ddots & \end{bmatrix} \begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ \vdots \\ W_p \end{bmatrix} \right). \tag{16}$$

We will use this representation (or rather (15)) in the proof of the first bound in Theorem 26 (which is a restatement of Theorem 11).

It is also useful to write the linear function in standard form. Using Roth's lemma, we can write (15) as

$$\text{vec}(H_n^{\text{out}}) = \psi\Big(\big[W_{n-1}^\top \otimes U_n | W_n^\top \otimes M_n | W_{n+1}^\top \otimes O_n \big] \, \text{vec}\Big(\big[H_{n-1} | H_n | H_{n+1} \big] \Big) \Big). \tag{17}$$

Now, concatenating over n yields the expression $\psi(WH)$ from (18) for the entire layer,

we can write (14) as (details in Appendix C)

$$H^{\text{out}} = \psi(W H^{\text{in}}), \tag{18}$$

where $H^{\text{in}} = \text{vec}([H_0^{\text{in}} | H_1^{\text{in}} | \cdots | H_p^{\text{in}}]) \in \mathbb{R}^N$, $N = \sum_{n=0}^p S_n d_n$, $H^{\text{out}} = \text{vec}([H_0^{\text{out}} | H_1^{\text{out}} | \cdots | H_p^{\text{out}}]) \in \mathbb{R}^M$ with $M = \sum_{n=0}^p S_n m$, and

$$W = \begin{bmatrix} W_0^\top \otimes M_0 & W_1^\top \otimes O_0 & & \\ W_0^\top \otimes U_1 & W_1^\top \otimes M_1 & W_2^\top \otimes O_1 & \\ & W_1^\top \otimes U_2 & W_2^\top \otimes M_2 & W_3^\top \otimes O_2 \\ & & & \ddots \end{bmatrix}. \tag{19}$$

We study the number of linear regions of the function (18) with ReLU based on the matrix $W \in \mathbb{R}^{M \times N}$. For each of the output coordinates $i \in \{1, \ldots, M\}$, the ReLU splits the input space $\mathbb{R}^N$ into two regions separated by a hyperplane $\{H^{\text{in}} \in \mathbb{R}^N \colon W_{i:} H^{\text{in}} = 0\}$ with normal $W_{i:}^\top \in \mathbb{R}^N$.

In order to count the total number of regions, we will use results from the theory of hyperplane arrangements. Zaslavsky (1975, Theorem A) shows that the number of regions $r(\mathcal{A})$ defined by an arrangement $\mathcal{A}$ of hyperplanes in $\mathbb{R}^N$ is

$$r(\mathcal{A}) = (-1)^N \chi_{\mathcal{A}}(-1),$$

where $\chi_{\mathcal{A}}$ is the characteristic polynomial of the arrangement. By virtue of a theorem of Whitney (see Stanley 2004, Theorem 2.4 and Orlik & Terao 1992, Lemma 2.55), it can be written as $\chi_{\mathcal{A}}(t) = \sum (-1)^{|\mathcal{B}|} t^{N - \text{rank}(\mathcal{B})}$, where the sum runs over subarrangements $\mathcal{B} \subseteq \mathcal{A}$ that are central (hyperplanes in $\mathcal{B}$ have a nonempty intersection), and $\text{rank}(\mathcal{B})$ denotes the dimension spanned by the normals to the hyperplanes in $\mathcal{B}$. In our case, $\mathcal{A}$ is a central arrangement with normals given by the rows of the matrix W in (19). Hence:

Lemma 23. *The number of linear regions of the function (18) with $W \in \mathbb{R}^{M \times N}$ and ψ being ReLU is equal to*

$$r(\mathcal{A}) = \sum_{B \subseteq \{1, \ldots, M\}} (-1)^{|B| - \text{rank}(W_{B:})},$$

where $W_{B:}$ denotes the submatrix of rows $i \in B$.

This formula counts the linear regions of any particular function represented by our layer. Some interesting cases can be computed explicitly.

Proposition 24. *Consider some $K \leq N$. If $\text{rank}(W_{B:}) = \min\{|B|, K\}$ for any B, then $r(\mathcal{A}) = 2 \sum_{j=0}^{K-1} \binom{M-1}{j}$.*

Proof. This result is known in theory of partial orders and hyperplane arrangements. We include a proof which illustrates the evaluation of the characteristic polynomial. If there is K so that $\text{rank}(W_{B:}) = \min\{|B|, K\}$ for all B, then Lemma 23 can be evaluated as

$$r(\mathcal{A}) = \sum_B (-1)^{|B| - \min\{|B|, K\}}$$

$$
\begin{aligned}
&= \sum_{j=0}^{K} \sum_{B \in \binom{\{1,\dots,M\}}{j}} 1 + \sum_{j=K+1}^{M} \sum_{B \in \binom{\{1,\dots,M\}}{j}} (-1)^{j-K} \\
&= \sum_{j=0}^{K} \binom{M}{j} + (-1)^{M-K} \sum_{j=0}^{M-(K+1)} \binom{M}{j} (-1)^{j} \\
&= \sum_{j=0}^{K} \binom{M}{j} + (-1) \binom{M-1}{K} \\
&= 2 \sum_{j=0}^{K-1} \binom{M-1}{j},
\end{aligned}
$$

which is what was claimed. □

We can characterize cases where the hypothesis of Proposition 24 is satisfied for particular values of K. For instance, using the tridiagonal block structure of W in (19):

Proposition 25. *If $m \leq d_n$ and $\operatorname{rank}(O_n) = S_n$ for $n = 0, \dots, p$, then $\operatorname{rank}(W_{B:}) \geq \min\{|B|, K\}$, where $K = N - m \cdot \max\{S_0, \dots, S_p\}$.*

Proof. Since matrix W from (19) is lower block triangular, we have

$$
\operatorname{rank}(W) \geq \operatorname{rank}\left([W_0^\top \otimes M_0 | W_1^\top \otimes O_0]\right) + \sum_{n=1}^{p-2} \operatorname{rank}\left(W_{n+1}^\top \otimes O_n\right) + \operatorname{rank}\left(\begin{bmatrix} W_p^\top \otimes O_{p-1} \\ W_p^\top \otimes M_p \end{bmatrix}\right).
$$

Since the matrix W is also upper block triangular, we have a similar bound in terms of the U_n. Using the rank formula for Kronecker products $\operatorname{rank}(W_n^\top \otimes M_n) = \operatorname{rank}(W_n^\top)\operatorname{rank}(M_n)$, we see that if all the factor matrices are full rank, then $\operatorname{rank}(W_{B:}) = \sum_{n=0}^{p-1} \min\{d_n, m\} S_n$ for any $B \subseteq \{1, \dots, \sum_{n=0}^{p-1} m S_n\}$. Recall that $W_{B:}$ denotes the submatrix of W composed of the rows $i \in B$. □

With the above proposition, we would obtain the following bounds for counting the linear regions of an MPSN layer. This is a restatement of Theorem 11. Here we give more details of the meaning and proof of the bounds.

