# Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting

Giorgos Bouritsas*[1], Fabrizio Frasca†[1,2], Stefanos Zafeiriou ‡[1], and Michael M. Bronstein§[1,2]

[1]Imperial College London, UK
[2]Twitter, UK

**Abstract**

While Graph Neural Networks (GNNs) have achieved remarkable results in a variety of applications, recent studies exposed important shortcomings in 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-Leman (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, e.g. in network science and bioinformatics, that substructures are often intimately related to downstream tasks. To this end, we propose "Graph Substructure Networks" (GSN), a topologically-aware message passing scheme based on substructure encoding. We theoretically analyse the expressive power of our architecture, showing that it is strictly more expressive than the WL test, and provide sufficient conditions for universality. Importantly, we do not attempt to adhere to the WL hierarchy; this allows us to retain multiple attractive properties of standard GNNs such as locality and linear network complexity, while being able to disambiguate even hard instances of graph isomorphism. We perform an extensive experimental evaluation on graph classification and regression tasks and obtain state-of-the-art results in diverse real-world settings including molecular graphs and social networks. The code is publicly available at `https://github.com/gbouritsas/graph-substructure-networks`.

## 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 ranging from social networks and recommender systems [1] to bioinformatics [2, 3], chemistry [4, 5, 6] and physics [7, 8], to name a few. Most GNN architectures are based on message passing [5], where the representation of each node is iteratively updated by aggregating information from its neighbours.

A crucial difference from traditional neural networks operating on grid-structured data is the absence of canonical ordering of the nodes in a graph. To address this, the aggregation function is constructed to be invariant to neighbourhood permutations and, as a consequence, to graph isomorphism. This kind of symmetry is not always desirable and thus different inductive biases that disambiguate the neighbours have been proposed. For instance, in geometric graphs, such as 3D molecular graphs and meshes, directional biases are usually employed in order to model the positional information of the nodes [9, 10, 11, 12, 13]; for proteins, ordering information is used to disambiguate amino-acids at different positions in the sequence [14]; in multi-relational knowledge graphs, a different aggregation is performed for each relation type [15].

*g.bouritsas@imperial.ac.uk
†ffrasca@twitter.com
‡s.zafeiriou@imperial.ac.uk
§mbronstein@twitter.com

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, since message-passing GNNs are at most as powerful as the Weisfeiler Leman test (WL) [16, 17], they are limited in their abilities to adequately exploit the graph structure, e.g. by counting substructures [18, 19]. This uncovers a crucial limitation of GNNs, as substructures have been widely recognised as important in the study of complex networks. For example, in molecular chemistry, functional groups and rings are related to a plethora of chemical properties, while cliques are related to protein complexes in Protein-Protein Interaction networks and community structure in social networks, respectively [20, 21].

Therefore, three major questions arise when designing GNN architectures: (a) How to go beyond *isotropic*, i.e., locally symmetric, aggregation functions ? (b) How to ensure that GNNs are aware of the *structural chatacteristics* of the graph? (c) How to achieve the above two without sacrificing *invariance to isomorphism* and hence the ability of GNNs to generalise?

In this work we attempt to simultaneously provide an answer to the above. We propose to break local symmetries by introducing structural information in the aggregation function, hence addressing (a) and (b). In particular, the contribution of each neighbour (*message*) is transformed differently depending on its structural relationship with the central node. This relationship is expressed by counting the appearance of certain substructures. Since substructure counts are *vertex invariants*, i.e. they are invariant to vertex permutations, it is easy to see that the resulting GNN will be invariant to isomorphism, hence also addressing (c). Moreover, by choosing the substructures, one can provide the model with different inductive biases, based on the graph distribution at hand.

We characterise the expressivity of our message-passing scheme, coined as *Graph Substructure Network (GSN)*, showing that GSN is strictly more expressive than traditional GNNs for the vast majority of substructures, while retaining the locality of message passing, as opposed to higher-order methods [22, 23, 24, 17] that follow the WL hierarchy (see Section 2). In the limit, our model can yield a unique representation for every isomorphism class and is thus universal. We provide an extensive experimental evaluation on hard instances of graph isomorphism testing (strongly regular graphs), as well as on real-world networks from the social and biological domains, including the recently introduced large-scale benchmarks [25, 26]. We observe that when choosing the structural inductive biases based on domain-specific knowledge, GSN achieves state-of-the-art results.

# 2 Preliminaries

Let $G = (\mathcal{V}_G, \mathcal{E}_G)$ be a graph with vertex set $\mathcal{V}_G$ and edge set $\mathcal{E}_G$, directed or undirected. 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., $\mathcal{E}_{G_S} = \mathcal{E}_G \cap \left(\mathcal{V}_{G_S} \times \mathcal{V}_{G_S}\right)$, the subgraph is said to be *induced*.

## 2.1 Isomorphism & Automoprhism

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 forms the *automorphism group* of the graph, denoted as $\mathrm{Aut}(H)$, which contains 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 endpoint 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: $\mathrm{Orb}(v) = \{u \in \mathcal{V}_H \mid \exists g \in \mathrm{Aut}(H) \text{ s.t. } g(u) = v\}$, and the set of all orbits $H \setminus \mathrm{Aut}(H) = \{\mathrm{Orb}(v) \mid v \in \mathcal{V}_H\}$ 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* $\{O^E_{H,1}, O^E_{H,2}, \ldots, O^E_{H,d_H}\}$.

### 2.2 Weisfeiler-Leman tests

The *Weisfeiler-Leman graph-isomorphism test* [28], also known as naive vertex refinement, *1-WL*, or just *WL*), is a fast heuristic to decide if two graphs are isomorphic or not. 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:

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

where $\{\!\{\cdot\}\!\}$ denotes a multiset (a set that allows element repetitions) and $\mathcal{N}(v)$ is the neighbourhood of $v$. The WL algorithm terminates when the colours stop changing, and outputs a histogram of colours. Two graphs with different histograms are non-isomorphic; if the histograms are identical, the graphs are possibly, but not necessarily, isomorphic. Note that the neighbour aggregation in the WL test is a form of message passing, and GNNs are the learnable analogue.

A series of works on improving GNN expressivity mimic the higher-order generalisations of WL, known as $k$-WL and $k$-Folklore WL (WL hierarchy) and operate on $k$-tuples of nodes (see Appendix B.1). The $(k+1)$-FWL is strictly stronger than $k$-FWL, $k$-FWL is as strong as $(k+1)$-WL and 2-FWL is strictly stronger than the simple 1-WL test.

## 3 Graph Substructure Networks

Graphs 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 [29] or positional encodings in language models for sequential data [30, 31, 32]. Nevertheless, GNNs are usually unaware of the nodes' different structural roles, since all nodes are treated equally when performing local operations. Despite the initial intuition that the neural network would be able to discover these roles by constructing deeper architectures, it has been shown that GNNs are ill-suited for this purpose and are blind to the existence of structural properties, e.g. triangles or larger cycles [19, 18].

To this end, we propose to explicitly encode structural roles as part of message passing, in order to capture richer topological properties. Our method draws inspiration from [33], where it was shown that GNNs become universal when the nodes in the graph are uniquely identified, i.e when they are equipped with different features. However, it is not clear how to choose these identifiers in a permutation equivariant way. Structural roles, when treated as identifiers, although not necessarily unique, are not only permutation equivariant, but also more amenable to generalisation due to their repetition across different graphs. Thus, they can constitute a trade-off between uniqueness and generalisation.

### 3.1 Structural features

Structural roles are encoded into features by counting the appearance of certain substructures. Define a set of small (connected) graphs $\mathcal{H} = \{H_1, H_2 \ldots H_K\}$, for example cycles of fixed length or cliques. For each graph $H \in \mathcal{H}$, we first find its isomorphic subgraphs in $G$ denoted as $G_S$. 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 $H$, $\text{Orb}_H(f(v))$. By counting all the possible appearances of different orbits in $v$, we obtain the *vertex structural feature* $\mathbf{x}^V_H(v)$ of $v$, defined as follows. For all $i \in \{1, \ldots, d_H\}$:

[1] Note that the edge automorphism group is larger than that of induced automorphisms, but strictly larger only for 3 trivial cases [27]. However, induced automorphisms provide a more natural way to express edge structural roles.

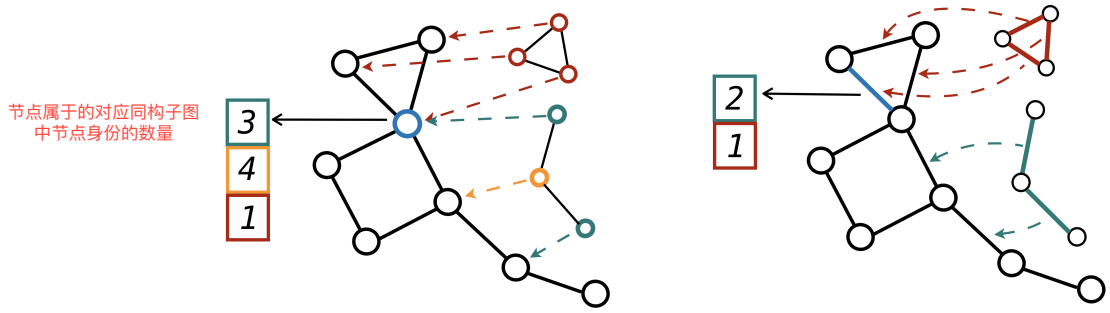


Figure 1: *Node* (left) and *edge* (right) induced subgraph counting for a 3-cycle and a 3-path. Counts are reported for the blue node on the left and for the blue edge on the right. Different colors depict orbits.

$$x^V_{H,i}(v) = \left| \left\{ G_S \simeq H \;\middle|\; v \in \mathcal{V}_{G_S},\ f(v) \in O^V_{H,i} \right\} \right|. \tag{2}$$

Note that there exist $|\mathrm{Aut}(H)|$ different functions $f$ that can map a subgraph $G_S$ to $H$, but any of those can be used to determine the orbit mapping of each node $v$. By combining the counts from different substructures in $\mathcal{H}$ and different orbits, we obtain the feature vector $\mathbf{x}^V_v = [\mathbf{x}^V_{H_1}(v), \ldots, \mathbf{x}^V_{H_K}(v)] \in \mathbb{N}^{D\times 1}$ of dimension $D = \sum_{H_i \in \mathcal{H}} d_{H_i}$.

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

$$x^E_{H,i}(u,v) = \left| \left\{ G_S \simeq H \;\middle|\; (u,v) \in \mathcal{E}_{G_S},\ (f(u),f(v)) \in O^E_{H,i} \right\} \right|, \tag{3}$$

and the combined edge features $\mathbf{x}^E_{u,v} = [\mathbf{x}^E_{H_1}(u,v), \ldots, \mathbf{x}^E_{H_K}(u,v)]$. An example of vertex and edge structural features is illustrated in Figure 1.

