WL meet VC

Christopher Morris[*] RWTH Aachen University
Floris Geerts[*] University of Antwerp
Jan Tönshof, Martin Grohe RWTH Aachen University

Abstract

Recently, many works investigated the expressive power of graph neural networks (GNNs) by linking it to the 1-dimensional Weisfeiler–Leman algorithm (1-WL). Here, the 1-WL is a well-studied heuristic for the graph isomorphism problem, which iteratively colors or partitions a graph's vertex set. While this connection has led to significant advances in understanding and enhancing GNNs' expressive power, it does not provide insights into their generalization performance, i.e., their ability to make meaningful predictions beyond the training set. In this paper, we study GNNs' generalization ability through the lens of Vapnik–Chervonenkis (VC) dimension theory in two settings, focusing on graph-level predictions. First, when no upper bound on the graphs' order is known, we show that the bitlength of GNNs' weights tightly bounds their VC dimension. Further, we derive an upper bound for GNNs' VC dimension using the number of colors produced by the 1-WL. Secondly, when an upper bound on the graphs' order is known, we show a tight connection between the number of graphs distinguishable by the 1-WL and GNNs' VC dimension.

1 Introduction

There are numerous approaches for machine learning for graph-structured, most notably those based on *graph kernels* [1, 2] or *graph neural networks* (GNNs) [3–5]. Here, graph kernels [6] based on the 1-*dimensional Weisfeiler–Leman algorithm* (1-WL) [7], a well-studied heuristic for the graph isomorphism problem, and corresponding GNNs [8, 9], have recently advanced the state-of-the-art in supervised vertex- and graph-level learning [5]. Further, based on the k-*dimensional Weisfeiler–Leman algorithm* (k-WL), 1-WL's more powerful generalization, several works generalized GNNs to *higher-order GNNs* (k-GNNs), resulting in provably more expressive architectures, e.g., Geerts and Reutter [10], Maron et al. [11], Morris et al. [12]. While devising provably expressive GNN-like architectures is a meaningful endeavor, it only partially addresses the challenges of machine learning with graphs. That is, expressiveness results reveal little about an architecture's ability to generalize to graphs outside the training set. Surprisingly, only a few notable contributions study GNNs' generalization behaviors, e.g., Garg et al. [13], Kriege et al. [14], Liao et al. [15], Maskey et al. [16] and Scarselli et al. [17]. However, these approaches express GNN's generalization ability using only classical graph parameters, e.g., maximum degree, number of vertices, or edges, which cannot fully capture the complex structure of real-world graphs. Further, most approaches study generalization in the *non-uniform regime*, i.e., assuming that the GNNs operate on graphs of a pre-specified order. Further, they only investigate the case $k = 1$, i.e., standard GNNs, ignoring more expressive generalizations.

Present work. This paper investigates the influence of graph structure and the parameters' encoding lengths on GNNs' generalization by tightly connecting 1-WL's expressivity and GNNs' Vapnik–Chervonenkis (VC) dimension. Specifically, our contributions are:

1. In the non-uniform regime, we prove *tight* bounds on GNNs' VC dimension. We show that GNNs' VC dimension depends tightly on the number of equivalence classes computed by the 1-WL over a set of graphs; see Propositions 2.1 and 2.2. Moreover, our results easily extend to the k-WL and many recent expressive GNN extensions.
2. In the uniform regime, i.e., when graphs can have arbitrary order, we show that GNNs' VC dimension is *lower* and *upper bounded* by the largest bitlength of its weights; see Proposition 2.5.

[*]Equal contribution.

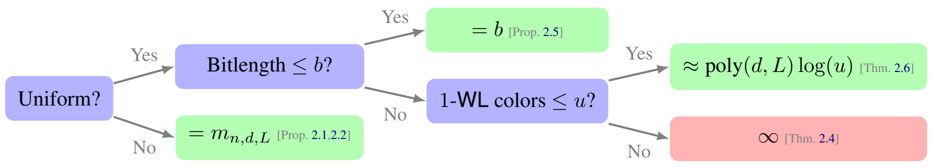


Figure 1: Overview of our results for bounded-width GNNs. Green and red boxes denote VC dimension bounds. Here, $m_{n,d,L}$ denotes the number of graphs of order at most n with boolean d-dimensional features distinguishable by 1-WL after L iterations.

3. In both the uniform and non-uniform regimes, GNNs' VC dimension depends *logarithmically on the number of colors* computed by the 1-WL and polynomially on the number of parameters; see Theorem 2.6.

Main insight. Overall, our results provide new insights into GNNs' generalization behavior and how graph structure and parameters influence it. Specifically, our results imply that a complex graph structure, captured by 1-WL, results in worse generalization performance. The same holds for increasing the encoding length of the GNN's parameters. *Importantly, our theory provides the first link between expressivity results and generalization ability.* Moreover, our results establish the *first* lower bounds for GNNs' VC dimension. See Figure 1 for a high-level overview of our results. See Appendix A for a discussion on related work.

1.1 Background

Let $\mathbb{N} := \{1, 2, 3, \dots\}$. For $n \geq 1$, let $[n] := \{1, \dots, n\} \subset \mathbb{N}$. We use $\{\!\!\{\dots\}\!\!\}$ to denote multisets, i.e., the generalization of sets allowing for multiple instances for each of its elements. Throughout the paper, we use standard notations, e.g., we denote the *neighborhood* of a vertex v by $N(v)$ and $\ell(v)$ denotes its discrete vertex label, and so on; see Appendix B for details.

The Weisfeiler–Leman algorithm. We here describe the 1-WL and refer to Appendix C for the k-WL. The 1-WL or color refinement is a well-studied heuristic for the graph isomorphism problem, originally proposed by Weisfeiler and Leman [7]. Formally, let $G = (V(G), E(G), \ell)$ be a labeled graph. In each iteration, $t > 0$, the 1-WL computes a vertex coloring $C^1_t \colon V(G) \to \mathbb{N}$, depending on the coloring of the neighbors. That is, in iteration $t > 0$, we set

$$C^1_t(v) := \mathsf{RELABEL}\Big(\big(C^1_{t-1}(v), \{\!\!\{C^1_{t-1}(u) \mid u \in N(v)\}\!\!\}\big)\Big),$$

for all vertices v in $V(G)$, where $\mathsf{RELABEL}$ injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C^1_0 := \ell$. To test if two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of vertices colored c in $\mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. Moreover, if the number of colors between two iterations, t and $(t + 1)$, does not change, i.e., the cardinalities of the images of C^1_t and C^1_{i+t} are equal, or, equivalently,

$$C^1_t(v) = C^1_t(w) \iff C^1_{t+1}(v) = C^1_{t+1}(w),$$

for all vertices v and w in $V(G)$, then the algorithm terminates. For such t, we define the *stable coloring* $C^1_\infty(v) = C^1_t(v)$, for v in $V(G)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [18]. We define the *color complexity* of a graph G as the number of colors computed by the 1-WL after $|V(G)|$ iterations on G. See Appendix C for details on the k-WL, 1-WL's more expressive generalization.

Graph neural networks. Formally, let $G = (V(G), E(G), \ell)$ be a labeled graph with initial vertex features $\mathbf{h}^{(0)}_v$ in $\mathbb{R}^d$ that are *consistent* with ℓ. That is, each vertex v is annotated with a feature $\mathbf{h}^{(0)}_v$ in $\mathbb{R}^d$ such that $\mathbf{h}^{(0)}_v = \mathbf{h}^{(0)}_u$ if and only $\ell(v) = \ell(u)$, e.g., a one-hot encoding of the labels $\ell(u)$ and $\ell(v)$. Following, Gilmer et al. [4] and Scarselli et al. [19], in each layer, $t > 0$, we compute vertex features

$$\mathbf{h}^{(t)}_v := \mathsf{UPD}^{(t)}\Big(\mathbf{h}^{(t-1)}_v, \mathsf{AGG}^{(t)}\big(\{\!\!\{\mathbf{h}^{(t-1)}_u \mid u \in N(v)\}\!\!\}\big)\Big) \tag{1}$$

in $\mathbb{R}^d$, where $\mathsf{UPD}^{(t)}$ and $\mathsf{AGG}^{(t)}$ may be differentiable parameterized functions, e.g., neural networks. In the case of graph-level tasks, e.g., graph classification, one additionally uses

$$\mathbf{h}_G \coloneqq \mathsf{READOUT}\big(\{\!\!\{\mathbf{h}_v^{(L)} \mid v \in V(G)\}\!\!\}\big) \in \mathbb{R}, \tag{2}$$

to compute a single vectorial representation based on learned vertex features after iteration L.[2] Again, $\mathsf{READOUT}$ may be a differentiable parameterized function. See Appendix D for a definition of (higher-order) k-GNNs.

Notation. In the subsequent sections, we use the following notation. We denote the class of all (labeled) graphs by $\mathcal{G}$, the class of all graphs with d-dimensional, real-valued vertex features by $\mathcal{G}_d$, the class of all graphs with d-dimensional boolean vertex features by $\mathcal{G}_d^{\mathbb{B}}$, the class of all graphs with an order of at most n and d-dimensional vertex features by $\mathcal{G}_{d,n}$, and the class of all graphs with d-dimensional vertex features and of color complexity at most u by $\mathcal{G}_{d,\leq u}$. Further, we consider the following classes of GNNs. We denote the class of all GNNs consisting of L layers with $(L+1)$th layer readout layer by $\mathsf{GNN}(L)$, the subset of $\mathsf{GNN}(L)$ but whose aggregate, update and readout functions have a width at most d by $\mathsf{GNN}(d, L)$, and the subset of $\mathsf{GNN}(L)$ but whose aggregation function is a summation and update and readout functions are single layer perceptrons of width at most d by $\mathsf{GNN}_{\mathsf{slp}}(d, L)$. More generally, we consider the class $\mathsf{GNN}_{\mathsf{mlp}}(d, L)$ of GNNs using summation for aggregation and such that update and readout functions are *multilayer perceptrons* (MLPs), all of width of at most d. We refer to elements in $\mathsf{GNN}_{\mathsf{mlp}}(d, L)$ as *simple GNNs*. See Appendix F for details. We stress that simple GNNs are already expressive enough to be equivalent to the 1-WL in distinguishing non-isomorphic graphs.

VC dimension of GNNs. For a class $\mathcal{C}$ of GNNs and class $\mathcal{X}$ of attributed graphs, see Appendix B, $\mathsf{VC\text{-}dim}_{\mathcal{X}}(\mathcal{C})$ is the maximal number m of graphs $\mathbf{G}_1, \dots, \mathbf{G}_m$ in $\mathcal{X}$ that can be shattered by $\mathcal{C}$. Here, $\mathbf{G}_1, \dots, \mathbf{G}_m$ are *shattered* if for any $\boldsymbol{\tau}$ in $\{0, 1\}^m$ there exists a GNN gnn in $\mathcal{C}$ such that for all i in $[m]$:

$$\mathsf{gnn}(\mathbf{G}_i) = \begin{cases} \geq 2/3 & \text{if } \tau_i = 1, \text{ and} \\ \leq 1/3 & \text{if } \tau_i = 0. \end{cases} \tag{3}$$

The above definition can straightforwardly be generalized to k-GNNs. Bounding the VC dimension directly implies an upper bound on the generalization error; see Appendix E and [20, 21] for details.

Bitlength of GNNs. Below we study the dependence of GNNs' VC dimension on the *bitlength* of its weights. Assume an L-layered GNN with a set of parameters Θ, then the GNN's bitlength is the maximum number of bits needed to encode each weight in Θ and the parameters specifying the activation functions. We define the bitlength of a class of GNNs as the maximum bitlength across all GNNs in the class.

2 WL meet VC 🤝

We first consider the non-uniform regime, i.e., we assume an upper bound on the graphs' order. Given the connection between GNNs and the 1-WL [8, 22], GNNs' ability to shatter a set of graphs can easily be related to distinguishability by the 1-WL. For example, let us first consider the VC dimension of GNNs on the class $\mathcal{G}_{d,n}^{\mathbb{B}}$ consisting of graphs of an order of at most n with d-dimensional boolean features. Further, let $m_{n,d,L}$ be the maximal number of graphs in $\mathcal{G}_{d,n}^{\mathbb{B}}$ distinguishable by 1-WL after L iterations. The following result shows that $m_{n,d,L}$ is also the maximal number of graphs in $\mathcal{G}_{d,n}^{\mathbb{B}}$ that L-layer GNNs can shatter.

Proposition 2.1. *For all n, d and L, it holds* $\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,n}^{\mathbb{B}}}\big(\mathsf{GNN}(L)\big) \leq m_{n,d,L}$.

This upper bound holds regardless of the choice of aggregation, update, and readout functions used in the GNNs. We next show a matching lower bound for the VC dimension of GNNs on graphs in $\mathcal{G}_{d,n}^{\mathbb{B}}$. In fact, the lower bound already holds for simple GNNs of width $\mathcal{O}(nm_{n,d,L})$.

Proposition 2.2. *For all n, d, and L, all $m_{n,d,L}$ 1-WL-distinguishable graphs of order at most n with d-dimensional boolean features can be shattered by sufficiently wide L-layer GNNs. Hence,* $\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,n}^{\mathbb{B}}}\big(\mathsf{GNN}(L)\big) = m_{n,d,L}$.

We note that the above two results can be straightforwardly generalized to k-GNNs, i.e., their VC dimension is tightly connected to the expressive power of the k-WL, and other recent extensions of GNNs; see Appendix D.1.

[2]For simplicity, we assume GNNs to return scalars on graphs. This makes the definition of VC dimension more concise.

We now consider the uniform regime, i.e., we assume no upper bound on the graphs' order. Since the number $m_{n,d,L}$ of 1-WL distinguishable graphs increases for growing n, Proposition 2.2 implies that the VC dimension of L-layered GNNs on the class $\mathcal{G}_d^{\mathbb{B}}$ of all graphs with d-dimensional boolean features but of arbitrary order is unbounded.

Corollary 2.3. *For all d and $L \geq 1$, it holds that* $\mathsf{VC\text{-}dim}_{\mathcal{G}_d^{\mathbb{B}}}(\mathsf{GNN}(L)) = \infty$.

The proof of this result requires update and readout functions in GNNs of *unbounded width*. Using a different "bit extraction" proof technique, we can strengthen the previous result such that *fixed-width* GNNs can be considered.

Theorem 2.4. *For all d, L at least two, it holds that* $\mathsf{VC\text{-}dim}_{\mathcal{G}_d^{\mathbb{B}}}(\mathsf{GNN}(d, L)) = \infty$.

Again, this result holds even for the class of simple GNNs. The theorem relies on the following result, which is of independent interest.

Proposition 2.5. *There exists a family $\mathcal{F}_b$ of simple 2-layer GNNs of width two and bitlength $\mathcal{O}(b)$ using piece-wise linear activation functions such that its VC dimension is* exactly b.

We now fix the width of the GNNs and consider $\mathcal{G}_{d,\leq u}$. Note that $\mathcal{G}_{d,\leq u}$ may contain graphs of arbitrary order. For example, all regular graphs (of the same degree) belong to $\mathcal{G}_{d,\leq 1}$. We also remark that there is no upper bound on the number of 1-WL distinguishable graphs for $\mathcal{G}_{d,\leq u}$, because we only bound the number of colors appearing in a single graph and not the number of colors appearing in all graphs of the class. As such, the bounds obtained earlier do not apply. Finally, in the bound below, input graphs can have real features.

