Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting

Giorgos Bouritsas [1] **Fabrizio Frasca** [2] **Stefanos Zafeiriou** [1] **Michael M. Bronstein** [1 2]

Abstract

Graph Neural Networks (GNNs) have been recently found to suffer from important limitations regarding their ability to capture the structure of the underlying graph. It has been shown that the expressive power of standard GNNs is bounded by the Weisfeiler-Lehman (WL) graph isomorphism test, from which they inherit proven limitations such as the inability to detect and count graph substructures. On the other hand, there is significant empirical evidence that substructures are often informative for downstream tasks, suggesting that it is desirable to design GNNs capable of leveraging this important source of information. To this end, we propose a novel topologically-aware message passing scheme based on subgraph isomorphism counting. We show that our architecture allows incorporating domain-specific inductive biases and that it is strictly more expressive than the WL test, while being able to disambiguate even hard instances of graph isomorphism. In contrast to recent works, our approach does not attempt to adhere to the WL hierarchy and therefore retains the locality and linear complexity of standard GNNs.

1. Introduction

The field of graph representation learning has undergone a rapid growth in the past few years. In particular, Graph Neural Networks (GNNs), a family of neural architectures designed for irregularly structured data, have been successfully applied to problems spanning a plethora of applications (Ying et al., 2018a; Fout et al., 2017; Duvenaud et al., 2015; Kipf et al., 2018; Battaglia et al., 2016). Most GNN architectures are based on message passing (Gilmer et al., 2017), where at each layer the nodes update their hidden representations by aggregating information from their neighbours via a permutation invariant function (as a consequence they are invariant to graph isomorphism). This kind of symmetry is not always desirable, and thus different inductive biases that disambiguate the neighbours have been proposed, such as directional biases for geometric graphs (Masci et al.; Monti et al., 2017; Bouritsas et al., 2019; Klicpera et al., 2020; de Haan et al., 2020), or positional biases for protein sequences (Ingraham et al., 2019).

The structure of the graph itself does not usually explicitly take part in the aggregation function. In fact, most models rely on multiple message passing steps as a means for each node to discover the global structure of the graph. However, this is not generally feasible, since it was proven that GNNs are at most as powerful as the Weisfeiler Lehman (WL) test, that limiting their abilities to adequately exploit the graph structure, e.g. by counting substructures (Arvind et al., 2019; Chen et al., 2020). This uncovers a crucial limitation of GNNs, as substructures have been widely recognised as important in the study of complex networks (e.g. functional groups and rings in molecules or cliques in Protein-Protein Interaction networks and social networks).

In this work, we design a topologically-aware graph neural model by enhancing the message passing scheme with structural biases; each message is transformed differently depending on the topological relationship between the endpoint nodes. In order to achieve this, we construct structural identifiers that are assigned to either the vertices or the edges of the graph and are extracted by subgraph isomorphism counting. Intuitively, in this way we partition the nodes or the edges of each graph in different equivalence classes reflecting topological characteristics that are shared both between nodes in each graph individually and across different graphs. Our approach leverages domain-specific knowledge by choosing a collection of substructures that are known to be of importance in the graphs at hand. In this way, we model the most discriminative structural biases, which at the same time are amenable to generalisation. We show that our model is at least as powerful as traditional GNNs, while being strictly more expressive for the vast of majority of substructures. In the limit, i.e. when the substructures are

[1]Department of Computing, Imperial College London, United Kingdom [2]Twitter, London, United Kingdom. Correspondence to: Giorgos Bouritsas <g.bouritsas@imperial.ac.uk>.

Graph Representation Learning and Beyond (GRL+) Workshop at the 37^{th} International Conference on Machine Learning, Vienna, Austria.

allowed to be the size of the graph, our model can yield a unique representation for every isomorphism class and is thus universal.

2. Preliminaries

Let $G = (\mathcal{V}_G, \mathcal{E}_G)$ be a graph with vertex set $\mathcal{V}_G$ and undirected edge set $\mathcal{E}_G$. A subgraph $G_S = (\mathcal{V}_{G_S}, \mathcal{E}_{G_S})$ of G is any graph with $\mathcal{V}_{G_S} \subseteq \mathcal{V}_G, \mathcal{E}_{G_S} \subseteq \mathcal{E}_G$. When $\mathcal{E}_{G_S}$ includes all the edges of G with endpoints in $\mathcal{V}_{G_S}$, i.e. $\mathcal{E}_{G_S} = \{(v,u) \in \mathcal{E} : v, u \in \mathcal{V}_{G_S}\}$, the subgraph is said to be *induced*.

Isomorphisms Two graphs G, H are *isomorphic* (denoted $H \simeq G$), if there exists an adjacency-preserving bijective mapping (*isomorphism*) $f : \mathcal{V}_G :\to \mathcal{V}_H$, i.e., $(v,u) \in \mathcal{E}_G$ iff $(f(v), f(u)) \in \mathcal{E}_H$. Given some small graph H, the *subgraph isomorphism* problem amounts to finding a subgraph G_S of G such that $G_S \simeq H$. An *automorphism* of H is an isomorphism that maps H onto itself. The set of all the unique automorphisms form the *automorphism group* of the graph, denoted as $\mathrm{Aut}(H)$ containing all the possible symmetries of the graph. The automorphism group yields a partition of the vertices into disjoint subsets of $\mathcal{V}_H$ called *orbits*. Intuitively, this concept allows us to group the vertices based on their *structural roles*, e.g. the end vertices of a path, or all the vertices of a cycle (see Figure 1). Formally, the orbit of a vertex $v \in \mathcal{V}_H$ is the set of vertices to which it can be mapped via an automorphism: $\mathrm{Orb}(v) = \{u \in \mathcal{V}_H : \exists g \in \mathrm{Aut}(H) \text{ s.t. } g(u) = v\}$, and the set of all orbits $H \setminus \mathrm{Aut}(H) = \{\mathrm{Orb}(v) : v \in \mathcal{V}_H\}$ is usually called the *quotient* of the automorphism when it acts on the graph H. We are interested in the unique elements of this set that we will denote as $\{O^V_{H,1}, O^V_{H,2}, \ldots, O^V_{H,d_H}\}$, where d_H is the cardinality of the quotient.

Analogously, we define edge structural roles via *edge automorphisms*, i.e. bijective mappings from the edge set onto itself, that preserve edge adjacency (two edges are adjacent if they share a common endpoint). In particular, every vertex automorphism g induces an edge automorphism by mapping each edge $\{u,v\}$ to $\{g(u), g(v)\}$.[1] In the same way as before we construct the edge automorphism group, from which we deduce the partition of the edge set in *edge orbits* $\{O^E_{H,1}, O^E_{H,2}, \ldots, O^E_{H,d_H}\}$.

3. Graph Substructure Networks

Complex networks consist of nodes (or edges) with repeated structural roles. Thus, it is natural for a neural network to treat them in a similar manner, akin to weight sharing between local patches in CNNs for images (LeCun et al., 1989) or positional encodings in language models for sequential data (Sukhbaatar et al., 2015; Gehring et al., 2017; Vaswani et al., 2017).

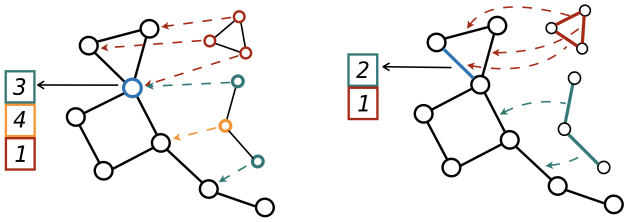

Figure 1: *Node* (left) and *edge* (right) subgraph counting for a 3-cycle and a 3-path, w.r.t the blue node and the blue edge, respectively. Different colors depict orbits.

However, contrary to Euclidean domains, e.g. regular 2D and 1D grids, the diversity in the topology of such networks prohibits modelling each structural role independently (that would require a large amount of training data). To address this, we suggest a simplification where nodes are described by *vertex invariants*, features that are invariant to isomorphism but also possibly shared between different roles.

Such invariants are naturally extracted by GNNs themselves, but it can be argued that they might be oversimplified, as nodes will be blind to the existence of e.g. triangles or larger cycles in their neighbourhoods (Chen et al., 2020; Arvind et al., 2019). Hence, we extend GNNs towards a stronger invariant that captures richer topological properties. We observe that the neighborhood of each node can be partially described by its substructures, thus by counting their appearances, one obtains an approximate characterisation of the node's structural role. This information can be subsequently passed to a GNN and act in a complementary manner.