### 3.2 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) [5], where now the messages from the neighbouring nodes also contain the structural information. In particular, each node $v$ updates its state $\mathbf{h}^t_v$ by combining its previous state with the aggregated messages:

$$\mathbf{h}^{t+1}_v = \mathrm{UP}^{t+1}\left(\mathbf{h}^t_v, \mathbf{m}^{t+1}_v\right) \tag{4}$$

$$\mathbf{m}^{t+1}_v = \begin{cases} M^{t+1}\left(\wr(\mathbf{h}^t_v, \mathbf{h}^t_u, \mathbf{x}^V_v, \mathbf{x}^V_u, \mathbf{e}_{u,v})\wr_{u\in\mathcal{N}(v)}\right) \text{ (\textbf{GSN-v})} \\ \qquad\qquad \text{or} \\ M^{t+1}\left(\wr(\mathbf{h}^t_v, \mathbf{h}^t_u, \mathbf{x}^E_{u,v}, \mathbf{e}_{u,v})\wr_{u\in\mathcal{N}(v)}\right) \text{ (\textbf{GSN-e}),} \end{cases} \tag{5}$$

where $\mathrm{UP}^{t+1}$ is an arbitrary function approximator (e.g. a MLP), $M^{t+1}$ is the neighborhood aggregation function, i.e. an arbitrary function on multisets (e.g., of the form $\sum_{u\in\mathcal{N}(v)} \mathrm{MLP}(\cdot)$) and $\mathbf{e}_{u,v}$ are the edge features. The two variants, named *GSN-v* and *GSN-e*, correspond to vertex- or edge-counts, respectively, which are analogous to absolute and relative positional encodings in language models [34, 35].

It is important to note here that contrary to identifier-based GNNs [33, 36, 37] that obtain universality at the expense of permutation equivariance (since the identifiers are arbitrarily chosen with the sole requirement of being unique), GSNs retain this property, hence they are by construction ivariant to isomorphism. This stems from the fact that the process generating our structural identifiers (i.e. subgraph isomorphism) is permutation equivariant itself (proof provided in the Appendix A.1).

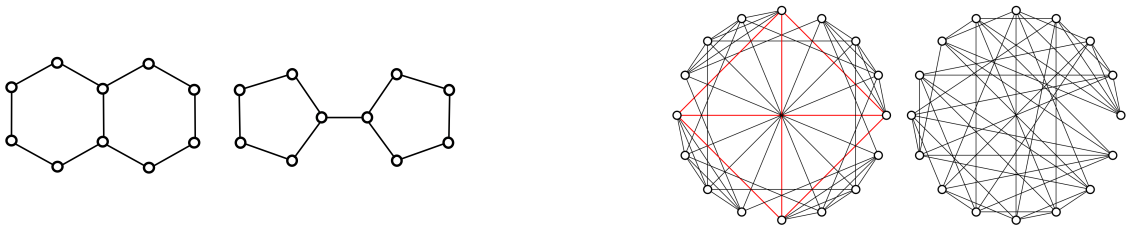

Figure 2: (Left) *Decalin* and *Bicyclopentyl*: Non-isomorphic molecular graphs than can be distinguished by GSN, but not the by the WL test [38] (nodes represent carbon atoms and edges chemical bonds). (Right) *Rook's 4x4 graph* and the *Shrikhande graph*: the smallest pair of strongly regular non-isomorphic graphs with the same parameters SR(16,6,2,2). GSN can distinguish them with 4-clique counts, while 2-FWL fails.

### 3.3 How powerful are GSNs?

We now turn to the expressive power of GSNs in comparison to MPNNs and the WL tests, a key tool for the theoretical analysis of the expressivity of graph neural networks so far. Since GSN is a generalisation of MPNNs, it is easy to see that it is at least as powerful. Importantly, 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 [18] and its extension to MPNNs [19] (for proofs, see Appendices A.2-A.4).

**Theorem 3.1.** *GSN is strictly more powerful than MPNN and the 1-WL test when one of the following holds:*

- *$H$ is any graph except for star graphs of any size, and structural features are inferred by subgraph matching, i.e. we count all subgraphs $G_S \cong H$ for which it holds that $\mathcal{E}_{G_S} \subseteq \mathcal{E}_G$. Or,*
- *$H$ is any graph except for single edges and single nodes, and structural features are inferred by **induced** subgraph matching, i.e. we count all subgraphs $G_S \cong H$ for which it holds that $\mathcal{E}_{G_S} = \mathcal{E}_G \cap (\mathcal{V}_{G_S} \times \mathcal{V}_{G_S})$.*

*Proof.* It is easy to see that GSN model class contains MPNNs, and is thus at least as expressive. We can also show that GSN is at least as expressive as the 1-WL test by repurposing the proof of Theorem 3 in [16] (see Appendix A.2)

Given the first part of the proposition, in order to show that GSNs are strictly more expressive than the 1-WL test, it suffices to show that GSN can distinguish a pair of graphs that 1-WL deems isomorphic. [18] 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, [19] 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.

If $H$ is a substructure that 1-WL 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 1-WL deems 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), where the two non-isomorphic graphs are distinguishable by GSN via e.g. cycle counting, but not by 1-WL. □

**Universality.** A natural question that emerges is what are the sufficient conditions under which GSN can solve graph isomorphism. This would entail that GSN is a universal approximator of functions defined on graphs [37, 39]. To address this, we can examine whether there exists a specific substructure collection that can completely characterise each graph. 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. However, the *Reconstruction Conjecture* [40, 41], states that a graph with size $n \geq 3$ can be reconstructed from its vertex-deleted subgraphs (proven for $n \leq 11$ [42]). Consequently, (proof in the Appendix A.3):

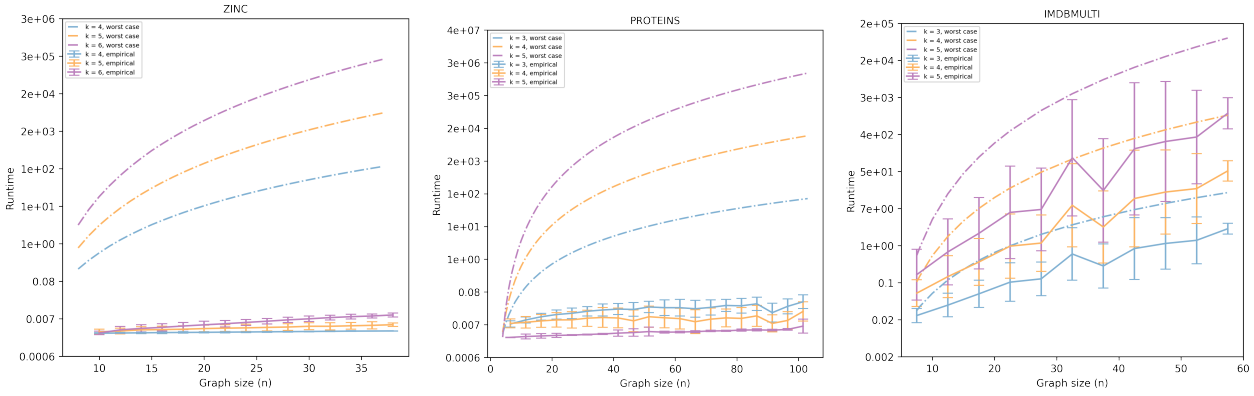


Figure 3: Empirical (solid) vs worst case (dashed) runtime for different graph distributions (in seconds, log scale). For each distribution we count the best performing (and frequent) substructures of increasing sizes $k$. Computational complexity for real-life graphs is significantly better than the worst case.

**Corollary 3.2.** *If the Reconstruction Conjecture holds and the substructure collection $\mathcal{H}$ contains all graphs of size $k = n - 1$, then GSN can distinguish all non-isomorphic graphs of size $n$ and is therefore universal.*

**GSN-v vs GSN-e.** We can also examine the expressive power of the two proposed variants. A crucial observation that we make is that for each graph $H$ in the collection, the vertex structural identifiers can be reconstructed by the corresponding edge identifiers. Thus, we can show that for every GSN-v there exists a GSN-e that can simulate the behaviour of the former (proof provided in the Appendix A.4).

**Theorem 3.3.** *For a given subgraph collection $\mathcal{H}$, let $C^V$ the set of functions that can be expressed by a GSN-v with arbitrary depth and with, and $C^E$ the set of functions that can be expressed by a GSN-e with the same properties. Then, it holds that $C^E \supseteq C^V$, or in other words GSN-e is at least as expressive as GSN-v.*

**Comparison with higher-order WL tests.** Finally, the expressive power of GSN can be compared to higher-order versions of the WL test. In particular, for each $k$-th order Folklore WL test in the hierarchy, it is known that there exists a family of graphs that will make the test fail. These are known in the literature as $k$-isoregular graphs [43], and the most well-known example is the *Strongly Regular* (SR) graph family, for $k = 2$ (more details can be found in Appendix B).

Hence, if we can find a substructure collection that allows GSN to distinguish certain pairs from these families, then this guarantees that the corresponding $k$-FWL test is no stronger than GSN. In this work, we identify such counterexamples for the 2-FWL test. Formally:

**Proposition 3.4.** *There exist substructure families with $\mathcal{O}(1)$ size, i.e., independent of the size of the graph $n$, such that 2-FWL is no stronger than GSN.*

We provide numerous counterexamples that prove this claim. Figure 2 (right) provides a typical pair of SR graphs that can be distinguished with a 4-clique, while in section 5.1 this is extended to a large-scale study, where other constant size substructures (paths, cycles and cliques) can achieve similar results.

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, despite the increase in expressivity, k-WL tests are not only more computationally involved, but they also process the graph in a non-local fashion. However, locality is presumed to be a strong inductive bias of GNNs and key to their excellent performance in real-world scenarios.

### 3.4 How to choose the substructures?

**Expressivity.** The Reconstruction Conjecture provides a sufficient, albeit impractical condition for universality. This motivates us to analyse the constant size case $k = \mathcal{O}(1)$ for practical scenarios, similar to the argument put forward for hard instances of graph isomorphism (Proposition 3.4).

In particular, one can count only the most discriminative subgraphs, i.e. the ones that can achieve the maximum possible vertex disambiguation, similarly to identifier-based approaches. Whenever these subgraph counts can provide a unique identification of the vertices, then universality will also hold (Corollary 3.1. in [33]).

We conjecture, that in real-world scenarios the number of subgraphs needed for unique, or near-unique identification, are far fewer than those dictated by Corollary 3.2. This is consistent with our experimental findings, where we observed that certain small substructures such as paths and trees, significantly improve vertex disambiguation, compared to the initial vertex features (see Figure 5 (left) and Table 2 in the appendix). As expected this allows for better fitting of the training data, which validates our claim that GNN expressivity improves.

**Generalisation.** However, none of the above claims can guarantee good generalisation in unseen data. For example, in Figure 5, we observe that the test set performance does not follow the same trend with train performance when choosing substructures with strong vertex disambiguation. Aiming at better generalisation, it is desirable to make use of substructures for which there is prior knowledge of their importance in certain network distributions and have been observed to be intimately related to various properties. For example, small substructures (graphlets) have been extensively analysed in protein-protein interaction networks [44], triangles and cliques characterise the structure of ego-nets and social networks in general [20], simple cycles (rings) are central in molecular distributions, directed and temporal motifs have been shown to explain the working mechanisms of gene regulatory networks, biological neural networks, transportation networks and food webs [45, 46, 47].

In Figure 5 (right), we showcase the importance of these inductive biases: a cycle-based GSN predicting molecular properties achieves smaller generalisation gap compared to a traditional MPNN, while at the same time generalising better with less training data. Choosing the best substructure collection is still an open problem that does not admit a straightforward solution due to its combinatorial nature. Alternatively, various heuristics can be used, e.g., motif frequencies or feature selection strategies. Answering this question is left for future work.

### 3.5 Complexity

The complexity of GSN comprises two parts: precomputation (substructure counting) and training/testing. The key appealing property is that training and inference are linear w.r.t the number of edges, $\mathcal{O}(|\mathcal{E}|)$, as opposed to higher-order methods [24, 17] with $\mathcal{O}(n^k)$, and [48] with $\mathcal{O}(n^2)$ training complexity and relational pooling [49] with $\mathcal{O}(n!)$ training complexity in absence of approximations.

