Ordered Subgraph Aggregation Networks

Chendi Qian[*]
Department of Computer Science
TU Munich

Gaurav Rattan[*]
Department of Computer Science
RWTH Aachen University

Floris Geerts
Department of Computer Science
University of Antwerp

Christopher Morris
Department of Computer Science
RWTH Aachen University

Mathias Niepert
Department of Computer Science
University of Stuttgart

Abstract

Numerous subgraph-enhanced graph neural networks (GNNs) have emerged recently, provably boosting the expressive power of standard (message-passing) GNNs. However, there is a limited understanding of how these approaches relate to each other and to the Weisfeiler–Leman hierarchy. Moreover, current approaches either use all subgraphs of a given size, sample them uniformly at random, or use hand-crafted heuristics instead of learning to select subgraphs in a data-driven manner. Here, we offer a unified way to study such architectures by introducing a theoretical framework and extending the known expressivity results of subgraph-enhanced GNNs. Concretely, we show that increasing subgraph size always increases the expressive power and develop a better understanding of their limitations by relating them to the established k-WL hierarchy. In addition, we explore different approaches for learning to sample subgraphs using recent methods for backpropagating through complex discrete probability distributions. Empirically, we study the predictive performance of different subgraph-enhanced GNNs, showing that our data-driven architectures increase prediction accuracy on standard benchmark datasets compared to non-data-driven subgraph-enhanced graph neural networks while reducing computation time.

1 Introduction

Graph-structured data are ubiquitous across application domains ranging from chemo- and bioinformatics [Barabasi and Oltvai, 2004, Jumper et al., 2021, Stokes et al., 2020] to image [Simonovsky and Komodakis, 2017] and social-network analysis [Easley and Kleinberg, 2010]. Numerous approaches for graph–based machine learning have been proposed, most notably those based on *graph kernels* [Borgwardt et al., 2020, Kriege et al., 2020] or using *graph neural networks* (GNNs) [Chami et al., 2020, Gilmer et al., 2017, Morris et al., 2021]. Here, graph kernels based on the 1*-dimensional Weisfeiler–Leman algorithm* (1-WL) [Weisfeiler and Leman, 1968], a simple heuristic for the graph isomorphism problem, and corresponding GNNs [Morris et al., 2019, Xu et al., 2019], have recently advanced the state-of-the-art in supervised vertex- and graph-level learning. However, the 1-WL and GNNs operate via local neighborhood aggregation, missing crucial patterns in the given data while more expressive architectures based on the k*-dimensional Weisfeiler–Leman algorithm* (k-WL) [Azizian and Lelarge, 2020, Maron et al., 2019, Morris et al., 2020b, 2021, 2022] may not scale to larger graphs.

[*] These authors contributed equally.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

https://doi.org/10.52202/068431-1529

Hence, several approaches such as Bevilacqua et al. [2021], Cotta et al. [2021], Li et al. [2020], Papp et al. [2021], Thiede et al. [2021], You et al. [2021] and Zhao et al. [2021] have enhanced the expressive power of GNNs, by removing, extracting, or marking (small) subgraphs, so as to allow GNNs to leverage more structural patterns within the given graph, essentially breaking symmetries induced by the GNNs' local aggregation function. We henceforth refer to these approaches as *subgraph-enhanced GNNs*.

Present work First, to bring some order to the multitude of recently proposed subgraph-enhanced GNNs, we introduce a theoretical framework to study these approaches' expressive power in a unified setting. Concretely,

- we introduce *k-ordered subgraph aggregation networks* (k-OSANs) and show that they capture most of the recently proposed subgraph-enhanced GNNs.
- We show that any k-OSAN is upper bounded by $(k+1)$-WL in terms of expressive power and show that k-OSANs and k-WL are incomparable in terms of expressive power. Consequently, we obtain new upper bounds on the expressive power of recently proposed subgraph-enhanced GNNs.
- We show that increasing k, i.e., using larger subgraphs, always leads to an increase in expressive power, effectively showing that k-OSANs form a hierarchy.

Second, most approaches consider all subgraphs or use hand-crafted heuristics to select them, e.g., by deleting vertices or edges. Instead, we leverage recent progress in back-propagating through discrete structures using perturbation-based differentiation [Domke, 2010, Niepert et al., 2021] to sample subgraphs in a *data-driven* fashion, automatically adapting to the given data distribution. Concretely,

- we explore different strategies to sample subgraphs leveraging the I-MLE framework [Niepert et al., 2021], resulting in the data-driven k-OSAN architecture.
- We show, empirically, that data-driven k-OSANs increase prediction accuracy on standard benchmark datasets compared to non-data-driven subgraph-enhanced GNNs while vastly reducing computation time.

1.1 Related work

In the following, we discuss related work relevant to the present work; see Appendix A for an extended discussion.