Theorem 2.6. *Assume d and L in $\mathbb{N}$, and GNNs in $\mathsf{GNN}_{\mathsf{slp}}(d, L)$ using piece-wise polynomial activation functions with $p > 0$ pieces and degree $\delta \geq 0$. Let $P = d(2dL + L + 1) + 1$ be the number of parameters in the GNNs. For all u in $\mathbb{N}$,*

$$\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,\leq u}}(\mathsf{GNN}_{\mathsf{slp}}(d, L)) \leq \begin{cases} \mathcal{O}(P \log(puP)) & \textit{if } \delta = 0, \\ \mathcal{O}(LP \log(puP)) & \textit{if } \delta = 1, \\ \mathcal{O}(LP \log(puP) + L^2 P \log(\delta)) & \textit{if } \delta > 1. \end{cases}$$

We note that the above result can be straightforwardly generalized to k-GNNs. These upper bounds, concerning the dependency on u, cannot be improved by more than a constant factor. Also, note that any graph of order at most n has at most n 1-WL colors, and hence $\mathcal{G}_{d,n} \subseteq \mathcal{G}_{d,\leq n}$. The above bound thus complements the upper bound on $\mathcal{G}_{d,n}^{\mathbb{B}}$ given earlier, but now for fixed-width GNNs.

Implications, limitations, and future work. Our results include the first lower bounds of GNNs' VC dimension and the first inherent connection between GNNs' VC dimension and the expressive power of the 1-WL. We show that the VC dimension bounds for GNNs can be tightened when considering graphs of low color complexity. This connection to the number of colors indicates that graphs with complex structures, captured by the 1-WL, may have a higher VC dimension. Moreover, if the 1-WL can distinguish a large set of graphs of a given dataset, our results imply that a sufficiently expressive GNN will require a large set of training samples to generalize well. Therefore, we can use the 1-WL to assess GNNs' generalization ability on a given dataset quickly. Although the experimental results in Appendix H suggest that our VC dimension bounds hold in practice to some extent, it is well known that they do not explain the generalization behavior of deep neural networks in the over-parameterized regime Bartlett et al. [23], trained with variants of stochastic gradient descent. Therefore, it is a future challenge to understand how graph structure influences the generalization properties of over-parameterized GNNs, trained with variants of stochastic gradient descent.

3 Conclusion

We investigated GNNs' generalization capabilities through the lens of VC dimension theory in different settings. Specifically, when not assuming a bound on the graphs' order, we showed that the VC dimension tightly depends on the bitlength of the GNNs' weights. We further showed that the number of colors computed by the 1-WL, besides the number of parameters and layers, influences the VC dimension. When a bound on the graphs' order is known, we upper and lower bounded GNNs' VC dimension via the maximal number of graphs distinguishable by the 1-WL. *Thus, our theory provides the first link between expressivity results and generalization.* Further, our theory also applies to a large set of recently proposed GNN enhancements.

Acknowledgements

Christopher Morris is partially funded by a DFG Emmy Noether grant (468502433) and RWTH Junior Principal Investigator Fellowship under Germany's Excellence Strategy. Martin Grohe is partially funded by the European Union (ERC, SymSim, 101054974). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Karsten M. Borgwardt, M. Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *Foundations and Trends in Machine Learning*, 13(5–6), 2020. 1, 12

[2] N. M. Kriege, F. D. Johansson, and C. Morris. A survey on graph kernels. *Applied Network Science*, 5(1):6, 2020. 1, 12

[3] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *ArXiv preprint*, 2020. 1

[4] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017. 2, 12

[5] C. Morris, Y. L., H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt. Weisfeiler and Leman go machine learning: The story so far. *ArXiv preprint*, 2021. 1, 12, 13, 14

[6] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, pages 2539–2561, 2011. 1, 24

[7] B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968. English translation by G. Ryabov is available at `https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf`. 1, 2, 12

[8] 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 Conference on Artificial Intelligence*, pages 4602–4609, 2019. 1, 3, 12, 14, 16, 17

[9] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 1, 12, 16

[10] Floris Geerts and Juan L. Reutter. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022. 1, 12

[11] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pages 2153–2164, 2019. 1, 12

[12] Christopher Morris, Gaurav Rattan, Sandra Kiefer, and Siamak Ravanbakhsh. SpeqNets: Sparsity-aware permutation-equivariant graph networks. In *International Conference on Machine Learning*, pages 16017–16042, 2022. 1, 12

[13] Vikas K. Garg, Stefanie Jegelka, and Tommi S. Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430, 2020. 1, 12

[14] Nils M. Kriege, Christopher Morris, Anja Rey, and Christian Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *International Joint Conference on Artificial Intelligence*, pages 2348–2354, 2018. 1, 12

[15] Renjie Liao, Raquel Urtasun, and Richard S. Zemel. A PAC-Bayesian approach to generalization bounds for graph neural networks. In *International Conference on Learning Representations*, 2021. 1, 12

[16] S. Maskey, Y. Lee, R. Levie, and G. Kutyniok. Generalization analysis of message passing neural networks on large random graphs. In *Advances in Neural Information Processing Systems*, 2022. 1, 12

[17] F. Scarselli, A. C. Tsoi, and M. Hagenbuchner. The Vapnik-Chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, pages 248–259, 2018. 1, 12

[18] M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017. 2, 12

[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 2, 12

[20] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995. 3, 12

[21] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018. 3, 12, 15, 20

[22] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5449–5458, 2018. 3

[23] Peter L. Bartlett, Philip M. Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *ArXiv preprint*, 2019. 4

[24] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015. 12

[25] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. 12

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 12

[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and deep locally connected networks on graphs. In *International Conference on Learning Representation*, 2014. 12

[28] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3837–3845, 2016. 12

[29] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2019. 12

[30] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 12

[31] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1): 97–109, 2019. 12

[32] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5425–5434, 2017. 12

[33] I. I. Baskin, V. A. Palyulin, and N. S. Zefirov. A neural device for searching direct correlations between structures and properties of chemical compounds. *Journal of Chemical Information and Computer Sciences*, 37(4):715–721, 1997. 12

[34] Christoph Goller and Andreas Küchler. Learning task-dependent distributed representations by backpropagation through structure. In *International Conference on Neural Networks*, pages 347–352, 1996. 12

[35] D. B. Kireev. Chemnet: A novel neural network based method for graph/property mapping. *Journal of Chemical Information and Computer Sciences*, 35(2):175–180, 1995. 12

[36] C. Merkwirth and T. Lengauer. Automatic generation of complementary descriptors with molecular graph networks. *Journal of Chemical Information and Modeling*, 45(5):1159–1168, 2005. 12

[37] A. Micheli and A. S. Sestito. A new neural network model for contextual processing of graphs. In *Italian Workshop on Neural Nets Neural Nets and International Workshop on Natural and Artificial Immune Systems*, pages 10–17, 2005. 12

[38] A. Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009. 12

[39] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–35, 1997. 12

[40] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020. 12

[41] Floris Geerts, Filip Mazowiecki, and Guillermo A. Pérez. Let's agree to degree: Comparing graph convolutional networks in the message-passing framework. In *International Conference on Machine Learning*, pages 3640–3649, 2021. 12

[42] W. Azizian and M. Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2021. 12

[43] Floris Geerts. The expressive power of kth-order invariant graph networks. *ArXiv preprint*, 2020. 12

[44] C. Morris, G. Rattan, and P. Mutzel. Weisfeiler and Leman go sparse: Towards higher-order graph embeddings. In *Advances in Neural Information Processing Systems*, 2020. 12, 14, 17, 21

[45] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, pages 15868–15876, 2019. 12

[46] T. Maehara and H. NT. A simple proof of the universality of invariant/equivariant graph neural networks. *ArXiv preprint*, 2019. 12

[47] A. Aamand, J. Y. Chen, P. Indyk, S. Narayanan, R. Rubinfeld, N. Schiefer, S. Silwal, and T. Wagner. Exponentially improving the complexity of simulating the Weisfeiler-Lehman test with graph neural networks. *ArXiv preprint*, 2022. 12

[48] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pages 4663–4673, 2019. 12

[49] Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *Advances in Neural Information Processing Systems*, 2020. 12

[50] R. Abboud, İ. İ. Ceylan, M. Grohe, and T. Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *Joint Conference on Artificial Intelligence*, pages 2112–2118, 2021. 12

[51] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. In *International Joint Conference on Artificial Intelligence*, pages 2126–2132, 2020.

[52] R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks. In *SIAM International Conference on Data Mining*, pages 333–341, 2021. 12

[53] Omri Puny, Derek Lim, Bobak Toussi Kiani, Haggai Maron, and Yaron Lipman. Equivariant polynomials for graph neural networks. *ArXiv preprint*, 2023. 12

[54] Pablo Barceló, Floris Geerts, Juan L. Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In *Advances in Neural Information Processing Systems*, pages 25280–25293, 2021. 12

[55] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *ArXiv preprint*, 2020. 14

[56] Hoang Nguyen and Takanori Maehara. Graph homomorphism convolution. In *International Conference on Machine Learning*, pages 7306–7316, 2020. 12

[57] Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, pages 599–608, 2021. 12

[58] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F. Montúfar, Pietro Lió, and Michael M. Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pages 1026–1037, 2021. 12

[59] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael M. Bronstein. Weisfeiler and Lehman go cellular: CW networks. In *Advances in Neural Information Processing Systems*, pages 2625–2640, 2021. 12

[60] Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten M. Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations*, 2022. 12

[61] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph learning with 1D convolutions on random walks. *ArXiv preprint*, 2021. 12

[62] Karolis Martinkus, Pál András Papp, Benedikt Schesch, and Roger Wattenhofer. Agent-based graph neural networks. *ArXiv preprint*, 2022. 12

[63] Rajat Talak, Siyi Hu, Lisa Peng, and Luca Carlone. Neural trees for learning on graphs. *ArXiv preprint*, 2021. 12

[64] Pablo Barceló, Mikhail Galkin, Christopher Morris, and Miguel A. Romero Orth. Weisfeiler and leman go relational. *ArXiv preprint*, 2022. 12

[65] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems*, 2020. 12

[66] Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In *International Conference on Machine Learning*, pages 748–758, 2021. 12

[67] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022. 12, 14

[68] L. Cotta, C. Morris, and B. Ribeiro. Reconstruction for powerful graph representations. In *Advances in Neural Information Processing Systems*, pages 1713–1726, 2021.

[69] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang. How powerful are k-hop message passing graph neural networks. In *Advances in Neural Information Processing Systems*, 2022.

[70] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and extending subgraph GNNs by rethinking their symmetries. *CoRR*, 2022.

[71] Y. Huang, X. Peng, J. Ma, and M. Zhang. Boosting the cycle counting power of graph neural networks with I^2-GNNs. *ArXiv preprint*, 2022.

[72] P. A. Papp, L. Faber K. Martinkus, and R. Wattenhofer. DropGNN: Random dropouts increase the expressiveness of graph neural networks. In *Advances in Neural Information Processing Systems*, 2021.

[73] Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. In *International Conference on Machine Learning*, pages 17323–17345, 2022.

[74] C. Qian, G. Rattan, F. Geerts, C. Morris, and M. Niepert. Ordered subgraph aggregation networks. In *Advances in Neural Information Processing Systems*, 2022. 14

[75] E. H. Thiede, W. Zhou, and R. Kondor. Autobahn: Automorphism-based graph neural nets. In *Advances in Neural Information Processing Systems*, pages 29922–29934, 2021.

[76] A. Wijesinghe and Q. Wang. A new perspective on "how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2022.

[77] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *AAAI Conference on Artificial Intelligence*, pages 10737–10745, 2021.

[78] M. Zhang and P. Li. Nested graph neural networks. In *Advances in Neural Information Processing Systems*, pages 15734–15747, 2021.

[79] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022.

[80] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. *CoRR*, abs/2302.07090, 2023. 12

[81] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of gnns via graph biconnectivity. *ArXiv preprint*, 2023. 12

[82] Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *ArXiv preprint*, 2022. 12

[83] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek. Attending to graph transformers. *CoRR*, abs/2302.04181, 2023. 12

[84] Martin Grohe. The descriptive complexity of graph neural networks. *ArXiv preprint*, 2023. 12

[85] Eran Rosenbluth, Jan Tönshoff, and Martin Grohe. Some might say all you need is sum. *ArXiv preprint*, 2023. 12

[86] Marek Karpinski and Angus Macintyre. Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, 54(1):169–176, 1997. 12

[87] Haotian Ju, Dongyue Li, Aneesh Sharma, and Hongyang R. Zhang. Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion. *ArXiv preprint*, 2023. 12

[88] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *International Conference on Knowledge Discovery & Data Mining*, pages 1539–1548, 2019. 12

[89] Pascal Mattia Esser, Leena C. Vankadara, and Debarghya Ghoshdastidar. Learning theory can (sometimes) explain generalisation in graph neural networks. In *Advances in Neural Information Processing Systems*, pages 27043–27056, 2021. 12

[90] Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its applications. In *Annual Conference on Learning Theory*, pages 157–171, 2007. 12

[91] Ilya O. Tolstikhin and David Lopez-Paz. Minimax lower bounds for realizable transductive classification. *ArXiv preprint*, 2016. 12

[92] O. Goldreich. Introduction to testing graph properties. In *Property Testing*. Springer, 2010. 12

[93] Gilad Yehudai, Ethan Fetaya, Eli A. Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986, 2021. 12

[94] B. Weisfeiler. *On Construction and Identification of Graphs*. Springer, 1976. 12

[95] László Babai. Lectures on graph isomorphism. University of Toronto, Department of Computer Science. Mimeographed lecture notes, October 1979, 1979. 12

[96] N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988*, pages 59–81, 1990. 12

[97] A. Atserias and E. N. Maneva. Sherali-adams relaxations and indistinguishability in counting logics. *SIAM Journal on Computing*, 42(1):112–137, 2013. 12

[98] M. Grohe and M. Otto. Pebble games and linear equations. *Journal of Symbolic Logic*, 80(3): 797–844, 2015.