Structural features Let $\mathcal{H} = \{H_1, H_2 \ldots H_K\}$ be a set of small (connected) graphs, for example cycles of fixed length or cliques. For each graph $H \in \mathcal{H}$, we first find its isomorphic subgraphs in G; let f be a subgraph isomorphism between H and G_S. For each node $v \in \mathcal{V}_{G_S}$, we infer its role w.r.t H by obtaining the orbit of its mapping $f(v)$ in H, $\mathrm{Orb}_H\big(f(v)\big)$. By counting all the possible appearances of $\mathrm{Orb}_H\big(f(v)\big)$ in v, we obtain the *structural feature* $\mathbf{x}^V_H(v)$ of v, defined as follows:

$$x^V_{H,i}(v) = \frac{|\{G_S \simeq H : v \in \mathcal{V}_{G_S} \text{ s.t. } f(v) \in O^V_{H,i}\}|}{|\mathrm{Aut}(H)|}, \qquad (1)$$

where $i = 1, \ldots, d_H$. We divide the counts by the number of the automorphisms of H, since for every matched subgraph G_S there will always be $|\mathrm{Aut}(H)|$ different ways to map it to H, thus these repetitions will be uninformative. By combining the counts from different substructures in $\mathcal{H}$ and different orbits, we obtain the feature vector

[1] Note that the edge automorphism group is larger than that of induced automorphisms, but strictly larger only for 3 trivial cases (Whitney, 1932). However, induced automorphisms provide a more natural way to express structural roles.

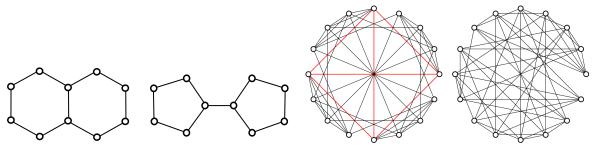

Figure 2: (Left) *Decalin* and *Bicyclopentyl*: non-isomorphic molecular graphs than can be distinguished by GSN, but not by 1-WL (Sato, 2020). (Right) *Rook's 4x4 graph* and the *Shrikhande graph*: the smallest pair of SR non-isomorphic graphs with the same parameters SR(16,6,2,2). GSN can distinguish them with 4-clique counts, while 2-FWL fails.

$$\mathbf{x}_v^V = [\mathbf{x}_{H_1}^V(v), \dots, \mathbf{x}_{H_K}^V(v)] \in \mathbb{N}^{D\times 1}\,,\, D = \textstyle\sum_{H\in\mathcal{H}} d_{H_i}.$$

Similarly, we define *edge structural features* $\mathbf{x}_{H,i}^E(\{u,v\})$ by counting occurrences of edge automorphism orbits:

$$x_{H,i}^E(\{u,v\}) = \frac{|\,\{G_S \simeq H : \{u,v\}\in\mathcal{E}_{G_S} \text{ s.t. } \{f(u),f(v)\}\in O_{H,i}^E\}|}{|\mathrm{Aut}(H)|}, \tag{2}$$

and the combined edge features $\mathbf{x}_{\{u,v\}}^E = [\mathbf{x}_{H_1}^E(\{u,v\}), \dots, \mathbf{x}_{H_K}^E(\{u,v\})]$. An example of counting features is illustrated in Figure 1.

Structure-aware message passing The key building block of our architecture is the graph substructure layer, defined in a general manner as a Message Passing Neural Network (MPNN) (Gilmer et al., 2017), where now the messages from the neighbouring nodes also contain the structural information. In particular, each node v updates its state $\mathbf{h}_v^t$ by combining its previous state with the aggregated messages: $\mathbf{h}_v^{t+1} = \mathrm{UP}^{t+1}(\mathbf{h}_v^t, \mathrm{MSG}^{t+1}(v))$, where the UP^{t+1} function is a neural network (MLP) and the message aggregation is a summation of features transformed by an MLP M^{t+1} as follows:

$$\mathrm{MSG}^{t+1}(v) = \sum_{u\in\mathcal{N}(v)} M^{t+1}\big(\mathbf{h}_v^t, \mathbf{h}_u^t, \mathbf{x}_v^V, \mathbf{x}_u^V, \mathbf{e}_{\{u,v\}}\big) \tag{3}$$

$$\mathrm{MSG}^{t+1}(v) = \sum_{u\in\mathcal{N}(v)} M^{t+1}\big(\mathbf{h}_v^t, \mathbf{h}_u^t, \mathbf{x}_{\{u,v\}}^E, \mathbf{e}_{\{u,v\}}\big), \tag{4}$$

where the two variants, named GSN-v and GSN-e, correspond to vertex- or edge-counts, respectively, and $\mathbf{e}_{\{u,v\}}$ denotes edge features.

How powerful are GSNs? We now turn to the expressive power of GSNs in comparison to the classical WL graph isomorphism tests (description of the WL hierarchy and proofs are provided in the supplementary material).

Proposition 3.1. *GSNs are at least as powerful as MPNNs and the 1-WL test.*

Furthermore, we can state that GSNs have the capacity to learn functions that traditional MPNNs cannot learn. The following observation derives directly from the analysis of the counting abilities of the 1-WL test (Arvind et al., 2019) and its extension to MPNNs (Chen et al., 2020).

Proposition 3.2. *GSNs are strictly more powerful than MPNNs and the 1-WL test when H is any induced subgraph except from single edges and single nodes, or any not necessarily induced subgraph except from star graphs of any size.*

Although our method does not attempt to align with the WL hierarchy, we observe that it has the capacity to distinguish graphs where the 2-Folklore Weisfeiler-Lehman (2-FWL) (Maron et al., 2019a) fails, which can be stated as:

Proposition 3.3. *2-FWL is not stronger than GSN.*

It is sufficient to find an example of two non-isomorphic graphs that are distinguishable by GSN but not 2-FWL, for which purpose we consider the *Strongly Regular (SR)* graph family: A *SR(n,d,λ,μ)-graph* is a regular graph with n nodes and degree d, where every two adjacent vertices have always λ mutual neighbours, while every two non-adjacent vertices have always μ mutual neighbours. The graphs in Figure 2 (right) are examples of non-isomorphic strongly regular graphs, on which 2-FWL (and thus 1-WL) is known to fail (Arvind et al., 2019). On the other hand, the examples of Figure 2 can be distinguished by a GSN by e.g. counting 4-cliques: there is at least one in Rook's 4x4 graph contrary to the Shrikhande graph that has none.

In Section 4, we empirically show that small-sized substructures are usually adequate to tell these graphs apart. Although it is not clear if there exists a certain substructure collection that results in GSNs that align with the WL hierarchy, we stress that this is not a necessary condition in order to design more powerful GNNs. In particular, the advantages offered by k-WL might not be able to outweigh the disadvantage of the larger complexity introduced. For example, a 2-FWL equivalent GNN will still fail to count 4-cliques or 8-cycles (Fürer, 2017; Arvind et al., 2019).

How large should the substructures be? As of today, we are not aware of any results in graph theory that can guarantee the reconstruction of a graph from a smaller collection of its subgraphs. In fact, the Reconstruction Conjecture (Kelly et al., 1957; Ulam, 1960), states that a graph with size $n \geq 3$ can be reconstructed from its vertex-deleted subgraphs, which in our case amounts to using all the substructures of size $k = n-1$. Therefore, if the Reconstruction Conjecture holds, GSN can distinguish all non-isomorphic graphs when using substructures of size $k = n-1$. However, the Reconstruction Conjecture has only been proven for $n \leq 11$ (McKay, 1997) and still remains open for larger graphs, while to the best of our knowledge, there is no similar hypothesis for smaller values of k. We hypothesise that for ML tasks, small structures of size $k = \mathcal{O}(1)$ are practically sufficient. This is validated by the experiments where strong empirical performance is observed for small and relatively frequent subgraph structures.

Table 1: Graph classification accuracy on various social and biological networks from the TUD Datasets collection. The top three performance scores are highlighted as: **First**, **Second**, **Third**. For GSN, we show the best performing substructure collection. * denotes Graph Kernel methods.