GNNs Recently, GNNs [Gilmer et al., 2017, Scarselli et al., 2009] emerged as the most prominent graph representation learning architecture. Notable instances of this architecture include, e.g., Duvenaud et al. [2015], Hamilton et al. [2017] and Veličković et al. [2018], which can be subsumed under the message-passing framework introduced in Gilmer et al. [2017]. In parallel, approaches based on spectral information were introduced in, e.g., Defferrard et al. [2016], Bruna et al. [2014], Kipf and Welling [2017] and Monti et al. [2017]—all of which descend from early work in Baskin et al. [1997], Kireev [1995], Micheli and Sestito [2005], Merkwirth and Lengauer [2005], Micheli [2009], Scarselli et al. [2009] and Sperduti and Starita [1997].

Limits of GNNs and more expressive architectures Recently, connections between GNNs and Weisfeiler–Leman type algorithms have been shown [Azizian and Lelarge, 2020, Barceló et al., 2020, Chen et al., 2019b, Geerts et al., 2020, Geerts, 2020, Geerts and Reutter, 2022, Maehara and NT, 2019, Maron et al., 2019, Morris et al., 2019, 2022, Xu et al., 2019]. Specifically, Morris et al. [2019] and Xu et al. [2019] showed that the expressive power of any possible GNN architecture is limited by the 1-WL in terms of distinguishing non-isomorphic graphs.

Recent works have extended the expressive power of GNNs, e.g., by encoding vertex identifiers [Murphy et al., 2019, Vignac et al., 2020], using random features [Abboud et al., 2020, Dasoulas et al., 2020, Sato et al., 2020], homomorphism and subgraph counts [Barceló et al., 2021, Bouritsas et al., 2020, NT and Maehara, 2020], spectral information [Balcilar et al., 2021], simplicial and cellular complexes [Bodnar et al., 2021b,a], persistent homology [Horn et al., 2021], random walks [Tönshoff et al., 2021], graph decompositions [Talak et al., 2021], or distance [Li et al., 2020] and directional information [Beaini et al., 2020]. See Morris et al. [2021] for an in-depth survey on this topic.

Subgraph-enhanced GNNs Most relevant to the present work are *subgraph-enhanced GNNs*. Cotta et al. [2021] and Papp et al. [2021] showed how to make GNNs more expressive by removing one or more vertices from a given graph and using standard GNN architectures to learn vectorial representations of the

https://doi.org/10.52202/068431-1529

resulting subgraphs. The approaches either consider all possible subgraphs or utilize random sampling to arrive at more scalable architectures. Cotta et al. [2021] showed that by removing one or two vertices, such architectures can distinguish graphs the 1-WL and 2-WL, respectively, are not able to distinguish. Extensions and refinements of the above were proposed in Bevilacqua et al. [2021], Papp and Wattenhofer [2022], Thiede et al. [2021], You et al. [2021], Zhang and Li [2021] and Zhao et al. [2021], see Papp and Wattenhofer [2022] for an overview. For example, Bevilacqua et al. [2021] generalized several ideas discussed above and proposed the ESAN framework in which each graph is represented as a multiset of its subgraphs and processed them using an equivariant architecture based on the DSS architecture [Maron et al., 2020] and GNNs. The authors proposed several simple subgraph selection policies, e.g., edge removal, ego networks, or vertex removal, and showed that the architecture surpasses the expressive power of the 1-WL. Moreover, Frasca et al. [2022] presented a novel symmetry analysis unifying a series of subgraph-enhanced GNNs, allowing them to upper-bound their expressive power and to define a systematic framework to conceive novel architectures in this family. We note here that the above works, unlike the present one, mostly do not study the approaches' expressive power beyond (folklore or non-oblivious) 2-WL and do not compare at all to the (folklore or non-oblivious) 3-WL, while our analysis works for the whole k-WL hierarchy.

See Appendix A for a detailed overview of recent progress in differentiating through discrete structures.

2 Preliminaries

As usual, for $n \geq 1$, let $[n] \coloneqq \{1, \dots, n\} \subset \mathbb{N}$. We use $\{\!\!\{ \dots \}\!\!\}$ to denote multisets, i.e., the generalization of sets allowing for multiple instances of each of its elements.

A *graph* G is a pair $(V(G), E(G))$ with *finite* sets of *vertices* $V(G)$ and *edges* $E(G) \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$. If not otherwise stated, we set $n \coloneqq |V(G)|$. For ease of notation, we denote the edge $\{u, v\}$ in $E(G)$ by (u, v) or (v, u). In the case of *directed graphs*, $E \subseteq \{(u, v) \in V \times V \mid u \neq v\}$. A *labeled graph* G is a triple (V, E, l) with *(vertex) coloring* or *label function* $l \colon V(G) \to \mathbb{N}$. Then $l(v)$ is a *label* of v for v in $V(G)$. The *neighborhood* of v in G is denoted by $N_G(v) \coloneqq \{u \in V(G) \mid \{v, u\} \in E(G)\}$ and the *degree* of a vertex v is $|N_G(v)|$. For $S \subseteq V(G)$, the graph $G[S] = (S, E_S)$ is the *subgraph induced by* S, where $E_S \coloneqq \{(u, v) \in E(G) \mid u, v \in S\}$.