[99] P. N. Malkin. Sherali–Adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, pages 73–97, 2014. 12

[100] H. Dell, M. Grohe, and G. Rattan. Lovász meets Weisfeiler and Leman. In *International Colloquium on Automata, Languages, and Programming*, pages 40:1–40:14, 2018. 12

[101] A. Atserias, L Mancinska, D. E. Roberson, R. Sámal, S. Severini, and A. Varvitsiotis. Quantum and non-signalling graph isomorphisms. *Journal of Combinatorial Theory, Series B*, pages 289–328, 2019. 12

[102] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992. 12, 14

[103] Sandra Kiefer. *Power and Limits of the Weisfeiler-Leman Algorithm*. PhD thesis, Department of Computer Science, RWTH Aachen University, 2020. 12

[104] V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On the power of color refinement. In *International Symposium on Fundamentals of Computation Theory*, pages 339–350, 2015. 13

[105] S. Kiefer, P. Schweitzer, and E. Selman. Graphs identified by logics with counting. In *International Symposium on Mathematical Foundations of Computer Science*, pages 319–330, 2015. 13

[106] Christoph Berkholz, Paul S. Bonsma, and Martin Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory of Computing Systems*, 60(4):581–614, 2017. 13

[107] S. Kiefer and B. D. McKay. The iteration number of Colour Refinement. In *International Colloquium on Automata, Languages, and Programming*, pages 73:1–73:19, 2020. 13

[108] S. Kiefer and P. Schweitzer. Upper bounds on the quantifier depth for graph differentiation in first-order logic. In *Symposium on Logic in Computer Science*, pages 287–296, 2016. 13

[109] M. Lichter, I. Ponomarenko, and P. Schweitzer. Walk refinement, walk logic, and the iteration number of the Weisfeiler-Leman algorithm. In *Symposium on Logic in Computer Science*, pages 1–13, 2019. 13

[110] V. Arvind, F. Fuhlbrück, J. Köbler, and O. Verbitsky. On Weisfeiler-Leman invariance: Subgraph counts and related graph properties. In *International Symposium on Fundamentals of Computation Theory*, pages 111–125, 2019. 13

[111] Martin Fürer. On the combinatorial power of the Weisfeiler-Lehman algorithm. In *International Conference on Algorithms and Complexity*, pages 260–271, 2017. 13

[112] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems*, 2020. 13

[113] S. Kiefer, I. Ponomarenko, and P. Schweitzer. The Weisfeiler-Leman dimension of planar graphs is at most 3. *Journal of the ACM*, 66(6):44:1–44:31, 2019. 13

[114] L. Babai. Graph isomorphism in quasipolynomial time. In *Symposium on Theory of Computing*, pages 684–697, 2016. 13

[115] M. Grohe, P. Schweitzer, and D. Wiebking. Deep Weisfeiler Leman. *ArXiv preprint*, 2020. 13

[116] M. Grohe. Word2vec, Node2vec, Graph2vec, X2vec: Towards a theory of vector embeddings of structured data. In *Symposium on Principles of Database Systems*, pages 1–16, 2020. 13

[117] M. Grohe. The logic of graph neural networks. In *Symposium on Logic in Computer Science*, pages 1–17, 2021. 13, 17

[118] Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20:63:1–63:17, 2019. 22, 23, 24

[119] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Supplement 1):i47–i56, 2005. 24

[120] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, pages D431–3, 2004. 24

[121] Xifeng Yan, Hong Cheng, Jiawei Han, and Philip S. Yu. Mining significant graph patterns by leap search. In *International Conference on Management of Data*, pages 433–444, 2008. 24

[122] J. Kazius, R. McGuire, and R. Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal Medicinal Chemistry*, 48(13):312–320, 2005. 24

[123] Kaspar Riesen and Horst Bunke. IAM graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop*, pages 287–297, 2008. 24

[124] N. Wale, I. A. Watson, and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008. 24

[125] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *ArXiv preprint*, 2020. 24

[126] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015. 24, 25

[127] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 24

[128] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *International Conference on Learning Representations, Workshop on Representation Learning on Graphs and Manifolds*, 2019. 24

A Related work

In the following, we discuss relevant related work.

GNNs. Recently, GNNs [4, 19] emerged as the most prominent graph representation learning architecture. Notable instances of this architecture include, e.g., Duvenaud et al. [24], Hamilton et al. [25], and Veličković et al. [26], which can be subsumed under the message-passing framework introduced in Gilmer et al. [4]. In parallel, approaches based on spectral information were introduced in, e.g., Bruna et al. [27], Defferrard et al. [28], Gama et al. [29], Kipf and Welling [30], Levie et al. [31], and Monti et al. [32]—all of which descend from early work in Scarselli et al. [19], Baskin et al. [33], Goller and Küchler [34], Kireev [35], Merkwirth and Lengauer [36], Micheli and Sestito [37], Micheli [38], and Sperduti and Starita [39].

Limits of GNNs and more expressive architectures. Recently, connections between GNNs and Weisfeiler–Leman type algorithms have been shown [8, 9, 40, 41]. Specifically, Morris et al. [8] and Xu et al. [9] showed that the 1-WL limits the expressive power of any possible GNN architecture in terms of distinguishing non-isomorphic graphs. In turn, these results have been generalized to the k-WL, see, e.g., Morris et al. [8], Maron et al. [11], Morris et al. [12], Azizian and Lelarge [42], Geerts [43], Morris et al. [44], and connected to permutation-equivariant functions approximation over graphs, see, e.g., Geerts and Reutter [10], Azizian and Lelarge [42], Chen et al. [45], Maehara and NT [46]. Further, Aamand et al. [47] devised an improved analysis using randomization. Recent works have extended the expressive power of GNNs, e.g., by encoding vertex identifiers [48, 49], using random features [50–52], equivariant graph polynomials [53], homomorphism and subgraph counts [54–56], spectral information [57], simplicial [58] and cellular complexes [59], persistent homology [60], random walks [61, 62], graph decompositions [63], relational [64], distance [65] and directional information [66], subgraph information [5, 67–80], and biconnectivity [81]. See Morris et al. [5] for an in-depth survey on this topic. Geerts and Reutter [10] devised a general approach for bounding the expressive power of a large variety of GNNs utilizing the 1-WL or k-WL. Recently, Kim et al. [82] showed that transformer architectures [83] can simulate the 2-WL. Grohe [84] showed tight connections between GNNs' expressivity and circuit complexity. Moreover, Rosenbluth et al. [85] investigated the expressive power of different aggregation functions beyond sum aggregation.

GNN's generalization capabilities. Scarselli et al. [17] used classical techniques from learning theory [86] to show that GNNs' VC dimension [20] with piece-wise polynomial activation functions on a *fixed* graph, under various assumptions, is in $\mathcal{O}(P^2 n \log n)$, where P is the number of parameters and n is the order of the input graph. We note here that Scarselli et al. [17] analyzed a different type of GNN not aligned with modern GNN architectures [4]. Garg et al. [13] showed that the empirical Rademacher complexity, e.g., [21], of a specific, simple GNN architecture, using sum aggregation, is bounded in the maximum degree, the number of layers, Lipschitz constants of activation functions, and parameter matrices' norms. We note here that their analysis assumes weight sharing across layers. Liao et al. [15] refined these results via a PAC-Bayesian approach, further refined in Ju et al. [87]. Maskey et al. [16] used random graphs models to show that GNNs' generalization ability depends on the (average) number of vertices in the resulting graphs. Verma and Zhang [88] studied the generalization abilities of 1-layer GNNs in a transductive setting based on algorithmic stability. Similarly, Esser et al. [89] used stochastic block models to study the transductive Rademacher complexity [90, 91] of standard GNNs. Moreover, [14] leveraged results from graph property testing [92] to study the sample complexity of learning to distinguish various graph properties, e.g., planarity or triangle freeness, using graph kernels [1, 2]. We stress that all of the above approaches only consider classical graph parameters to bound the generalization abilities of GNNs. Finally, [93] showed negative results for GNNs' ability to generalize to larger graphs. However, the generalization properties of GNNs and their connection to expressivity is understood to a lesser extent.

Expressive power of k-WL. The Weisfeiler–Leman algorithm constitutes one of the earliest and most natural approaches to isomorphism testing [7, 94], and the theory community has heavily investigated it over the last few decades [18]. Moreover, the fundamental nature of the k-WL is evident from various connections to other fields such as logic, optimization, counting complexity, and quantum computing. The power and limitations of the k-WL can be neatly characterized in terms of logic and descriptive complexity [95, 96], Sherali-Adams relaxations of the natural integer linear optimization problem for the graph isomorphism problem [97–99], homomorphism counts [100], and quantum isomorphism games [101]. In their seminal paper, Cai et al. [102] showed that, for each k, a pair of non-isomorphic graphs of size $\mathcal{O}(k)$ exists not distinguished by the k-WL. Kiefer [103] gives a thorough survey of more background and related results concerning the expressive power of the k-WL. For $k = 1$, the power of the algorithm has been completely characterized [104, 105]. Moreover, upper bounds on the running time [106] and the number of iterations for $k = 1$ [107] and the non-oblivious $k = 2$ [108, 109] have been shown. For k in $\{1, 2\}$, Arvind et al. [110] studied the abilities of the (non-oblivious) k-WL to detect and count fixed subgraphs, extending the work of Fürer [111]. The former was refined in [112]. Kiefer et al. [113] showed that the non-oblivious 3-WL completely captures the structure of planar graphs. The algorithm for logarithmic k plays a prominent role in the recent result of [114] improving the best-known running time for the graph isomorphism problem. Recently, Grohe et al. [115] introduced the framework of Deep Weisfeiler–Leman algorithms, which allow the design of a more powerful graph isomorphism test than Weisfeiler–Leman type algorithms. Finally, the emerging connections between the Weisfeiler–Leman paradigm and graph learning are described in two recent surveys [5, 116].

B Extended notation

A *graph* G is a pair $(V(G), E(G))$ with *finite* sets of *vertices* or *nodes* $V(G)$ and *edges* $E(G) \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$. If not otherwise stated, we set $n := |V(G)|$, and the graph is of *order* n. We also call the graph G an n-order graph. For ease of notation, we denote the edge $\{u, v\}$ in $E(G)$ by (u, v) or (v, u). In the case of *directed graphs*, the set $E(G) \subseteq \{(u, v) \in V(G) \times V(G) \mid u \neq v\}$ and a *directed acyclic graph* (DAG) is a directed graph with no directed cycles. A *(vertex-)labeled graph* G is a triple $(V(G), E(G), \ell)$ with a (vertex-)label function $\ell\colon V(G) \to \mathbb{N}$. Then $\ell(v)$ is a *label* of v, for v in $V(G)$. An *attributed graph* G is a triple $(V(G), E(G), a)$ with a graph $(V(G), E(G))$ and (vertex-)attribute function $a\colon V(G) \to \mathbb{R}^{1\times d}$, for some $d > 0$. That is, contrary to labeled graphs, we allow for vertex annotations from an uncountable set. Then $a(v)$ is an *attribute* or *feature* of v, for v in $V(G)$. Equivalently, we define an n-order attributed graph $G := (V(G), E(G), a)$ as a pair $\mathbf{G} = (G, \mathbf{L})$, where $G = (V(G), E(G))$ and $\mathbf{L}$ in $\mathbb{R}^{n\times d}$ is a *vertex feature matrix*. Here, we identify $V(G)$ with $[n]$. For a matrix $\mathbf{L}$ in $\mathbb{R}^{n\times d}$ and v in $[n]$, we denote by $\mathbf{L}_{v\cdot}$ in $\mathbb{R}^{1\times d}$ the vth row of $\mathbf{L}$ such that $\mathbf{L}_{v\cdot} := a(v)$. We also write $\mathbb{R}^d$ for $\mathbb{R}^{1\times d}$.

The *neighborhood* of v in $V(G)$ is denoted by $N(v) := \{u \in V(G) \mid (v, u) \in E(G)\}$ and the *degree* of a vertex v is $|N(v)|$. In case of directed graphs, $N^+(u) := \{v \in V(G) \mid (v, u) \in E(G)\}$ and $N^-(u) := \{v \in V(G) \mid (u, v) \in E(G)\}$. The *in-degree* and *out-degree* of a vertex v are $|N^+(v)|$ and $|N^-(v)|$, respectively. Two graphs G and H are *isomorphic* and we write $G \simeq H$ if there exists a bijection $\varphi\colon V(G) \to V(H)$ preserving the adjacency relation, i.e., (u, v) is in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $E(H)$. Then φ is an *isomorphism* between G and H. In the case of labeled graphs, we additionally require that $l(v) = l(\varphi(v))$ for v in $V(G)$, and similarly for attributed graphs.

C Oblivious k-WL

Intuitively, to surpass the limitations of the 1-WL, the k-WL colors ordered subgraphs instead of a single vertex.[3] More precisely, given a graph G, the k-WL colors the tuples from $V(G)^k$ for $k \geq 2$ instead of the vertices. By defining a neighborhood between these tuples, we can define a coloring similar to the 1-WL. Formally, let G be a graph, and let $k \geq 2$. In each iteration, $t \geq 0$, the algorithm, similarly to the 1-WL, computes a *coloring* $C^k_t\colon V(G)^k \to \mathbb{N}$. In the first iteration, $t = 0$, the tuples $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ get the same color if they have the same atomic type, i.e., $C^k_0(\mathbf{v}) := \text{atp}(\mathbf{v})$. Here, we define the atomic type atp: $V(G)^k \to \mathbb{N}$, for $k > 0$, such that $\text{atp}(\mathbf{v}) = \text{atp}(\mathbf{w})$ for $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ if and only if the mapping $\varphi\colon V(G)^k \to V(G)^k$ where $v_i \mapsto w_i$ induces a partial isomorphism, i.e., we have $v_i = v_j \iff w_i = w_j$ and $(v_i, v_j) \in E(G) \iff (\varphi(v_i), \varphi(v_j)) \in E(G)$. Then, for each layer, $t > 0$, C^k_t is defined by

$$C^k_t(\mathbf{v}) := \mathsf{RELABEL}\big(C^k_{t-1}(\mathbf{v}), M_t(\mathbf{v})\big),$$

with $M_t(\mathbf{v})$ the multiset

$$M_t(\mathbf{v}) := \big(\{\!\!\{C^k_{t-1}(\phi_1(\mathbf{v}, w)) \mid w \in V(G)\}\!\!\}, \dots, \{\!\!\{C^k_{t-1}(\phi_k(\mathbf{v}, w)) \mid w \in V(G)\}\!\!\}\big),$$

and where

$$\phi_j(\mathbf{v}, w) := (v_1, \dots, v_{j-1}, w, v_{j+1}, \dots, v_k).$$

[3]There exists two definitions of the k-WL, the so-called oblivious k-WL and the folklore or non-oblivious k-WL; see Grohe [117]. There is a subtle difference in how they aggregate neighborhood information. Within the graph learning community, it is customary to abbreviate the oblivious k-WL as k-WL, a convention we follow in this paper.

That is, $\phi_j(\mathbf{v}, w)$ replaces the j-th component of the tuple $\mathbf{v}$ with the vertex w. Hence, two tuples are *adjacent* or *j-neighbors* if they are different in the jth component (or equal, in the case of self-loops). Hence, two tuples $\mathbf{v}$ and $\mathbf{w}$ with the same color in iteration $(t-1)$ get different colors in iteration t if there exists a j in $[k]$ such that the number of j-neighbors of $\mathbf{v}$ and $\mathbf{w}$, respectively, colored with a certain color is different.

We run the k-WL algorithm until convergence, i.e., until for t in $\mathbb{N}$

$$C^k_t(\mathbf{v}) = C^k_t(\mathbf{w}) \iff C^k_{t+1}(\mathbf{v}) = C^k_{t+1}(\mathbf{w}),$$

for all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$, holds. For such t, we define $C^k_\infty(\mathbf{v}) = C^k_t(\mathbf{v})$ for $\mathbf{v}$ in $V(G)^k$. At convergence, we call the partition of $V(G)^k$ induced by C^k_t the *stable partition*. We set $C^k_\infty(v) := C^k_\infty(v, \dots, v)$ and refer to this as the color of the vertex v.