Theorem 26 (Number of linear regions of an MPSN layer)**.** *With the above settings, the maximum number of linear regions of the functions represented by a ReLU MPSN layer (18) is upper bounded by*

$$
R_{\text{MPSN}} \leq \prod_{n=0}^{p} \left(2 \sum_{i=0}^{d_{n-1}+d_n+d_{n+1}-1} \binom{m-1}{i} \right)^{S_n}, \tag{20}
$$

where we set $d_{-1} = d_{p+1} = 0$. We also note the 'trivial' upper bound, with $N := \sum_{n=0}^{p} S_n d_n$ and $M := \sum_{n=0}^{p} S_n m$, $R_{\text{MPSN}} \leq 2 \sum_{j=0}^{N-1} \binom{M-1}{j}$. Moreover, if $m \leq d_n$ and $\operatorname{rank}(O_n) = S_n$ for all n, writing $S = \max\{S_0, \dots, S_p\}$, we have the lower bound

$$
R_{\text{MPSN}} \geq 2 \sum_{j=0}^{N-(mS)-1} \binom{M-1}{j}.
$$

We note that the MPSN lower bound (26) surpasses the SCNN upper bound (22), with order $\prod_n (\sum_{n'} m S_{n'})^{d_n S_n}$ and $\prod_n (m S_n)^{d_n S_n}$, respectively, when $d_n \geq m$. The GNN bound (21) is a special case of the SCNN bound (22) with $p = 0$, and has asymptotic order $m^{d_0 S_0}$.

It is not difficult to obtain minor case by case improvements of the bounds in Theorem 26 by conducting a more careful analysis of the row independencies in matrix W for specific values of the input feature dimensions $d_0, \dots, d_p$, output feature dimension m, numbers of simplices $S_0, \dots, S_p$, and the structure of the matrices U_n, M_n, O_n, $n = 0, \dots, p$.

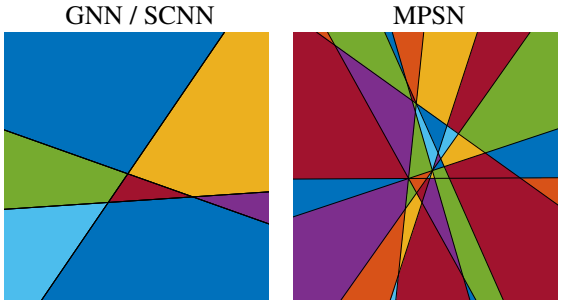

Figure 7: Shown is a 2D slice of the input vertex feature space of SCNN and MPSN layers with $S_0 = S_1 = 3$, $d_0 = d_1 = 1$, $m = 3$, computing output vertex features, with input edge features set as a random linear function of the input vertex features.

Proof of Theorem 26. The proof of the fist upper bound is analogous to Theorem 22. The difference is now we consider also the faces and cofaces that interact with an n-simplex in the MPSN. We use the expression (15). The difference compared with Theorem 22 lies in the number of input features for each n, which here results in

$$\prod_{n=0}^{p} \left(2 \sum_{i=0}^{d_{n-1}+d_n+d_{n+1}-1} \binom{m-1}{i} \right)^{S_n}, \tag{21}$$

which is the first upper bound. The second upper bound is the trivial upper bound, which is the maximum possible number of regions of a central arrangement of M hyperplanes in $\mathbb{R}^N$ from Theorem 20. The lower bound follows from inserting Proposition 25 into Proposition 24. □

The regions for the three network architectures are illustrated in Figure 2 for a complex with $S_0 = S_1 = 3$ and $S_2 = 1$, each input dimension 1 and output dimension $m = 3$. It shows that from GNN, SCNN to MPSN the number of linear regions increases in turn, consistent with the theory.

MPSN with populated higher-features To conclude this section, we consider a situation of interest, where we are given a simplicial complex but only vertex features are available. To still exploit the structure of the simplicial complex, we can populate the higher features as linear functions of the vertex features. In this case (18) becomes $\psi(W' \operatorname{vec}(H_0^{\text{in}}))$, where $W' = WC$ for some matrix $C \in \mathbb{R}^{(\sum_n d_n S_n) \times (d_0 S_0)}$. We show that this strategy indeed can increase the complexity of the represented functions.

Proposition 27 (MPSN with populated higher-features)**.** *Consider an arbitrary simplicial complex and an MPSN layer mapping $\mathbb{R}^{S_0 \times d_0} \to \mathbb{R}^{S_0 \times m}; H_0^{\text{in}} \mapsto H_0^{\text{out}}$, whereby higher-dimensional input features are populated as linear functions of the input vertex features. Consider further the corresponding SCNN layer which computes $H_0^{\text{out}} = \psi(L_0 H_0^{\text{in}} W_0)$. Then, $R_{\text{MPSN}} \geq R_{\text{SCNN}}$. Moreover, for certain simplicial complexes and feature dimensions $d_0, \ldots, d_p, m$, the above inequality is strict.*

Proof. Focusing on the $\mathcal{S}_0$ output features, the relevant matrices are $[W_0^\top \otimes M_0]$ for the simplicial network and $[W_0^\top \otimes M_0 | W_1^\top \otimes O_0]C$ for the MSPN with populated higher-dimensional features. Both matrices have format $mS_0 \times dS_0$ and rank $\min\{m, d_0\} \operatorname{rank}(M_0)$. However, subsets of rows have different ranks in both cases. For illustration, if $d = 1$ and l is the smallest number of nonzero entries of any row in M_0, then $[W_0^\top M_0]$ has an m row submatrix of rank $\min\{m, l\}$. In contrast, an m row submatrix of the augmented matrix will have rank $\min\{m, l + l'\}$, where l' is the smallest number of nonzero entries of a row in O_0. □

The regions for the two cases are illustrated in Figure 7. It shows MPSN (Right) has more regions than GNN/SCNN (Left) and has a higher complexity for populated case.