Two graphs G and H are *isomorphic* and we write $G \simeq H$ if there exists a bijection $\varphi \colon V(G) \to V(H)$ that preserves the adjacency relation, i.e., (u, v) is in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $E(H)$. Then φ is an *isomorphism* between G and H. Moreover, we call the equivalence classes induced by the relation $\simeq$ *isomorphism types*. In the case of labeled graphs, we additionally require that $l(v) = l(\varphi(v))$ for v in $V(G)$. We further define the atomic type $\mathsf{atp} \colon V(G)^k \to \mathbb{N}$ such that $\mathsf{atp}(\mathbf{v}) = \mathsf{atp}(\mathbf{w})$ for $\mathbf{v}, \mathbf{w} \in V(G)^k$ if and only if the mapping $\varphi \colon V(G)^k \to V(G)^k$ where $v_i \mapsto w_i$ induces a partial isomorphism, i.e., we have $v_i = v_j \iff w_i = w_j$ and $(v_i, v_j) \in E(G) \iff (\varphi(v_i), \varphi(v_j)) \in E(G)$. Let $\mathbf{v}$ be a *tuple* in $V(G)^k$ for $k > 0$, then $G[\mathbf{v}]$ is the *ordered k-vertex subgraph* induced by the multiset of elements of $\mathbf{v}$, where the vertices are labeled with integers from $[k]$ corresponding to their positions in $\mathbf{v}$. Moreover, let $\mathsf{t}(G[\mathbf{v}]) \coloneqq \mathbf{v}$, i.e., the k-tuple $\mathbf{v}$ underlying the ordered k-vertex subgraph $G[\mathbf{v}]$. We denote the set of all ordered k-vertex subgraphs of a graph G by G_k. Finally, let $\mathcal{G}$ be the set of all vertex-labeled graphs.

The 1-WL and the k-WL The 1-WL or color refinement is a simple heuristic for the graph isomorphism problem, originally proposed by Weisfeiler and Leman [1968].[2] Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling vertices. Given an initial coloring or labeling of the vertices of both graphs, e.g., their degree or application-specific information, in each iteration, two vertices with the same label get different labels if the number of identically labeled neighbors is not equal. If, after some iteration, the number of vertices annotated with a specific label is different in both graphs, the algorithm terminates and a stable coloring (partition) is obtained. We can then conclude that the two graphs are not isomorphic. It is easy to see that the algorithm cannot distinguish all non-isomorphic graphs [Cai et al., 1992]. Nonetheless, it is a powerful heuristic that can successfully test isomorphism for a broad class of graphs [Babai and Kucera, 1979].

Formally, let $G = (V, E, l)$ be a labeled graph. In each iteration, $i > 0$, the 1-WL computes a vertex coloring $C^1_i \colon V(G) \to \mathbb{N}$, which depends on the coloring of the neighbors. That is, in iteration $i > 0$, we

[2] Strictly speaking, the 1-WL and color refinement are two different algorithms. That is, the 1-WL considers neighbors and non-neighbors to update the coloring, resulting in a slightly higher expressive power when distinguishing vertices in a given graph, see Grohe [2021] for details. For brevity, we consider both algorithms to be equivalent.

set

$$C^1_i(v) := \mathsf{RELABEL}\Big(\big(C^1_{i-1}(v), \{\!\!\{C^1_{i-1}(u) \mid u \in N_G(v)\}\!\!\}\big)\Big),$$

where RELABEL injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C^1_0 := l$. To test if two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of vertices colored c in $\mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. Moreover, if the number of colors between two iterations, i and $(i+1)$, does not change, i.e., the cardinalities of the images of C^1_i and C^1_{i+1} are equal, the algorithm terminates. For such i, we define the *stable coloring* $C^1_\infty(v) = C^1_i(v)$ for v in $V(G)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [Grohe, 2017].

Due to the shortcomings of the 1-WL or color refinement in distinguishing non-isomorphic graphs, several researchers [Babai, 1979, 2016, Immerman and Lander, 1990], devised a more powerful generalization of the former, today known as the *k-dimensional Weisfeiler-Leman algorithm* (k-WL); see Appendix B for a detailed description.

Graph Neural Networks Intuitively, GNNs learn a vectorial representation, i.e., a d-dimensional vector, representing each vertex in a graph by aggregating information from neighboring vertices. Formally, let $G = (V, E, l)$ be a labeled graph with initial vertex features $\mathbf{h}^{(0)}_v \in \mathbb{R}^d$ that are *consistent* with l. That is, each vertex v is annotated with a feature $\mathbf{h}^{(0)}_v \in \mathbb{R}^d$ such that $\mathbf{h}^{(0)}_u = \mathbf{h}^{(0)}_v$ if $l(u) = l(v)$, e.g., a one-hot encoding of the labels $l(u)$ and $l(v)$. Alternatively, $\mathbf{h}^{(0)}_v$ can be an arbitrary real-valued feature vector or attribute of the vertex v, e.g., physical measurements in the case of chemical molecules. A GNN architecture consists of a stack of neural network layers, i.e., a composition of parameterized functions. Similarly to 1-WL, each layer aggregates local neighborhood information, i.e., the neighbors' features, around each vertex and then passes this aggregated information on to the next layer.