The worst-case complexity of subgraph isomorphism of fixed size $k$ is $\mathcal{O}(n^k)$, by examining all the possible $k$-tuples in the graph. However, for specific types of subgraphs, such as paths and cycles, the problem can be solved even faster (see e.g. [50]). Approximate counting algorithms are also widely used, especially for counting frequent network motifs [51, 52, 53, 54], and can provide a considerable speed-up. Furthermore, recent neural approaches [55, 56] provide fast approximate counting.

In our experiments, we performed exact counting using the common isomorphism algorithm VF2 [57]. Although its worst case complexity is $\mathcal{O}(n^k)$, it scales better in practice, for instance when the candidate subgraph is infrequently matched or when the graphs are sparse, and is also trivially parallelisable. In Figure 3, we show a quantitative analysis of the empirical runtime of the counting algorithm against the worst case, for three different graph distributions: molecules, protein contact maps, social networks. It is easy to see that when the graphs are sparse (for the first two cases) and the number of matches is small, the algorithm is significantly faster than the worst case, while it scales better with the size of the graph $n$. Even, in the case of social networks, where several examples are near-complete graphs, both the runtime and the growth w.r.t

both $n$ and $k$ are better than the worst case. Overall, the preprocessing computational burden in most of the cases remains negligible for relatively small and sparse graphs, as it is the case of molecules.

# 4 Related Work

## 4.1 Expressive power of GNNs

**WL hierarchy.** The seminal results in the theoretical analysis of the expressivity of GNNs [16] and k-GNNs [17] established that traditional message passing-based GNNs are at most as powerful as the 1-WL test. [39] showed that graph isomorphism is equivalent to universal invariant function approximation. Higher-order Invariant Graph Networks (IGNs) have been studied in a series of works [22, 23, 24, 39, 58, 59], establishing connections with the WL hierarchy, similarly to [17, 60]. The main drawbacks of these methods are the training and inference time complexity and memory requirements of $\mathcal{O}(n^k)$ and the super-exponential number of parameters (for linear IGNs) making them impractical, as well as their non-local nature making them more prone to overfitting. Finally, [61] also analysed the expressive power of MPNNs and other more powerful variants and provided generalisation bounds.

**Unique identifiers.** From a different perspective, [62] and [33] showed the connections between GNNs and distributed local algorithms [63, 64, 65] and suggested more powerful alternatives based on either local orderings or unique global identifiers (in the form of random features in [36]) that make GNNs universal. Similarly, [37] propose to use random colorings in order to uniquely identify the nodes. However, these methods lack a principled permutation equivariant way to choose orderings/identifiers. To date this is an open problem in graph theory called *graph canonisation* and it is at least as hard as solving graph isomorphism itself. A possible workaround is proposed in [49], where the authors take into account all possible vertex permutations. However, obviously this quickly becomes intractable ($\mathcal{O}(n!)$) even when considering small-sized graphs.

**More expressive permutation equivariant GNNs.** Concurrently with our work, other more expressive GNNs have been proposed using equivariant message passing. In [66], the authors propose to linearly transform each message with a different kernel based on the local isomorphism class of the corresponding edge (similar to our definition of structural roles). However, as also noted by the authors, taking into account all possible local isomorphism classes leads to insufficient weight sharing and hence to overfitting. In contrast, in GSN, usually the substructure collection is small ($\sim$5-10 graphs) and the substructures are repetitive in the graph distribution, and as a result generalisation improves. Vignac et al. [48] propose a message passing scheme where matrices of order equal to the size of the graph are propagated instead of vectors. This can be perceived as a practical unique identification scheme, but the neural network complexity becomes quadratic in the number of nodes. Finally, [67] and [68] enhance the aggregation function with distance encodings (a strategy more relevant for vertex-level tasks) and graph eigenvectors respectively as alternative symmetry breaking mechanisms. In the experimental section, GSN is compared against these methods in real-world scenarios.

**Quantifying expressivity.** Solely quantifying the expressive 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 [69]. As a result, there have been several efforts to analyse the power of $k$-WL tests in comparison to other graph invariants [70, 71, 18, 72], while recently [19] approached GNN expressivity by studying their ability to count substructures.

## 4.2 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 for directed graphs [73]. The seminal paper of [45] coined the term *network motifs* as over-represented subgraph patterns that were shown to characterise certain functional properties of complex networks in systems biology. The closely related concept of *graphlets* [44, 74, 75, 76], different from motifs in being induced subgraphs, has been used to analyse the distribution of real-world networks and as

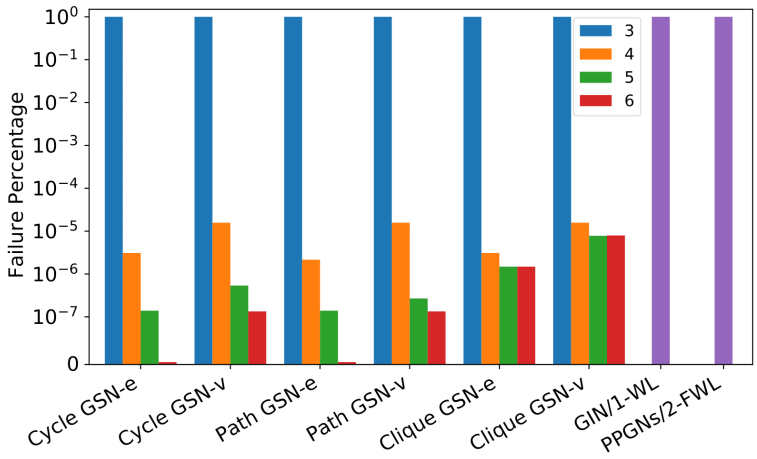


Figure 4: SR graphs isomorphism test (log scale, smaller values are better). Different colours indicate different substructure sizes.

a topological signature for network similarity. Our work is similar in spirit with the *graphlet degree vector* (GDV) [74], a node-wise descriptor based on graphlet counting.

Substructures have been also used in the context of ML. In particular, subgraph patterns have been used to define Graph Kernels (GKs) [77, 78, 79, 80, 81], with the most prominent being the graphlet kernel [78]. Motif-based node embeddings [82, 83] and diffusion operators [84, 85, 86] that employ adjacency matrices weighted according to motif occurrences, have recently been proposed for graph representation learning. Our formulation provides a unifying framework for these methods and it is the first to analyse their expressive power. Finally, GNNs that operate in larger induced neighbourhoods [87, 88] or higher-order paths [89] have prohibitive complexity since the size of these neighbourhoods typically grows exponentially.

# 5 Experimental Evaluation

In the following section we evaluate GSN in comparison to the state-of-the-art in a variety of datasets from different domains. We are interested in practical scenarios where the collection of subgraphs, as well as their size, are kept small. Depending on the dataset domain we experimented with typical substructure families (*cycles, paths* and *cliques*) and maximum substructure size $k$ (note that for each setting, our substructure collection consists of all the substructures of the family with size $\leq k$). We also experimented with both graphlets and motifs and observed similar performance in most cases. To showcase that structural features can be used as an off-the-shelf strategy to boost GNN performance, we usually choose a base message passing architecture and minimally modify it into a GSN. Unless otherwise stated, the base architecture is a general-purpose MPNN with MLPs used in the message and update functions. Additional implementation details can be found in the Appendix C.

## 5.1 Synthetic Graph Isomorphism test

We tested the ability of GSNs to decide if two graphs are non-isomorphic on a collection of Strongly Regular graphs of size up to 35 nodes, attempting to disambiguate pairs with the same number of nodes (for different sizes the problem becomes trivial). As we are only interested in the bias of the architecture itself, we 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 $\epsilon$. Figure 4 shows the failure percentage of our isomorphism test when using different graphlet substructures (*cycles, paths*, and *cliques*) of varying size $k$. Interestingly, the number of failure cases of GSN decreases rapidly as we increase $k$; cycles

and paths of maximum length $k = 6$ 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. Observe also the discrepancy between GSN-v and GSN-e. In particular, vertex-wise counts do not manage to distinguish all graphs, although missing only a few instances, which is in accordance with Theorem 3.3. Finally, 1-WL [16] and 2-FWL [24] equivalent models demonstrate 100% failure, as expected from theory.

## 5.2 TUD Graph Classification Benchmarks

We evaluate GSN on datasets from the classical TUD benchmarks. We use seven datasets from the domains of bioinformatics and computational social science and compare against various GNNs and Graph Kernels. The base architecture that we used is GIN [16]. We follow the same evaluation protocol of [16], performing 10-fold cross-validation and then reporting the performance at the epoch with the best average accuracy across the 10 folds. Table 1 lists all the methods evaluated with the split of [90]. We select our model by tuning architecture and optimisation hyperparameters and substructure related parameters, that is: (i) $k$, (ii) motifs against graphlets. Following domain evidence we choose the following substructure families: *cycles* for molecules, *cliques* for social networks. Best performing substructures both for GSN-e and GSN-v are reported. As can be seen, our model obtains state-of-the-art performance in most of the datasets, with a considerable margin against the main GNN baselines in some cases.

Table 1: Graph classification accuracy on TUD Dataset. **First**, **Second**, **Third** best methods are highlighted. For GSN, the best performing structure is shown. *Graph Kernel methods.

| Dataset | MUTAG | PTC | Proteins | NCI1 | Collab | IMDB-B | IMDB-M |
|---|---|---|---|---|---|---|---|
| RWK* [91] | 79.2±2.1 | 55.9±0.3 | 59.6±0.1 | >3 days | N/A | N/A | N/A |
| GK* (k=3) [78] | 81.4±1.7 | 55.7±0.5 | 71.4±0.31 | 62.5±0.3 | N/A | N/A | N/A |
| PK* [92] | 76.0±2.7 | 59.5±2.4 | 73.7±0.7 | 82.5±0.5 | N/A | N/A | N/A |
| WL kernel* [93] | 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* [94] | 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 [95] | 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 [90] | 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 [22] | 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 [16] | 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 [24] | **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 |
| Natural GN [66] | 89.4±1.60 | 66.8±1.79 | 71.7±1.04 | 82.7±1.35 | N/A | 74.8±2.01 | 51.3±1.50 |
| WEGL [96] | N/A | **67.5±7.7** | 76.5±4.2 | N/A | 80.6±2.0 | 75.4±5.0 | 52.3±2.9 |
| GIN+GraphNorm [97] | **91.6 ± 6.5** | 64.9 ± 7.5 | **77.4 ± 4.9** | 82.7 ± 1.7 | 80.2 ± 1.0 | 76.0 ± 3.7 | N/A |
| **GSN-e** | **90.6±7.5**<br>6 (cycles) | **68.2±7.2**<br>6 (cycles) | **76.6±5.0**<br>4 (cliques) | **83.5± 2.3**<br>15 (cycles) | **85.5±1.2**<br>3 (triangles) | **77.8±3.3**<br>5 (cliques) | **54.3±3.3**<br>5 (cliques) |
| **GSN-v** | **92.2±7.5**<br>12 (cycles) | 67.4±5.7<br>10 (cycles) | 74.59±5.0<br>4 (cliques) | **83.5±2.0**<br>3 (triangles) | **82.7±1.5**<br>3 (triangles) | **76.8±2.0**<br>4 (cliques) | **52.6±3.6**<br>3 (triangles) |

## 5.3 ZINC Molecular graphs

We evaluate GSN on the task of regressing the "penalized water-octanol partition coefficient - logP" (see [98, 99, 100] for details) of molecules from the ZINC database [101, 25]. We use structural features obtained with $k$-cycle counting and report the result of the best performing substructure w.r.t. the validation set.