D RELATIONSHIP TO CONVOLUTIONS

Background on Hodge Laplacian The simplicial convolutions described in this section rely on the Hodge Laplacians of the simplicial complex. Such a Laplacian operator L_p is associated with each dimension p of the complex. The operator L_p has a special structure that depends on the boundary operators of the corresponding dimensions B_p and B_{p+1}. These matrices are the discrete equivalent of the boundary operators encountered in algebraic topology (see Schaub et al. (2020) for more details). They essentially describe things such as the fact that boundary of a (filled) triangle is formed by its edges with some particular choice of orientation (positive or negative). The Laplacian can be written as a function of these boundary matrices as $L_p = B_p^T B_p + B_{p+1} B_{p+1}^T$. We denote the first term by $L_p^{\downarrow}$ and the second by $L_p^{\uparrow}$, because they encode (orientation-aware) up and down adjacencies between the p-simplicies of the complex. Additionally, let us recall the important relation $B_p B_{p+1} = \mathbf{0}$ (i.e. the zero map). This equation, which is fundamental in homology theory, formally specifies that the boundary of a simplex has no boundary (e.g. the boundary of a filled triangle has no boundary because it forms a loop and, therefore, it has no endpoints).

We emphasise that the boundary matrices and the Laplacian and, consequently, the *convolutions which are based on these Laplacian operators depend on an arbitrary choice of orientation for the simplicial complex*. These operators then cannot be directly applied to tasks involving multiple simplicial complexes (unless their orientation is given) because one has to find a consistent orientation between them or find a way to relate the different orientation basis. Therefore, unlike MPSNs, these convolutional operators cannot be directly employed in general to clique complexes or other types of simplicial complexes derived from arbitrary graphs.

Simplicial Neural Networks Ebli et al. (2020) presented Simplicial Neural Networks (SNNs), neural network models extending convolutional layers to attributed simplicial complexes. The theoretical construction closely resembles the one by Defferrard et al. (2016), where the standard graph Laplacian is simply replaced by the more general Hodge p-Laplacian L_p, i.e. the generalization of the Laplacian operator to simplices of order p. The p-th order SNN convolutional layer is

$$\mathcal{F}_p^{-1}(\phi_W) *_p c = \psi\left(\sum_{r=0}^{R} W_r L_p^r c\right), \tag{22}$$

where ϕ_W is a convolutional filter with learnable parameters W, $c \in C^p(K)$ is a p-cochain on input simplicial complex K (i.e. a real valued function over the set of p-simplices in K) and ψ accounts for the application of bias and non-linearity. In particular, the convolutional filter ϕ_W is parameterized as an R-degree polynomial of the Hodge p-Laplacian L_p. By imposing a small degree R, it is possible to guarantee spatial filter localization similarly as in graphs.

This proposed convolutional layer can easily be rewritten in terms of our message passing scheme. Let us first conveniently introduce the following Lemma:

Lemma 28. *The r-th power of the Hodge p-Laplacian, L_p^r, is equivalent to the sum of the r-th powers of its constituent upper and lower components, that is: $L_p^r = (L_p^{\downarrow} + L_p^{\uparrow})^r = (L_p^{\downarrow})^r + (L_p^{\uparrow})^r$.*

Proof. We prove the lemma by induction on the power exponent r. As the base case, we consider exponent $r = 1$, for which the equivalence clearly holds as $L_p^1 = (L_p^{\downarrow} + L_p^{\uparrow})^1 = L_p^{\downarrow} + L_p^{\uparrow} = (L_p^{\downarrow})^1 + (L_p^{\uparrow})^1$.

For the induction step, we assume that $L_p^{r-1} = (L_p^{\downarrow})^{r-1} + (L_p^{\uparrow})^{r-1}$ holds true and prove that $L_p^r = (L_p^{\downarrow})^r + (L_p^{\uparrow})^r$. L_p^r is defined as $L_p^r = L_p L_p^{r-1} = (L_p^{\downarrow} + L_p^{\uparrow})(L_p^{\downarrow} + L_p^{\uparrow})^{r-1}$.

By the induction hypothesis: $(L_p^{\downarrow} + L_p^{\uparrow})(L_p^{\downarrow} + L_p^{\uparrow})^{r-1} = (L_p^{\downarrow} + L_p^{\uparrow})\big((L_p^{\downarrow})^{r-1} + (L_p^{\uparrow})^{r-1}\big) = L_p^{\downarrow}(L_p^{\downarrow})^{r-1} + L_p^{\downarrow}(L_p^{\uparrow})^{r-1} + L_p^{\uparrow}(L_p^{\downarrow})^{r-1} + L_p^{\uparrow}(L_p^{\uparrow})^{r-1} = (L_p^{\downarrow})^r + L_p^{\uparrow}(L_p^{\downarrow})^{r-1} + L_p^{\downarrow}(L_p^{\uparrow})^{r-1} + (L_p^{\uparrow})^r$. The second term $L_p^{\uparrow}(L_p^{\downarrow})^{r-1}$ is rewritten as $B_{p+1}B_{p+1}^T(B_p^T B_p)^{r-1} = B_{p+1}B_{p+1}^T \underbrace{B_p^T B_p}_{(r-1)\text{ times}} = \mathbf{0}$, since $B_{p+1}^T B_p^T = (B_p B_{p+1})^T$, which equals $\mathbf{0}^T$ by definition. Similarly, the third term $L_p^{\downarrow}(L_p^{\uparrow})^{r-1}$ is rewritten as $B_p^T B_p (B_{p+1}B_{p+1}^T)^{r-1} = B_p^T B_p \underbrace{B_{p+1}B_{p+1}^T}_{(r-1)\text{ times}} = \mathbf{0}$, since, again, $B_p B_{p+1} = \mathbf{0}$. □

We now proceed to prove Theorem 7 that our MPSN can be reduced to SCNN of Ebli et al. (2020) or Bunch et al. (2020). We split the proofs in two individuals.