Following, Gilmer et al. [2017] and Scarselli et al. [2009], in each layer, $i > 0$, we compute vertex features

$$\mathbf{h}^{(i+1)}_v := \mathsf{UPD}^{(i+1)}\Big(\mathbf{h}^{(i)}_v, \mathsf{AGG}^{(i+1)}\big(\{\!\!\{\mathbf{h}^{(i)}_u \mid u \in N_G(v)\}\!\!\}\big)\Big) \in \mathbb{R}^d,$$

where $\mathsf{UPD}^{(i+1)}$ and $\mathsf{AGG}^{(i+1)}$ may be differentiable parameterized functions, e.g., neural networks.[3] In the case of graph-level tasks, e.g., graph classification, one uses

$$\mathbf{h}_G := \mathsf{READOUT}\big(\{\!\!\{\mathbf{h}^{(T)}_v \mid v \in V(G)\}\!\!\}\big) \in \mathbb{R}^d, \tag{1}$$

to compute a single vectorial representation based on learned vertex features after iteration T. Again, READOUT may be a differentiable parameterized function. To adapt the parameters of the above three functions, they are optimized end-to-end, usually through a variant of stochastic gradient descent, e.g., [Kingma and Ba, 2015], together with the parameters of a neural network used for classification or regression.

The Weisfeiler–Leman hierarchy and permutation-invariant function approximation The Weisfeiler–Leman hierarchy is a purely combinatorial algorithm for testing graph isomorphism. However, the graph isomorphism function, mapping non-isomorphic graphs to different values, is the hardest to approximate permutation-invariant function. Hence, the Weisfeiler–Leman hierarchy has strong ties to GNNs' capabilities to approximate permutation-invariant or equivariant functions over graphs. For example, Morris et al. [2019], Xu et al. [2019] showed that the expressive power of any possible GNN architecture is limited by 1-WL in terms of distinguishing non-isomorphic graphs. Azizian and Lelarge [2020] refined these results by showing that if an architecture is capable of simulating k-WL and allows the application of universal neural networks on vertex features, it will be able to approximate any permutation-equivariant function below the expressive power of k-WL; see also Chen et al. [2019b]. Hence, if one shows that one architecture distinguishes more graphs than another, it follows that the corresponding GNN can approximate more functions. These results were refined in Geerts and Reutter [2022] for color refinement and taking into account the number of iterations of k-WL.

[3] Strictly speaking, Gilmer et al. [2017] consider a slightly more general setting in which vertex features are computed by $\mathbf{h}^{(i+1)}_v := \mathsf{UPD}^{(i+1)}\Big(\mathbf{h}^{(i)}_v, \mathsf{AGG}^{(i+1)}\big(\{\!\!\{(\mathbf{h}^{(i)}_v, \mathbf{h}^{(i)}_u, l(v,u)) \mid u \in N_G(v)\}\!\!\}\big)\Big)$.

https://doi.org/10.52202/068431-1529

3 Ordered subgraph Weisfeiler–Leman and MPNNs

In the following, we introduce a variant of 1-WL, denoted *k-ordered subgraph WL* (k-OSWL). Essentially, the k-OSWL labels or marks ordered subgraphs and then executes 1-WL on top of the marked graphs, followed by an aggregation phase. Although unordered subgraphs are also possible, ordered ones lead to more expressive architectures and also encompass the unordered case; see Appendix F.1 for a discussion. To make the procedure permutation-invariant, we consider all possible ordered subgraphs. Based on the ideas of k-OSWL, we then introduce k-OSANs, which can be seen as a neural variant of the former, allowing us to analyze various subgraph-enhanced GNNs.

3.1 Ordered subgraph WL

We now describe the algorithm formally. Let G be a graph, and let $\mathbf{g} \in G_k$ be an ordered k-vertex subgraph. Then k-OSWL computes a vertex coloring, similarly to 1-WL, with the main distinction that it can use structural graph information related to the ordered subgraph $G[(v, \mathsf{t}(\mathbf{g}))]$, where v is a vertex in $V(G)$.

More precisely, at each iteration $i \geq 0$, k-OSWL computes a coloring $C_i \colon V(G) \times G_k \to \mathbb{N}$ where we interpret elements $(v, \mathbf{g}) \in V(G) \times G_k$ as a vertex v along with an ordered k-vertex subgraph. Given an ordered k-vertex subgraph $\mathbf{g} \in G_k$, initially, for $i = 0$, we set $C_0(v, \mathbf{g}) := \mathsf{atp}(v, \mathsf{t}(\mathbf{g}))$, and for $i > 0$, we set

$$C_{i+1}(v, \mathbf{g}) := \mathsf{RELABEL}\Big(\big(C_i(v, \mathbf{g}), \{\!\{C_i(u, \mathbf{g}) \mid u \in \square\}\!\}\big)\Big),$$

where $\square$ is either $N_G(v)$ or $V(G)$. We compute the stable partition analogously to 1-WL. Finally, to compute a single color for a vertex v, we aggregate all ordered k-vertex subgraphs, i.e., we compute

$$C(v) := \mathsf{RELABEL}\big(\{\!\{C_\infty(v, \mathbf{g}) \mid \mathbf{g} \in G_k\}\!\}\big). \tag{2}$$

In other words, one can regard the k-OSWL as running 1-WL in parallel over n^k graphs, one for each ordered k-vertex subgraph $\mathbf{g} \in G_k$, followed by combining the colors of each vertex in all these graphs. Furthermore, by restricting the number of considered subgraphs, the algorithm allows for more fine-grained control over the trade-off between scalability and expressivity. Note that 0-OSWL is equal to 1-WL. We also define a variation of the k-OSWL, denoted *vertex-subgraph k*-OSWL, which, unlike Equation (2), first computes a color $C(\mathbf{g})$ for each ordered k-vertex subgraph $\mathbf{g}$ by aggregating over vertices; see Appendix E for details.