As dictated by the evaluation protocol of [25], the total number of parameters of the model is approximately 100K, which is achieved by selecting an appropriate network width.[2] The data split is obtained from [25] and the evaluation metric is the Mean Absolute Error (MAE). We compare against a variety of baselines, ranging from traditional message passing NNs to recent more expressive architectures [102, 68, 48, 103] and a molecular-specific one which is based on the junction tree molecular decomposition [104]. Wherever possible,

[2] A larger version of GSN using 500K parameters attains **0.101 ± 0.010** test MAE.

Table 2: MAE in *ZINC*

| Method | MAE | MAE (EF) |
|---|---|---|
| GCN [107] | 0.469±0.002 | - |
| GIN [16] | 0.408±0.008 | - |
| GraphSage[108] | 0.410±0.005 | - |
| GAT [109] | 0.463±0.002 | - |
| MoNet[10] | 0.407±0.007 | - |
| GatedGCN [110] | 0.422±0.006 | 0.363±0.009 |
| MPNN | 0.254±0.014 | 0.209±0.018 |
| MPNN-r | 0.322±0.026 | 0.279±0.023 |
| PNA[102] | 0.320±0.032 | 0.188±0.004 |
| DGN[68] | **0.219±0.010** | 0.168±0.003 |
| GNNML[103] | **0.161±0.006** | - |
| HIMP[104] | - | **0.151±0.006** |
| SMP[48] | **0.219±** | **0.138±** |
| **GSN** | **0.140±0.006** | **0.115±0.012** |

Table 3: Test and Validation ROC-AUC in OGB-MOLHIV.

| Method | Test ROC-AUC | Validation ROC-AUC |
|---|---|---|
| GIN+VN[16] | 0.7707 ± 0.0149 | **0.8479 ± 0.0068** |
| DeeperGCN[111] | 0.7858 ± 0.0117 | 0.8427 ± 0.0063 |
| HIMP[104] | 0.7880 ± 0.0082 | - |
| GCN+GraphNorm[97] | 0.7883 ± 0.0100 | 0.7904 ± 0.0115 |
| PNA[102] | 0.7905 ± 0.0132 | **0.8519 ± 0.0099** |
| PHC-GNN[112] | 0.7934 ± 0.0116 | 0.8217 ± 0.0089 |
| DeeperGCN+FLAG[113] | **0.7942 ± 0.0120** | 0.8425 ± 0.0061 |
| DGN + eigenvectors [68] | **0.7970 ± 0.0097** | 0.8470 ± 0.0047 |
| P-WL [114] | **0.8039 ± 0.0040** | 0.8279 ± 0.0059 |
| **GSN** (GIN+VN base) | 0.7799±0.0100 | **0.8658±0.0084** |
| **GSN** (DGN + substructures) | **0.8039 ± 0.0090** | 0.8473 ± 0.0096 |

we compare two variants, one that does not take edge features into account and one that does. In both cases, GSN achieves state-of-the-art results outperforming all the baseline architectures.

## 5.4 OGB-MOLHIV

We use `ogbg-molhiv` from the Open Graph Benchmark - OGB - [26] as a graph-level binary classification task, where the aim is to predict if a molecule inhibits HIV replication or not. We obtain baseline results from a plethora of different methods[3], ranging from algorithms specific to molecular graphs to general purpose architectures. Following the same rationale as in the previous experiments, we choose a base architecture and modify it into a GSN variant by introducing structural features in the aggregation function (cycle counts, similar to other molecular datasets). Here we use the following two base architectures: (a) *GIN-VN*, a variation of GIN that allows for edge features and is extended with a *virtual node*, i.e. a node connected to every node in the graph. (b) Directional Graph Networks (DGN), a GNN that propagates messages in an anisotropic manner, based on a predefined graph vector field. Observe that the vector field is an alternative way to break local symmetries. The authors of DGN use vector fields defined by the eigenvectors of the graph, while in our case the vector field is defined by graph substructures. More information can be found in the supplementary material.

Using the evaluator provided by the authors, we report the ROC-AUC metric at the epoch with the best validation performance (substructures are also chosen based on the validation set). By examining the results in Table 3 the following observations can be made, (a) general purpose GNNs benefit from symmetry breaking mechanisms, either in the form of eigenvectors or in the form of substructures. (b) Cyclical substructures are a good inductive bias when learning on molecules (e.g. P-WL is a graph kernel based on topological features that contain information of graph cycles, similar to GSN). (c) Further evidence for that is provided by observing the performance of molecular fingerprints methods. In specific, a method based on the extended-connectivity fingerprints [105], which mainly focuses on the structure of the molecule reports 0.8060 ± 0.0010 test performance, while one that additionally uses the MACCS fingerprints [106], which mainly encode the presence of certain functional groups (i.e. both structure and attributes), reports 0.8232 ± 0.0047 test performance. These methods although not directly comparable to ours, currently achieve the best results in this dataset, thus this further motivates research on making GNNs structure-aware.

[3]the most representative ones from the OGB public leaderboard: `https://ogb.stanford.edu/docs/leader_graphprop/#ogbg-molhiv`

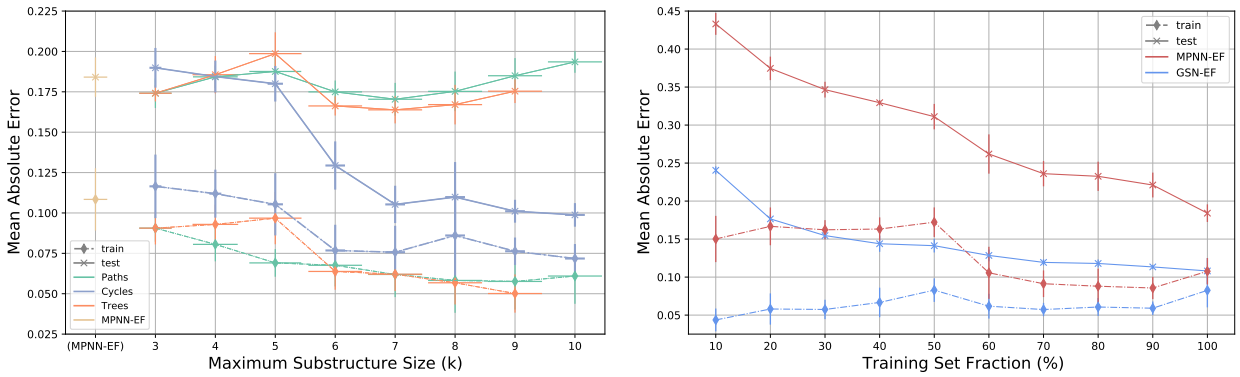


Figure 5: (Left) Train (dashed) and test (solid) MAEs for Path-, Tree- and Cycle-GSN-*EF* as a function of the maximum substructure size $k$. Vertical bars indicate standard deviation; horizontal bars depict disambiguation scores $\delta$. (Right) Train (dashed) and test (solid) MAEs for GSN-*EF*(blue) and MPNN-*EF*(red) as a function of the dataset fraction used for training

## 5.5 Ablation Studies

### 5.5.1 Comparison between substructure collections

In Figure 5 (left), we compare the training and test error for different substructure families (cycles, paths and non-isomorphic trees – for each experiment we use all the substructures of size $\leq k$ in the family). Additionally, we measure the "uniqueness" of the identifiers each substructure yields as follows: for each graph $G$ in the dataset, we measure the number of unique vertex features $u_G$ (input vertex features concatenated with vertex structural identifiers for GSN-v). Then, we sum them up over the entire training set and divide by the total number of nodes, yielding the disambiguation score $\delta = \frac{\sum_G u_G}{\sum_G |V_G|}$. The disambiguation scores for the different types of substructures are illustrated as horizontal bars in Figure 5 (the exact values can be found in Appendix C.2, Table 2).

A first thing to notice is that the training error is tightly related to the disambiguation score. As identifiers become more discriminative, the model gains expressive power. On the other hand, the test error is not guaranteed to decrease when the identifiers become more discriminative. For example, although cycles have smaller disambiguation scores, they manage to generalise much better than the other substructures, the performance of which is similar to the baseline architecture (MPNN with MLPs). This is also observed when comparing against [36] (*MPNN-r* method in Table 2), where, akin to unique identifiers, random features are used to strengthen the expressivity of GNN architectures. This approach also fails to improve the baseline architecture in terms of the performance in the test set. This validates our intuition that unique identifiers can be hard to generalise when chosen in a non-permutation equivariant way and motivates once more the importance of choosing the identifiers not only based on their discriminative power, but also in a way that allows incorporating the appropriate inductive biases. Finally, we observe a substantial jump in performance when using GSN with cycles of size $k \geq 6$. This is not surprising, as cyclical patterns of such sizes (e.g. aromatic rings) are very common in organic molecules.

### 5.5.2 Generalisation

We repeat the experimental evaluation on ZINC using different fractions of the training set and compare the vanilla MPNN model against GSN. In Figure 5 (right), we plot the training and test errors of both methods. Regarding the training error, GSN consistently performs better, following our theoretical analysis on its expressive power. More importantly, GSN manages to generalise much better even with a small fraction of the training dataset. Observe that GSN requires only 20% of the samples to achieve approximately the same test error that MPNN achieves when trained on the entire training set.

#### 5.5.3 Structural Features & Message Passing:

We perform an ablation study on the abilities of the structural features to predict the task at hand, when given as input to a graph-agnostic network. In particular, we compare our best performing GSN with a DeepSets model [115] that treats the input features and the structural identifiers as a set. For fairness of evaluation the same hyperparameter search is performed for both models (see Appendix C.5. Interestingly, as we show in Table 4, our baseline attains particularly strong performance across a variety of datasets and often outperforms other traditional message passing baselines. This demonstrates the benefits of these additional features and motivates their introduction in GNNs, which are unable to compute them. As expected, we observe that applying message passing on top of these features, brings performance improvements in the vast majority of the cases, sometimes considerably, as in the ZINC dataset.

Table 4: Comparison between DeepSets and GSN with the same structural features

| Dataset | DeepSets | # params | GSN | # params |
|---|---|---|---|---|
| MUTAG | **93.3±6.9** | 3K | 92.8±7.0 | 3K |
| PTC | 66.4±6.7 | 2K | **68.2±7.2** | 3K |
| Proteins | **77.8±4.2** | 3K | **77.8±5.6** | 3K |
| NCI1 | 80.3 ±2.4 | 10K | **83.5± 2.0** | 10K |
| Collab | 80.9 ±1.6 | 30K | **85.5±1.2** | 52K |
| IMDB-B | 77.1 ±3.7 | 51K | **77.8±3.3** | 65K |
| IMDB-M | 53.3 ±3.2 | 68K | **54.3±3.3** | 66K |
| ZINC | 0.288 ±0.003 | 366K | **0.108 ±0.018** | 385K |
| ogbg-molhiv | 77.34±1.46 | 3.4M | **77.99±1.00** | 3.3M |

# 6 Conclusion

In this paper, we propose a novel way to design structure-aware graph neural networks. Motivated by the limitations of traditional GNNs to capture important topological properties of the graph, we formulate a message passing scheme enhanced with structural features that are extracted by subgraph isomorphism. 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. In future work, we will further explore the expressivity of GSNs as an alternative to the $k$-WL tests, as well as their generalisation capabilities. Another important direction is to infer prominent substructures directly from the data and explore the ability of graph neural networks to compose substructures.

# 7 Acknowledgements

This research was partially supported by the ERC Consolidator Grant No. 724228 - LEMAN (GB and MB). The work of GB is partially funded by a PhD scholarship from the Department of Computing, Imperial College London. SZ acknowledges support from the EPSRC Fellowship DEFORM: Large Scale Shape Analysis of Deformable Models of Humans (EP/S010203/1) and a Google Faculty award. MB acknowledges support from Google Faculty awards and the Royal Society Wolfson Research Merit award.