Similarly to the 1-WL, to test whether two graphs G and H are non-isomorphic, we run the k-WL in "parallel" on both graphs. Then, if the two graphs have a different number of vertices colored c, for c in $\mathbb{N}$, the k-WL *distinguishes* the graphs as non-isomorphic. By increasing k, the algorithm gets more powerful in distinguishing non-isomorphic graphs, i.e., for each $k \geq 2$, there are non-isomorphic graphs distinguished by $(k+1)$-WL but not by k-WL [102]. For a finite set of graphs $S \subset \mathcal{G}$, we run the algorithm in "parallel" over all graphs in the set S.

D k-order GNNs

By generalizing Equation (1) in Section 1.1, following [5, 8, 44], we can derive k-GNNs computing features for all k-tuples $V(G)^k$, for $k > 0$, defined over the set of vertices of an attributed graph $G = (V(G), E(G), a)$ with features from $\mathbb{R}^d$. Concretely, in each layer, $t > 0$, for each k-tuple $\mathbf{v} = (v_1, \dots, v_k)$ in $V(G)^k$, we compute a feature

$$\mathbf{h}^{(t)}_{\mathbf{v}} := \mathsf{UPD}^{(t)}\Big(\mathbf{h}^{(t-1)}_{\mathbf{v}}, \mathsf{AGG}^{(t)}\big(\{\!\!\{\mathbf{h}^{(t-1)}_{\mathbf{v}}(\phi_1(\mathbf{v}, w)) \mid w \in V(G)\}\!\!\}, \dots,$$
$$\{\!\!\{\mathbf{h}^{(t-1)}_{\mathbf{v}}(\phi_k(\mathbf{v}, w)) \mid w \in V(G)\}\!\!\}\big)\Big).$$

Initially, for $t = 0$, we set

$$\mathbf{h}^{(0)}_{\mathbf{v}} := \mathsf{UPD}([\text{atp}(\mathbf{v}), a(v_1), \dots, a(v_k)]) \in \mathbb{R}^d,$$

i.e., the atomic type and the attributes of a given k-tuple determine the initial feature of a k-tuple's vertices. In the above, UPD, $\mathsf{UPD}^{(t)}$, and $\mathsf{AGG}^{(t)}$ may be differentiable parameterized functions, e.g., neural networks. In the case of graph-level tasks, e.g., graph classification, one additionally uses

$$\mathbf{h}_G := \mathsf{READOUT}\big(\{\!\!\{\mathbf{h}^{(L)}_{\mathbf{v}} \mid \mathbf{v} = (v, \dots, v), v \in V(G)\}\!\!\}\big) \in \mathbb{R}^d,$$

to compute a single vectorial graph representation based on the learned k-tuple features after iteration L.

D.1 Transfering VC bounds from GNNs to k-order GNNs and other more expressive architectures

In the following, we briefly sketch how Propositions 2.1 and 2.2, Corollary 2.3, and Theorem 2.6 can be lifted to k-GNNs. First, observe that we can simulate the computation of a k-GNN via a GNN on a sufficiently defined auxiliary graph. That is, the auxiliary graph contains a vertex for each k-tuple, and an edge connects two k-tuples j if they are j-neighbors for j in $[k]$; see Morris et al. [5] for details. Using a 1-WL equivalent GNN taking edge labels into account, we can extend Propositions 2.1 and 2.2 and Corollary 2.3 to k-GNNs. Similar reasoning applies to Theorem 2.6, i.e., we can apply the proof technique from Appendix G.3 to this auxiliary graph.

D.1.1 Architectures based on subgraph information

Further, we note that Propositions 2.1 and 2.2 also easily extend to recent GNN enhancements, e.g., subgraph-based [55] or subgraph-enhanced GNNs [67, 74]. Since suitably defined variations of the 1-WL, incorporating subgraph information at initialization, upper bound the architectures' expressive power, we can easily apply the reasoning behind the proofs of Propositions 2.1 and 2.2 to these cases. Hence, the architectures' VC dimensions are also tightly related to the number of graphs distinguishable by respective 1-WL variants.

E Relationship between VC dimension and generalization error

If we can bound the VC dimension of a hypothesis class $\mathcal{C}$ of GNNs, we directly get insights into its generalization ability, i.e., the difference of the empirical error $R_S(h)$ and the true error $R_{\mathcal{D}}(h)$ for $h \in \mathcal{C}$ and a data generating distribution $\mathcal{D}$.

Theorem E.1. *Let $\mathcal{C}$ be a class of GNNs, with finite VC dimension $\mathsf{VC\text{-}dim}(\mathcal{C}) = d$. Then for $\mathcal{C}$, for all $\varepsilon > 0$ and $\delta \in (0,1)$, using*

$$m = \mathcal{O}\bigg(\frac{1}{\varepsilon^2}\bigg(d \ln\bigg(\frac{d}{\varepsilon}\bigg) + \ln\bigg(\frac{1}{\delta}+1\bigg)\bigg)\bigg)$$

samples, for all data generating distributions $\mathcal{D}$, we have

$$\Pr_{S \simeq \mathcal{D}^m}(\forall h \in \mathcal{C} : |R_S(h) - R_{\mathcal{D}}(h)| \leq \varepsilon) \geq 1 - \delta.$$

This result was first proven by Vladimir Vapnik and Alexey Chervonenkis in 1960's; see, e.g., Mohri et al. [21] for a proof.

F Simple GNNs

We here provide more detail on the simple GNNs mentioned in Section 1.1. That is, for given d and L in $\mathbb{N}$, we define the class $\mathsf{GNN}_{\mathsf{mlp}}(d, L)$ of simple GNNs as L-layer GNNs for which, according to Equation (1), for each t in $[L]$, the aggregation function $\mathsf{AGG}^{(t)}$ is simply summation and the update function $\mathsf{UPD}^{(t)}$ is a multilayer perceptron $\mathsf{mlp}^{(t)} : \mathbb{R}^{2d} \to \mathbb{R}^d$ of width at most d. Similarly, the readout function in Equation (2) consists of a multilayer perceptron $\mathsf{mlp} : \mathbb{R}^d \to \mathbb{R}$ applied on the sum of all vertex features computed in layer L.[4] More specifically, GNNs in $\mathsf{GNN}_{\mathsf{mlp}}(d, L)$ compute on a graph $(G, \mathbf{L})$ in $\mathcal{G}_d$, for each $v \in V(G)$,

$$\mathbf{h}_v^{(t)} := \mathsf{mlp}^{(t)}\Big(\mathbf{h}_v^{(t-1)}, \sum_{u \in N(v)} \mathbf{h}_u^{(t-1)}\Big) \in \mathbb{R}^d, \tag{4}$$

for t in $[L]$ and $\mathbf{h}_v^{(0)} := \mathbf{L}_{v,\cdot}$, and

$$\mathbf{h}_G := \mathsf{mlp}\Big(\sum_{v \in V(G)} \mathbf{h}_v^{(L)}\Big) \in \mathbb{R}. \tag{5}$$

We also consider an even simpler class $\mathsf{GNN}_{\mathsf{slp}}(d, L)$ of $\mathsf{GNN}_{\mathsf{mlp}}(d, L)$ in which the multilayer perceptrons are in fact single layer perceptrons. That is, Equation (4) is replaced by

$$\mathbf{h}_v^{(t)} := \sigma_t\Big(\mathbf{h}_v^{(t-1)}\mathbf{W}_1^{(t)} + \sum_{u \in N(v)} \mathbf{h}_u^{(t-1)}\mathbf{W}_2^{(t)} + \mathbf{b}^{(t)}\Big) \in \mathbb{R}^d, \tag{6}$$

where $\mathbf{W}_1^{(t)}$ in $\mathbb{R}^{d \times d}$ and $\mathbf{W}_2^{(t)}$ in $\mathbb{R}^{d \times d}$ are weight matrices, and $\mathbf{b}^{(t)}$ in $\mathbb{R}^{1 \times d}$ is a bias vector, and $\sigma_t \colon \mathbb{R} \to \mathbb{R}$ is an activation function, for t in $[L]$. Similarly, Equation (5) is replaced by

$$\mathbf{h}_G := \sigma_{L+1}\Big(\sum_{v \in V(G)} \mathbf{h}_v^{(L)}\mathbf{w} + b\Big) \in \mathbb{R}. \tag{7}$$

with $\mathbf{w}$ in $\mathbb{R}^{d \times 1}$ a weight vector and b in $\mathbb{R}$ a bias value of the final readout layer. Also, $\sigma_{L+1} : \mathbb{R} \to \mathbb{R}$ is an activation function. We can thus represent elements in $\mathsf{GNN}_{\mathsf{slp}}(d, L)$ more succinctly by the following tuple of parameters,

$$\mathit{\Theta} = \Big(\mathbf{W}_1^{(1)}, \mathbf{W}_2^{(1)}, \mathbf{b}^{(1)}, \dots, \mathbf{W}_1^{(L)}, \mathbf{W}_2^{(L)}, \mathbf{b}^{(L)}, \mathbf{w}, b\Big),$$

together with the tuple of activation functions $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_L, \sigma_{L+1})$. We can equivalently view $\mathit{\Theta}$ as an element in $\mathbb{R}^{d(2dL+L+1)+1}$. Each $\mathit{\Theta}$ in $\mathbb{R}^{d(2dL+L+1)+1}$ and $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_{L+1})$ induces a permutation-invariant *graph function*

$$\mathsf{gnn}_{\mathit{\Theta},\boldsymbol{\sigma}} \colon \mathcal{G}_d \to \mathbb{R} \colon (G, \mathbf{L}) \mapsto \mathsf{gnn}_{\mathit{\Theta},\boldsymbol{\sigma}}(G, \mathbf{L}) := \mathbf{h}_G,$$

with $\mathbf{h}_G$ as defined in Equation (7).

[4]For simplicity we assume that all feature dimensions of the layers are fixed to d in $\mathbb{N}$ and also assume that the readout layer returns a scalar.

G Missing proofs

In the following, we outline missing proofs from the main paper.

G.1 Proofs of Proposition 2.1 and Proposition 2.2

We start with the general upper bound on the VC dimension in terms of the number of 1-WL-indistinguishable graphs.

Proposition G.1 (Proposition 2.1 in the main text)**.** *For all n, d, and L, the maximal number of graphs of order at most n with d-dimensional boolean features that can be shattered by L-layer GNNs is bounded by the maximal number ($m_{n,d,L}$) of 1-WL-distinguishable graphs. That is,*

$$\mathsf{VC\text{-}dim}_{\mathcal{G}^{\mathbb{B}}_{d,n}}\big(\mathsf{GNN}(L)\big) \leq m_{n,d,L}.$$

Proof. Clearly, every set $\mathcal{S}$ of $m_{n,d,L}+1$ graphs from $\mathcal{G}^{\mathbb{B}}_{d,n}$ contains at least two graphs $\mathbf{G}$ and $\mathbf{G}'$ not distinguishable by the 1-WL. Since GNNs cannot distinguish 1-WL-indistinguishable graphs [8, 9], they cannot tell $\mathbf{G}$ and $\mathbf{G}'$ apart and hence cannot not shatter $\mathcal{S}$. Hence, the VC dimension can be at most $m_{n,d,L}$. □

We next show a corresponding lower bound. In fact, the lower bound already holds for the class of simple GNNs of arbitrary width, that is for GNNs in $\mathsf{GNN}_{\mathsf{mlp}}(L) \coloneqq \bigcup_{d\in\mathbb{N}} \mathsf{GNN}_{\mathsf{mlp}}(d,L)$.

Proposition G.2 (Proposition 2.2 in the main paper)**.** *For all n, d, and L, all $m_{n,d,L}$ 1-WL-distinguishable graphs of order at most n with d-dimensional boolean features can be shattered by sufficiently wide L-layer GNNs. Hence,*

$$\mathsf{VC\text{-}dim}_{\mathcal{G}^{\mathbb{B}}_{d,n}}\big(\mathsf{GNN}(L)\big) = m_{n,d,L}.$$

Proof. For all i in $[m_{n,d,L}]$, choose $\mathbf{G}_i$ in $\mathcal{G}^{\mathbb{B}}_{d,n}$ such that $\mathcal{S} = \{\mathbf{G}_1,\dots,\mathbf{G}_{m_{n,d,L}}\}$ consists of the maximum number of graphs in $\mathcal{G}^{\mathbb{B}}_{d,n}$ pairwise distinguishable by the 1-WL after L iterations.

We next show that the class of simple GNNs which are wide enough, that is, $\mathsf{GNN}_{\mathsf{mlp}}(d',L)$ for large enough d', is sufficiently rich to shatter $\mathcal{S}$. That is, we show that for each $\mathcal{T} \subseteq \mathcal{S}$ there is a $\mathsf{gnn}_{\mathcal{T}}$ in $\mathsf{GNN}_{\mathsf{mlp}}(d',L)$ such that for all i in $[m_{n,d,L}]$:

$$\mathsf{gnn}_{\mathcal{T}}(\mathbf{G}_i) = \begin{cases} 1 & \text{if } \mathbf{G}_i \in \mathcal{T}\text{, and} \\ 0 & \text{otherwise.} \end{cases}$$

This shows that $\mathcal{S}$ is shattered by $\mathsf{GNN}_{\mathsf{mlp}}(d',L)$ and hence its VC dimension is at least $|\mathcal{S}| = m_{n,d,L}$, as desired.

Overview of the construction Intuitively, we will show that $\mathsf{GNN}_{\mathsf{mlp}}(d',L)$, with d' large enough, is powerful enough to return a one-hot encoding of the color histograms of graphs in $\mathcal{S}$. That is, there is a simple GNN gnn in $\mathsf{GNN}_{\mathsf{mlp}}(d',L)$ which in the MLP in its readout layer embeds a graph $\mathbf{G}_i$ in $\mathcal{S}$ as a vector $\mathbf{h}_{\mathbf{G}}$ in $\{0,1\}^{m_{n,d,L}}$ satisfying $(h_{\mathbf{G}})_i = 1$ if and only if $\mathbf{G}_i$ in $\mathcal{T}$ and $i \in [m_{n,d,L}]$. Then, we extend the readout multilayer perceptron of gnn by one more layer such that on input $\mathbf{G}$ the revised GNN evaluates to the scalar

$$g_{\mathbf{G}} \coloneqq \mathsf{sign}(\mathbf{h}_{\mathbf{G}} \cdot \mathbf{w}^{\mathsf{T}} - 1) \in \{0,1\},$$

with $\mathbf{w}$ in $\mathbb{R}^{d'\times 1}$. We observe that given $\mathcal{T} \subseteq \mathcal{S}$ it suffices to let the parameter vector $\mathbf{w}$ be the indicator vector for $\mathcal{T}$. Indeed, this ensures that $g_{\mathbf{G}} = 1$ if and only if $\mathbf{G}$ is in a color class included in $\mathcal{T}$. We can explore all such subsets $\mathcal{T}$ of $\mathcal{S}$ by varying $\mathbf{w}$; hence, this GNN will shatter $\mathcal{S}$.