We remark that in contrast to k-WL, which has to update the coloring of all n^k ordered k-vertex subgraphs in a complicated manner, the computation of k-OSWL's coloring relies on the simple and easy-to-implement 1-WL. Furthermore, k-OSWL's computation can be either done in parallel or sequentially across all n vertices and n^k graphs. Despite its simplicity, in Section 3.2, we will show that the k-OSWL has high expressivity.

3.2 Ordered subgraph MPNNs

In the following, to study the expressivity of subgraph-enhanced GNNs, we introduce k-ordered subgraph MPNNs (k-OSANs), which can be viewed as neural variants of the k-OSWL. At initialization, k-OSANs learn two features for each element in G_k and each vertex v

$$\mathbf{h}_{v,\mathbf{g}}^{(0)} := \mathsf{UPD}(\mathsf{atp}(v, \mathsf{t}(\mathbf{g}))) \in \mathbb{R}^{\mathsf{d}}, \quad \text{and} \quad \boldsymbol{\pi}_{v,\mathbf{g}} := \mathsf{UPD}_{\boldsymbol{\pi}}(\mathsf{atp}(v, \mathsf{t}(\mathbf{g}))),$$

where UPD and $\mathsf{UPD}_{\boldsymbol{\pi}}$ are differentiable, parameterized function, e.g., a neural network. Additional vertex features can be concatenated to the first feature. We use the second feature $\boldsymbol{\pi}_{v,\mathbf{g}}$ to select admissible ordered subgraphs for the vertex v; see below. Now in each layer $(i + 1)$, we update the feature of a vertex v with regard to the k-ordered subgraph $\mathbf{g}$ as

$$\mathbf{h}_{v,\mathbf{g}}^{(i+1)} := \mathsf{UPD}^{(i+1)}\Big(\mathbf{h}_{v,\mathbf{g}}^{(i)}, \mathsf{AGG}^{(i+1)}\big(\{\!\{\mathbf{h}_{u,\mathbf{g}}^{(i)} \mid u \in \square\}\!\}\big)\Big),$$

where $\square$ is either $N_G(v)$ or $V(G)$. After T such layers, for each vertex v, we then learn a joint feature over all k-ordered subgraphs, i.e., we apply subgraph aggregation

$$\mathbf{h}_v^{(T)} := \mathsf{SAGG}\big(\{\!\{\mathbf{h}_{v,\mathbf{g}}^{(T)} \mid \mathbf{g} \in G_k \text{ s.t. } \boldsymbol{\pi}_{v,\mathbf{g}} \neq \mathbf{0}\}\!\}\big). \tag{3}$$

https://doi.org/10.52202/068431-1529

Here, we leverage $\boldsymbol{\pi}_{v,\mathbf{g}} \neq \mathbf{0}$ to select a subset of the set of k-ordered subgraphs. Finally, analogous to GNNs, we use a $\mathsf{READOUT}$ layer to compute a single graph feature. Again, $\mathsf{AGG}^{(i+1)}$, $\mathsf{UPD}^{(i+1)}$, $\mathsf{READOUT}$, and SAGG are differentiable, parameterized functions, e.g., neural networks.

We also define a variation of k-OSANs, denoted *vertex-subgraph k-OSANs*, which, unlike Equation (3), first compute a color for each ordered k-vertex subgraph; see Appendix E for details.

Expressive power of k-OSANs In the following, we study the expressive power of k-OSANs. The first result shows that any possible k-OSAN has at most the expressive power of k-OSWL in terms of distinguishing non-isomorphic graphs. Further, k-OSANs are in principle capable of reaching k-OSWL's expressive power. Hence, the k-OSWL upper bounds k-OSANs ability to represent permutation-invariant functions.

Proposition 1. For all $k \geq 1$, it holds that k-OSANs are upper bounded by k-OSWL in terms of distinguishing non-isomorphic graphs. Further, there exists a k-OSAN instance that has exactly the same expressive power as the k-OSWL.

The following result shows that any possible k-OSAN is upper-bounded by the $(k+1)$-WL in terms of distinguishing non-isomorphic graphs while the expressive power of k-OSANs and the k-WL are incomparable. That is, there exist non-isomorphic graphs k-WL cannot distinguish while k-OSANs can and vice versa.

Proposition 2. For all $k \geq 1$, it holds that $(k+1)$-WL is *stricly more* expressive than k-OSANs and there exist non-isomorphic graphs k-WL cannot distinguish while k-OSANs can and vice versa.