Dataset	MUTAG	PTC	Proteins	NCI1	Collab	IMDB-B	IMDB-M
GK* (k=3) (Shervashidze et al., 2009)	81.4±1.7	55.7±0.5	71.4±0.31	62.5±0.3	N/A	N/A	N/A
WL kernel* (Shervashidze et al., 2011)	**90.4±5.7**	59.9±4.3	75.0±3.1	**86.0±1.8**	78.9±1.9	73.8±3.9	50.9±3.8
GNTK* (Du et al., 2019a)	90.0±8.5	**67.9±6.9**	75.6±4.2	**84.2±1.5**	**83.6±1.0**	**76.9±3.6**	**52.8±4.6**
DGCNN (Zhang et al., 2018)	85.8±1.8	58.6±2.5	75.5±0.9	74.4±0.5	73.8±0.5	70.0±0.9	47.8±0.9
IGN (Maron et al., 2019b)	83.9±13.0	58.5±6.9	**76.6±5.5**	74.3±2.7	78.3±2.5	72.0±5.5	48.7±3.4
GIN (Xu et al., 2019)	89.4±5.6	64.6±7.0	**76.2±2.8**	82.7±1.7	80.2±1.9	75.1±5.1	52.3±2.8
PPGNs (Maron et al., 2019a)	**90.6±8.7**	66.2±6.6	**77.2±4.7**	83.2±1.1	81.4±1.4	73.0±5.8	50.5±3.6
GSN-e	**90.6±7.5** 6 (cycles)	**68.2±7.2** 6 (cycles)	**76.6±5.0** 4 (cliques)	**83.5± 2.3** 15 (cycles)	**85.5±1.2** 3 (triangles)	**77.8±3.3** 5 (cliques)	**54.3±3.3** 5 (cliques)
GSN-v	**92.2±7.5** 12 (cycles)	**67.4±5.7** 10 (cycles)	74.6±5.0 4 (cliques)	**83.5±2.0** 3 (triangles)	**82.7±1.5** 3 (triangles)	**76.8±2.0** 4 (cliques)	**52.6±3.6** 3 (triangles)

4. Experimental Evaluation

We evaluate GSN on synthetic and real-world datasets. Depending on the dataset domain we experiment with different substructure families (*cycles*, *paths* and *cliques*) and maximum substructure size (k). Please refer to the supplementary materials for additional details on the experimental setup.

Strongly Regular graphs We tested the ability of GSNs to decide if two graphs are non-isomorphic. We use a collection of Strongly Regular graphs[2] with up to 35 nodes and attempt to disambiguate pairs with same number of nodes (for different sizes the problem becomes trivial). At this stage we are only interested in the bias of the architecture itself, we thus use GSN with random weights to compute graph representations. Two graphs are deemed isomorphic if the Euclidean distance of their representations is smaller than a predefined threshold ϵ. Figure 3 shows the failure percentage when using different induced subgraphs (*cycles*, *paths*, and *cliques*) of varying size k. While 1-WL and 2-FWL equivalent models mark 100% failure (as expected from theory), the number of failure cases of GSN decreases rapidly as we increase k; cycles and paths of length 6 are enough to tell apart all the graphs in the dataset. Note that the performance of cliques saturates, possibly because the largest clique in our dataset has 5 nodes. We hypothesise GSN-e outperforms GSN-v because edge counts allow GSN to create a finer partition of the nodes in the graph.

TUD Datasets We use seven datasets from the TUD benchmark[3] from the domains of bioinformatics and social science. Table 1 lists all the methods evaluated with the split of (Zhang et al., 2018). We compare against various GNNs and Graph Kernels and our main GNN baselines are GIN (Xu et al., 2019) and PPGN (Maron et al., 2019a). We follow the same evaluation protocol of (Xu et al., 2019)

[2] http://users.cecs.anu.edu.au/~bdm/data/graphs.html

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

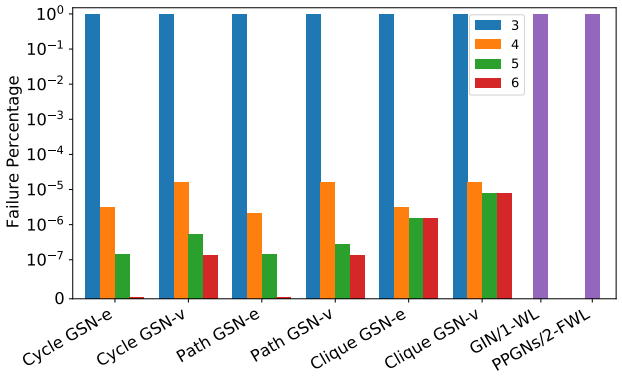

Figure 3: SR graphs isomorphism test. Different colours indicate different substructure sizes.

and perform model selection by tuning architecture and optimisation hyperparameters, and substructure related parameters: 1) k, 2) *motifs* (not necessarily induced subgraphs) against *graphlets* (induced subgraphs), following domain evidence: cycles for molecules, cliques for social networks. Our model obtains state-of-the art performance and outperforms all its GNN counterparts in the vast majority of the datasets, demonstrating strong generalisation capabilities in addition to its theoretically proven expressive power.

5. Conclusion

In this paper, we propose a novel way to design structure-aware graph neural networks. Motivated by the limitations of traditional GNNs to grasp important topological properties of the graph, we formulate a message passing scheme enhanced with structural features that are extracted by counting the appearances of substructures. We show both theoretically and empirically that our construction leads to improved expressive power and attains state-of-the-art performance in real-world scenarios.

References

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *International Conference on Machine Learning, ICML*, volume 97, pp. 242–252. PMLR, 2019.

Angluin, D. Local and global properties in networks of processors. In *ACM Symposium on Theory of Computing (STOC)*, pp. 82–93, 1980.

Arvind, V., Fuhlbrück, F., Köbler, J., and Verbitsky, O. On weisfeiler-leman invariance: Subgraph counts and related graph properties. In *Fundamentals of Computation Theory (FCT)*, 2019.

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1993–2001, 2016.

Babai, L., Erdos, P., and Selkow, S. M. Random graph isomorphism. *SIAM Journal on computing*, 9(3):628–635, 1980.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing systems (NIPS)*, pp. 4502–4510, 2016.

Benson, A. R., Gleich, D. F., and Leskovec, J. Higher-order organization of complex networks. *Science*, 353(6295): 163–166, 2016.

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Bouritsas, G., Bokhnyak, S., Ploumpis, S., Bronstein, M., and Zafeiriou, S. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 7213–7222, 2019.

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

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 15868–15876, 2019.

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

Costa, F. and De Grave, K. Fast neighborhood subgraph pairwise distance kernel. In *International Conference on Machine Learning (ICML)*, pp. 255–262, 2010.

Dareddy, M. R., Das, M., and Yang, H. motif2vec: Motif aware node representation learning for heterogeneous networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1052–1059, 2019.

Dasoulas, G., Santos, L. D., Scaman, K., and Virmaux, A. Coloring graph neural networks for node disambiguation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

de Haan, P., Weiler, M., Cohen, T., and Welling, M. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020.

Dell, H., Grohe, M., and Rattan, G. Lovász meets weisfeiler and leman. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 40:1–40:14, 2018.

Deshpande, M., Kuramochi, M., Wale, N., and Karypis, G. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1036–1050, 2005.

Du, S. S., Hou, K., Salakhutdinov, R., Póczos, B., Wang, R., and Xu, K. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5724–5734, 2019a.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *International Conference on Machine Learning, ICML*, volume 97, pp. 1675–1685. PMLR, 2019b.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2224–2232, 2015.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Flam-Shepherd, D., Wu, T., Friederich, P., and Aspuru-Guzik, A. Neural message passing on high order paths. *Advances in Neural Information Processing Systems Workshops (NeurIPSW)*, 2020.

Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. Protein interface prediction using graph convolutional networks.

In *Advances in Neural Information Processing Systems (NIPS)*, pp. 6530–6539, 2017.

Fürer, M. On the power of combinatorial and spectral invariants. *Linear algebra and its applications*, 2010.

Fürer, M. On the combinatorial power of the weisfeiler-lehman algorithm. In *International Conference on Algorithms and Complexity (CIAC)*, pp. 260–271, 2017.