Encoding 1-WL colors via GNNs We proceed with the construction of the required GNN. For simplicity of exposition, in the description below we will construct GNN layers of non-uniform width. One can easily obtain uniform width by padding each layer. First, by Morris et al. [8, Theorem 2], there exists a GNN architecture with feature dimension (at most) n and consisting of L layers such that for each $\mathbf{G}_i$ in $\mathcal{S}$ it computes 1-WL-equivalent vertex features $\mathbf{f}_v$ in $\mathbb{R}^{1\times d}$ for $v \in V(G_i)$. That is, for vertices v and w in $V(G_i)$ it holds that

$$\mathbf{f}_v = \mathbf{f}_w \iff C^1_L(v) = C^1_L(w).$$

We note here that we can construct a single GNN architecture for all graphs by applying [8, Theorem 2] over the disjoint union the graphs in $\mathcal{S}$. This increases the width from n to $nm_{n,d,L}$.

Encoding 1-WL histograms via GNNs Moreover, again by [44, Theorem 2] there exists $\mathbf{W}$ in $\mathbb{R}^{nm_{n,d,L} \times nm_{n,d,L}}$ and $\mathbf{b}$ in $\mathbb{R}^{nm_{n,d,L}}$ such that

$$\sigma\Big(\sum_{v \in V(G)} \mathbf{f}_v \mathbf{W} + \mathbf{b}\Big) = \sigma\Big(\sum_{v \in V(H)} \mathbf{f}_v \mathbf{W} + \mathbf{b}\Big) \iff h_{\mathbf{G}} = h_{\mathbf{H}},$$

for graphs $\mathbf{G}$ and $\mathbf{H}$ in $\mathcal{S}$. We use ReLU as activation function σ here, just as in [8]. Other activation functions could be used as well [117]. Hence, for each graph in $\mathcal{S}$, we have a vector in $\mathbb{R}^{1 \times nm_{n,d,L}}$ uniquely encoding it. Since the number of vertices n is fixed, there exists a number M in $\mathbb{N}$ such that $M\sigma(\sum_{v \in V(G)} \mathbf{W}\mathbf{f}_v)$ is in $\mathbb{N}^{1 \times nm_{n,d,L}}$ for all $\mathbf{G}$ in $\mathcal{S}$. Moreover, observe that there exists a matrix $\mathbf{W}'$ in $\mathbb{N}^{nm_{n,d,L} \times 2m_{n,d,L}}$ such that

$$M\sigma\Big(\sum_{v \in V(G)} \mathbf{f}_v \mathbf{W} + \mathbf{b}\Big)\mathbf{W}' = M\sigma\Big(\sum_{v \in V(H)} \mathbf{f}_v \mathbf{W} + \mathbf{b}\Big)\mathbf{W}' \iff h_{\mathbf{G}} = h_{\mathbf{H}},$$

for graphs $\mathbf{G}$ and $\mathbf{H}$ in $\mathcal{S}$. For example, we can set

$$\mathbf{W}' = \begin{bmatrix} K^{nm_{n,L}-1} & \cdots & K^{nm_{n,L}-1} \\ \vdots & \cdots & \vdots \\ K^0 & \cdots & K^0 \end{bmatrix} \in \mathbb{N}^{nm_{n,d,L} \times 2m_{n,d,L}}$$

for sufficiently large $K > 1$. Hence, the above GNN architecture computes a vector $\mathbf{k}_{\mathbf{G}}$ in $\mathbb{N}^{2m_{n,d,L}}$ containing $2m_{n,d,L}$ occurrences of a natural number uniquely encoding each color histogram for each graph $\mathbf{G}$ in $\mathcal{S}$.

We next turn $\mathbf{k}_{\mathbf{G}}$ into our desired $\mathbf{h}_{\mathbf{G}}$ as follows. We first define an intermediate vector $\mathbf{h}'_{\mathbf{G}}$ whose entries will be used to check which color histogram is returned. More specifically, we define

$$\mathbf{h}'_{\mathbf{G}} = \mathsf{lsig}(\mathbf{k}_{\mathbf{G}} \cdot (\mathbf{w}'')^{\intercal} + \mathbf{b}),$$

with $\mathbf{w}'' = (1, -1, 1, -1, \ldots, 1, -1) \in \mathbb{R}^{2m_{n,L}}$ and $\mathbf{b} = (-c_1 - 1, c_1 + 1, -c_2 - 1, c_2 + 1, \ldots, -c_{m_{n,d,L}} - 1, c_{m_{n,d,L}} + 1) \in \mathbb{R}^{2m_{n,d,L}}$ with c_i the number encoding the ith color histogram. We note that for odd i,

$$(h'_{\mathbf{G}})_i \coloneqq \mathsf{lsig}(\mathsf{col}(G) - c_i - 1) = \begin{cases} 1 & \mathsf{col}(G) \geq c_i \\ 0 & \text{otherwise.} \end{cases}$$

and for even i,

$$(h'_{\mathbf{G}})_i \coloneqq \mathsf{lsig}(-\mathsf{col}(G) + c_i + 1) = \begin{cases} 1 & \mathsf{col}(G) \leq c_i \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $((h'_{\mathbf{G}})_i, (h'_{\mathbf{G}})_{i+1})$ are both 1 if and only if $\mathsf{col}(G) = c_i$. We thus obtain $\mathbf{h}_{\mathbf{G}}$ by combining $((h'_{\mathbf{G}})_i, (h'_{\mathbf{G}})_{i+1})$ using an "AND" encoding (e.g., $\mathsf{lsig}(x + y - 1)$) applied to pairs of consecutive entries in $\mathbf{h}'_{\mathbf{G}}$. That is,

$$\mathbf{h}_{\mathbf{G}} \coloneqq \mathsf{lsig}\left(\mathbf{h}'_{\mathbf{G}} \cdot \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \end{pmatrix} - (1, 1, \ldots, 1) \right) \in \mathbb{R}^{m_{d,n,L}}$$

We thus see that a 3-layer MLP suffices for the readout layer of the simple GNN, finishing the proof. We remark that the maximal width is $2nm_{n,d,L}$, so we can take $d' = 2nm_{n,d,L}$. □

G.2 Proof of Proposition 2.5

We now prove Proposition 2.5.

Proposition G.3 (Proposition 2.5 in the main text)**.** *There exists a family $\mathcal{F}_b$ of simple 2-layer GNNs of width two and of bitlength $\mathcal{O}(b)$ using a piece-wise linear activation such that its VC dimension is* exactly *b.*

Proof. We first show the lower bound. We fix some $n \geq 1$. We shall construct a family of GNNs whose weights have bitlength $O(n)$ and a family of n graphs shattered by these GNNs. Thereto, for all $\mathbf{x} = (x_1, \ldots, x_n)$ in $\{0,1\}^n$, we let

$$\rho(\mathbf{x}) \coloneqq \sum_{i=1}^{n} (2^{-2i+1} + x_i 2^{-2i}).$$

Written in binary, we have

$$\rho(\mathbf{x}) = 0.1x_1 1x_2 1x_3 \ldots 1x_n.$$

Observe that

$$\frac{1}{2} \leq \rho(\mathbf{x}) \leq 1. \tag{8}$$

For $1 \leq k \leq n$, we let

$$\rho_k(\mathbf{x}) \coloneqq \rho\big((x_{k+1}, \ldots, x_{k+n})\big) = \sum_{i=1}^{n} (2^{-2i+1} + x_{k+i} 2^{-2i}),$$

where $x_{k+i} \coloneqq 0$ for $k + i > n$. Then it follows from (8) that

$$\frac{1}{2} \leq \rho_k(\mathbf{x}) \leq 1. \tag{9}$$

We claim that

$$\rho_k(\mathbf{x}) = 2^{2k}\rho(\mathbf{x}) - \underbrace{\left(\sum_{i=1}^{k} 2^{2(k-i)+1} - \sum_{i=1}^{k} 2^{2(k-n-i)+1}\right)}_{=:a_k} - \underbrace{\sum_{i=1}^{k-1} 2^{2(k-i)} x_i}_{:=b_k(\mathbf{x})} - x_k \tag{10}$$

Indeed, we have

$$\begin{aligned}
2^{2k}\rho(\mathbf{x}) &= \sum_{i=1}^{n} \left(2^{2(k-i)+1} + x_i 2^{2k-2i}\right) \\
&= \sum_{i=1}^{n+k} \left(2^{2(k-i)+1} + x_i 2^{2(k-i)}\right) - \sum_{i=n+1}^{n+k} 2^{2(k-i)+1} \\
&= \sum_{i=1}^{k} 2^{2(k-i)+1} - \sum_{i=n+1}^{n+k} 2^{2(k-i)+1} + \sum_{i=1}^{k} x_i 2^{2(k-i)} + \sum_{i=k+1}^{n+k} \left(2^{2(k-i)+1} + x_i 2^{2(k-i)}\right) \\
&= \sum_{i=1}^{k} 2^{2(k-i)+1} - \sum_{i=1}^{k} 2^{2(k-n-i)+1} + \sum_{i=1}^{k} x_i 2^{2(k-i)} + \sum_{i=1}^{n} \left(2^{-2i+1} + x_{k+i} 2^{-2i}\right) \\
&= \sum_{i=1}^{k} 2^{2(k-i)+1} - \sum_{i=1}^{k} 2^{2(k-n-i)+1} + \sum_{i=1}^{k-1} x_i 2^{2(k-i)} + x_k + \rho_k(\mathbf{x}) \\
&= a_k + b_k(\mathbf{x}) + x_k + \rho_k(\mathbf{x}),
\end{aligned}$$

which proves (10). Now let

$$c_k(\mathbf{x}) \coloneqq b_k(\mathbf{x}) + a_k + 1.$$

Then by (9) and (10), we have

$$x_k - \frac{1}{2} \leq 4^k \rho(\mathbf{x}) - c_k(\mathbf{x}) \leq x_k. \tag{11}$$

For $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ in $\{0,1\}^n$, we write $\mathbf{x} \neq_k \mathbf{y}$ if $x_i \neq y_i$ for some $i < k$. Observe that $\mathbf{x} \neq_k \mathbf{y}$ implies $\big|b_k(\mathbf{x}) - b_k(\mathbf{y})\big| \geq 4$ and thus

$$\big|c_k(\mathbf{x}) - c_k(\mathbf{y})\big| \geq 4. \tag{12}$$

Let $A : \mathbb{R} \to \mathbb{R}$ be the continuous piecewise-linear function defined by

$$A(x) := \begin{cases} 0 & \text{if } x < 0, \\ 2x & \text{if } 0 \leq x < \frac{1}{2}, \\ 1 & \text{if } \frac{1}{2} \leq x < 1, \\ 3 - 2x & \text{if } 1 \leq x < \frac{3}{2} \\ 0 & \text{if } \frac{3}{2} \leq x. \end{cases}$$

Since $x_k \in \{0, 1\}$, by (11) we have

$$x_k = A\big(4^k \rho(\mathbf{x}) - c_k(\mathbf{x})\big). \tag{13}$$

If follows from (12) that for $\mathbf{y}$ with $\mathbf{y} \neq_k \mathbf{x}$ we have

$$A\big(4^k \rho(\mathbf{x}) - c_k(\mathbf{y})\big) = 0. \tag{14}$$

Let

$$\mathcal{C}_k := \Big\{ c_k(\mathbf{y}) \ \Big|\ \mathbf{y} \in \{0,1\}^n \Big\}.$$

Then

$$x_k = \sum_{c \in \mathcal{C}_k} A\big(4^k \rho(\mathbf{x}) - c\big). \tag{15}$$

Note that the only dependence on $\mathbf{x}$ of the right-hand side of (15) is in $\rho(\mathbf{x})$, because $\mathcal{C}_k$ does not depend on $\mathbf{x}$.

Observe that $|\mathcal{C}_k| = 2^{k-1}$, because $c_k(\mathbf{y})$ only depends on $y_1, \dots, y_{k-1} \in \{0,1\}$ and is distinct for distinct values of the y_i. We have

$$a_k = \sum_{i=1}^{k} 2^{2(k-i)+1} - \underbrace{\sum_{i=1}^{k} 2^{2(k-n-i)+1}}_{=:s \leq 1} = 2 \sum_{i=0}^{k-1} 4^i - s = \frac{2}{3}\big(4^k - 1\big) - s.$$

Thus

$$\frac{2}{3}\big(4^k - 1\big) - 1 \leq a_k \leq \frac{2}{3}\big(4^k - 1\big).$$

Furthermore,

$$0 \leq b_k(\mathbf{x}) \leq \sum_{i=1}^{k-1} 2^{2(k-i)} = \sum_{i=1}^{k-1} 4^{k-i} = 4 \sum_{i=0}^{k-2} 4^i = \frac{4}{3}\big(4^{k-1} - 1\big).$$

Thus

$$\frac{2}{3}\big(4^k - 1\big) \leq c \leq \frac{2}{3}\big(4^k - 1\big) + \frac{4}{3}\big(4^{k-1} - 1\big) + 1 = 4^k - 1. \tag{16}$$

Now for each ρ in $\mathbb{R}$ we construct a 2-layer GNN $\mathfrak{G}_\rho$ as follows:

- Initially, all nodes v carry the 1-dimensional feature $\mathbf{h}_v^{(0)} := 1$.
- The first layer computes the 2-dimensional feature $\begin{pmatrix} h_{v,1}^{(1)} \\ h_{v,2}^{(1)} \end{pmatrix}$ defined by

$$h_{v,1}^{(1)} := \sum_{w \in N(v)} \rho \cdot \mathbf{h}_w^{(0)} - \rho,$$
$$h_{v,2}^{(1)} := \sum_{w \in N(v)} \mathbf{h}_w^{(0)} - 1.$$

- The second layer computes the 1-dimensional feature $\mathbf{h}_v^{(2)}$ defined by

$$\mathbf{h}_v^{(2)} = A\left(h_{v,1}^{(1)} - \sum_{w \in N(v)} h_{w,2}^{(1)} \right).$$

- The readout functions just takes the sum of all the $\mathbf{h}_v^{(2)}$.

We define a graph F_k as follows. The graph F_k is a forest of height 2.

- F_k has a root node r_c for every $c \in \mathcal{C}_k$.
- Each r_c has a child s_c and 4^k additional children $t_{c,1}, \ldots, t_{c,4^k}$.
- The $t_{c,i}$ are leaves.
- Each s_c has children $u_{c,1}, \ldots, u_{c,c}$.
- The $u_{c,i}$ are leaves.

Now we run the GNN $\mathfrak{G}_\rho$ on F_k with $\rho = \rho(\mathbf{x})$ for some $\mathbf{x} = (x_1, \ldots, x_n)$ in $\{0,1\}^n$.