Finally, the following results shows that increasing the size of the subgraphs always leads to a strictly more expressive k-OSANs.

Theorem 3. For all $k \geq 1$, it holds that $(k+1)$-OSANs is *strictly more* expressive than k-OSANs.

Subgraph-enhanced GNNs captured by k-OSANs To exemplify the power and generality of k-OSANs, we show how k-OSANs cover most subgraph-enhanced GNNs; see Appendix F for a thorough overview. We say that k-OSANs *capture* a subgraph-enhanced GNN G if there exists a k-OSAN instance that is at least as expressive as G.

The first results shows that k-OSANs capture k-marked GNNs (k-mGNNs) [Papp and Wattenhofer, 2022] and k-reconstruction GNNs (k-recGNNs) [Cotta et al., 2021]. For both approaches, the sets of k vertices to be marked or deleted correspond to unordered k-vertex subgraphs. It then suffices to ensure that the update and aggregation functions in the k-OSANs treat the vertices in the selected k-vertex subgraphs as being marked or deleted.

Proposition 4. For $k \geq 1$, k-OSANs capture k-mGNNs and vertex-subgraph k-OSANs capture k-recGNNs.

Further, 1-OSANs capture identity-aware GNNs (idGNNs) [You et al., 2021], GNN As Kernel (kernelGNNs) [Zhao et al., 2021], and nested GNNs (nestedGNNs) [Zhang and Li, 2021]. Intuitively, in these approaches GNNs are used locally around each vertex. It thus suffices to ensure that the update and aggregation functions in the 1-OSANs use the selected single vertex subgraph to only pass messages locally.

Proposition 5. 1-OSANs capture idGNNs, kernelGNNs, and nestedGNNs.

Finally, k-OSANs capture the DS-GNNs with the vertex-deleted policy [Bevilacqua et al., 2021].[4]

Proposition 6. Vertex-subgraph k-OSANs capture DS-GNNs with the k-vertex-deleted policy.

We note that the above result can be further extended to accommodate the edge-deleted and ego-networks policy from Bevilacqua et al. [2021]; see Appendix F.

Importantly, by viewing existing subgraph-enhanced GNNs as k-OSANs we immediately gain insights into their expressive power. Previous results primarily focused on showing more expressivity than 1-WL.

[4] We note here that it is an open question if vertex-subgraph k-OSANs also capture the more general DSS-GNNs [Bevilacqua et al., 2021].

https://doi.org/10.52202/068431-1529

4 Data-driven Subgraph-enhanced GNNs

In the above section, we thoroughly investigated the expressive power of subgraph-enhanced GNNs. Specifically, we showed that they are strictly limited by the $(k+1)$-WL and that they can distinguish graphs which are not distinguishable by k-WL. As indicated by Proposition 1 and to reach maximal expressive power, however, we need to consider all possible ordered subgraphs, resulting in an exponential running time. Hence, in this section, we leverage the I-MLE framework [Niepert et al., 2021], to sample ordered subgraphs in a data-driven fashion. We first address the problem of learning the parameters of a probability distribution over ordered subgraphs using a GNN. Secondly, we show how to approximately sample from this intractable distribution. Subsequently, these subgraphs are used within a k-OSAN to compute a graph representation. Finally, we propose a gradient estimation scheme that allows us to use backpropagation in the resulting discrete-continuous architecture.

Parameterizing probability distributions over subgraphs Contrary to existing approaches, which often consider all possible subgraphs or sample a fraction of subgraphs uniformly at random, our method maintains a probability distribution over (ordered) subgraphs. Let G be a graph where each vertex v has an initial feature $\mathbf{h}_v^{(0)}$, which we stack row-wise over all vertices into the feature matrix $\mathbf{H} \in \mathbb{R}^{n \times d}$. Further, let $h_{\mathbf{W}_1} : \mathcal{G} \times \mathbb{R}^{n \times d} \to \mathbb{R}^{m \times n}$ be a permutation-equivariant function, e.g., a message-passing GNN, parameterized by $\mathbf{W}_1$, called *upstream model*, mapping a graph G and its initial features $\mathbf{H}$ to a parameter matrix

$$\boldsymbol{\theta} := h_{\mathbf{W}_1}(G, \mathbf{H}) \in \mathbb{R}^{m \times n}.$$

Intuitively, each parameter θ_{ij} is an unnormalized prior probability of vertex j being part of the ith sampled subgraph of G. Let $\boldsymbol{\theta}_i := (\theta_{i1}, \dots, \theta_{in})$ for $i \in [m]$. We use these to parameterize m probability distributions $p(\boldsymbol{z}; \boldsymbol{\theta}_i)$, for $i \in [m]$, over vector encodings of ordered k-vertex subgraphs of G, i.e.,