Gärtner, T., Flach, P. A., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In Schölkopf, B. and Warmuth, M. K. (eds.), *Computational Learning Theory and Kernel Machines (COLT)*, volume 2777, pp. 129–143. Springer, 2003.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *International Conference on Machine Learning, (ICML)*, volume 70, pp. 1243–1252, 2017.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pp. 1263–1272, 2017.

Giscard, P., Kriege, N. M., and Wilson, R. C. A general purpose algorithm for counting simple cycles and simple paths of any length. *Algorithmica*, 81(7):2716–2737, 2019.

Holland, P. W. and Leinhardt, S. Local structure in social networks. *Sociological methodology*, 7:1–45, 1976.

Horváth, T., Gärtner, T., and Wrobel, S. Cyclic pattern kernels for predictive graph mining. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 158–167, 2004.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 15794–15805, 2019.

Ivanov, S. and Burnaev, E. Anonymous walk embeddings. In Dy, J. G. and Krause, A. (eds.), *International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2191–2200. PMLR, 2018.

Jacot, A., Hongler, C., and Gabriel, F. Neural tangent kernel: Convergence and generalization in neural networks. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8580–8589, 2018.

Kelly, P. J. et al. A congruence theorem for trees. *Pacific Journal of Mathematics*, 7(1):961–968, 1957.

Keriven, N. and Peyré, G. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7090–7099, 2019.

Kim, C., Moon, H., and Hwang, H. J. Near: Neighborhood edge aggregator for graph classification. *arXiv preprint arXiv:1909.02746*, 2019.

Kipf, T. N., Fetaya, E., Wang, K., Welling, M., and Zemel, R. S. Neural relational inference for interacting systems. In *International Conference on Machine Learning, (ICML)*, pp. 2693–2702, 2018.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

Kondor, R., Son, H. T., Pan, H., Anderson, B. M., and Trivedi, S. Covariant compositional networks for learning graphs. In *International Conference on Learning Representations Workshops ICLRW*. OpenReview.net, 2018.

Kramer, S., De Raedt, L., and Helma, C. Molecular feature mining in hiv data. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 136–143, 2001.

Kriege, N. M. and Mutzel, P. Subgraph matching kernels for attributed graphs. In *International Conference on Machine Learning, (ICML)*, 2012.

Kuramochi, M. and Karypis, G. Frequent subgraph discovery. In *IEEE International Conference on Data Mining (ICDM)*, pp. 313–320, 2001.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Lee, J. B., Rossi, R., Kong, X., Kim, S., Koh, E., and Rao, A. Graph convolutional networks with motif-based attention. In *28th ACM International Conference on Information and Knowledge Management (CIKM)*, 2019.

Li, M. L., Dong, M., Zhou, J., and Rush, A. M. A hierarchy of graph neural networks based on learnable local features. *arXiv preprint arXiv:1911.05256*, 2019.

Linial, N. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

Liu, X., Pan, H., He, M., Song, Y., and Jiang, X. Neural subgraph isomorphism counting. *arXiv preprint arXiv:1912.11589*, 2019.

Liu, Z., Nalluri, S. K. M., and Stoddart, J. F. Surveying macrocyclic chemistry: from flexible crown ethers to rigid cyclophanes. *Chemical Society Reviews*, 46(9):2459–2478, 2017.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations (ICLR)*, 2020.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019a.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *International Conference on Learning Representations, (ICLR)*, 2019b.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In *International Conference on Machine Learning (ICML)*, 2019c.

Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P. Geodesic convolutional neural networks on riemannian manifolds. In *IEEE International Conference on Computer Vision Workshops, (ICCVW)*, pp. 832–840.

McKay, B. D. Small graphs are reconstructible. *Australasian J. Combinatorics*, 15:123–126, 1997.

Milenković, T. and Pržulj, N. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6:CIN–S680, 2008.

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

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5425–5434, 2017.

Monti, F., Otness, K., and Bronstein, M. M. Motifnet: A motif-based graph convolutional network for directed graphs. In *2018 IEEE Data Science Workshop (DSW)*, pp. 225–228, 2018.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 4602–4609, 2019.

Murphy, R. L., Srinivasan, B., Rao, V. A., and Ribeiro, B. Relational pooling for graph representations. In *International Conference on Machine Learning (ICML)*, volume 97, pp. 4663–4673, 2019.

Naor, M. and Stockmeyer, L. J. What can be computed locally? In Kosaraju, S. R., Johnson, D. S., and Aggarwal, A. (eds.), *ACM Symposium on Theory of Computing (STOC)*, pp. 184–193.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In Balcan, M. and Weinberger, K. Q. (eds.), *International Conference on Machine Learning, ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2014–2023. JMLR.org, 2016.

Paranjape, A., Benson, A. R., and Leskovec, J. Motifs in temporal networks. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 601–610, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

Pržulj, N. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

Pržulj, N., Corneil, D. G., and Jurisica, I. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18): 3508–3515, 2004.

Rossi, R. A., Ahmed, N. K., Koh, E., Kim, S., Rao, A., and Abbasi-Yadkori, Y. Hone: Higher-order network embeddings. In *arXiv:1801.09303*, 2018.

Sankar, A., Zhang, X., and Chang, K. C.-C. Meta-gnn: metagraph neural network for semi-supervised learning in attributed heterogeneous information networks. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 137–144, 2019.

Sarajlić, A., Malod-Dognin, N., Yaveroğlu, Ö. N., and Pržulj, N. Graphlet-based characterization of directed networks. *Scientific reports*, 6:35098, 2016.

Sato, R. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

Sato, R., Yamada, M., and Kashima, H. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4083–4092, 2019.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research (JMLR)*, 2011.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 29–38. IEEE Computer Society, 2017.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2440–2448, 2015.

Ulam, S. M. *A collection of mathematical problems*, volume 8. Interscience Publishers, 1960.

Valiant, L. G. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

Verma, S. and Zhang, Z. Hunt for the unique, stable, sparse and fast feature learning on graphs. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems (NIPS)*, pp. 88–98, 2017.

Whitney, H. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning, (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5449–5458. PMLR, 2018.

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

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G. (eds.), *ACM International Conference on Knowledge Discovery and Data Mining SIGKDD*, pp. 1365–1374. ACM, 2015.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, 2018a.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems NeurIPS*, pp. 4805–4815, 2018b.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*, 2018.

Appendix

A. Background

Weisfeiler-Lehman tests The *Weisfeiler-Lehman (WL) graph-isomorphism test*, or naive vertex refinement (we will refer to it as *1-WL* or just *WL*), is a fast heuristic used to decide if two graphs are isomorphic. The WL-test proceeds as follows: every vertex v is initially assigned a colour $c^0(v)$ that is later iteratively refined by aggregating neighbouring information:

$$c^{t+1}(v) = \text{HASH}\Big(c^t(v),\ \{\!\{c^t(u)\}\!\}_{u\in\mathcal{N}(v)}\Big), \tag{5}$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset (a set that allows element repetitions) and $\mathcal{N}(v)$ is the neigborhood of v. Note that the neighbour aggregation in WL-test is a form of message passing, and GNNs are the learnable analogue.

Most of the research in improving GNN expressivity has focused on models that mimic the generalisations of WL, known as the WL hierarchy. Briefly, here we describe the so-called *Folklore WL* family (k-FWL), as referred to by Maron et al. (Maron et al., 2019a).[4] The k-FWL operates on k-tuples of nodes $\mathbf{v} = (v_1, v_2, \dots, v_k)$ to which an initial colour $c^0(\mathbf{v})$ is assigned based on their *isomorphism types*, which can loosely be thought of as a generalisation of isomorphism that also preserves the ordering of the nodes in the tuple (see section B.3). Then, at each iteration the colour is refined as follows:

$$c^{t+1}(\mathbf{v}) = \text{HASH}\Big(c^t(\mathbf{v}),\ \{\!\{\big(c^t(\mathbf{v}_{u,1}), c^t(\mathbf{v}_{u,2}), \dots, c^t(\mathbf{v}_{u,k})\big)\}\!\}_{u\in\mathcal{V}}\Big) \tag{6}$$

where $\mathbf{v}_{u,j} = (v_1, v_2, \dots, v_{j-1}, u, v_{j+1} \dots, v_k)$. The multiset $\{\!\{\big(c^t(\mathbf{v}_{u,1}), c^t(\mathbf{v}_{u,2}), \dots, c^t(\mathbf{v}_{u,k})\big)\}\!\}_{u\in\mathcal{V}}$ can be perceived as a form of generalised neighborhood. Note here, that information is saved in all possible tuples in the graph, thus each k-tuple receives information *from the entire graph*, contrary to the *local* nature of the 1-WL test.