# References

[1] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 974–983. ACM, 2018.

[2] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6530–6539, 2017.

[3] Pablo Gainza, Freyr Sverrisson, Federico Monti, Emanuele Rodola, D Boscaini, MM Bronstein, and BE Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.

[4] 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 (NIPS)*, pages 2224–2232, 2015.

[5] 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 (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[6] Benjamin Sanchez-Lengeling, Jennifer N Wei, Brian K Lee, Richard C Gerkin, Alán Aspuru-Guzik, and Alexander B Wiltschko. Machine learning for scent: Learning generalizable perceptual representations of small molecules. *arXiv preprint arXiv:1910.10685*, 2019.

[7] Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning, (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2693–2702. PMLR, 2018.

[8] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing systems (NIPS)*, pages 4502–4510, 2016.

[9] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *IEEE International Conference on Computer Vision Workshops, (ICCVW)*, pages 832–840. IEEE Computer Society, 2015.

[10] 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 (CVPR)*, pages 5425–5434, 2017.

[11] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. 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)*, pages 7213–7222, 2019.

[12] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[13] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020.

[14] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15794–15805, 2019.

[15] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018.

[16] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[17] 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. AAAI Press, 2019.

[18] Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. In *Fundamentals of Computation Theory (FCT)*, volume 11651 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2019.

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

[20] Mark Granovetter. The strength of weak ties: A network theory revisited. In *Sociological Theory*, pages 105–130, 1982.

[21] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences (PNAS)*, 99(12):7821–7826, 2002.

[22] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2019.

[23] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 4363–4371. PMLR, 2019.

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

[25] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[26] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

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

[28] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series 2*, 9:12–16, 1968. English translation is available at `https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf`.

[29] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[30] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2440–2448, 2015.

[31] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning, (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 2017.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[33] Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[34] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics, 2018.

[35] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics, 2019.

[36] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

[37] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[38] Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

[39] 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 (NeurIPS)*, pages 15868–15876, 2019.

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

[41] Stanislaw M Ulam. *A collection of mathematical problems*, volume 8. Interscience Publishers, 1960.

[42] Brendan D McKay. Small graphs are reconstructible. *Australasian J. Combinatorics*, 15:123–126, 1997.

[43] Brendan L Douglas. The weisfeiler-lehman method and graph isomorphism testing. *arXiv preprint arXiv:1101.5211*, 2011.

[44] Nataša Pržulj, Derek G Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[45] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[46] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM*, pages 601–610. ACM, 2017.

[47] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[48] Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[49] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 4663–4673. PMLR, 2019.

[50] Pierre-Louis Giscard, Nils M. Kriege, and Richard C. Wilson. A general purpose algorithm for counting simple cycles and simple paths of any length. *Algorithmica*, 81(7):2716–2737, 2019.

[51] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[52] Sebastian Wernicke. A faster algorithm for detecting network motifs. In Rita Casadio and Gene Myers, editors, *Algorithms in Bioinformatics, 5th International Workshop, WABI*, volume 3692 of *Lecture Notes in Computer Science*, pages 165–177. Springer, 2005.

[53] Sebastian Wernicke. Efficient detection of network motifs. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 3(4):347–359, 2006.

[54] Sebastian Wernicke and Florian Rasche. FANMOD: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[55] Rex Ying, Andrew Z. Wang, Jiaxuan You, and Jure Leskovec. Frequent subgraph mining by walking in order embedding space. In *International Conference on Machine Learning Workshops (ICMLW)*, 2020.

[56] Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020.

[57] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, 2004.

[58] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7090–7099, 2019.

[59] Omri Puny, Heli Ben-Hamu, and Yaron Lipman. From graph low-rank global attention to 2-fwl approximation. *arXiv preprint arXiv:2006.07846*, 2020.

[60] Christopher Morris, Gaurav Rattan, and Mutzel Petra. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[61] Vikas K. Garg, Stefanie Jegelka, and Tommi S. Jaakkola. Generalization and representational limits of graph neural networks. 2020.

[62] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4083–4092, 2019.

[63] Dana Angluin. Local and global properties in networks of processors. In *ACM Symposium on Theory of Computing (STOC)*, pages 82–93. ACM, 1980.

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

[65] Moni Naor and Larry J. Stockmeyer. What can be computed locally? In *ACM Symposium on Theory of Computing (STOC)*, pages 184–193. ACM, 1993.

[66] Pim de Haan, Taco Cohen, and Max Welling. Natural graph networks. *arXiv preprint arXiv:2007.08349*, 2020.

[67] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding–design provably more powerful gnns for structural representation learning. 2020.

[68] Dominique Beaini, Saro Passaro, Vincent Létourneau, William L Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *International Conference on Machine Learning, (ICML)*, Proceedings of Machine Learning Research, 2021.

[69] László Babai, Paul Erdos, and Stanley M Selkow. Random graph isomorphism. *SIAM Journal on computing*, 9(3):628–635, 1980.

[70] Martin Fürer. On the power of combinatorial and spectral invariants. *Linear algebra and its applications*, 432(9):2373–2380, 2010.

[71] Martin Fürer. On the combinatorial power of the weisfeiler-lehman algorithm. In *International Conference on Algorithms and Complexity (CIAC)*, volume 10236 of *Lecture Notes in Computer Science*, pages 260–271. Springer, 2017.

[72] Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets weisfeiler and leman. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[73] Paul W Holland and Samuel Leinhardt. Local structure in social networks. *Sociological methodology*, 7:1–45, 1976.

[74] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):177–183, 2007.

[75] Tijana Milenković and Nataša Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6:257–273, 2008.

[76] Anida Sarajlić, Noël Malod-Dognin, Ömer Nebil Yaveroğlu, and Nataša Pržulj. Graphlet-based characterization of directed networks. *Scientific reports*, 6:35098, 2016.

[77] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 158–167. ACM, 2004.

[78] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics (AISTATS)*, volume 5 of *Proceedings of Machine Learning Research*, pages 488–495. PMLR, 2009.

[79] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *International Conference on Machine Learning (ICML)*, pages 255–262. Omnipress, 2010.

[80] Nils M. Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *International Conference on Machine Learning, (ICML)*, page 291–298. Omnipress, 2012.

[81] Hoang NT and Takanori Maehara. Graph homomorphism convolution. In *International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2020.

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

[83] Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi Yadkori. Hone: Higher-order network embeddings. *arXiv preprint arXiv:1801.09303*, 2018.

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

[85] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. 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)*, pages 137–144. ACM, 2019.

[86] John Boaz Lee, Ryan Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. Graph convolutional networks with motif-based attention. In *28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 499–508. ACM, 2019.

[87] Michael Lingzhi Li, Meng Dong, Jiawei Zhou, and Alexander M Rush. A hierarchy of graph neural networks based on learnable local features. *arXiv preprint arXiv:1911.05256*, 2019.

[88] Cheolhyeong Kim, Haeseong Moon, and Hyung Ju Hwang. Near: Neighborhood edge aggregator for graph classification. *arXiv preprint arXiv:1909.02746*, 2019.

[89] Daniel Flam-Shepherd, Tony Wu, Pascal Friederich, and Alan Aspuru-Guzik. Neural message passing on high order paths. *Advances in Neural Information Processing Systems Workshops (NeurIPSW)*, 2020.

[90] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*, pages 4438–4445. AAAI Press, 2018.

[91] Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Computational Learning Theory and Kernel Machines (COLT)*, volume 2777, pages 129–143. Springer, 2003.

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

[93] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research (JMLR)*, 12:2539–2561, 2011.

[94] Simon S. Du, Kangcheng Hou, Ruslan Salakhutdinov, Barnabás Póczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5724–5734, 2019.

[95] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1993–2001, 2016.

[96] Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2021.

[97] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning, (ICML)*, Proceedings of Machine Learning Research, 2021.

[98] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[99] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1945–1954. PMLR, 2017.

[100] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2328–2337. PMLR, 2018.

[101] John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

[102] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[103] 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, (ICML)*, Proceedings of Machine Learning Research, 2021.

[104] Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

[105] Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6):1273–1280, 2002.

[106] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[107] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2017.

[108] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1024–1034. Curran Associates, Inc., 2017.

[109] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2018.

[110] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[111] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[112] Tuan Le, Marco Bertolini, Frank Noé, and Djork-Arné Clevert. Parameterized hypercomplex graph neural networks for graph classification. *arXiv preprint arXiv:2103.16584*, 2021.

[113] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Flag: Adversarial data augmentation for graph neural networks. *arXiv preprint arXiv:2010.09891*, 2020.

[114] Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning*, pages 5448–5458. PMLR, 2019.

[115] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

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

[117] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[118] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[119] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[120] 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, (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 5449–5458. PMLR, 2018.

[121] Zhichang Liu, Siva Krishna Mohan Nalluri, and J Fraser Stoddart. Surveying macrocyclic chemistry: from flexible crown ethers to rigid cyclophanes. *Chemical Society Reviews*, 46(9):2459–2478, 2017.

[122] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8580–8589, 2018.

[123] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning, (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 2019.

[124] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 2019.

# A Deferred Proofs

## A.1 GSN is permutation equivariant

*Proof.* Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ the adjacency matrix of the graph, $\mathbf{H}_0 \in \mathbb{R}^{n\times d_{in}^V}$ the input vertex features, $\mathbf{E} \in \mathbb{R}^{n\times n\times d_{in}^E}$ the input edge features and $\mathbf{E}_i \in \mathbb{R}^{n\times n}$ the edge features at dimension $i$. Let $S^V(\mathbf{A}) \in \mathbb{N}^{n\times d_V}, S^E(\mathbf{A}) \in \mathbb{N}^{n\times n\times d_E}$ the functions generating the vertex and edge structural identifiers respectively.

It is obvious that subgraph isomorphism is invariant to the ordering of the vertices, i.e. we will always obtain the same matching between subgraphs $G_S$ and graphs $H$ in the subgraph collection. Thus, each vertex $v$ (edge $(v,u)$) in the graph will be assigned the same structural identifiers $\mathbf{x}_v^V$ ($\mathbf{x}_{v,u}^E$) regarless of the vertex ordering, and $S^V$ and $S^E$ are permutation equivariant, i.e., for any permutation matrix $\mathbf{P} \in \{0,1\}^{n\times n}$ it holds that

$$S^V(\mathbf{PAP}^T) = \mathbf{P}S^V(\mathbf{A})$$
$$S^E(\mathbf{PAP}^T) = \mathbf{P}[S_1^E(\mathbf{A}); \ldots; S_{d_E}^E(\mathbf{A})]\mathbf{P}^T,$$

where the permutation is applied at each slice $S_i^E(\mathbf{A}) \in \mathbb{N}^{n\times n}$ of the tensor $S^E(\mathbf{A})$.