$$p(\boldsymbol{z}; \boldsymbol{\theta}_i) := \begin{cases} \exp\left(\langle \boldsymbol{z}, \boldsymbol{\theta}_i \rangle - A(\boldsymbol{\theta}_i)\right) & \text{if } \boldsymbol{z} \in \mathcal{Z}, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product and $A(\boldsymbol{\theta}_i)$ is the *log-partition function* defined as $A(\boldsymbol{\theta}_i) := \log\left(\sum_{\boldsymbol{z} \in \mathcal{Z}} \exp\left(\langle \boldsymbol{z}, \boldsymbol{\theta}_i \rangle\right)\right)$. Furthermore, for a distribution over *unordered* k-vertex subgraphs, $\mathcal{Z}$ is the set of all binary n-component vectors with exactly k non-zero entries indicating which vertices are part of a subgraph of G. Hence, there is a bijection between $\mathcal{Z}$ and the set of unordered k-vertex subgraphs of G. For a distribution over *ordered* k-vertex subgraphs, the set $\mathcal{Z}$ is the set of all vectors in $[k]^n$ with k non-zero entries. For each non-zero entry $\boldsymbol{z}_i$ for $\boldsymbol{z} \in \mathcal{Z}$ it holds that $\boldsymbol{z}_i = j$ if and only if vertex i has *rank* $k+1-j$ in the ordered subgraph, encoding the position in the ordered graph. This encoding is required for the gradient computation we perform later. For instance, in an ordered 5-vertex subgraph, if a node has rank 1 but should have rank 5 to obtain a lower loss, then the gradient of the downstream loss is proportional to $5-1=4$. Similarly, if a node i is not part of the ordered subgraph, that is, $\boldsymbol{z}_i = 0$ but should be in position 1 of the ordered subgraph, then the gradient of a downstream loss is proportional to $0-5$. For any $i, j \in [k]$ with $i \neq j$ we have that $\boldsymbol{z}_i \neq \boldsymbol{z}_j$. Hence, again, there is a bijection between $\mathcal{Z}$ and the set of ordered k-vertex subgraphs of G.

Efficient approximate sampling of subgraphs Computing the log-partition function and sampling exactly from the probability distribution in Equation (4) is intractable for both ordered and unordered graphs. Since it is tractable, however, to compute a configuration with a highest probability, a maximum a posteriori (MAP) configuration $\boldsymbol{z}^*(\boldsymbol{\theta}_i)$, we can use perturb-and-MAP [Papandreou and Yuille, 2011, Niepert et al., 2021] to sample approximately. For unordered graphs, determining the top-k values in $\boldsymbol{\theta}_i$ suffices, while for ordered graphs, we additionally require their rank. Therefore, the worst-case running time of computing $\boldsymbol{z}^*(\boldsymbol{\theta}_i)$ for ordered graphs of size k is in $O(n + k \log k)$. That is, we first use a selection algorithm to find the kth largest element E in the list of weights, taking time $\mathcal{O}(n)$, e.g., using the Quickselect algorithm. Now, we go through the list again and select all entries larger to E, taking time $\mathcal{O}(n)$. Finally, we sort the k values.

Now, to use perturb-and-MAP to *approximately* sample the i-th ordered k-vertex subgraph $\mathbf{g}_i$ from the above probability distributions, we compute

$$\mathbf{g}_i := \mathtt{adj}\left(\boldsymbol{z}^*(\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i)\right) \quad \text{with} \quad \boldsymbol{\epsilon}_i \sim \rho(\boldsymbol{\epsilon}),$$

where $\rho(\boldsymbol{\epsilon})$ is a noise distribution such as the Gumbel distribution and `adj` converts the above vector encoding of the (ordered) subgraph to an $n \times n$ adjacency matrix as follows. The jth row or column encodes the vertex of the ordered subgraph with rank j and its incident edges within the ordered subgraph. All other

https://doi.org/10.52202/068431-1529

entries are set to 0, i.e., they are masked out. We therefore sample a multiset of (ordered) subgraphs $S \coloneqq \{\!\{ \boldsymbol{g}_1, ..., \boldsymbol{g}_m \}\!\} \subseteq G_k$, which act as the input to a k-OSAN instance $f_{\mathbf{W}_2}$, called *downstream model*, where $\boldsymbol{\pi}_{v,\mathbf{g}} \neq \mathbf{0}$ for $v \in V(G)$ if $\mathbf{g} \in S$, to compute the target outputs

$$f_{\mathbf{W}_2}(G, \mathbf{H}, \{\!\{ \boldsymbol{g}_1, ..., \boldsymbol{g}_m \}\!\}).$$

Backpropagating through the subgraph distribution Now that we have outlined a way to approximately sample subgraphs, we still need to learn the parameters $\boldsymbol{\omega} = (\mathbf{W}_1, \mathbf{W}_2)$ of the upstream and downstream model. Hence, given a set of examples $\{(G_j, \mathbf{H}_j, \hat{\boldsymbol{y}}_j)\}_{j=1}^{N}$, we are concerned with finding approximate solutions to $\min_{\boldsymbol{\omega}} {}^1\!/\!{}_N \sum_j L(G, \mathbf{H}, \hat{\boldsymbol{y}}_j; \boldsymbol{\omega})$, where L is the expected training error

$$L(G, \mathbf{H}, \hat{\boldsymbol{y}}_j; \boldsymbol{\omega}) \coloneqq \mathbb{E}_{\boldsymbol{g}_i \sim p(\boldsymbol{z}; \boldsymbol{\theta}_i)} \left[\ell \left(f_{\mathbf{W}_2} \left(G, \mathbf{H}, \{\!\{ \boldsymbol{g}_1, ..., \boldsymbol{g}_m \}\!\} \right), \hat{\boldsymbol{y}} \right) \right], \tag{5}$$