B. Proofs

B.1. Proof of Proposition 3.1

Proof. The first part of the proof is trivial since the GSN model class contains MPNNs and is thus at least as expressive. For the 1-WL test, one can repurpose the proof of Theorem 3 in (Xu et al., 2019) and demand the injectivity of the update function (w.r.t. both the hidden state $\mathbf{h}_v^t$ and the message aggregation $\text{MSG}^{t+1}(v)$), and the injectivity of the message aggregation w.r.t. the multiset of the hidden states of the neighbours $\{\!\{\mathbf{h}_u^t\}\!\}_{u\in\mathcal{N}(v)}$. It suffices then to show that if injectivity is preserved then GSNs are at least as powerful as the 1-WL.

In more detail, we will show that GSN is at least as powerful as 1-WL for node-labelled graphs, since traditionally the 1-WL test does not take into account edge labels.[5] We can rephrase the statement as follows: *If GSN deems two graphs G_1, G_2 as isomorphic, then also 1-WL deems them isomorphic*. Given that the graph-level representation is extracted by a readout function that receives the multiset of the node colours in its input (i.e. the graph-level representation is the node colour histogram at some iteration t), then it suffices to show that if for the two graphs the multiset of the node colours that GSN infers is the same, then also 1-WL will infer the same multiset for the two graphs.

Consider the case where the two multisets that GSN extracts are the same: i.e. $\{\!\{\mathbf{h}_v^t\}\!\}_{v\in\mathcal{V}_{G_1}} = \{\!\{\mathbf{h}_u^t\}\!\}_{u\in\mathcal{V}_{G_2}}$. Then both multisets contain the same distinct colour/hidden representations with the exact same multiplicity. Thus, it further suffices to show that if two nodes v, u (that may belong to the same or to different graphs) have the same GSN hidden representations $\mathbf{h}_v^t = \mathbf{h}_u^t$ at any iteration t, then they will also have the same colours $c^t(v) = c^t(u)$, extracted by 1-WL. Intuitively, this means that GSN creates a partition of the nodes of each graph that is at least as fine-grained as the one created by 1-WL. We prove by induction (similarly to (Xu et al., 2019)) that GSN model class contains a model where this holds (w.l.o.g. we show that for GSN-v; same proof applies to GSN-e).

For $t = 0$ the statement holds since the initial node features are the same for both GSN and 1-WL, i.e. $\mathbf{h}_v^0 = c^0(v), \quad \forall v \in$

[4] In the majority of papers on GNN expressivity (Morris et al., 2019; Maron et al., 2019a; Chen et al., 2020), another family of WL tests is discussed, under the terminology k-*WL* with expressive power equal to $(k-1)$-FWL. In contrast, in most graph theory papers on graph isomorphism (Cai et al., 1992; Fürer, 2017; Arvind et al., 2019) the k-WL term is used to describe the algorithms referred to as k-FWL in GNN papers. Here, we follow the k-FWL convention to align with the work mostly related to ours.

[5] if one considers a simple 1-WL extension that concatenates edge labels to neighbour colours, then the same proof applies.

$\mathcal{V}_{G_1} \cup \mathcal{V}_{G_2}$. Suppose the statement holds for $t-1$, i.e. $\mathbf{h}_v^{t-1} = \mathbf{h}_u^{t-1} \Rightarrow c^{t-1}(v) = c^{t-1}(u)$. Then we show that it also holds for t.

Every node hidden representation at step t is updated as follows: $\mathbf{h}_v^t = \mathrm{UP}^t\big(\mathbf{h}_v^{t-1}, MSG^t(v)\big)$. Assuming that the update function UP^t is injective, we have the following: if $\mathbf{h}_v^t = \mathbf{h}_u^t$, then:

1. $\mathbf{h}_v^{t-1} = \mathbf{h}_u^{t-1}$, which from the induction hypothesis implies that $c^{t-1}(v) = c^{t-1}(u)$.

2. $MSG^t(v) = MSG^t(u)$, where the message function is defined as in Eq. 3 of the main paper: $MSG^t(v) = \sum_{w \in \mathcal{N}(v)} M^t\big(\mathbf{h}_v^{t-1}, \mathbf{h}_w^{t-1}, \mathbf{x}_v^V, \mathbf{x}_w^V\big)$. Additionally here we require MSG^t to be injective w.r.t. the multiset of the hidden representations of the neighbours. In fact, using Lemma 5 from (Xu et al., 2019) we know that there always exists a function M^t, such that $MSG^t(v)$ is unique for each multiset $\wr\big(\mathbf{h}_v^{t-1}, \mathbf{h}_w^{t-1}, \mathbf{x}_v^V, \mathbf{x}_w^V\big)\int_{w \in \mathcal{N}_v}$, assuming that the domain from where the elements of the multiset originate is countable. Thus,

$$
\begin{aligned}
MSG^t(v) = MSG^t(u) \Rightarrow \\
\wr\big(\mathbf{h}_v^{t-1}, \mathbf{h}_w^{t-1}, \mathbf{x}_v^V, \mathbf{x}_w^V\big)\int_{w \in \mathcal{N}_v} = \wr\big(\mathbf{h}_u^{t-1}, \mathbf{h}_z^{t-1}, \mathbf{x}_u^V, \mathbf{x}_z^V\big)\int_{z \in \mathcal{N}_u} \Rightarrow \\
\wr\mathbf{h}_w^{t-1}\int_{w \in \mathcal{N}_v} = \wr\mathbf{h}_z^{t-1}\int_{z \in \mathcal{N}_u}
\end{aligned}
$$

From the induction hypothesis we know that $\mathbf{h}_w^{t-1} = \mathbf{h}_z^{t-1}$ implies that $c^{t-1}(w) = c^{t-1}(z)$ for any $w \in \mathcal{N}_v, z \in \mathcal{N}_u$, thus $\wr c^{t-1}(w)\int_{w \in \mathcal{N}_v} = \wr c^{t-1}(z)\int_{z \in \mathcal{N}_u}$.

Concluding, given the update rule of 1-WL: $c^t(v) = \mathrm{HASH}\Big(c^{t-1}(v),\ \wr c^{t-1}(w)\int_{w \in \mathcal{N}_v}\Big)$, it holds that $c^t(v) = c^t(u)$.

□

B.2. Proof of Proposition 3.2

Proof. Arvind et al. (Arvind et al., 2019) showed that 1-WL, and consequently MPNNs, can count only *forests of stars*. Thus, if the subgraphs are required to be connected, then they can only be star graphs of any size (note that this contains single nodes and single edges). In addition, Chen et al. (Chen et al., 2020), showed that 1-WL, and consequently MPNNs, cannot count any connected induced subgraph with 3 or more nodes, i.e. any connected subgraph apart from single nodes and single edges.

Given proposition 3.1, in order to show that GSNs are strictly more expressive than MPNNs and the 1-WL test, it suffices to show that GSN can distinguish a pair of graphs that MPNNs and the 1-WL test deem isomorphic. If H is a substructure that MPNNs cannot learn to count, i.e. the ones mentioned above, then there is at least one pair of graphs with different number of counts of H, that MPNNs deem isomorphic. Thus, by assigning counting features to the nodes/edges of the two graphs based on appearances of H, a GSN can obtain different representations for G_1 and G_2 by summing up the features. Hence, G_1, G_2 are deemed non-isomorphic. An example is depicted in Figure 2 (left) in the main paper, where the two non-isomorphic graphs are distinguishable by GSN via e.g. cycle counting, but not by 1-WL.

□

B.3. Proposition 3.3: Why does 2-FWL fail on strongly regular graphs?

Proof. Below we provide a proof for this known statement in order to give further intuition in the limitations of the 2-FWL. We first rigorously describe what an isomorphism type is. Two k-tuples $\mathbf{v}^a = \{v_1^a, v_2^a, \ldots, v_k^a\}$, $\mathbf{v}^b = \{v_1^b, v_2^b, \ldots, v_k^b\}$ will have the same isomorphism type iff:

- $\forall\, i, j \in \{0, 1, \ldots, k\}, \quad v_i^a = v_j^a \Leftrightarrow v_i^b = v_j^b$
- $\forall\, i, j \in \{0, 1, \ldots, k\}, \quad v_i^a \sim v_j^a \Leftrightarrow v_i^b \sim v_j^b$, where $\sim$ means that the vertices are adjacent.