Let $f(\mathbf{A},\mathbf{H},\mathbf{E}) \in \mathbb{R}^{n\times d_{out}}$ a GSN layer. We will show that $f$ is permutation equivariant. We need to show that if $\mathbf{Y} = f(\mathbf{A},\mathbf{H},\mathbf{E})$ the output of the GSN layer, then $\mathbf{PY} = f(\mathbf{PAP}^T, \mathbf{PH}, \mathbf{P}[\mathbf{E}_1;\ldots;\mathbf{E}_{d_E}]\mathbf{P}^T)$ for any permutation matrix $\mathbf{P}$. It is easy to see that GSN-v can can be expressed as a traditional MPNN $g(\mathbf{A},\mathbf{H},\mathbf{E}) \in \mathbb{R}^{n\times d_{out}}$ (similar to Eq. (5) of the main paper) by replacing the vertex features $\mathbf{H}$ with the concatenation of the input vertex features and the vertex structural identifiers, i.e. $\mathbf{Y} = f(\mathbf{A},\mathbf{H},\mathbf{E}) = g(\mathbf{A},[\mathbf{H};S^V(\mathbf{A})],\mathbf{E})$. Thus,

$$\begin{aligned}
&f(\mathbf{PAP}^T, \mathbf{PH}, \mathbf{P}[\mathbf{E}_1;\ldots;\mathbf{E}_{d_E}]\mathbf{P}^T,) = \\
&= g(\mathbf{PAP}^T, [\mathbf{PH}; S^V(\mathbf{PAP}^T)], \mathbf{P}[\mathbf{E}_1;\ldots;\mathbf{E}_{d_E}]\mathbf{P}^T) \\
&= g(\mathbf{PAP}^T, \mathbf{P}[\mathbf{H}; S^V(\mathbf{A})], \mathbf{P}[\mathbf{E}_1;\ldots;\mathbf{E}_{d_E}]\mathbf{P}^T) \\
&= \mathbf{P}g(\mathbf{A}, [\mathbf{H}; S^V(\mathbf{A})], \mathbf{E}) \\
&= \mathbf{PY}
\end{aligned}$$

where in the last step we used the permutation equivariant property of MPNNs. Similarly, we can show that a GSN-e layer is permutation equivariant, since it can be expressed as a traditional MPNN layer by replacing the edge features with the concatenation of the original edge features and the edge structural identifiers $f(\mathbf{A},\mathbf{H},\mathbf{E}) = g(\mathbf{A},\mathbf{H},[\mathbf{E};S^E(A)])$.

Overall, a GSN network is permutation equivariant as composition of permutation equivariant functions, or permutation invariant when composed with a permutation invariant layer at the end, i.e. the READOUT function.

□

## A.2 Proof of Theorem 3.1: GSN is at least as powerful as the 1-WL test

*Proof.* To show that GSN it is at least as expressive as the 1-WL test, we will repurpose the proof of Theorem 3 in [16] and demand the injectivity of the update function (w.r.t. both the hidden state $\mathbf{h}_v^t$ and the message $\mathbf{m}_v^{t+1}$), and the injectivity of the aggregation w.r.t. the multiset of the hidden states of the neighbours $\wr\mathbf{h}_u^t\int_{u\in\mathcal{N}(v)}$. It suffices then to show that if injectivity is preserved then GSNs are at least as powerful as the 1-WL.

We will show the above statement for vertex-labelled graphs, since traditionally the 1-WL test does not take into account edge labels.[4] We can rephrase the statement as follows: *If GSN deems two graphs $G_1$,*

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

*$G_2$ 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 vertex colours in its input (i.e. the graph-level representation is the vertex colour histogram at some iteration $t$), then it suffices to show that if for the two graphs the multiset of the vertex 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. $\wr \mathbf{h}_v^t \int_{v \in \mathcal{V}_{G_1}} = \wr \mathbf{h}_u^t \int_{u \in \mathcal{V}_{G_2}}$. Then both multisets contain the same distinct colour/hidden representations with the exact same multiplicity. Thus, it further suffices to show that if two vertices $v, u$ (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 $c_v^t = c_u^t$, extracted by 1-WL. Intuitively, this means that GSN creates a partition of the vertices of each graph that is at least as fine-grained as the one created by 1-WL. We prove by induction (similarly to [16]) 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 vertex features are the same for both GSN and 1-WL, i.e. $\mathbf{h}_v^0 = c_v^0, \quad \forall v \in \mathcal{V}_{G_1} \cup \mathcal{V}_{G_2}$. Suppose the statement holds for $t-1$, i.e. $\mathbf{h}_v^{t-1} = \mathbf{h}_u^{t-1} \Rightarrow c_v^{t-1} = c_u^{t-1}$. Then we show that it also holds for $t$. Every vertex hidden representation at step $t$ is updated as follows: $\mathbf{h}_v^t = \mathrm{UP}^t\big(\mathbf{h}_v^{t-1}, \mathbf{m}_v^t\big)$. Assuming that the update function $\mathrm{UP}^t$ is injective, we have the following: if $\mathbf{h}_v^t = \mathbf{h}_u^t$, then:

- $\mathbf{h}_v^{t-1} = \mathbf{h}_u^{t-1}$, which from the induction hypothesis implies that $c_v^{t-1} = c_u^{t-1}$.
- $\mathbf{m}_v^t = \mathbf{m}_u^t$, where the message function is defined as in Eq. (5) of the main paper. Additionally here we require $M^t$ to be injective w.r.t. the multiset of the hidden representations of the neighbours:[5]

$$\mathbf{m}_v^t = \mathbf{m}_u^t \Rightarrow \wr \mathbf{h}_w^{t-1} \int_{w \in \mathcal{N}_v} = \wr \mathbf{h}_z^{t-1} \int_{z \in \mathcal{N}_u}$$

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

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

□

## A.3 Proof of Corollary 3.2

*Proof.* In order to prove the universality of GSN, we will show that when the substructure collection contains all graphs of size $n-1$, then there exists a parametrisation of GSN that can infer the isomorphism classes of all vertex-deleted subgraphs of the graph $G$ (the *deck* of $G$).

The reconstruction conjectures states that two graphs with at least three vertices are isomorphic if and only if they have the same deck. Thus, the deck is sufficient to distinguish all non-isomorphic graphs. The deck can be defined as follows:

Let $\mathcal{H} = \{H_1, H_2, \cdots, H_K\}$ the set of all possible graphs of size $n-1$. The vertex-deleted subgraphs of $G$ are by definition all the induced subgraphs of $G$ with size $n-1$, which we denote as

$$\mathcal{G}_{n-1} = \{G_S \text{: induced subgraph of } G \text{ with } |\mathcal{V}_{G_S}| = n-1\}.$$

Then, the deck $D(G)$ can be defined as a vector of size $|\mathcal{H}|$, where at the $j$-th dimension

$$\begin{aligned} D_j(G) &= \left| \left\{ G_S \in \mathcal{G}_{n-1} \mid G_S \simeq H_j \right\} \right| \\ &= \sum_{G_S \in \mathcal{G}_{n-1}} \mathbb{1}[G_S \simeq H_j], \end{aligned}$$

[5] Lemma 5 from [16] states that such a function always exists assuming that the elements of the multiset originate from a countable domain

where $\mathbb{1}[\cdot]$ the indicator function. The structural feature $x^V_{H_j,i}(v)$ for each substructure $H_j$ and orbit $i$ are computed as follows:

$$\begin{aligned} x^V_{H_j,i}(v) &= \left| \left\{ G_S \simeq H_j \mid v \in \mathcal{V}_{G_S},\ f(v) \in O^V_{H_j,i} \right\} \right| \\ &= \sum_{G_S \in \mathcal{G}_{n-1}} \mathbb{1}[v \in \mathcal{V}_{G_S}] \mathbb{1}[G_S \simeq H_j] \mathbb{1}[f_{G_S}(v) \in O^V_{H_j,i}] \end{aligned}$$

where $f_{G_S}(\cdot) = \emptyset$ if $G_S \not\simeq H_j$, otherwise it is the bijective mapping function. The deck can be inferred as follows:

$$\begin{aligned} \sum_{v \in \mathcal{V}_G} \sum_{i=1}^{d_{H_j}} x^V_{H_j,i}(v) &= \sum_{v \in \mathcal{V}_G} \sum_{i=1}^{d_{H_j}} \sum_{G_S \in \mathcal{G}_{n-1}} \mathbb{1}[v \in \mathcal{V}_{G_S}] \mathbb{1}[G_S \simeq H_j] \mathbb{1}[f_{G_S}(v) \in O^V_{H_j,i}] \\ &= \sum_{v \in \mathcal{V}_G} \sum_{G_S \in \mathcal{G}_{n-1}} \mathbb{1}[v \in \mathcal{V}_{G_S}] \mathbb{1}[G_S \simeq H_j] \sum_{i=1}^{d_{H_j}} \mathbb{1}[f_{G_S}(v) \in O^V_{H_j,i}] \\ &= \sum_{G_S \in \mathcal{G}_{n-1}} \mathbb{1}[G_S \simeq H_j] \sum_{v \in \mathcal{V}_G} \mathbb{1}[v \in \mathcal{V}_{G_S}] \\ &= \sum_{G_S \in \mathcal{G}_{n-1}} (n-1) \mathbb{1}[G_S \simeq H_j] \\ &= (n-1) D_j(G) \end{aligned}$$

where we used that $\sum_{i=1}^{d_{H_j}} \mathbb{1}[f_{G_S}(v) \in O^V_{H_j,i}] = 1$, since each vertex can be mapped to a single orbit only. Thus, the deck can be inferred by a simple GSN-v parametrisation (a linear layer with depth equal to $|\mathcal{H}|$ that performs orbit-wise summation and division by the constant $n-1$ for each vertex separately, followed by a sum readout). Since GSN-v can be inferred by GSN-e (Theorem 3.3), then GSN-e is also universal. □

## A.4 Proof of Theorem 3.3

*Proof.* Without loss of generality we will show Theorem 3.3 for the case of a single substructure $H$. In order to show that GSN-e can express GSN-v, we will first prove the following: *the vertex identifier of a vertex $v$ can be inferred by the edge identifiers of its incident edges.*

To simplify notation we define the following: For an orbit $\mathrm{Orb}^V(v)$, denote the *orbit neighbourhood* as the multiset of the orbits of the neighbours of $v$ and the *orbit degree* as the degree of any vertex in $\mathrm{Orb}^V(v)$ :

$$\mathcal{N}\big(\mathrm{Orb}^V(v)\big) = \wr \mathrm{Orb}^V(u) \mid u \in \mathcal{N}(v) \wr \ \text{ and } \ \deg(v) = \deg(\mathrm{Orb}^V(v)) = |\mathcal{N}\big(\mathrm{Orb}^V(v)\big)|.$$

For brevity we will use the following notation: vertex orbits are indexed as follows $O^V_1, O^V_2, \dots O^V_{d_V}$ and edge orbits $O^E_{1,1}, O^E_{1,2}, \dots O^E_{d_V,d_V}$ with $O^E_{i,j} = \big(O^V_i, O^V_j\big)$. Structural features are denoted accordingly: $x^V_i(v)$ and $x^E_{ij}(v,u)$

Let's assume that there exists only one matched subgraph $G_S \simeq H$ and the bijection between $\mathcal{V}_{G_S}$ and $\mathcal{H}$ is denoted as $f$. Then, for an abitrary vertex $v$ and a vertex orbit $O^V_i$, one of the following holds:

- $v \notin \mathcal{V}_{G_S}$. Then $x^V_i(v) = 0$ and $x^E_{ij}(v,u) = 0,\ \forall u \in \mathcal{N}(v)$,
- $v \in \mathcal{V}_{G_S}$ and $\mathrm{Orb}(f(v)) \neq O^V_i$. Then, $x^V_i(v) = 0$ and $x^E_{ij}(v,u) = 0,\ \forall u \in \mathcal{N}(v)$. Note that here the directionality of the edge is important, otherwise we cannot determine the value of $x^E_{ij}(v,u)$ unless we also know the orbit of $f(u)$.