- We have $\mathbf{h}_v^{(0)} = 1$ for all v in $V(F_k)$.
- We have

$$\begin{aligned}\mathbf{h}_{t_{c,i},1}^{(1)} &= \mathbf{h}_{u_{c,i},1}^{(1)},\\ \mathbf{h}_{s_c,1}^{(1)} &= c\rho,\\ \mathbf{h}_{r_c,1}^{(1)} &= 4^k\rho,\end{aligned}$$

and

$$\begin{aligned}\mathbf{h}_{t_{c,i},2}^{(1)} &= \mathbf{h}_{u_{c,i},1}^{(1)} = 0,\\ \mathbf{h}_{s_c,2}^{(1)} &= c,\\ \mathbf{h}_{r_c,2}^{(1)} &= 4^k.\end{aligned}$$

- We have

$$\begin{aligned}\mathbf{h}_{t_{c,i}}^{(2)} &= A\big(-4^k\big) = 0\\ \mathbf{h}_{u_{c,i}}^{(2)} &= A\big(-c\big) = 0\\ \mathbf{h}_{s_c}^{(2)} &= A\big(c\rho - 4^k\big) = 0\\ \mathbf{h}_{r_c}^{(2)} &= A\big(4^k\rho - c\big) = \begin{cases} x_k & \text{if } c = c_k(\mathbf{x})\\ 0 & \text{otherwise}\end{cases} \qquad \text{by (13) and (14).}\end{aligned}$$

To see that the first three equalities hold, recall that $A(x) \neq 0$ only if $0 < x < \frac{3}{2}$. Thus $A(-4^k) = 0$. Moreover, by (16) we have $2 \leq c$ and thus $A(c) = 0$. Finally, $A\big(c\rho - 4^k\big) = 0$ because $\rho < 1$ and $c \leq 4^k - 1$ by (16) and therefore $c\rho - 4^k < 0$.

- As there is exactly one node r_c with $c = c_k(\mathbf{x})$, the readout is $\sum_{v \in V(F_k)} \mathbf{h}_v^{(2)} = x_k$.

Hence

$$\mathfrak{G}_{\rho(\mathbf{x})}(F_k) = x_i$$

Thus the GNNs $\mathfrak{G}_{\rho(\mathbf{x})}$ for $\mathbf{x} \in \{0,1\}^n$ shatter the set $\{F_1, \ldots, F_n\}$. Since the bitlength is upper bounded by $\mathcal{O}(b)$ and the number of parameters in the above construction is constant, the hypothesis set is finite, and the upper bound follows from standard learning-theoretic results; see, e.g., [21]. □

G.3 Proof of Theorem 2.6

In the following, we outline the proof of Theorem 2.6. First, we define feedforward neural networks and show how simple GNNs can be interpreted as such.

Feedforward neural networks. A *feedforward neural network* (FNN) is specified by a tuple $N = (\mathcal{N}, \beta, \gamma)$ where $\mathcal{N}$ describes the underlying *architecture* and where β and γ define the *parameters* or *weights*. More specifically, $\mathcal{N} = \big(V^{\mathcal{N}}, E^{\mathcal{N}}, i_1, \ldots, i_p, o_1, \ldots, o_q, \alpha^{\mathcal{N}}\big)$ where $(V^{\mathcal{N}}, E^{\mathcal{N}})$ is a *finite* DAG with p *input nodes* $i_1, \ldots, i_p$ of in-degree 0, and q *output nodes* $o_1, \ldots, o_q$ of out-degree 0. No other nodes have in- or out-degree zero. Moreover, $\alpha^{\mathcal{N}}$ is a function assigning to each node $v \in V^{\mathcal{N}} \setminus \{i_1, \ldots, i_p\}$ an activation function $\alpha(v) : \mathbb{R} \to \mathbb{R}$. Furthermore, the function $\beta \colon V^{\mathcal{N}} \setminus \{i_1, \ldots, i_p\} \to \mathbb{R}$ is a function assigning biases to nodes, and finally, the function $\gamma \colon E^{\mathcal{N}} \to \mathbb{R}$ assigns weights to edges. For an FNN N, we define its *size* s as the number of biases and weights, that is $s = |V^{\mathcal{N}}| - p + |E^{\mathcal{N}}|$.

Given an FNN $N = (\mathcal{N}, \beta, \gamma)$, we get a function $\mathsf{fnn}_N \colon \mathbb{R}^p \to \mathbb{R}^q$ defined as follows. For all v in $V^{\mathcal{N}}$, we define a function $h_v^N \colon \mathbb{R}^p \to \mathbb{R}$ such for $\mathbf{a} = (a_1, \dots, a_p)$ in $\mathbb{R}^p$,

$$h_v^N(\mathbf{a}) \coloneqq \begin{cases} a_j & \text{if } v = i_j \text{ for } j \in [p], \\ \alpha^{\mathcal{N}}(v)\Big(\sum_{u \in N^+(v)} \gamma(u,v) h_u^N(\mathbf{a}) + \beta(v)\Big) & \text{otherwise.} \end{cases}$$

Finally, $\mathsf{fnn}_N \colon \mathbb{R}^p \to \mathbb{R}^q$ is defined as $\mathbf{a} \mapsto \mathsf{fnn}_N(\mathbf{a}) \coloneqq \big(h_{o_1}^N(\mathbf{a}), \dots, h_{o_q}^N(\mathbf{a})\big)$.

Simple GNNs as FNNs. We next connect simple GNNs in $\mathsf{GNN}_{\mathsf{slp}}(d, L)$ to FNNs. As described in Section F such GNNs are specified by $L + 1$ activation functions $\boldsymbol{\sigma} \coloneqq (\sigma_1, \dots, \sigma_{L+1})$ and a weight vector Θ in $\mathbb{R}^{d(2dL+L+1)+1}$ describing weight matrices and bias vectors in all the layers. We show that for any attributed graph of order at most n $\mathbf{G} = (G, \mathbf{L})$ in $\mathcal{G}_{n,d}$ with $G = (V(G), E(G))$ and $\mathbf{L}$ in $\mathbb{R}^{n \times d}$ there exists an architecture $\mathcal{N}_G(\boldsymbol{\sigma})$ such that for any weight assignment Θ in $\mathbb{R}^{d(2dL+L+1)+1}$ of the GNN, there exists $\beta_\Theta \colon V^{\mathcal{N}_G} \to \mathbb{R}$ and $\gamma_\Theta \colon E^{\mathcal{N}_G} \to \mathbb{R}$, satisfying

$$\mathsf{gnn}_{\Theta,\boldsymbol{\sigma}}(G, \mathbf{L}) = \mathsf{fnn}_{N=(\mathcal{N}_G(\boldsymbol{\sigma}), \beta_\Theta, \gamma_\Theta)}(\mathbf{L}'), \tag{17}$$

where $\mathbf{L}'$ in $\mathbb{R}^{nd}$ is the (column-wise) concatenation of the rows of the matrix $\mathbf{L}$. Moreover, $\mathcal{N}_G(\boldsymbol{\sigma})$ is of polynomial size in the number of vertices and edges in G, the feature dimension d, and the number of layers L. Furthermore, $\mathcal{N}_G(\boldsymbol{\sigma})$ has only a single output node o.

The idea behind the construction of $\mathcal{N}_G(\boldsymbol{\sigma})$ is to consider the tree unraveling or unrolling, see, e.g., [44], of the computation of $\mathsf{gnn}_{\Theta,\boldsymbol{\sigma}}(G, \mathbf{L})$ but instead of a tree we represent the computation more concisely as a DAG. The DAG $\mathcal{N}_G(\boldsymbol{\sigma})$ is defined as follows.

- The node set $V^{\mathcal{N}_G}$ consists of the following nodes.
  - We have input nodes $i_{v,j}$ for v in $V(G)$ and j in $[d]$ which will take the vertex labels L_{vj} in $\mathbb{R}$ as value.
  - For each t in $[L]$, we include the following nodes: $n_{v,j}^{(t)}$ for v in $V(G)$, j in $[d]$.
  - Finally, we have a single output node o.

  We thus have $d(L + 1)|V(G)| + 1$ nodes in total.
- The edge set $E^{\mathcal{N}_G}$ consists of the following edges.
  - We have edges encoding the adjacency structure of the graph G in every layer. More specifically, we have an edge $e_{u,j,v,k,t} \coloneqq (n_{u,j}^{(t-1)}, n_{v,k}^{(t)})$ whenever u in $N(v) \cup \{v\}$ and where u and v are in $V(G)$, j and k in $[d]$, and t in $\{2, \dots, L\}$.
  - We also have edges from the input nodes $i_{u,j}$ to $n_{v,k}^{(1)}$ for all u in $N_G(v) \cup \{v\}$ and where u and v are in $V(G)$ and j and k in $[d]$.
  - Finally, we have edges connecting the last layer nodes to the output, i.e., edges $e_{v,j} \coloneqq (n_{v,j}^{(L)}, o)$ for all v in $V(G)$ and j in $[d]$.

  We thus have $d|V(G)| + d^2((L-1)(E(G) + V(G)) + (E(G) + V(G)))$ edges in total.
- Finally, we define the activation functions.
  - $\alpha^{\mathcal{N}}(n_{v,j}^{(t)}) \coloneqq \sigma_t$ for all v in $V(G)$, j in $[d]$ and t in $[L]$, and $\alpha^{\mathcal{N}}(o) \coloneqq \sigma_{L+1}$.

This fixes the architecture $\mathcal{N}_G(\boldsymbol{\sigma})$. We next verify Equation (17). Let Θ in $\mathbb{R}^{d(2dL+L+1)+1}$ and $\mathbf{G} = (G, \mathbf{L})$ in $\mathcal{G}^{n,d}$. Let $\mathcal{N}_G(\boldsymbol{\sigma})$ be the architecture defined above for the graph G. We define β_Θ and γ_Θ, as follows.

- $\beta_\Theta \coloneqq V^{\mathcal{N}_G} \to \mathbb{R}$ is such that $\beta_\Theta(n_{v,j}^{(t)}) = b_j^{(t)}$ for all v in $V(G)$, j in $[d]$ and t in $[L]$. We also set $\beta_\Theta(o) = b$.
- $\gamma_\Theta \colon E^{\mathcal{N}_G} \to \mathbb{R}$ is such that $\gamma_\Theta(e_{u,j,v,k,t}) \coloneqq W_{jk}^{(2,t)}$ if $u \neq v$ and $\gamma_\Theta(e_{u,j,v,k,t}) \coloneqq W_{jk}^{(1,t)}$ otherwise, and $\gamma^\Theta(e_{v,j}) = w_j$, for u and v in $V(G)$, j and k in $[d]$, and t in $[L]$.

Note that we share weights across edges that correspond to the same edge in the underlying graph.

Now, if we denote by $\mathbf{f}_v^{(t)}$ the feature vector in $\mathbb{R}^d$ computed in the tth layer by the GNN $\mathsf{gnn}_{\Theta,\boldsymbol{\sigma}}(G, \mathbf{L})$, then it is readily verified, by induction on the layers, that for $N = (\mathcal{N}_G, \alpha_{\boldsymbol{\sigma}}, \beta_\Theta, \gamma_\Theta)$:

$$h_{n_{v,j}^{(t)}}^N = \mathbf{f}_{v,j}^{(t)} \text{ and thus } h_o^N \coloneqq \sigma_{L+1}\left(\sum_{v \in V(G)} \sum_{j \in [d]} w_j \mathbf{f}_{vj}^{(L)} + b\right),$$

from which Equation (17) follows.

We next expand the construction by obtaining an FNN that simulates GNNs on multiple input graphs. More specifically, consider a set $\mathfrak{G}$ consisting of m graphs $\mathbf{G}_1 = (G_1, \mathbf{L}_1), \ldots, \mathbf{G}_m = (G_m, \mathbf{L}_m)$ in $\mathcal{G}_{n,d}$ and a GNN in $\mathsf{GNN}_{\mathsf{slp}}(d, L)$ using activation functions $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{L+1})$ in its layers. We first construct an FNN architecture $\mathcal{N}_{G_i}(\boldsymbol{\sigma})$ for each graph separately, as explained above, such that for every Θ in $\mathbb{R}^P$, there exists β_Θ and γ_Θ such that

$$\mathsf{gnn}_{\Theta,\boldsymbol{\sigma}}(G_i, \mathbf{L}_i) = \mathsf{fnn}_{N_{G_i} := (\mathcal{N}_{G_i}(\boldsymbol{\sigma}), \beta_\Theta, \gamma_\Theta)}(\mathbf{L}'_i),$$

with $\mathbf{L}'_i$ is the concatenation of rows in $\mathbf{L}_i$, as before.

Then, let $\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma})$ be the FNN architecture obtained as the disjoint union of $\mathcal{N}_{G_1}(\boldsymbol{\sigma}), \ldots, \mathcal{N}_{G_m}(\boldsymbol{\sigma})$. If we denote by o_i the output node of $\mathcal{N}_{G_i}(\boldsymbol{\sigma})$ in $\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma})$, then we have again that for every Θ in $\mathbb{R}^P$, there exists β_Θ and γ_Θ such that

$$\mathsf{gnn}_{\Theta,\boldsymbol{\sigma}}(\mathbf{G}_i) = h_{o_i}^{N_{\mathfrak{G}} := (\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma}), \beta_\Theta, \gamma_\Theta)}(\mathbf{L}')$$

for all i in $[m]$, where $\mathbf{L}' \coloneqq (\mathbf{L}'_1, \ldots, \mathbf{L}'_m)$.

We recall that, for t in $[L]$, the nodes in $\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma})$ are of the form $\nu_{v,j}^{(t),g}$ for v in V_{G_g}, j in $[d]$ and g in $[m]$. In layer $L+1$, we have the output nodes $o_1, \ldots, o_m$. If the order of the graphs in $\mathfrak{G}$ is at most n, then every layer, except the last one, has ndm nodes. The last layer only has m nodes.

Piece-wise polynomial activation functions. A *piece-wise polynomial activation function* $\sigma_{p,\delta} : \mathbb{R} \to \mathbb{R}$ is specified by a partition of $\mathbb{R}$ into p intervals I_j and corresponding polynomials $p_j(x)$ of degree at most δ, for j in $[p]$. That is, $\sigma_{p,\delta}(x) = p_j(x)$ if x in I_j. Examples of $\sigma_{p,\delta}(x)$ are: $\mathsf{sign}(x) \colon \mathbb{R} \to \mathbb{R} \colon x \mapsto \mathbf{1}_{x \geq 0}$ for which $p = 2$ and $\delta = 0$, $\mathsf{relu}(x) \colon \mathbb{R} \to \mathbb{R} \colon x \mapsto \max(0, x)$ for which $p = 2$ and $\delta = 1$, and $\mathsf{lsig}(x) \colon \mathbb{R} \to \mathbb{R} \colon x \mapsto \max(0, \min(1, x))$ for which $p = 3$ and $\delta = 1$. *Piece-wise linear activation functions* are of the form $\sigma_{p,1}$, i.e., they are defined in terms of linear polynomials. The parameters of an activation function $\sigma_{p,\delta}$ consist of the coefficients of the polynomials involved and the boundary points (numbers) of the intervals in the partition of $\mathbb{R}$.

Proof of Theorem 2.6. We next derive upper bounds on the VC dimension of GNNs by the approach used in Bartlett et al. [118], where they used it for bounding the VC dimension of FNNs using piecewise polynomial activation functions. Their approach allows for recovering known bounds on the VC dimension of FNNs in a unified manner. As we will see, the bounds by Bartlett et al. [118] for FNNs naturally translate to bounds for GNNs.

Assume d and L in $\mathbb{N}$. In this section, we will consider the subclass of GNNs in $\mathsf{GNN}_{\mathsf{slp}}(d, L)$ that use *piece-wise polynomial activation functions* with $p > 0$ pieces and degree $\delta \geq 0$. As explained in Section F, $d(2dL + L + 1) + 1$ is the total number of (learnable) parameters for our GNNs in $\mathsf{GNN}_{\mathsf{slp}}(d, L)$. As shorthand notation, we define $P \coloneqq d(2dL + L + 1) + 1$. We first bound $\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,n}}\big(\mathsf{GNN}_{\mathsf{slp}}(d, L)\big)$ and then use this bound to obtain a bound for $\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,\leq u}}\big(\mathsf{GNN}_{\mathsf{slp}}(d, L)\big)$.