Proof of Theorem 7 in reference to Ebli et al. (2020). We first rephrase Equation 22 for a generic layer and multi-dimensional input and output p-simplicial representations

$$H^{t+1} = \psi\Big(\sum_{r=0}^{R} L_p^r H^t W_r^{t+1}\Big) = \psi\Big(H^t W_0^{t+1} + \sum_{r=1}^{R} L_p^r H^t W_r^{t+1}\Big).$$

The convolutional operation for p-simplex v can be rewritten as

$$\begin{aligned} h_v^{t+1} &= \psi\Big(h_v^t W_0^{t+1} + \sum_{r=1}^{R} (L_p^r)_v H^t W_r^{t+1}\Big) \\ &= \psi\Big(h_v^t W_0^{t+1} + \sum_{r=1}^{R} \sum_{w \in \mathcal{S}_p} (L_p^r)_{v,w} h_w^t W_r^{t+1}\Big), \end{aligned}$$

where h_v^{t+1} denotes the v-th row of matrix H^{t+1}, $(L_p^r)_v$ denotes the v-th row of operator L_p^r and $(L_p^r)_{v,w}$ its entry at position v, w. We now leverage on Lemma 28 to rewrite the convolution operation on simplex v as

$$\begin{aligned} h_v^{t+1} &= \psi\Big(\sum_{r=1}^{R} \sum_{w \in \mathcal{S}_p} \left(((L_p^{\downarrow})^r)_{v,w} + ((L_p^{\uparrow})^r)_{v,w}\right) h_w^t W_r^{t+1} + h_v^t W_0^{t+1}\Big) \\ &= \psi\Big(\sum_{w \in \mathcal{S}_p} \sum_{r=1}^{R} ((L_p^{\downarrow})^r)_{v,w} h_w^t W_r^{t+1} + \sum_{w \in \mathcal{S}_p} \sum_{r=1}^{R} ((L_p^{\uparrow})^r)_{v,w} h_w^t W_r^{t+1} + h_v^t W_0^{t+1}\Big). \end{aligned}$$

Considering that matrices $L_p^{\downarrow}$ and $L_p^{\uparrow}$ only convey the notions of, respectively, lower and upper simplex adjacency, the equation above is easily interpreted in terms of our message passing scheme by setting

$$\begin{aligned} M_{\uparrow}^{t+1}\big(h_v^t, h_w^t, h_{v \cup w}^t\big) &= \sum_{r=1}^{R} ((L_p^{\uparrow})^r)_{v,w} h_w^t W_r^{t+1} \\ M_{\downarrow}^{t+1}\big(h_v^t, h_w^t, h_{v \cap w}^t\big) &= \sum_{r=1}^{R} ((L_p^{\downarrow})^r)_{v,w} h_w^t W_r^{t+1} \\ U^{t+1}\Big(h_v^t, \{m_i^t(v)\}_{i=\downarrow,\uparrow}\Big) &= \psi\Big(h_v^t W_0^{t+1} + m_{\uparrow}{}_v^{t+1} + m_{\downarrow}{}_v^{t+1}\Big), \end{aligned}$$

and by letting AGG be the summation over the extended notion of upper and lower R-neighborhoods, that is neighborhoods comprising p-simplices at a distance from v which is at most R (w is at distance d from v if there exists a sequence of upper- (respectively, lower-) adjacent p-simplices $[\nu_0, \nu_1, \ldots, \nu_d]$ such that $\nu_0 = v, \nu_d = w$). □

It is noteworthy that, contrary to our general proposed framework, the two message functions $M_{\uparrow}^{t+1}$ and $M_{\downarrow}^{t+1}$ share the same learnable parameters $\{W_r^{t+1}\}_{r=1}^{R}$, and that no signal of order lower or higher than p is involved in the computation.

SC-Conv Bunch et al. (2020) proposed a convolutional operator that can be applied on 2-dimensional simplicial complexes. The construction is based on the canonical normalised Hodge Laplacians defined by Schaub et al. (2020); starting from the operators, the authors generalize the Graph Convolutional Network model proposed in Kipf & Welling (2017) by defining the corresponding adjacency matrices with added self-loops:

$$X_0^{t+1} = \sigma\left(D_1^{-1} B_1 X_1^t W_{0,1}^t + \tilde{A}_0^u X_0^t W_{0,0}^t\right)$$

$$X_1^{t+1} = \sigma\left(B_2 D_3 X_2^t W_{1,2}^t + (\tilde{A}_1^d + \tilde{A}_1^u) X_1^t W_{1,1}^t + D_2 B_1^T D_1^{-1} X_0^t W_{1,0}^t\right)$$
$$X_2^{t+1} = \tilde{A}_2^d X_2^t W_{2,1}^t + D_4 B_2^T D_5^{-1} X_1^t W_{2,0}^t.$$

We defer readers to Section 2.1 of the original paper for the definitions of the $\{\tilde{A}_i^\alpha\}_{i=0}^2$ and $\{D\}_{i=1}^5$ matrices in the above equations. Differently from (Ebli et al., 2020), this scheme models the interactions between signals defined at different dimensions. It can, nonetheless, be rewritten in terms of our message passing framework.

Proof of Theorem 7 in reference to Bunch et al. (2020). We report here the derivation for message passing on 1-simplices (edges) as it is the most general. The derivation on 0- and 2-simplices can simply be obtained as a special case of this last. We first denote, for simplicity,

$$\Delta_{1,1} = \tilde{A}_1^d + \tilde{A}_1^u, \quad \Delta_{1,0} = D_2 B_1^T D_1^{-1}, \quad \Delta_{1,2} = B_2 D_3.$$

We note that the convolutional operation for a generic 1-simplex e can be rewritten as

$$\begin{aligned}
x_{1,e}^{t+1} &= \sigma\Big((\Delta_{1,2})_e X_2^t W_{1,2}^t + (\Delta_{1,1})_e X_1^t W_{1,1}^t + (\Delta_{1,0})_e X_0^t W_{1,0}^t\Big) \\
&= \sigma\Big(\sum_{u\in\mathcal{S}_2} (\Delta_{1,2})_{e,u} x_{2,u}^t W_{1,2}^t + \sum_{f\in\mathcal{S}_1} (\tilde{A}_1^u)_{e,f} x_{1,f}^t W_{1,1}^t \\
&\qquad + \sum_{f\in\mathcal{S}_1} (\tilde{A}_1^d)_{e,f} x_{1,f}^t W_{1,1}^t + \sum_{v\in\mathcal{S}_0} (\Delta_{1,0})_{e,v} x_{0,v}^t W_{1,0}^t\Big) \\
&= \sigma\Big(\sum_{u\in\mathcal{C}(e)} (\Delta_{1,2})_{e,u} x_{2,u}^t W_{1,2}^t + \Big(\sum_{f\in\mathcal{N}_\uparrow(e)} (\tilde{A}_1^u)_{e,f} x_{1,f}^t \\
&\qquad + \sum_{f\in\mathcal{N}_\downarrow(e)} (\tilde{A}_1^d)_{e,f} x_{1,f}^t + (\Delta_{1,1})_{e,e} x_{1,e}^t\Big) W_{1,1}^t + \sum_{v\in\mathcal{F}(e)} (\Delta_{1,0})_{e,v} x_{0,v}^t W_{1,0}^t\Big).
\end{aligned}$$