Note that this is a stronger condition than isomorphism, since the mapping between the vertices of the two tuples needs to preserve order. In case the graph is employed with edge and vertex features, they need to be preserved as well (see (Chen et al., 2020)) for the extended case).

For the 2-FWL test, when working with simple undirected graphs without self-loops, we have the following 2-tuple isomorphism types:

- $\mathbf{v} = \{v_1, v_1\}$: *vertex type*. Mapped to the color $c^0(\mathbf{v}) = c_\alpha$
- $\mathbf{v} = \{v_1, v_2\}$ and $v_1 \not\sim v_2$: *non-edge type*. Mapped to the color $c^0(\mathbf{v}) = c_\beta$
- $\mathbf{v} = \{v_1, v_2\}$ and $v_1 \sim v_2$: *edge type*. Mapped to the color $c^0(\mathbf{v}) = c_\gamma$

For each 2-tuple $\mathbf{v} = \{v_1, v_2\}$, a generalised "neighbour" is the following tuple: $(\mathbf{v}_{u,1}, \mathbf{v}_{u,2}) = \big((u, v_2), (v_1, u)\big)$, where u is an arbitrary vertex in the graph.

Now, let us consider a strongly regular graph SR(n,d,λ,μ). We have the following cases:

- generalised neighbour of a *vertex type* tuple: $(\mathbf{v}_{u,1}, \mathbf{v}_{u,2}) = \big((u, v_1), (v_1, u)\big)$. The corresponding neighbor colour tuples are:
  - (c_α, c_α) if $v_1 = u$,
  - (c_β, c_β) if $v_1 \not\sim u$,
  - (c_γ, c_γ) if $v_1 \sim u$.

  The update of the 2-FWL is: $c^1(\mathbf{v}) = \mathrm{HASH}\Big((c_\alpha, \{\!\{\underbrace{(c_\alpha, c_\alpha)}_{1 \text{ time}}, \underbrace{(c_\beta, c_\beta)}_{n-1-d \text{ times}}, \underbrace{(c_\gamma, c_\gamma)}_{d \text{ times}}\}\!\}\Big)$

  same for all *vertex type* 2-tuples.
- generalised neighbour of a *non-edge type* tuple: $(\mathbf{v}_{u,1}, \mathbf{v}_{u,2}) = \big((u, v_2), (v_1, u)\big)$. The corresponding neighbor colour tuples are:
  - (c_α, c_β) if $v_2 = u$,
  - (c_β, c_α) if $v_1 = u$,
  - (c_γ, c_β) if $v_2 \sim u$ and $v_1 \not\sim u$,
  - (c_β, c_γ) if $v_1 \sim u$ and $v_2 \not\sim u$,
  - (c_β, c_β) if $v_1 \not\sim u$ and $v_2 \not\sim u$,
  - (c_γ, c_γ) if $v_1 \sim u$ and $v_2 \sim u$.

  The update of the 2-FWL is:
  $c^1(\mathbf{v}) = \mathrm{HASH}\Big((c_\beta, \{\!\{\underbrace{(c_\alpha, c_\beta)}_{1 \text{ time}}, \underbrace{(c_\beta, c_\alpha)}_{1 \text{ time}}, \underbrace{(c_\gamma, c_\beta)}_{d-\mu \text{ times}}, \underbrace{(c_\beta, c_\gamma)}_{d-\mu \text{ times}}, \underbrace{(c_\beta, c_\beta)}_{n-2-(2d-\mu) \text{ times}}, \underbrace{(c_\gamma, c_\gamma)}_{\mu \text{ times}}\}\!\}\Big)$

  same for all *non-edge type* 2-tuples.
- generalised neighbour of an *edge type* tuple:
  - (c_α, c_γ) if $v_2 = u$,
  - (c_γ, c_α) if $v_1 = u$,
  - (c_γ, c_β) if $v_2 \sim u$ and $v_1 \not\sim u$,
  - (c_β, c_γ) if $v_1 \sim u$ and $v_2 \not\sim u$,
  - (c_β, c_β) if $v_1 \not\sim u$ and $v_2 \not\sim u$,
  - (c_γ, c_γ) if $v_1 \sim u$ and $v_2 \sim u$.

  The update of the 2-FWL is:
  $c^1(\mathbf{v}) = \mathrm{HASH}\Big((c_\gamma, \{\!\{\underbrace{(c_\alpha, c_\gamma)}_{1 \text{ time}}, \underbrace{(c_\gamma, c_\alpha)}_{1 \text{ time}}, \underbrace{(c_\gamma, c_\beta)}_{d-\lambda \text{ times}}, \underbrace{(c_\beta, c_\gamma)}_{d-\lambda \text{ times}}, \underbrace{(c_\beta, c_\beta)}_{n-2-(2d-\lambda) \text{ times}}, \underbrace{(c_\gamma, c_\gamma)}_{\lambda \text{ times}}\}\!\}\Big)$

  same for all *edge type* 2-tuples.

From the analysis above, it is clear that all 2-tuples in the graph of the same initial type are assigned the same colour in the 1st iteration of 2-FWL. In other words, the vertices cannot be furthered partition, so the algorithm terminates. Therefore, if two SR graphs have the same parameters n,d,λ,μ then 2-FWL will yield the same colour distribution and thus the graphs will be deemed isomorphic. □

C. Complexity Analysis of GSNs

In the general case, subgraph isomorphism is a NP-complete problem, while its counting version is $\#P$-Complete (Valiant, 1979). However, for fixed k values, the setting we are interested in, the problem can be solved in $\mathcal{O}(n^k)$ by examining all the possible k-tuples in the graph. For specific types of subgraphs, such as paths and cycles, the problem can be solved even faster (see e.g. (Giscard et al., 2019)). Moreover, the computationally expensive part of the algorithm is done as a pre-processing step and thus does not affect network training and inference that remain linear w.r.t the number of edges, $\mathcal{O}(|\mathcal{E}|)$. This is opposed to k-WL equivalents (Maron et al., 2019a; Morris et al., 2019) with $\mathcal{O}(n^k)$ training complexity and relational pooling (Murphy et al., 2019) with $\mathcal{O}(n!)$ training complexity in absence of approximations.

D. Experiments

In the following appendix we give the implementation details of the experimental section. All experiments were performed on a server equipped with 8 Tesla V100 16 gb GPUs, except for the Collab dataset where a Tesla V100 32 gb GPU was used due to larger memory requirements (a large percentage of Collab graphs are dense or even nearly complete in some cases). Experimental tracking and hyper-parameter optimisation were done via the Weights & Biases platform (wandb) (Biewald, 2020). Our implementation is based on native PyTorch sparse operations (Paszke et al., 2019) in order to ensure complete reproducibility of the results. PyTorch Geometric (Fey & Lenssen, 2019) was used for additional operations (such as preprocessing and data loading).

Throughout the experimental evaluation the structural identifiers $\mathbf{x}_u^V$ and $\mathbf{x}_{\{u,v\}}^E$ are one-hot encoded, by taking into account the unique count values present in the dataset. Other more sophisticated methods can be used, e.g. transformation to continuous features via some normalisation scheme, binning etc. We found that the unique values in our datasets were usually of small or of average cardinality, which is a good indication of recurrent structural roles, thus such methods were not necessary.

D.1. Synthetic Experiment

For the Strongly Regular graphs dataset we use all the available families of graphs with size of at most 35 nodes. These are the following:

- SR(16,6,2,2): 2 graphs
- SR(25,12,5,6): 15 graphs
- SR(26,10,3,4): 10 graphs
- SR(28,12,6,4): 4 graphs
- SR(29,14,6,7): 41 graphs
- SR(35,16,6,8: 3854 graphs
- SR(35,18,9,9): 227 graphs

The total number of non-isomorphic pairs of the same size is $\approx 7 * 10^7$. We used a simple 2-layer architecture with width 64. The message aggregation was performed as in the general formulation of Eq. 3 and 4 of the main paper, where the update and the message functions are MLPs. The prediction is inferred by applying a sum readout function in in the last layer (i.e. a graph-level representation is obtained by summing the node-level representations) and then passing the output through a MLP.