We take $\mathfrak{G}$ consisting of m graphs $\mathbf{G}_1 = (G_1, \mathbf{L}_1), \ldots, \mathbf{G}_m = (G_m, \mathbf{L}_m)$ in $\mathcal{G}_{n,d}$ and consider the FNN architecture $\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma})$ define above with output nodes $o_1, \ldots, o_m$. Let $\mathsf{tresh} : \mathbb{R} \to \mathbb{R}$ such that $\mathsf{tresh}(x) = 1$ if $x \geq 2/3$ and $\mathsf{tresh}(x) = 0$ if $x \leq 1/3$. We will bound

$$K' \coloneqq \big|\big\{\big(\mathsf{tresh}(h_{o_1}^{N_{\mathfrak{G}}}(\mathbf{L}')), \ldots, \mathsf{tresh}(h_{o_m}^{N_{\mathfrak{G}}}(\mathbf{L}'))\big) \colon N_{\mathfrak{G}} \coloneqq (\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma}), \beta_\Theta, \gamma_\Theta), \Theta \in \mathbb{R}^P\big\}\big|,$$

as this number describes how many $0/1$ patterns can occur when Θ ranges over $\mathbb{R}^P$. These $0/1$ patterns correspond, by the construction of $N_{\mathfrak{G}}$ and the semantics of its output nodes, to how many of the input graphs in $\mathfrak{G}$ can be shattered. To bound K' using the approach in Bartlett et al. [118] we need to slightly change the activation function σ_{L+1} used in the FNN architecture. The reason is that Bartlett et al. use the sign function to turn a real-valued function into a $0/1$-valued function. In contrast, we use the tresh function described above.

Let $\boldsymbol{\sigma}' \coloneqq (\sigma_1, \ldots, \sigma_L, \sigma_{L+1} - 1/3)$. We will bound K' by bounding

$$K \coloneqq \big|\big\{\big(\mathsf{sign}(h_{o_1}^{N_{\mathfrak{G}}}(\mathbf{L}')), \ldots, \mathsf{sign}(h_{o_m}^{N_{\mathfrak{G}}}(\mathbf{L}'))\big) \colon N_{\mathfrak{G}} \coloneqq (\mathcal{N}_{\mathfrak{G}}(\boldsymbol{\sigma}'), \beta_\Theta, \gamma_\Theta), \Theta \in \mathbb{R}^P\big\}\big|.$$

Note that $K' \leq K$ because if $\mathsf{tresh}(\sigma_{L+1})(\mathbf{x}) = 1$ then $\sigma_{L+1}(\mathbf{x}) \geq 2/3$ and hence $\sigma_{L+1}(\mathbf{x}) - 1/3 > 0$ and hence $\mathsf{sign}(\sigma_{L+1}(\mathbf{x}) - 1/3) = 1$. Similarly, $\mathsf{tresh}(\sigma_{L+1})(\mathbf{x}) = 0$ then $\sigma_{L+1}(\mathbf{x}) \leq 1/3$ and hence $\sigma_{L+1}(\mathbf{x}) - 1/3 \leq 0$ and hence $\mathsf{sign}(\sigma_{L+1}(\mathbf{x}) - 1/3) = 0$.

Then, if $\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,n}}\big(\mathsf{GNN}_{\mathsf{slp}}(d,L)\big) = m$ then $K \geq 2^m$. We thus bound K in terms of a function κ in m and then use $2^m \leq \kappa(m)$ to find an upper bound for m, i.e., an upper bound for $\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,n}}\big(\mathsf{GNN}_{\mathsf{slp}}(d,L)\big)$. To bound K we can now use the approach of Bartlett et al. [118]. In a nutshell, the entire parameter space $\mathbb{R}^P$ is partitioned into pieces $S_1,\dots,S_\ell$ such that whenever Θ and Θ' belong to the same piece (i) they incur the same sign pattern in $\{0,1\}^m$; and (ii) each $h_{o_1}^{N_{\mathfrak{G}}}(\mathbf{L}')$ is a polynomial of degree at most $1+L\delta^L$. For $\delta=0$, these are polynomials in $d+1$ variables, for $\delta>0$, the number of variables is P. Crucial in Bartlett's approach is the following lemma.

Lemma G.4 (Lemma 17 in Bartlett et al. [118]). *Let $p_1(\mathbf{x}),\dots,p_r(\mathbf{x})$ be polynomials of degree at most δ and in variables $\mathbf{x}$ satisfying $|\mathbf{x}|\leq r$, where $|\cdot|$ denotes the number of components of a vector. Then*

$$\Big|\big\{\big(\mathsf{sign}(p_1(\Theta)),\dots,\mathsf{sign}(p_r(\Theta))\big) \mid \Theta\in\mathbb{R}^{|\mathbf{x}|}\big\}\Big| \leq 2\left(\frac{2er\delta}{|\mathbf{x}|}\right)^{|\mathbf{x}|}.$$

Given property (ii) of the pieces $S_1,\dots,S_\ell$, we can apply the above lemma to the polynomials $h_{o_1}^{N_{\mathfrak{G}}}(\mathbf{L}'),\dots,h_{o_m}^{N_{\mathfrak{G}}}(\mathbf{L}')$ and, provided that the number of variables is at most m, obtain a bound for K by $\ell 2\left(\frac{2em}{d+1}\right)^{d+1}$, when $\delta=0$, and $K\leq \ell 2\left(\frac{2em(1+L\delta^L)}{P}\right)^{P}$ for $\delta>0$.

It then remains to bound the number of parts ℓ. Bartlett et al. show how to do this inductively (on the number of layers), again using Lemma G.4. More precisely, every node in the FNN architecture is associated with a number of polynomials. In layer t we have nmd nodes (number of computation nodes), and we associate with each node p polynomials (number of breakpoints of activation function) of degree at most $1+(t-1)\delta^{t-1}$ and have $(2d+1)d$ variables for $\delta=0$ and $(2d+1)dt$ variables for $\delta>0$. We then get, for $\delta=0$,

$$K \leq 2^L\left(\left(\frac{2edmnp}{(2d+1)d}\right)^{(2d+1)d}\right)^{L} 2\left(\frac{2em}{d+1}\right)^{d+1}, \tag{18}$$

and for $\delta>0$,

$$K \leq \prod_{t=1}^{L} 2\left(\frac{2edmnp(1+(t-1)\delta^{t-1})}{(2d+1)dt}\right)^{(2d+1)dt} 2\left(\frac{2em(1+L\delta^L)}{P}\right)^{P}. \tag{19}$$

These are precisely the bounds given in Bartlett et al. [118] applied to our FNN. It is important, however, to note that this upper bound is only valid when Lemma G.4 can be applied, and hence, the number of variables must be smaller than the number of polynomials. For t in $[L]$, we must have that the number of variables is less than the number of polynomials. We have $nmdp$ polynomials, and up to layer t we have $(2d+1)dt$ parameters (variables). Hence, we must have $(2d+1)dt\leq nmdp$ or $(2d+1)t\leq nmp$, and also $D\leq m$ or $P\leq nmdp$ for $\delta>0$. For $\delta=0$, we need $(2d+1)d\leq nmdp$ (or $(2d+1)\leq nmp$) and $d+1\leq m$. The following conditions are sufficient

$$P\leq m \text{ for } \delta>0, \text{ and } 2d+1\leq m \text{ for } \delta=0. \tag{†}$$

FNN size reduction based on 1-WL. We next bring in 1-WL into consideration by collapsing computation nodes in $N_{\mathfrak{G}}$ in each layer based on their equivalence with regards to 1-WL. In other words, if we assume that the graphs to be shattered have at most u vertex colors, then we have at most u—rather than n—computation nodes per graph. This implies that the parameter n in the above expression can be replaced by u.

As a consequence, following Bartlett et al. [118], using the weighted AM–GM inequality on the right-hand side of the inequalities (18) and (19), we obtain a bound for the VC dimension by finding maximal m satisfying, for $\delta=0$,

$$2^m \leq 2^{L+1}\left(\frac{2ep(udL+1)m}{P}\right)^{P}$$

and for $\delta>0$,

$$2^m \leq 2^{L+1}\left(\frac{2ep\big(ud\sum_{t=1}^{L}(1+(t-1)\delta^{t-1})+(1+L\delta^L)\big)m}{\frac{L(L+1)}{2}(2d+1)d+P}\right)^{\frac{L(L+1)}{2}(2d+1)d+P}.$$

Such m are found in [118], resulting in the following bounds.

Proposition G.5 ([118] modified for our FNN $N_{\mathfrak{G}}$)**.**

$$\mathsf{VC\text{-}dim}_{\mathcal{G}_{d,\leq u}}\big(\mathsf{GNN}_{\mathsf{slp}}(d,L)\big) \leq \begin{cases} \mathcal{O}(Ld^2\log(p(udL+1))) & \textit{if } \delta = 0 \\ \mathcal{O}(L^2d^2\log(p(udL+1))) & \textit{if } \delta = 1 \\ \mathcal{O}(L^2d^2\log(p(udL+1)) + L^3d^2\log(\delta)) & \textit{if } \delta > 1. \end{cases}$$

We can simplify this to $\mathcal{O}(P\log(puP))$, $\mathcal{O}(LP\log(puP))$, and $\mathcal{O}(LP\log(puP) + L^2P\log(\delta))$, respectively. We note that since these are larger than P, the condition (†) is satisfied.

H Experimental evaluation

In the following, we investigate how well the VC dimension bounds from the previous section hold in practice. Specifically, we answer the following questions.

Q1 How does the number of parameters influence GNNs' generalization performance?
Q2 How does the number of 1-WL-distinguishable graphs influence GNNs' generalization performance?
Q3 How does the bitlength influence a GNN's ability to fit random data?

The source code of all methods and evaluation procedures is available at `https://www.github.com/chrsmrrs/wl_vs_vc/`.

Datasets. To investigate questions **Q1** and **Q2**, we used the datasets ENZYMES [119, 120], MCF-7 [121], MCF-7H [121], MUTAGENICITY [122, 123], and NCI1 and NCI109 [6, 124] provided by Morris et al. [125]. See Table 3 for dataset statistics and properties. For question **Q3**, to investigate the influence of bitlength on GNN's VC dimension, we probed how well GNNs can fit random data. Hence, the experiments on these datasets aim at empirically verifying the VC dimension bounds concerning bitlength. To that, we created a synthetic dataset; see Appendix I.3. Since it is challenging to simulate different bitlengths without specialized hardware, we resorted to simulating an increased bitlength via an increased feature dimension; see Appendix I.4.

All experiments are therefore conducted with standard 32-bit precision. We also experimented with 64-bit precision but observed no clear difference. Furthermore, 16-bit precision proved numerically unstable in this setting.

Neural architectures. For the experiments regarding **Q1** and **Q2**, we used the simple GNN layer described in Appendix I.1 using a reLU activation function, ignoring possible edge labels. To answer question **Q1**, we fixed the number of layers to five and chose the feature dimension d in $\{4, 16, 256, 1\,024\}$. To answer **Q2**, we set the feature dimension d to 64 and choose the number of layers from $\{0, \dots, 6\}$. We used sum pooling and a two-layer MLP for all experiments for the final classification. To investigate **Q3**, we used the architecture described in Appendix I.2. In essence, we used a 2-layer MLP for the message generation function in each GNN layer and added batch normalization [126] before each non-linearity and fixed the number of layers to 3, and varied the feature dimension d in $\{4, 16, 64, 256\}$.

Experimental protocol and model configuration. For the experiments regarding **Q1** and **Q2**, we uniformly and at random choose 90% of a dataset for training and the remaining 10% for testing. We repeated each experiment five times and report mean test accuracies and standard deviations. We optimized the standard cross entropy loss for 500 epochs using the ADAM optimizer [127]. Moreover, we used a learning rate of 0.001 across all experiments and no learning rate decay or dropout. For **Q3**, we set the learning rate to 10^{-4} and the number of epochs to 100 000, and repeated each experiment 50 times. All architectures were implemented using PYTORCH GEOMETRIC [128] and executed on a workstation with 128GB RAM and an NVIDIA Tesla V100 with 32GB memory.

H.1 Results and discussion

In the following, we answer questions **Q1** to **Q3**.

Q1. See Table 1 and Figure 4. Increasing the feature dimension d increases the average difference between train and test accuracies across all datasets. For example, on the ENZYMES dataset, the difference increases from around 5% for $d = 4$ to more than 45% for $d = 1024$. However, we also observe that the difference does not increase when reaching near-perfect training accuracies, i.e., going from $d = 256$ to $d = 1\,024$ does not increase the difference. Hence, the results show that the number of parameters plays a crucial role in GNNs' generalization ability, in accordance with Theorem 2.6.

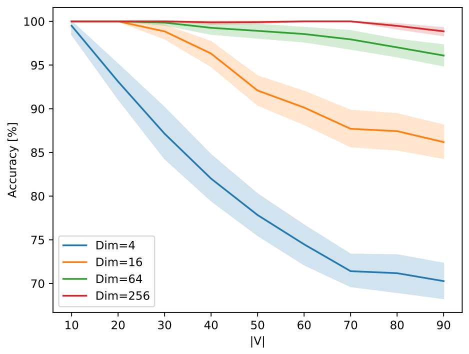


Figure 2: GNN's ability to fit the synthetic dataset for different feature dimensions in $\{4, 16, 64, 256\}$.

Q2. See Table 2. The results indicate that the number of 1-WL-distinguishable graphs ($m_{n,d,L}$) influence GNNs' generalization properties. For example, on the MUTAGENICITY dataset, after two iterations, the number of unique histograms computed by 1-WL stabilizes, and similarly, the generalization error stabilizes as well. Similar effects can be observed for the ENZYMES, NCI1, and NCI109 datasets. Hence, our results largely confirm Propositions 2.1 and 2.2.

Q3. See Figure 2. Increasing the feature dimension boosts the model's capacity to fit random class labels, indicating that increased bitlength implies an increased VC dimension. For example, for an order of 70, a GNN using a feature dimension of 4 cannot reach an accuracy of over 75%. In contrast, feature dimensions 64 and 256 can almost fit such data. Moreover, for larger graphs, up to order 90, a GNN with a feature dimension of 256 can almost perfectly fit random class labels, with a feature dimension of 64 only slightly worse, confirming Proposition 2.5.

I Additional experimental data and results

Here, we report on additional experimental details, results, and dataset generation.

I.1 Simple GNN layer used for Q1 and Q2

The simple GNN layer used in **Q1** and **Q2** updates the feature of vertex v at layer t via

$$\mathbf{f}_v^{(t)} := \sigma\Big(\mathbf{f}_v^{(t-1)}\mathbf{W}_1^{(t)} + \sum_{u \in N(v)} \mathbf{f}_u^{(t-1)}\mathbf{W}_2^{(t)}\Big) \in \mathbb{R}^d, \tag{20}$$

where $\mathbf{W}_1^{(t)}$ and $\mathbf{W}_2^{(t)} \in \mathbb{R}^{d \times d}$ are parameter matrices. In the experiments, we used reLU activation functions.