This equation is interpreted in terms of our message passing scheme by setting

$$\begin{aligned}
M_{1,\uparrow}^{t+1}\big(x_{1,e}^t, x_{1,f}^t, x_{2,e\cup f}^t\big) &= (\tilde{A}_1^u)_{e,f} x_{1,f}^t \\
M_{1,\downarrow}^{t+1}\big(x_{1,e}^t, x_{1,f}^t, x_{0,e\cap f}^t\big) &= (\tilde{A}_1^d)_{e,f} x_{1,f}^t \\
M_{1,\mathcal{C}}^{t+1}\big(x_{1,e}^t, x_{2,u}^t\big) &= (\Delta_{1,2})_{e,u} x_{2,u}^t \\
M_{1,\mathcal{F}}^{t+1}\big(x_{1,e}^t, x_{0,v}^t\big) &= (\Delta_{1,0})_{e,v} x_{0,v}^t \\
U_1^{t+1}\Big(x_{1,e}^t, \{m_i^t(e)\}_{i=\mathcal{F},\mathcal{C},\downarrow,\uparrow}\Big) &= \sigma\Big({W_{1,1}^t}^T\big((\Delta_{1,1})_{e,e} x_{1,e}^t + m_\uparrow(e)^{t+1} + m_\downarrow(e)^{t+1}\big) \\
&\qquad + {W_{1,2}^t}^T m_{\mathcal{C}}(e)^{t+1} + {W_{1,0}^t}^T m_{\mathcal{F}}(e)^{t+1}\Big),
\end{aligned}$$

and AGG $= \sum$. □

E EQUIVARIANCE AND INVARIANCE

One would expect MPSNs to be aware of the two symmetries of a simplicial complex: relabeling of the simplicies in the complex and, optionally, changes in the orientation of the complex if orientations are taken into account. We address these two below.

Let $\mathcal{K}$ be simplicial p-complex with boundary matrices B_i with $i \in \{0, \ldots, p\}$ and corresponding simplicial attributes X_i. Let $P_i \in \mathbb{R}^{S_i \times S_i}$ be some corresponding permutation matrices for the simplicies of dimension i. Let $P_i X_i$ and $P_{i-1} B_i P_i^T$ be the permuted feature matrices and boundary operators, respectively. Additionally, Let $\mathcal{P}$ be an operator acting on simplicial complexes that produces a new complex $\mathcal{P}\mathcal{K}$ permuted as above.

Definition 29 (Permutation equivariance and invariance)**.** *We say that a function f is (simplex) permutation equivariant if $f(\mathcal{P}\mathcal{K}) = \mathcal{P} f(\mathcal{K})$ for any $\mathcal{P}$ (i.e. for any set of permutation operators $\{P_i\}$). Similarly, we say that a function f is (simplex) permutation invariant if $f(\mathcal{P}\mathcal{K}) = f(\mathcal{K})$ for any $\mathcal{P}$.*

Remark 30. *Equivariant functions f must map the simplicial complex to another complex with the same structure, but with possibly different features, similarly to an MPSN layer. In contrast, an invariant function f is not restricted in its choice of the co-domain, similarly to an MPSN network ending with a readout.*

Theorem 31. *An MPSN layer is simplex permutation equivariant and an MPSN network with a final readout layer is simplex permutation invariant.*

Proof. It is sufficient to prove that an MPSN layer is permutation equivariant. Since the final readout layer is permutation invariant by definition, the invariance of the whole model follows directly. In the proof, we abuse the notation slightly and use $P_i(a)$ to denote the corresponding permutation function of P_i acting on indices.

We focus on a single simplex v of an arbitrary dimension n and the corresponding $w = P_n(v)$. Let h_v^{t+1} be the output feature of simplex v for an MPSN layer taking $\mathcal{K}$ as input and $\bar{h}_w^{t+1}$ the output features of simplex w for the same MPSN layer taking $\mathcal{PK}$ as input. We will now show they are equal by showing that the multi-set of features being passed to the message, aggregate and update functions are the same for the two simplicies.

The faces of the n-simplicies in $\mathcal{K}$ are given by the non-zero elements of B_n. Similarly, the lower adjacencies in $\mathcal{PK}$ are given by the non-zero elements of $P_{n-1}B_nP_n^T$ (i.e. the matrix where the rows and columns are permuted according to P_n). Therefore, we obtain

$$(B_n)_{a,b} = (P_{n-1}B_nP_n^T)_{P_{n-1}(a),P_n(b)},$$

In particular, this holds for $b = v, P_n(b) = w$. Because the feature matrices for the $(n-1)$-simplices in $\mathcal{PK}$ are also permuted with $P_{n-1}X_{n-1}$, v and w receive the same message from their faces. The proof follows similarly for coface adjacencies.

The lower adjacenices of the n-simplicies in $\mathcal{K}$ are given by the non-zero entries of $B_n^TB_n$. Similarly, the lower adjacencies in $\mathcal{PK}$ are given by the non-zero elements of

$$(P_{n-1}B_nP_n^T)^T(P_{n-1}B_nP_n^T) = P_nB_n^TP_{n-1}^TP_{n-1}B_nP_n^T = P_nB_n^TB_nP_n^T.$$

That is, the same adjacencies as in $\mathcal{K}$, but with the rows and columns accordingly permuted. Therefore,

$$(B_n^TB_n)_{a,b} = (P_nB_n^TB_nP_n^T)_{P_n(a),P_n(b)},$$

which, in particular, holds for $a = v, P_n(a) = w$. Since the feature matrices for the n-simplices in $\mathcal{PK}$ are also accordingly permuted with P_nX_n, v and w receive the same message from the lower adjacenct simplicies. This can be similarly shown for upper adjacencies. □

Another symmetry that we would like to preserve is orientation. For instance, we know that the homology of the complex is invariant to the particular orientation that was chosen. Therefore, for certain applications where orientations are of interest, we would like to design MPSN layers that are orientation equivariant and MPSN networks that are orientation invariant. We first define these notions. Let $T_i = \mathrm{diag}(t_{i,1}, \ldots, t_{i,S_i})$ a set of diagonal matrices where $t_{i,k} = \pm 1$ for any k and $i \in \{0, \ldots, p\}$. Additionally, we impose the constraint $T_0 = I$, since vertices have no orientation. Changing the orientation of the complex amounts to obtaining a new set of features T_iX_i and boundary matrices $T_{i-1}B_iT_i$. We denote the corresponding change of orientation for the whole complex by $\mathcal{TK}$ as above.