Regarding the substructures, we use *graphlet* counting, as certain *motifs* (e.g. cycles of length up to 7) are known to be unable to distinguish strongly regular graphs (since they can be counted by the 2-FWL (Fürer, 2017; Arvind et al., 2019)).

D.2. Graph Classification Benchmarks (TUD Benchmarks)

For this family of experiments, due to the usually small size of the datasets, we choose a parameter effective architecture, in order to reduce the risk of overfitting. Note that we aim to show that structural identifiers can be used off-the-shelf and

Table 2: Graph Classification accuracy on various social and biological networks from the TUD Datasets collection https://chrsmrrs.github.io/datasets/. Graph Kernel methods are denoted with an *. For completeness we also include methods that were evaluated on potentially different splits. The top three performance scores are highlighted as: **First**, **Second**, **Third**.

	Dataset	MUTAG	PTC	PROTEINS	NCI1	COLLAB	IMDB-B	IMDB-M
	size	188	344	1113	4110	5000	1000	1500
	classes	2	2	2	2	3	2	3
	avg num. nodes	17.9	25.5	39.1	29.8	74.4	19.7	13
different splits	DGK* (best) (Yanardag & Vishwanathan, 2015)	87.4 ±2.7	60.1±2.6	75.7±0.5	80.3 ±0.5	73.1 ±0.3	67.0±0.6	44.6±0.5
	FSGD* (Verma & Zhang, 2017)	92.1±	62.8±	73.4±	79.8±	80.0±	73.6±	52.4±
	AWE-FB*(Ivanov & Burnaev, 2018)	87.9±9.8	N/A	N/A	N/A	71.0±1.5	73.1±3.3	51.6±4.7
	AWE-DD*(Ivanov & Burnaev, 2018)	N/A	N/A	N/A	N/A	73.9±1.9	74.5±5.8	51.5±3.6
	ECC (Simonovsky & Komodakis, 2017)	76.1±	N/A	N/A	76.8±	N/A	N/A	N/A
	PSCN k=10^E (Niepert et al., 2016)	92.6±4.2	60.0±4.8	75.9±2.8	78.6±1.9	72.6±2.2	71.0±2.3	45.2±2.8
	DiffPool (Ying et al., 2018b)	N/A	N/A	76.2±	N/A	75.5 ±	N/A	N/A
	CCN (Kondor et al., 2018)	91.6±7.2	70.6±7.0	N/A	76.3±4.1	N/A	NA	N/A
	1-2-3 GNN (Morris et al., 2019)	86.1±	60.9±	75.5±	76.2±	N/A	74.2±	49.5±
same splits	RWK* (Gärtner et al., 2003)	79.2±2.1	55.9±0.3	59.6±0.1	>3 days	N/A	N/A	N/A
	GK* (k=3) (Shervashidze et al., 2009)	81.4±1.7	55.7±0.5	71.4±0.31	62.5±0.3	N/A	N/A	N/A
	PK* (Neumann et al., 2016)	76.0±2.7	59.5±2.4	73.7±0.7	82.5±0.5	N/A	N/A	N/A
	WL kernel* (Shervashidze et al., 2011)	**90.4±5.7**	59.9±4.3	75.0±3.1	**86.0±1.8**	78.9±1.9	73.8±3.9	50.9±3.8
	GNTK* (Du et al., 2019a)	90.0±8.5	**67.9±6.9**	75.6±4.2	**84.2±1.5**	**83.6±1.0**	**76.9±3.6**	**52.8±4.6**
	DCNN (Atwood & Towsley, 2016)	N/A	N/A	61.3±1.6	56.6±1.0	52.1±0.7	49.1±1.4	33.5±1.4
	DGCNN (Zhang et al., 2018)	85.8±1.8	58.6±2.5	75.5±0.9	74.4±0.5	73.8±0.5	70.0±0.9	47.8±0.9
	IGN (Maron et al., 2019b)	83.9±13.	58.5±6.9	**76.6±5.5**	74.3±2.7	78.3±2.5	72.0±5.5	48.7±3.4
	GIN (Xu et al., 2019)	89.4±5.6	64.6±7.0	**76.2±2.8**	82.7±1.7	80.2±1.9	75.1±5.1	52.3±2.8
	PPGNs (Maron et al., 2019a)	**90.6±8.7**	66.2±6.6	**77.2±4.7**	83.2±1.1	81.4±1.4	73.0±5.8	50.5±3.6
	GSN-e (Ours)	**90.6±7.5**	**68.2±7.2**	**76.6±5.0**	**83.5**± 2.3	**85.5±1.2**	**77.8±3.3**	**54.3±3.3**
		6 (cycles)	6 (cycles)	4 (cliques)	15 (cycles)	3 (triangles)	5 (cliques)	5 (cliques)
	GSN-v (Ours)	**92.2±7.5**	**67.4±5.7**	74.6±5.0	**83.5±2.0**	**82.7±1.5**	**76.8±2.0**	**52.6±3.6**
		12 (cycles)	10 (cycles)	4 (cliques)	3 (triangles)	3 (triangles)	4 (cliques)	3 (cliques)

are independent of the architecture used. At the same time we aim to suppress the effect of other confounding factors in the model performance, thus we build our model on top of an existing baseline architecture. In particular, we follow the simple GIN architecture and we concatenate structural identifiers to node or edge features depending on the variant. Then the hidden representation is updated as follows:

$$\mathbf{h}_v^{t+1} = \mathrm{UP}^{t+1}\Big([\mathbf{h}_v^t; \mathbf{x}_v^V] + \sum_{u \in \mathcal{N}_v} [\mathbf{h}_u^t; \mathbf{x}_u^V]\Big), \tag{7}$$

for GSN-v, and

$$\mathbf{h}_v^{t+1} = \mathrm{UP}^{t+1}\Big([\mathbf{h}_v^t; \mathbf{x}_{\{v,v\}}^E] + \sum_{u \in \mathcal{N}_v} [\mathbf{h}_u^t; \mathbf{x}_{\{u,v\}}^E]\Big), \tag{8}$$

for GSN-e, where $\mathbf{x}_{\{v,v\}}^E$ is a dummy variable used to distinguish self-loops from edges. We didn't find training the ϵ parameter used in GIN to make a difference. Note that this architecture, is less expressive than our general formulation. However, we found it to work well in practice for the TUD datasets, possibly due to its simplicity and small number of parameters.

We implement an architecture similar to GIN (Xu et al., 2019), i.e. 4 message passing layers, jumping knowledge from all the layers (Xu et al., 2018) (including the input), transformation of each intermediate graph-level representation by a linear layer, sum readout for biological and mean readout for social networks. Node features are one-hot encodings of the categorical node labels. Similarly to the baseline, the hyperparameters search space is the following: batch size in {32, 128} (except for Collab where only 32 was searched due to GPU memory limits), dropout in {0,0.5}, network width in {16,32} for biological networks, 64 for social networks, learning rate in {0.01, 0.001}, decay rate in {0.5,0.9} and decay steps in {10,50} (number of epochs after which the learning rate is reduced by multiplying with the decay rate). For social networks, since they are not attributed graphs, we also experimented with using the degree as a node feature, but in most cases the structural identifiers were sufficient.