- $v \in \mathcal{V}_{G_S}$ and $\text{Orb}(f(v)) = O_i^V$. Then, $x_i^V(v) = 1$ and since $f(v)$ has exactly $\deg(O_i^V)$ neighbours in $H$, then $v$ has exactly $\deg(O_i^V)$ neighbours in $G_S$ with vertex orbits $\mathcal{N}\big(O_i^V\big)$. In other words

$$\sum_{u \in \mathcal{N}(v)} \sum_{j: O_j^V \in \mathcal{N}(O_i^V)} x_{ij}^E(u,v) = \deg(O_i^V)$$

Thus, by induction, for $m$ matched subgraphs $G_S \simeq H$ with $v \in \mathcal{V}_{G_S}$ and $\text{Orb}(f(v)) = O_i^V$, it holds that $x_i^V(v) = m$ and $\sum_{u \in \mathcal{N}(v)} \sum_{j: O_j^V \in \mathcal{N}(O_i^V)} x_{ij}^E(u,v) = m * \deg(O_i^V)$. Then it follows that:

$$x_i^V(v) = \frac{1}{\deg(O_i^V)} \sum_{u \in \mathcal{N}(v)} \sum_{j: O_j^V \in \mathcal{N}(O_i^V)} x_{ij}^E(u,v) \tag{6}$$

The rest of the proof is straightforward: we will assume a GSN-v using substructure counts of the graph $H$, with $L$ layers and width $w$ defined as in the main paper (Eq. (5)) Then, there exists a GSN-e with $L+1$ layers, where the first layer has width $d_V^{in} + d_V$ and implements the following function: $\text{UP}^{t+1}\big(\mathbf{h}_v^t, \mathbf{m}_v^{t+1}\big) = [\mathbf{h}_v^t; \mathbf{m}_v^{t+1}]$, where:

$$\begin{aligned}
\mathbf{m}_v^{t+1} &= M^{t+1}\Big(\wr(\mathbf{h}_v^t, \mathbf{h}_u^t, \mathbf{x}_{v,u}^E, \mathbf{e}_{u,v})\wr_{u \in \mathcal{N}(v)}\Big) \\
&= [\frac{1}{\deg(O_1^V)} \sum_{u \in \mathcal{N}(v)} \sum_{j: O_j^V \in \mathcal{N}(O_1^V)} x_{1j}^E(u,v);\ \ldots\ ; \\
&\quad \frac{1}{\deg(O_{d_V}^V)} \sum_{u \in \mathcal{N}(v)} \sum_{j: O_j^V \in \mathcal{N}(O_{d_V}^V)} x_{d_V,j}^E(u,v)] \\
&= \mathbf{x}_v^V
\end{aligned}$$

Note that since $M$ is a universal multiset function approximator, then there exist parameters of $M$ with which the above function can be computed. The next $L$ layers of GSN-e can implement a traditional MPNN where now the input vertex features are $[\mathbf{h}_v^t; \mathbf{x}_v^V]$ (which is exactly the formulation of GSN-v) and this concludes the proof.

□

# B Comparison with higher-order Weisfeiler-Leman tests

## B.1 The WL hierarchy

Following the terminology introduced in [24], we describe the so-called *Folklore WL family (k-FWL)*. Note that, in the majority of papers on GNN expressivity [17, 24, 19] 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 [116, 71, 18] 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.

The $k$-FWL operates on $k$-tuples of vertices $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ to which an initial colour $c_{\mathbf{v}}^0$ is assigned based on their *isomorphism types* (see section B.2), which can loosely be thought of as a generalisation of isomorphism that also preserves the ordering of the vertices in the tuple. Then, at each iteration the colour is refined as follows:

$$c_{\mathbf{v}}^{t+1} = \text{HASH}\Big(c_{\mathbf{v}}^t, \wr\big(c_{\mathbf{v}_{u,1}}^t, c_{\mathbf{v}_{u,2}}^t, \ldots, c_{\mathbf{v}_{u,k}}^t\big)\wr_{u \in \mathcal{V}}\Big), \tag{7}$$

where $\mathbf{v}_{u,j} = (v_1, v_2, \ldots, v_{j-1}, u, v_{j+1}, \ldots, v_k)$.

The multiset $\wr\big(c_{\mathbf{v}_{u,1}}^t, c_{\mathbf{v}_{u,2}}^t, \ldots, c_{\mathbf{v}_{u,k}}^t\big)\wr_{u \in \mathcal{V}}$ can be perceived as a form of generalised neighbourhood. Observe that all possible tuples in the graph store information necessary for the updates, thus each $k$-tuple receives information *from the entire graph*, contrary to the *local* nature of the 1-WL test.

## B.2 Why does 2-FWL fail on strongly regular graphs?

*Proof.* We first formally define what an isomorphism type is and what are the properties of the SR family:

**Definition 1** (Isomorphism Types)**.** Two k-tuples $\mathbf{v}^a = \{v_1^a, v_2^a, \dots, v_k^a\}$, $\mathbf{v}^b = \{v_1^b, v_2^b, \dots, v_k^b\}$ will have the same isomorphism type iff:

- $\forall\, i, j \in \{0, 1, \dots, k\}, \quad v_i^a = v_j^a \Leftrightarrow v_i^b = v_j^b$
- $\forall\, i, j \in \{0, 1, \dots, k\}, \quad v_i^a \sim v_j^a \Leftrightarrow v_i^b \sim v_j^b$, 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, these need to be preserved as well (see [19]) for the extended case).

**Definition 2** (Strongly regular graph)**.** A *SR(n,d,λ,μ)-graph* is a regular graph with $n$ vertices and degree $d$, where every two adjacent vertices have always $\lambda$ mutual neighbours, while every two non-adjacent vertices have always $\mu$ mutual neighbours.

Now we can proceed to the details of the proof. 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 colour $c^0 = c_\alpha$
- $\mathbf{v} = \{v_1, v_2\}$ and $v_1 \not\sim v_2$: *non-edge type*. Mapped to the colour $c^0 = c_\beta$
- $\mathbf{v} = \{v_1, v_2\}$ and $v_1 \sim v_2$: *edge type*. Mapped to the colour $c^0 = c_\gamma$

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($n$,$d$,$\lambda$,$\mu$). 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 neighbour colour tuples are:
  - $(c_\alpha, c_\alpha)$ if $v_1 = u$,
  - $(c_\beta, c_\beta)$ if $v_1 \not\sim u$ ,
  - $(c_\gamma, c_\gamma)$ if $v_1 \sim u$.

  The update of the 2-FWL is: $c^1_{\mathbf{v}} = \text{HASH}\Big(c_\alpha,\ \{\!\{\underbrace{(c_\alpha, c_\alpha)}_{1 \text{ time}}, \underbrace{(c_\beta, c_\beta)}_{n-1-d \text{ times}}, \underbrace{(c_\gamma, c_\gamma)}_{d \text{ times}}\}\!\}\Big)$ 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 neighbour colour tuples are:
  - $(c_\alpha, c_\beta)$ if $v_2 = u$,
  - $(c_\beta, c_\alpha)$ if $v_1 = u$,
  - $(c_\gamma, c_\beta)$ if $v_2 \sim u$ and $v_1 \not\sim u$,
  - $(c_\beta, c_\gamma)$ if $v_1 \sim u$ and $v_2 \not\sim u$,
  - $(c_\beta, c_\beta)$ if $v_1 \not\sim u$ and $v_2 \not\sim u$,
  - $(c_\gamma, c_\gamma)$ if $v_1 \sim u$ and $v_2 \sim u$.

The update of the 2-FWL is:

$c^1_{\mathbf{v}} = \text{HASH}\Big(c_\beta,\ \wr(\underbrace{(c_\alpha, c_\beta)}_{1 \text{ time}}, \underbrace{(c_\beta, c_\alpha)}_{1 \text{ time}}, \underbrace{(c_\gamma, c_\beta)}_{d-\mu \text{ times}}, \underbrace{(c_\beta, c_\gamma)}_{d-\mu \text{ times}}, \underbrace{(c_\beta, c_\beta)}_{n-2-(2d-\mu) \text{ times}}, \underbrace{(c_\gamma, c_\gamma)}_{\mu \text{ times}})\int\Big)$ same for all *non-edge type* 2-tuples.

- generalised neighbour of an *edge type* tuple:
  - $(c_\alpha, c_\gamma)$ if $v_2 = u$,
  - $(c_\gamma, c_\alpha)$ if $v_1 = u$,
  - $(c_\gamma, c_\beta)$ if $v_2 \sim u$ and $v_1 \not\sim u$,
  - $(c_\beta, c_\gamma)$ if $v_1 \sim u$ and $v_2 \not\sim u$,
  - $(c_\beta, c_\beta)$ if $v_1 \not\sim u$ and $v_2 \not\sim u$,
  - $(c_\gamma, c_\gamma)$ if $v_1 \sim u$ and $v_2 \sim u$.

  The update of the 2-FWL is:

  $c^1_{\mathbf{v}} = \text{HASH}\Big(c_\gamma,\ \wr(\underbrace{(c_\alpha, c_\gamma)}_{1 \text{ time}}, \underbrace{(c_\gamma, c_\alpha)}_{1 \text{ time}}, \underbrace{(c_\gamma, c_\beta)}_{d-\lambda \text{ times}}, \underbrace{(c_\beta, c_\gamma)}_{d-\lambda \text{ times}}, \underbrace{(c_\beta, c_\beta)}_{n-2-(2d-\lambda) \text{ times}}, \underbrace{(c_\gamma, c_\gamma)}_{\lambda \text{ times}})\int\Big)$ 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 further partitioned, so the algorithm terminates. Therefore, if two SR graphs have the same parameters $n$,$d$,$\lambda$,$\mu$ then 2-FWL will yield the same colour distribution and thus the graphs will be deemed isomorphic.

□

# C Experimental Settings - Additional Details

In this section, we provide additional implementation details of our experiments. All experiments were performed on a server equipped with 8 Tesla V100 16 GB GPUs, except for the Collab dataset where a Tesla V100 GPU with 32 GB RAM was used due to larger memory requirements. Experimental tracking and hyperparameter optimisation were done via the Weights & Biases platform (wandb) [117]. Our implementation is based on native PyTorch sparse operations [118] in order to ensure complete reproducibility of the results. PyTorch Geometric [119] was used for additional operations (such as preprocessing and data loading).

In each one of the different experiments we aim to show that **structural identifiers can be used off-the-shelf and independently of the architecture.** At the same time we aim to suppress the effect of other confounding factors in the model performance, thus wherever possible we build our model on top of a baseline architecture. For more details, please see the relevant subsections. Interestingly, we observed that in most of the cases it was sufficient to replace only the first layer of the baseline architecture with a GSN layer, in order to obtain a boost in performance.

Throughout the experimental evaluation the structural identifiers $\mathbf{x}^V_v$ and $\mathbf{x}^E_{u,v}$ 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 a normalisation scheme or binning. However, we found that the number of unique values in our datasets were usually relatively small (which is a good indication of recurrent structural roles) and thus such methods were not necessary.

## C.1 Synthetic Experiment

For the Strongly Regular graphs dataset (available from `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`) we use all the available families of graphs with size of at most 35 vertices:

- 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. (5) of the main paper, where the update and the message functions are MLPs. The prediction is inferred by applying a sum readout function in the last layer 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 [71, 18]).

Given the adversities that strongly regular graphs pose in graph isomorphism testing, it would be interesting to see how this method can perform in other categories of hard instances, such as the classical *CFI* counter-examples for k-WL proposed in [116], and explore further its expressive power and combinatorial properties. We leave this direction to future work.

## C.2 TUD Graph Classification Benchmarks

For this family of experiments, due to the usually small size of the datasets, we choose a parameter-efficient architecture, in order to reduce the risk of overfitting. In particular, we follow the simple GIN architecture [16] and we concatenate structural identifiers to vertex or edge features depending on the variant. Then for GSN-v, the hidden representation is updated as follows:

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

and for GSN-e:

$$\mathbf{h}_v^{t+1} = \mathrm{UP}^{t+1}\Big([\mathbf{h}_v^t; \mathbf{x}_{v,v}^E] + \sum_{u \in \mathcal{N}_v} [\mathbf{h}_u^t; \mathbf{x}_{u,v}^E]\Big), \tag{9}$$

where $\mathbf{x}_{v,v}^E$ is a dummy variable (also one-hot encoded) used to distinguish self-loops from edges. Empirically, we did not find training the $\epsilon$ parameter used in GIN to make a difference.