Definition 32 (Orientation equivariance and invariance)**.** *We say that a function f is orientation equivariant if $f(\mathcal{TK}) = \mathcal{T}f(\mathcal{K})$ for any $\mathcal{T}$ (i.e. for any set of orientation operators $\{T_i\}$). Similarly, we say that a function f is orientation invariant if $f(\mathcal{TK}) = f(\mathcal{K})$ for any $\mathcal{T}$.*

Remark 33. *Orientation invariance can be trivially achieved by considering the absolute value of the features and by treating the complex as an unoriented one.*

However, it is (in general) desirable to use equivariance at the intermediate layers and make the network invariant with a final transformation (readout). To study such models requires making further assumptions about the structure of the message, update and aggregate functions. For instance, we can consider the model from (14) used in our linear regions analysis, with a convenient vectorised form

$$X_i^{\text{out}} = \psi\left(B_i^TB_iX_i^{\text{in}}W_1 + X_i^{\text{in}}W_2 + B_{i+1}B_{i+1}^TX_i^{\text{in}}W_3 + B_i^TX_{i-1}^{\text{in}}W_4 + B_{i+1}X_{i+1}^{\text{in}}W_5\right), \quad (23)$$

where we have split the upper and lower adjacencies in two. This corresponds to an MPSN with a message function that multiplies the linearly transformed features of the neighbour by the relative orientation (± 1), sum-based aggregation and an update function that adds the incoming messages to its linearly transformed features and passes the output through ψ.

Proposition 34. *When ψ is an odd activation function, the MPSN layer from Equation (23) is orientation equivariant.*

Proof. We denote with $X_i^{\text{out}} = f_i(B_i, B_{i+1}, X_i^{\text{in}}, X_{i-1}^{\text{in}}, X_{i+1}^{\text{in}})$ the application of one such MPSN layer on i-dimensional chains. For this MPSN layer to be equivariant, we need to show that $f_i(T_{i-1}B_iT_i, T_iB_{i+1}T_{i+1}, T_iX_i^{\text{in}}, T_{i-1}X_{i-1}^{\text{in}}, T_{i+1}X_{i+1}^{\text{in}}) = T_iX_i^{\text{out}}$. Because $T_iT_i = I$ and $T_i^T = T_i$ for all i, we can easily rewrite LHS as

$$\psi\Big(T_i\big(B_i^T B_i X_i^{\text{in}} W_1 + X_i^{\text{in}} W_2 + B_{i+1}B_{i+1}^T X_i^{\text{in}} W_3 + B_i^T X_{i-1}^{\text{in}} W_4 + B_{i+1} X_{i+1}^{\text{in}} W_5\big)\Big). \tag{24}$$

Notice that if ψ and T_i commute, then this becomes $T_iX_i^{\text{out}}$. We remark that they commute when ψ is an odd function and, in particular, when ψ is the identity. □

Remark 35. *A (permutation invariant) aggregation-based readout layer first applying an element-wise even function ψ to the elements of its input multiset is orientation invariant since*

$$AGG(\{\!\{\psi(x_i)\}\!\}) = AGG(\{\!\{\psi(\pm x_i)\}\!\}). \tag{25}$$

F Discussion and Related Work

F.1 Computational Complexity

Message Passing on arbitrary simplicial complexes A d-simplex σ of a p-complex has $d+1$ faces and there are $\binom{d+1}{2}$ upper adjacencies between its faces. Then, a message passing procedure relying on Theorem 1, which considers only these adjacencies, has a computational complexity $\Theta\big(\sum_{d=0}^{p}(d+1)S_d + \binom{d+1}{2}S_d\big) = \Theta\big(\sum_{d=0}^{p}\binom{d+1}{2}S_d\big)$. If we consider p to be a small constant, which is common for many real-world datasets, then the binomial coefficients can be absorbed in the bound, which results in a linear computational complexity in the size of the complex $\Theta(\sum_{d=0}^{p} S_d)$. While including cofaces does not increase this complexity, down adjacencies do. Because any d-complex can be down adjacent to any other d-complex, the worst case complexity is quadratic in the size of the complex $\mathcal{O}(\sum_{d=0}^{p} S_d^2)$.

Message passing on clique complexes When the input is a graph with n nodes, we also need to consider the complexity of the lifting operation. Additionally, we would like to evaluate the complexity of the message passing in terms of certain properties of the graph (e.g. size or maximum degree). First of all, based on Theorem 2, considering triangles is sufficient to get additional expressivity compared to the WL test. In sparse graphs, this can be done efficiently since the best triangle listing algorithms are known to have $\mathcal{O}(m^{3/2})$ complexity (Latapy, 2008), where m is the number of edges.

More generally, finding all the maximal cliques in a graph (Bron & Kerbosch, 1973; Tomita et al., 2006; Cazals & Karande, 2008) has an optimal worst-case complexity of $\mathcal{O}(3^{n/3})$. However, finding all the cliques of a certain maximum dimension in (sparse) real-world graphs can be expected to be significantly faster in practice. In our test, we use a simplex tree data-structure (Boissonnat & Maria, 2014) implemented in the topological data analysis library Gudhi (The GUDHI Project, 2021) to compute the clique complex. Empirically, in large-scale experiments involving graphs with 10^6 nodes, simplex trees are able to generate the clique complex up to a constant desired dimension in a computational cost that is linear in the number of simplicies in the complex (Boissonnat & Maria, 2014). Guarantees about this number can be obtained if the maximum degree of the graph is known. Recently, Chase (2020) has proven the Gan-Loh-Sudakov conjecture (Gan et al., 2015) stating that the maximum number of cliques of size t in a graph of maximum degree δ is $q\binom{\delta+1}{t} + \binom{r}{t}$, where $n = q(\delta+1) + r, 0 \leq r \leq \delta$. Thus, we expect practical performance to be good on real-world graphs, where δ is typically small compared to the size of the graph.