Model selection is done in two stages. First, we choose a substructure that we perceive as promising based on indications from the specific domain: *triangles* for social networks and Proteins, and *6-cycles (motifs)* for molecules. Under this setting we tune model hyperparameters for a GSN-e model. Then, we extend our search to the parameters related to the

Table 3: Chosen hyperparameters for each the two GSN variants for each dataset

	Dataset	MUTAG	PTC	PROTEINS	NCI1	COLLAB	IMDB-B	IMDB-M
GSN-e	batch size	32	128	32	32	32	32	32
	width	32	16	32	32	64	64	64
	decay rate	0.9	0.5	0.5	0.9	0.5	0.5	0.5
	decay steps	50	50	10	10	50	10	10
	dropout	0.5	0	0/5	0	0	0	0
	lr	10^{-3}	10^{-3}	10^{-2}	10^{-3}	10^{-2}	10^{-3}	10^{-3}
	degree	No	No	No	No	No	No	Yes
	substructure type	graphlets	motifs	same	graphlets	same	same	same
	substrucure family	cycles	cycles	cliques	cycles	clique	clique	cliques
	k	6	6	4	15	3	5	5
GSN-v	batch size	32	128	32	32	32	32	32
	width	32	16	32	32	64	64	64
	decay rate	0.9	0.5	0.5	0.9	0.5	0.5	0.5
	decay steps	50	50	10	10	50	10	10
	dropout	0.5	0	0.5	0	0	0	0
	lr	10^{-3}	10^{-3}	10^{-2}	10^{-3}	10^{-2}	10^{-3}	10^{-3}
	degree	No	No	No	No	No	Yes	Yes
	substructure type	graphlets	graphlets	same	same	same	same	same
	substrucure family	cycles	cycles	cliques	cycles	cliques	clique	cliques
	k	12	10	4	3	3	4	3

substructure collection: i.e. the maximum size k and motifs against graphlets. In all the molecular datasets we search cycles with $k = 3, \ldots, 12$, except for NCI1, where we also consider larger sizes due to the presence of large rings in the dataset (*macrocycles* (Liu et al., 2017)). For social networks, we searched cliques with $k = 3, 4, 5$. In Table 3 we report the hyperparameters chosen by our model selection procedure, including the best performing substructures.

The seven datasets[6] we chose are the intersection of the datasets used by the authors of our main baselines: the Graph Isomorphism Network (GIN) (Xu et al., 2019), a simple, yet powerful GNN with expressive power equal to the 1-WL test, and the Provably Powerful Graph Network (PPGN) (Maron et al., 2019a), a polynomial alternative to the Invariant Graph Network (Maron et al., 2019b), that increases its expressive power to match the 2-FWL. We also compare our results to other GNNs as well as Graph Kernel approaches. Our main baseline from the GK family is the Graph Neural Tangent Kernel (GNTK) (Du et al., 2019a), which is a kernel obtained from a GNN of infinite width. This operates in the Neural Tangent Kernel regime (Jacot et al., 2018; Allen-Zhu et al., 2019; Du et al., 2019b).

Table 2 is an extended version of Table 1 of the main paper, where the most prominent methods are reported, regardless of the splits they were evaluated on. For DGK (best variant) (Yanardag & Vishwanathan, 2015), FSGD (Verma & Zhang, 2017), AWE (Ivanov & Burnaev, 2018), ECC (Simonovsky & Komodakis, 2017), PSCN (Niepert et al., 2016), DiffPool (Ying et al., 2018b), CCN (Kondor et al., 2018) (slightly different setting since they perform a train, validation, test split), 1-2-3 GNN (Morris et al., 2019) and GNTK (Du et al., 2019a), we obtain the results from the original papers. For RWK (Gärtner et al., 2003), GK (Shervashidze et al., 2009), PK (Neumann et al., 2016), DCNN (Atwood & Towsley, 2016) and DGCNN (Zhang et al., 2018), we obtain the results from the DGCNN paper, where the authors reimplemented these methods and evaluated them with the same split. Similarly, we obtain the WLK (Shervashidze et al., 2011) and GIN (Xu et al., 2019) results from the GIN paper, and IGN (Maron et al., 2019b) and PPGN (Maron et al., 2019a) results from the PPGN paper.

E. Related Work

Expressive power of GNNs and the WL hierarchy The seminal results in the theoretical analysis of the expressivity of GNNs (Xu et al., 2019) and k-GNNs (Morris et al., 2019) established that traditional message passing based GNNs are at most as powerful as the 1-WL test. Chen et al. (Chen et al., 2019) showed that graph isomorphism is equivalent to universal invariant function approximation. Maron et al. (Maron et al., 2019b) studied Invariant Graph Networks (IGN) constructed to be invariant to symmetry groups, which were later shown that in order to be universal they must involve tensors of order no less than linear w.r.t the graph size n (Maron et al., 2019c; Keriven & Peyré, 2019). Similarly to (Morris et al., 2019), IGNs

[6] more details on the description of the datasets and the corresponding tasks can be found at (Xu et al., 2019).

were studied in the context of the WL hierarchy (Maron et al., 2019a); it was shown that k-order IGNs are as expressive as k-WL and a 3-WL equivalent variant based on matrix multiplication, instead of linear layers, has been also proposed. A similar construction is Ring-GNN, proposed in (Chen et al., 2019). The main drawback of these methods is the complexity and memory requirements of $\mathcal{O}(n^k)$, and the number of parameters for linear IGNs of $\mathcal{O}(B_k)$, making them impractical.

From a different perspective, Sato et al. (Sato et al., 2019) and Loukas (Loukas, 2020) showed the connections between GNNs and distributed local algorithms (Angluin, 1980; Linial, 1992; Naor & Stockmeyer) and suggested alternative models that employ either local orderings, making GNNs more powerful than the WL test, or unique global identifiers, making GNNs universal. It was later shown that random features (Sato et al., 2020) can serve this role and allow node disambiguation in GNNs. However, these methods lack a principled way to choose orderings/identifiers to be shared across graphs (that would require a graph canonisation procedure). Other proposed methods (Murphy et al., 2019; Dasoulas et al., 2020) take into account all possible node permutations and can therefore be intractable; required approximations are at the cost of compromising expressivity.

Solely quantifying the power of GNNs in terms of their ability to distinguish non-isomorphic graphs does not provide the necessary granularity: even the 1-WL test can distinguish almost all (in the probabilistic sense) non-isomorphic graphs (Babai et al., 1980). Chen et al. (Chen et al., 2020) approached GNN and IGN expressivity by studying their limitations in *counting substructures*, and showing that GNNs can count only nodes, edges and star-shaped graphs, while k-IGNs can count any substructure of size k at initialisation. Similarly, for the k-WL tests, there have been efforts to analyse their power as graph invariants (Fürer, 2010; 2017; Arvind et al., 2019; Dell et al., 2018). It was established, for example, that the 3-WL test is more powerful than spectral invariants, can count (not necessarily induced) cycles and paths of length up to 7, but not 4-cliques, while the k-WL test can count the number of subgraph homomorphisms of treewidth up to k. These characterizations of expressivity are more intuitive and informative and serve as our motivation for the design of more powerful architectures.

Substructures in Complex Networks The idea of analysing complex networks based on small-scale topological characteristics dates back to the 1970's and the notion of triad census (Holland & Leinhardt, 1976). The seminal paper of Milo et al. (Milo et al., 2002) coined the term *network motifs* as over-represented subgraph patterns shown to characterise certain functional properties of complex networks, and this idea was later used to explain higher order organisational patterns in many systems (Benson et al., 2016; Paranjape et al., 2017; Kuramochi & Karypis, 2001; Kramer et al., 2001; Deshpande et al., 2005). The closely related concept of *graphlets* (Pržulj et al., 2004; Pržulj, 2007; Milenković & Pržulj, 2008; Sarajlić et al., 2016), different from motifs in being induced subgraphs, has been successfully used as a topological signature for network similarity. Our work is similar in spirit with the *graphlet degree vector* (GDV) (Pržulj, 2007), a node descriptor based on graphlet counting. In our work, we endeavour to combine these ideas with message passing, so as to learn richer representations by diffusing structural descriptors along with node and edge features through the graph.

Substructures have been also used in ML. In particular, subgraph patterns have been used to define Graph Kernels (GKs) (Horváth et al., 2004; Shervashidze et al., 2009; Costa & De Grave, 2010; Kriege & Mutzel, 2012), with the most prominent being the graphlet kernel (Shervashidze et al., 2009), also based on counting subgraph occurrences. In comparison to GNNs, these methods usually only extract graph-level representations and not node-level. Motif-based node embeddings (Dareddy et al., 2019; Rossi et al., 2018) and diffusion operators (Monti et al., 2018; Sankar et al., 2019; Lee et al., 2019) that employ adjacency matrices weighted according to motif occurrences, have recently been proposed for graph representation learning; these can be expressed by our general formulation. Finally, GNNs that operate in larger induced neighbourhoods (Li et al., 2019; Kim et al., 2019) or higher-order paths (Flam-Shepherd et al., 2020) have prohibitive complexity since the size of these neighbourhoods grows exponentially. Remotely related to our work is also the suggestion to approximate the subgraph isomoprhism counting problem with neural networks trained specfically for this task (Liu et al., 2019).