We implement an architecture similar to GIN [16], i.e. *message passing layers*: 4 , *jumping knowledge* from all the layers [120] (including the input), *transformation of each intermediate graph-level representation*: linear layer, *readout*: sum for biological and mean for social networks. Vertex features are one-hot encodings of the categorical vertex 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 vertex 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 substructure collection: i.e. *the maximum size k* and *motifs vs graphlets*. In

Table 5: Chosen hyperparameters for each of 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 |

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* [121]). For social networks, we searched cliques with $k = 3, 4, 5$. In Table 5 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) [16], a simple, yet powerful GNN with expressive power equal to the 1-WL test, and the Provably Powerful Graph Network (PPGN) [24], a polynomial alternative to the Invariant Graph Network [22], 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) [94], which is a kernel obtained from a GNN of infinite width. This operates in the Neural Tangent Kernel regime [122, 123, 124].

## C.3 Graph Regression on ZINC

**Experimental Details:** The ZINC dataset includes 12k molecular graphs of which 10k form the training set and the remaining 2k are equally split between validation and test (splits obtained from `https://github.com/graphdeeplearning/benchmarking-gnns`). Molecule sizes range from 9 to 37 vertices/atoms. Vertex features encode the type of atoms and edge features the chemical bonds between them. Again, here vertex and edge features are one-hot encoded.

Our MPNN baseline model updates vertex representations as follows: $\mathbf{h}^{t+1}(v) = \text{MLP}^{t+1}(\mathbf{h}_v^t, \mathbf{m}_v^{t+1})$, $\mathbf{m}_v^{t+1} = \sum_{u \in \mathcal{N}(v)} \text{MLP}^t\big(\mathbf{h}_v^t, \mathbf{h}_u^t, \mathbf{e}_{u,v}\big)$. Our instantiation of GSN is a simple extension where structural identifiers are also given as input to the message MLP.

Following the same rationale as before, the network configuration is minimally modified w.r.t. the baselines provided in [25], while here no hyperparameter tuning is done and we use the default ones provided by the authors. In particular, the parameters are the following: *message passing layers*: 4, *transformation of the output of the last layer*: MLP, *readout*: sum, *batch size*: 128, *dropout*: 0.0, *network width*: 128, *learning rate*: 0.001. The learning rate is reduced by 0.5 (decay rate) after 5 epochs (*decay rate patience*) without

[6] more details on the description of the datasets and the corresponding tasks can be found at [16].

improvement in the validation loss. Training is stopped when the learning rate reaches the *minimum learning rate* value of $10^{-5}$. Validation and test metrics are inferred using the model at the last training epoch.

We select our best performing substructure related parameters based on the performance in the validation set in the last epoch. We search cycles with $k = 3, \ldots, 10$, *graphlets vs motifs*, and *GSN-v vs GSN-e*. The chosen hyperparameters for GSN are: *GSN-e, cycle graphlets of 10 vertices* and for GSN-*EF*: *GSN-v, cycle motifs of 8 vertices*. Once the model is chosen, we repeat the experiment 10 times with different seeds and report the mean and standard deviation of the test MAE in the last epoch.

**Disambiguation Scores:** In Table 6, we provide the disambiguation scores $\delta$ as defined in section 5.5.1 of the main paper for different types of substructures. These are computed based on vertex structural identifiers (GSN-v).

Table 6: Disambiguation scores $\delta$ on *ZINC* for different substructure families and their maximum size $k$. Size $k = 0$ refers to the use of the original vertex features only.

| k | Cycles | Paths | Trees |
|---|---|---|---|
| 0 | 0.196 | 0.196 | 0.196 |
| 3 | 0.199 | 0.540 | 0.540 |
| 4 | 0.200 | 0.746 | 0.762 |
| 5 | 0.256 | 0.866 | 0.875 |
| 6 | 0.327 | 0.895 | 0.897 |
| 7 | 0.330 | 0.900 | 0.900 |
| 8 | 0.330 | 0.901 | 0.901 |
| 9 | 0.330 | 0.901 | 0.901 |
| 10 | 0.330 | 0.901 | 0.901 |

## C.4 Graph Classification on `ogbg-molhiv`

The `ogbg-molhiv` dataset contains $\approx$ 41K graphs, with 25.5 vertices and 27.5 edges on average. As most molecular graphs, the average degree is small (2.2) and they exhibit a tree-like structure (average clustering coefficient 0.002). The average diameter is 12 (more details in [26]). Below we describe how we extend the two base architectures:

**GIN+VN.** We follow the design choices of the authors of [26] and extend their architectures to include structural identifiers. Initial vertex and edge features are multi-hot encodings passed through linear layers that project them in the same embedding space, i.e. $\mathbf{h}_v^0 = \mathbf{W}_h^0 \cdot \mathbf{h}_v^{in}$, $\mathbf{e}_{v,u}^t = \mathbf{W}_e^t \cdot \mathbf{e}_{u,v}^{in}$. The baseline model is a modification of GIN that allows for edge features: for each neighbour, the hidden representation is added to an embedding of its associated edge feature. Then the result is passed through a ReLU non-linearity which produces the neighbour's message. Formally, the aggregation is as follows:

$$\mathbf{h}_v^{t+1} = \mathrm{UP}^{t+1}\left(\mathbf{h}_v^t + \sum_{u \in \mathcal{N}_v} \sigma\left(\mathbf{h}_u^t + \mathbf{e}_{v,u}^t\right)\right) \tag{10}$$

In order to allow global information to be broadcasted to the vertices, a *virtual node* takes part in the message passing. The virtual node representation, denoted as $\mathbf{G}^t$, is initialised as a zero vector $\mathbf{G}^0$ and then Message Passing becomes:

$$\begin{aligned} &\tilde{\mathbf{h}}_v^t = \mathbf{h}_v^t + \mathbf{G}^t,\ \mathbf{h}_v^{t+1} = \mathrm{UP}^{t+1}\left(\tilde{\mathbf{h}}_v^t + \sum_{u \in \mathcal{N}_v} \sigma\left(\tilde{\mathbf{h}}_u^t + \mathbf{e}_{v,u}^t\right)\right), \\ &\mathbf{G}^{t+1} = MLP^{t+1}\left(\mathbf{G}^t + \sum_{u \in \mathcal{N}_v} \tilde{\mathbf{h}}_u^t\right) \end{aligned} \tag{11}$$

We modify this model, as follows: first the substructure counts are embedded into the same embedding space as the rest of the features. Then, for GSN-v, they are added to the corresponding vertex embeddings: $\acute{\mathbf{h}}_v^t = \mathbf{h}_v^t + \mathbf{W}_V^t \cdot \mathbf{x}_v^V$, or for GSN-e, they are added to the edge embeddings $\acute{\mathbf{e}}_{v,u}^t = \mathbf{e}_{v,u}^t + \mathbf{W}_E^t \cdot \mathbf{x}_{u,v}^E$.

**DGN + substructures.** We use the directional average operator as defined in [68]:

$$\mathbf{m}_v^{t+1} = [\sum_{u \in \mathcal{N}(v)} \alpha_{v,u}^1 \mathbf{h}_u^t; \ldots; \sum_{u \in \mathcal{N}(v)} \alpha_{v,u}^D \mathbf{h}_u^t], \tag{12}$$

where $\alpha_{v,u}^i$ are weighting average coefficients. In our case, each orbit $i$ induces a separate set of averaging coefficients. For example, for GSN-e $\alpha_{v,u} = \frac{|x_{v,u}^E|}{\epsilon + \sum_{u \in \mathcal{N}(v)} |x_{v,u}^E|}$, where $x_{v,u}^E$ denotes edge-wise substructure counts (the index of the orbit $i$ was dropped to simplify notation). Similarly, for GSN-v, $\alpha_{v,u} = \frac{|x_v^V - x_u^V|}{\epsilon + \sum_{u \in \mathcal{N}(v)} |x_v^V - x_u^V|}$. Subsequently, the vertex representation is updated as follows: $\mathbf{h}_v^{t+1} = \text{MLP}^{t+1}(\mathbf{m}_v^{t+1})$. Observe that this model is simpler than the aforementioned, in terms of both its parameter count and its expressive power. Since the MOLHIV dataset poses a significant challenge w.r.t. generalisation (the data splits reflect different molecular distributions), architectures biased towards simpler solutions usually perform better, sincey the mitigate the risk of overfitting.

In both cases we use the the same hyperparameters as the ones provided by the authors, and only select the substructure related parameters based on the highest validation ROC-AUC (choosing the best scoring epoch as in [26]). We search cycles with $k = 3, \ldots, 12$, *graphlets vs motifs*, and *GSN-v vs GSN-e*. The chosen hyperparameters are: *GSN-e, cycle graphlets of 6 vertices*. We repeat the experiment 10 times with different seeds and report the mean and standard deviation of the train, validation and test ROC-AUC, again by choosing the best scoring epoch w.r.t the validation set.

## C.5 Structural Features & Message Passing

The baseline architecture treats the input vertex and edge features, along with the structural identifiers, as a *set*. In particular, we consider each graph as a set of independent edges $(v, u)$ endowed with the features of the endpoint vertices $\mathbf{h}_v, \mathbf{h}_u$, the structural identifiers $\mathbf{x}_v^V, \mathbf{x}_u^V$ and the edge features $\mathbf{e}(v, u)$, and we implement a DeepSets universal set function approximator [115] to learn a prediction function:

$$f\Big(\Big\{\big(\mathbf{h}_v, \mathbf{h}_u, \mathbf{x}_v^V, \mathbf{x}_u^V, \mathbf{e}_{v,u}\big)\Big\}_{(v,u) \in \mathcal{E}_G}\Big) = \rho\Big(\sum_{(v,u) \in \mathcal{E}_G} \phi\big(\mathbf{h}_v, \mathbf{h}_u \mathbf{x}_v^V, \mathbf{x}_u^V, \mathbf{e}_{v,u}\big)\Big), \tag{13}$$

with $\mathcal{E}_G$ the edge set of the graph and $\rho, \phi$ MLPs. This baseline is naturally extended to the case where we consider edge structural identifiers by replacing $(\mathbf{x}_v^V, \mathbf{x}_u^V)$ with $\mathbf{x}_{v,u}^E$. For fairness of evaluation, we follow the exact same parameter tuning procedure as the one we followed for our GSN models for each benchmark, i.e. for the TUD datasets we first tune network and optimisation hyperaparameters (network width was set to be either equal to the ones we tuned for GSN, or such that the absolute number of learnable parameters was equal to those used by GSN; depth of the MLPs was set to 2) and subsequently we choose the substructure related parameters based on the evaluation protocol of [16]. For ZINC and ogbg-molhiv we perform only substructure selection, based on the performance on the validation set. Using the same widths as in GSN leads to smaller baseline models w.r.t the absolute number of parameters, and we interestingly observed this to lead to particularly strong performance in some cases, especially Proteins and MUTAG, where our DeepSets implementation attains state-of-art results. This finding motivated us to explore 'smaller' GSNs (with either reduced layer width or a single message passing layer). These GSN variants exhibited a similar trend, i.e. to perform better than their 'larger' counterparts over these two datasets. We hypothesise this phenomenon to be mostly due to the small size of these datasets, which encourages overfitting when using architectures with larger capacity. In Table 4 in the main paper, we report the result for the best performing architectures, along with the number of learnable parameters.