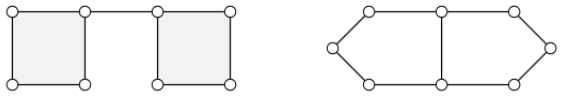

Figure 8: Example of two cubical complexes whose underlying graphs cannot be distinguished by 1-WL, but are not isomorphic as cubical complexes.

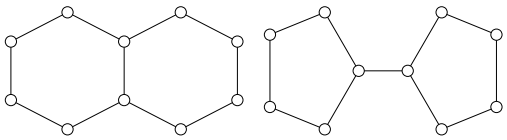

Figure 9: *Decalin* and *Bicyclopentyl*: Non-isomorphic molecular graphs that cannot be distinguished neither by WL nor by SWL, when based on clique complexes (here the nodes represent carbon atoms, and edges are chemical bonds).

F.2 Other Lifting Transformations and Cubical Complexes

We note that other graph lifting transformations could also be used for applying SWL and MPSNs to graph domains. While clique complexes are the commonest such transformation, many others exist (Ferri et al., 2018) and they could be used to emphasise motifs that are relevant for the task (Milo et al., 2002). More broadly, the transformation could target a much wider set of complexes such as cubical complexes (Kaczynski et al., 2004) or cell complexes (Hatcher, 2000), for which a message passing procedure has already been proposed (Hajij et al., 2020). Cell Complexes have a similar hierarchical structure to simplicial complexes, and the approach of Hajij et al. (2020) is very close to ours. At the same time, arbitrary cell complexes are too general for most applications. Here we discuss how our approach could be extended to cubical complexes, a type of cell complex that from a theoretical point of view is similar to simplicial complexes (Kaczynski et al., 2004), and which is used in applications. We plan to extend our approach to cubical complexes in future work.

Cubical complexes are cell complexes consisting of unions of points, edges, squares, cubes, and higher-dimensional hypercubes (Kaczynski et al., 2004). While they are less well-known than simplicial complexes, they can nevertheless be used in applications, and are better suited to the study of certain types of data sets, see, e.g. Wagner et al. (2011). Our approach to message passing can be directly implemented also for cubical complexes. One could similarly define a WL test for cubical complexes, however, the results of Section 2 do not carry over to cubical complexes, since there exist graphs that cannot be embedded into an n-dimensional cube for any n (Garey & Graham, 1975). It is therefore not possible to relate the expressive power for graphs in the WL test to cubical complexes.

In Figure 8 we provide an example of two cubical complexes that are not isomorphic, while their underlying graphs are isomorphic. We note that the clique complexes of the underlying graphs are also isomorphic. The examples in Figures 5 and 8 raise the question of whether we could design tests that are better suited to take into account the topological information encoded in simplicial and cubical complexes, such as homeomorphism tests, which have already been studied, see e.g. Baik & Miller (1990). In particular, such tests should be able to distinguish the graphs in Figure 9, a pair of real world molecular graphs that cannot be distinguished by the SWL test based on clique complexes.

A natural next step in our work is to perform tests on data sets that are more naturally modelled by cubical complexes, such as digital images, for which they can provide computational speed-ups compared to simplicial complexes (Wagner et al., 2011; Kaczynski et al., 2004).

F.3 Provably expressive GNNs

In order to overcome the limited expressive power of standard GNN architectures, several works have proposed variants inspired by the higher-order k-WL procedures (see Appendix). Maron et al. (2019) introduced a model equivalent in power to 3-WL, Morris et al. (2019) proposed k-GNNs, graph neural networks equivalents of set-based k-WL tests. By performing message passing on all possible

k-sets of nodes and across non-local neighborhoods, these models trade locality of computation for expressive power, and thus suffer from high spatial and temporal complexities. Local k-WL variants are introduced and characterized in their expressive power in a follow-up work (Morris et al., 2020). These approaches distinguish local and global neighbors and provably powerful neural counterparts are introduced. Although more efficient than standard higher-order procedures, in contrast to our method, these approaches still account for all possible node k-tuples in a graph, and do not model the relation between signals defined over different dimensions. An alternative approach to improving GNN expressivity has been adopted in Bouritsas et al. (2020), where isomorphism counting of graph substructures is employed as a symmetry breaking mechanism to disambiguate neighbors. Similarly to ours, this approach retains locality of operations; however, message passing is only performed at the node level, and thus does not account for signals defined over higher-dimensional objects.

G Additional Experiments and Details

G.1 Strongly Regular Graphs

In the experiments discussed in (3), we consider two graphs to be isomorphic if the Euclidean distance between their representations is below a fixed threshold ε. We set $\varepsilon = 0.01$ and embed graphs in a 16-dimensional space by running an untrained, 5-layer, MPSN model on the associated clique complexes. In particular, each graph is lifted to a $(k-1)$-dimensional simplicial complex, with k the size of the largest clique in the family it belongs to. Nodes are initialised with a constant, unitary signal, while simplices with the sum of the features of the constituent faces. The employed architecture is parameterised similarly to GIN (Xu et al., 2019b) and hence we refer to it as Simplicial Isomorphism Network (SIN). In particular, in accordance with Theorem 1, messages are only aggregated from faces and upper-adjacent simplices (for which we also include the representations of the shared cofaces). The following message passing operations are employed to compute the $t+1$ intermediate representation for p-simplex v:

$$h_v^{t+1} = \mathrm{MLP}_U^t\Big(\mathrm{MLP}^t\big((1+\epsilon)h_v^t + \sum_{w\in\mathcal{F}(v)} h_w^t\big) \parallel \mathrm{MLP}^t\big((1+\epsilon)h_v^t + \sum_{w\in\mathcal{N}(v)} M_\uparrow^t(h_w^t, h_{v\cup w}^t)\big)\Big)$$
$$M_\uparrow^t(h_w^t, h_{v\cup w}^t) = \mathrm{MLP}_M^t\big(h_w^t \parallel h_{v\cup w}^t\big) \tag{26}$$

where $\parallel$ indicates concatenation, MLP^t is a 2-Layer Perceptron and MLP_U^t, MLP_M^t consist of a dense layer followed by a non-linearity. Parameter ϵ is set to zero and is not optimised. Chain representations are pooled via summation and final complex embeddings are obtained by applying an 2-Layer Perceptron to the sum of chain representations at each dimension. The ELU Clevert et al. (2016) non-linearity is applied throughout the whole architecture. The experiments are performed for 10 different random weight-initialisations; in Figure 3 we report mean failure rate along with standard error (vertical error bars). The datasets can be found at the webpage `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`.