I.2 GNN architecture used for Q3

For the experiments on **Q3** we extend the simple GNN layer from Equation (20) used in **Q1** and **Q2**. We update the feature of vertex v at layer t via

$$\mathbf{f}_v^{(t)} := \sigma\Big(\mathrm{BN}\Big(\mathbf{f}_v^{(t-1)}\mathbf{W}^{(t)} + \sum_{u \in N(v)} \mathsf{mlp}^{(t)}(\mathbf{f}_u^{(t-1)})\Big)\Big) \in \mathbb{R}^d, \tag{21}$$

where BN is a batch normalization module [126] and $\mathsf{mlp}^{(t)}$ is a two-layer perceptron with architecture,

$$\text{Linear} \rightarrow \text{BN} \rightarrow \mathsf{reLU} \rightarrow \text{Linear}.$$

We, therefore, use a normalized 2-layer MLP to generate messages in each layer. We found this change necessary to ensure smooth convergence on the challenging synthetic task posed by **Q3**, where the GNN has to memorize an arbitrary binary graph labeling. Moreover, in the experiments, we used reLU activation functions.

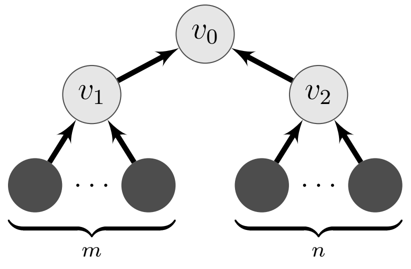


Figure 3: A visualization of a $T_{m,n}$.

I.3 Synthetic dataset generation

To address **Q3**, we aim to empirically estimate how well GNNs of different sizes can fit arbitrary binary labelings of graphs. We construct a synthetic dataset that focuses on a simple class of trees. Formally, for two natural numbers m and n in $\mathbb{N}_{\geq 0}$, we define the graph $T_{m,n} = (V_{m,n}, E_{m,n})$ as a directed tree with vertex set $V = \{v_0, \dots, v_{m+n+3}\}$. The root v_0 has two children v_1 and v_2. The remaining $m + n$ vertices form the leaves such that vertex v_1 has m children and v_2 has n children. Figure 3 provides a visual example.

For a chosen k in $\mathbb{N}$, $k \geq 4$, we define:

$$\mathcal{T}_k = \Big\{ T_{m,n} \mid 0 \leq m \leq \Big\lfloor \frac{(k-3)}{2} \Big\rfloor, n = k - 3 - m \Big\}.$$

Therefore, $\mathcal{T}_k$ contains all distinct graphs $T_{m,n}$ with $|V_{m,n}(T_{m,n})| = k$. In particular, we observe $|\mathcal{T}_k| = \lfloor \frac{(k-3)}{2} \rfloor$.

For $k \in \{10, 20, \dots, 90\}$, we aim to test how well a GNN with a given feature dimension d in $\{4, 16, 64, 256\}$ can learn binary labelings $y \colon \mathcal{T}_k \to \{0, 1\}$ of $\mathcal{T}_k$. The labeling y is obtained by sampling the label $y(T)$ uniformly at random for all T in $\mathcal{T}_k$. We then train a GNN model with stochastic gradient descent to minimize the binary cross entropy on the dataset $(\mathcal{T}_k, y)$. For each combination of k and d, we repeat the experiment 50 times. We resample a new labeling y and a new random initialization of the GNN model in each repetition.

I.4 Simulating bitlength via higher feature dimension

Here, we outline how to simulate a higher bitlength via a higher feature dimension. Assume the simple GNN layer of Equation (20). Clearly, we can express the matrices $\mathbf{W}_1^{(t)}$ and $\mathbf{W}_2^{(t)}$ as the sum of k matrix with smaller bitlength, e.g.,

$$\mathbf{W}_2^{(t)} = \mathbf{W}_2^{(1,t)} + \cdots + \mathbf{W}_2^{(k,t)}.$$

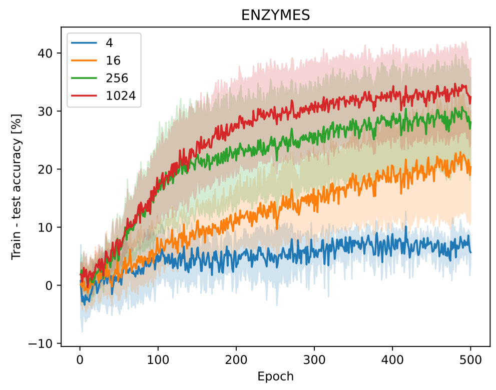
Hence, we can re-write the aggregation in Equation (20) as

$$\sum_{u \in N(v)} \mathbf{f}_u^{(t-1)} \big[\mathbf{W}_2^{(1,t)}, \cdots, \mathbf{W}_2^{(k,t)} \big] \cdot \mathbf{M} \in \mathbb{R}^d,$$

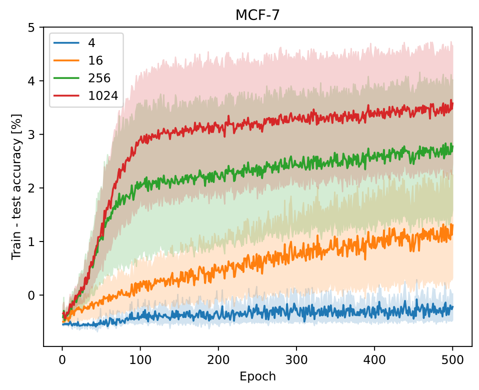
where $[\cdots]$ denotes column-wise matrix concatenation and $\mathbf{M}$ in $\{0,1\}^{kd \times d}$ is a matrix such that

$$\sigma\Big(\mathbf{f}_v^{(t-1)} \mathbf{W}_1^{(t)} + \sum_{u \in N(v)} \mathbf{f}_u^{(t-1)} \big[\mathbf{W}_2^{(1,t)}, \cdots, \mathbf{W}_2^{(k,t)} \big] \cdot \mathbf{M} \Big) = \sigma\Big(\mathbf{f}_v^{(t-1)} \mathbf{W}_1^{(t)} + \sum_{u \in N(v)} \mathbf{f}_u^{(t-1)} \mathbf{W}_2^{(t)} \Big),$$

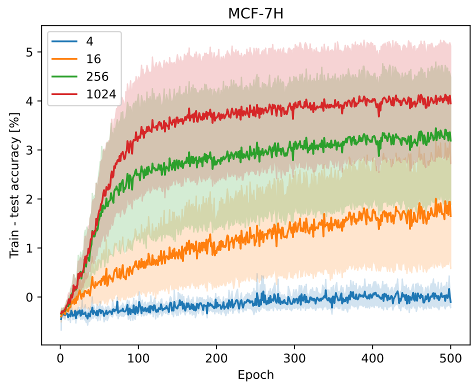
i.e., the matrix $\mathbf{M}$ sums together columns of the aggregated features such that they have feature dimension d.



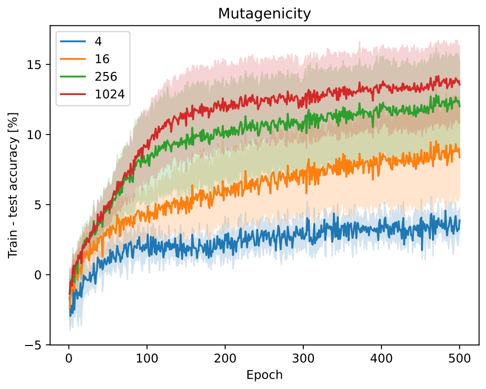
(a) ENZYMES



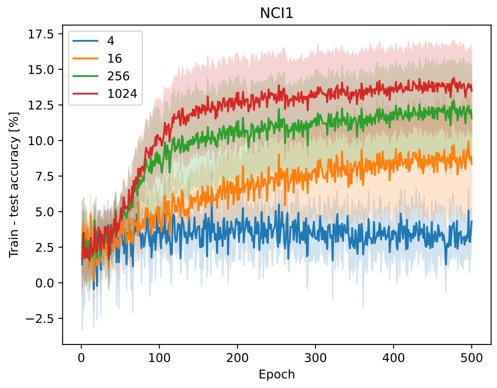
(b) MCF-7



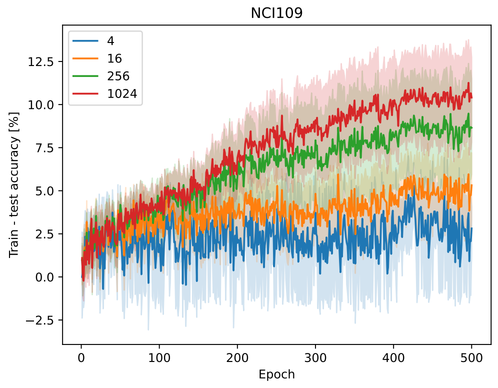
(c) MCF-7H



(d) MUTAGENICITY



(e) NCI1



(f) NCI109

Figure 4: Difference between train and test accuracy for different feature dimensions in $\{4, 16, 256, 1\,024\}$.

Table 1: Train and test classification accuracies with different numbers of parameters, using five layers, studying how the number of parameters influences generalization.

Dimension	Split	Dataset					
		ENZYMES	MCF-7	MCF-7H	MUTAGENICITY	NCI1	NCI109
4	Train	31.0 ±3.1	92.1 ±0.4	92.1 ±0.4	79.7 ±0.9	76.7 ±7.3	66.4 ±9.5
	Test	25.3 ±5.2	92.4 ±0.2	92.3 ±0.3	75.8 ±0.9	72.4 ±7.1	63.6 ±9.1
16	Train	76.8 ±6.4	96.0 ±0.1	96.5 ±0.3	92.7 ±2.7	88.4 ±6.2	86.4 ±0.9
	Test	41.7 ±9.4	93.2 ±0.5	93.1 ±0.4	79.8 ±2.2	76.1 ±2.1	78.6 ±1.9
256	Train	98.2 ±3.6	99.7 < 0.1	99.9 < 0.1	100.0 ±0.0	99.8 < 0.1	97.5 ±2.1
	Test	54.7 ±2.4	94.0 ±0.2	93.6 ±0.2	80.7 ±1.0	81.8 ±1.5	82.1 ±1.0
1 024	Train	99.8 ±0.2	99.8 < 0.1	99.8 ±0.1	99.9 ±0.2	99.8 ±0.1	98.6 ±1.0
	Test	54.3 ±2.3	93.8 ±0.3	93.6 ±0.2	81.7 ±0.8	80.5 ±1.0	82.9 ±0.9

Table 2: Train and test classification accuracies using different numbers of layers and a feature dimension of 64, studying how the number of different color histograms influences generalization.

Layers	Split	Dataset					
		ENZYMES	MCF-7	MCF-7H	MUTAGENICITY	NCI1	NCI109
0	Train	40.7 ±0.5	91.7 ±0.1	91.8 ±0.1	77.2 ±0.3	74.5 ±0.3	73.1 ±0.5
	Test	33.7 ±1.6	91.9 < 0.1	91.2 ±0.1	75.7 ±1.2	67.9 ±1.3	71.5 ±0.7
	Difference	7.0 ±1.9	-0.2 ±0.1	1.0 ±0.1	1.5 ±1.3	6.5 ±1.2	1.6 ±0.6
	# Histograms	385	11 533	19 625	2 819	2 889	2 929
1	Train	66.7 ±3.6	91.8 ±0.1	92.1 < 0.1	90.9 ±0.1	92.0 ±1.5	83.4 ±1.5
	Test	52.3 ±5.0	91.9 < 0.1	91.4 ±0.1	82.0 ±1.0	78.6 ±1.3	76.1 ±1.0
	Difference	14.4 ±5.2	<0.1 ±0.1	0.1 ±0.1	8.9 ±1.3	13.4 ±0.9	7.3 ±0.7
	# Histograms	595	25 417	26 037	3 624	3 906	3 950
2	Train	93.5 ±2.1	92.0 ±0.2	91.9 ±0.3	96.9 ±1.9	98.3 ±0.5	91.1 ±0.5
	Test	62.7 ±7.2	92.1 ±0.1	91.0 ±0.6	82.5 ±1.0	80.5 ±1.3	78.1 ±1.5
	Difference	39.9 ±5.5	-0.1 ±0.2	1.0 ±0.3	14.4 ±1.0	17.8 ±1.0	13.0 ±1.5
	# Histograms	595	26 872	27 353	4 239	4 027	4 055
3	Train	98.0 ±2.5	92.1 ±0.3	92.1 ±0.2	99.4 ±0.9	99.8 ±0.1	93.6 ±1.2
	Test	58.7 ±5.3	92.1 ±0.2	91.5 ±0.2	82.8 ±1.0	83.5 ±0.7	77.8 ±1.8
	Difference	39.4 ±2.8	0.1 ±0.2	1.0 ±0.1	16.6 ±1.0	16.3 ±0.7	15.8 ±1.4
	# Histograms	595	27 048	27 524	4 317	4 039	4 067
4	Train	99.8 ±0.3	92.0 ±0.1	92.2 ±0.2	99.1 ±0.2	99.8 < 0.1	96.9 ±1.0
	Test	62.7 ±2.5	92.1 ±0.1	91.5 ±0.2	82.7 ±0.8	83.2 ±0.4	79.8 ±1.2
	Difference	37.1 ±2.5	-0.1 ±0.1	1.0 ±0.1	16.4 ±0.7	16.6 ±0.4	17.2 ±0.8
	# Histograms	595	27 059	OOM	4 317	4 039	4 067
5	Train	98.9 ±1.9	92.1 ±0.2	92.4 ±0.2	99.9 ±0.2	99.8 ±0.0	97.7 ±0.9
	Test	57.0 ±3.9	92.3 ±0.2	91.6 ±0.2	83.0 ±0.8	84.1 ±1.1	79.6 ±0.5
	Difference	41.9 ±2.9	-0.2 ±0.2	1.0 ±0.2	16.9 ±0.7	15.7 ±1.1	18.1 ±0.5
	# Histograms	595	OOM	OOM	4 317	4 039	4 067
6	Train	99.4 ±0.8	92.0 ±0.2	92.2 ±0.2	99.1 ±1.9	99.6 ±0.6	95.2 ±1.9
	Test	54.0 ±2.3	92.2 ±0.2	91.4 ±0.4	83.5 ±1.0	83.4 ±1.3	79.2 ±1.3
	Difference	44.4 ±1.9	-0.2 ±0.1	1.0 ±0.2	15.6 ±1.2	16.2 ±0.9	16.0 ±2.1
	# Histograms	595	OOM	OOM	4 317	4 039	4 067

Table 3: Dataset statistics and properties.

Dataset	Properties					
	Number of graphs	Number of classes/targets	∅ Number of nodes	∅ Number of edges	Node labels	Edge labels
ENZYMES	600	6	32.6	62.1	✓	✗
MCF-7	27 770	2	26.4	28.5	✓	✓
MCF-7H	27 770	2	47.3	49.4	✓	✓
MUTAGENICITY	4 337	2	30.3	30.8	✓	✓
NCI1	4 110	2	29.9	32.3	✓	✗
NCI109	4 127	2	29.7	32.1	✓	✗