G.2 Trajectory Classification

For the trajectory classification task we generate the simplicial complex using an approach similar to Schaub et al. (2020). We generate 1,000 random points in the unit square, we perform a Delaunay triangulation of these points and then remove the triangles (and points) intersecting with two pre-defined regions of the plane to create the two holes. To generate the trajectories, we first randomly sample a random point from the top-left corner of the complex and an end point from the bottom-right corner of the complex. We then perform a random walk on the edges of the complex. With a probability of 0.9, the neighbour closest to the end-point is chosen, and with a probability 0.1, a random neighbour is chosen. To generate the two classes, we set random points either from the bottom-left corner or the top-right corner as an intermediate checkpoint. We generate 1,000 train trajectories and 200 test trajectories. We train all modes for 50 epochs and report the final test result.

Model-wise, we use an MPSN network with equivariant layers with identity activation function as in Equation (23). The final layer is an orientation-invariant readout layer, which first passes the features through the absolute value function and then performs a sum aggregation. For the MPNN baseline we use again a model like Equation (23), but with no upper adjacencies (no triangle awareness) and

with no awareness of the relative orientations. The MPNN is thus essentially performing message passing in the line graph.

Table 1: Graph classification accuracy on TUDatasets. The first section of the table includes graph kernel methods, while the second includes graph neural networks.

Dataset	Proteins	NCI1	IMDB-B	IMDB-M	RDT-B	RDT-M5K
size	1113	4110	1000	1500	2000	5000
#classes	2	2	2	3	2	5
Avg. #nodes	39.1	29.8	19.8	13.0	429.6	508.5
Avg. #Triangles	27.4	0.05	392.0	305.9	24.8	21.8
Median #Triangles	21.0	0.0	119.5	56.0	11.0	11.0
RWK (Gärtner et al., 2003)	59.6±0.1	>3 days	N/A	N/A	N/A	N/A
GK (k=3) (Shervashidze et al., 2009)	71.4±0.31	62.5±0.3	N/A	N/A	N/A	N/A
PK (Neumann et al., 2016)	73.7±0.7	82.5±0.5	N/A	N/A	N/A	N/A
WL kernel Shervashidze et al. (2011)	75.0±3.1	86.0±1.8	73.8±3.9	50.9±3.8	81.0±3.1	52.5±2.1
DCNN (Atwood & Towsley, 2016)	61.3±1.6	56.6±1.0	49.1±1.4	33.5±1.4	N/A	N/A
DGCNN (Zhang et al., 2018)	75.5±0.9	74.4±0.5	70.0±0.9	47.8±0.9	N/A	N/A
IGN (Maron et al., 2018)	76.6±5.5	74.3±2.7	72.0±5.5	48.7±3.4	N/A	N/A
GIN (Xu et al., 2019b)	76.2±2.8	82.7±1.7	75.1±5.1	52.3±2.8	92.4±2.5	57.5±1.5
PPGNs (Maron et al., 2019)	77.2±4.7	83.2±1.1	73.0±5.8	50.5±3.6	N/A	N/A
Natural GN (de Haan et al., 2020)	71.7±1.0	82.4±1.3	73.5±2.0	51.3±1.5	N/A	N/A
SIN (Ours)	76.4 ± 3.3	82.7 ± 2.1	75.6 ± 3.2	52.4 ± 2.9	92.2 ± 1.0	57.3 ± 1.6

G.3 Real-world Graph Classification

In this section, we study the practical impact of considering higher-order interactions via clique-complexes and report results for a few popular graph classification tasks commonly used for benchmarking GNNs[1]. We follow the same experimental setting and evaluation procedure described in Xu et al. (2019b). Accordingly, we report the best mean validation accuracy computed in a 10-fold cross-validation fashion, as well the related standard deviation. We employ a SIN model which applies the following message passing scheme to compute the $t+1$ intermediate representation for p-simplex v:

$$h_v^{t+1} = \mathrm{MLP}^t_{U,p}\Big(\mathrm{MLP}^t_{\mathcal{F},p}\big((1+\epsilon_{\mathcal{F}})h_v^t + \sum_{w\in\mathcal{F}(v)} h_w^t\big) \parallel \mathrm{MLP}^t_{\uparrow,p}\big((1+\epsilon_{\uparrow})h_v^t + \sum_{w\in\mathcal{N}(v)} h_w^t\big)\Big) \quad (27)$$

where $\parallel$ indicates concatenation, $\mathrm{MLP}^t_{\mathcal{F},p}$ and $\mathrm{MLP}^t_{\uparrow,p}$ are 2-Layers Perceptrons endowed with Batch Normalization (Ioffe & Szegedy, 2015) (BN) and ReLU activations, $\mathrm{MLP}^t_{U,p}$ is a dense layer followed by the application of BN and ReLU. The only exception is represented by Reddit datasets, on which BN was observed to cause instabilities in the training process and was not applied. Parameters $\epsilon_{\mathcal{F}}$ and $\epsilon_{\uparrow}$ are set to zero and are not optimised. As it is possible to notice in Equation (27), upper message $m_{\uparrow,p}(v)$ is computed as $m_{\uparrow,p}(v) = \sum_{w\in\mathcal{N}(v)} h_w^t$, thus explicitly disregarding the representation of shared cofaces $h_{v\cup w}^t$: this choice showed to yield better performance on these benchmarks. The overall architecture closely resembles the one adopted in Xu et al. (2019b): 4 message passing layers and a Jumping Knowledge readout scheme (Xu et al., 2018), which computes final p-chain embeddings by applying a non-linear dense layer to the concatenation of the representations obtained at each message passing iteration. Final complex representations are obtained by summing the chain embeddings read-out at dimensions 0 ('nodes') and 2 ('triangles'). One last dense layer is applied to output class predictions.

The performance of SIN are reported in Table 1, along with those of graph kernel methods and other GNNs. SIN generally performs on-par with state-of-the-art approaches. We observe that our model has the highest mean performance on the IMDB datasets, which have the largest mean and median number of triangles. At the same time, on datasets like NCI1, where the number of higher-order structures is close to zero, the model shows the same mean accuracy as GIN.

[1] https://chrsmrrs.github.io/datasets/