with $\boldsymbol{\theta} \coloneqq h_{\mathbf{W}_1}(G, \mathbf{H})$ and $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a point-wise loss function. The challenge of training a model as defined in Equation (5) is to compute $\nabla_{\boldsymbol{\theta}} L(G, \mathbf{H}, \hat{\boldsymbol{y}}_j; \boldsymbol{\omega})$, i.e., the gradient with respect to the parameters $\boldsymbol{\theta}$ of the probability distribution for the expected loss. In this work, we utilize implicit maximum likelihood learning, a recent framework that allows us to use algorithmic solvers of combinatorial optimization problems as black-box components [Rolinek et al., 2020, Niepert et al., 2021]. A particular instance of the framework uses implicit differentation via perturbation [Domke, 2010]. We derive the gradient computation for a single ordered subgraph $\mathbf{g}_i$ to simplify the notation. We compute the gradients of the downstream loss with respect to parameters $\boldsymbol{\theta}_i$ as

$$\nabla_{\boldsymbol{\theta}_i} L(G, \mathbf{H}, \hat{\boldsymbol{y}}; \boldsymbol{\omega}) \approx \mathbb{E}_{\boldsymbol{\epsilon}_i \sim \rho(\boldsymbol{\epsilon})} \left[{}^1\!/\!{}_\lambda \left(\boldsymbol{z}^* \left(\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i \right) - \boldsymbol{z}^* \left(\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i - \lambda \widehat{\nabla} \right) \right) \right],$$

where $\lambda > 0$ and $\widehat{\nabla}_v$, the approximated gradient for a single vertex v, is defined as

$$\widehat{\nabla}_v \coloneqq \mathbf{agg}\big(\{\!\{ \left[\nabla_{\boldsymbol{g}_\mathbf{i}} \ell \left(f_{\mathbf{W_2}} \left(G, \mathbf{H}, \{\!\{ \boldsymbol{g}_i \}\!\} \right) \right) \right]_{v,w} \mid w \in N_G(v) \}\!\} \big),$$

with $v \in V(G)$. Here, $\mathbf{agg}$ can be any aggregation function such as the element-wise sum or mean. Hence, to approximate the gradients with respect to the parameter $\boldsymbol{\theta}_{iv}$, which corresponds to vertex v of the input graph, we aggregate the gradients of the downstream loss with respect to all edges (v, w) incident to vertex v in the *original* graph.

Hence, the above techniques allow us to efficiently learn to sample subgraphs, which are then fed into a k-OSAN to learn a graph representation while optimizing the parameters of all components in an end-to-end fashion.

5 Experimental evaluation

Here, we aim to empirically investigate the learning performance and efficiency of data-driven subgraph-enhanced GNNs, instances of the k-OSAN framework, compared to non-data-driven ones. Specifically, we aim to answer the following questions.

Q1 Do data-driven subgraph-enhanced GNNs exhibit better predictive performance than non-data-driven ones?

Q2 Does the graph structure of the subgraphs sampled affect predictive performance?

Q3 Does data-driven sampling have an advantage in efficiency and predictive performance when used within state-of-the-art subgraph-enhanced GNNs?

All experimental results are fully reproducible from the source code provided at `https://github.com/Spazierganger/OSAN`.

Datasets To compare our data-driven, subgraph-enhanced GNNs to non-data-driven ones and standard GNN baselines, we used the ALCHEMY [Chen et al., 2019a], the QM9 [Ramakrishnan et al., 2014, Wu et al., 2018], OGBG-MOLESOL [Hu et al., 2020], and the ZINC [Dwivedi et al., 2020, Jin et al., 2017] graph-level regression datasets; see Table 16 in Appendix C for dataset statistics and properties. In addition, we used the EXP dataset [Abboud et al., 2020] to investigate the additional expressive power of subgraph-enhanced GNNs over standard ones. Following Morris et al. [2020b], we opted not to use the 3D-coordinates of the ALCHEMY dataset to solely show the benefits of the data-driven subgraph-enhanced

Table 1: Results on large-scale regression datasets, data-driven versus non-data-driven subgraph sampling.

(a) Results for the OGBG-MOLESOL dataset.

Method					RSMSE ↓
Baseline					1.193 ±0.083
	OPERAT.	TYPE	#	# SUBG.	
Random I-MLE	Delete	Vertex	1	3	1.215 ±0.095 1.053 ±0.080
Random I-MLE	Delete	Vertex	1	10	1.128 ±0.055 0.984 ±0.086
Random I-MLE	Delete	Vertex	2	1	1.283 ±0.080 0.968 ±0.102
Random I-MLE	Delete	Vertex	2	3	1.132 ±0.020 1.081 ±0.021
Random I-MLE	Delete	Vertex	5	3	0.992 ±0.115 1.115 ±0.076
Random I-MLE	Delete	Vertex	5	10	1.186 ±0.154 1.137 ±0.053
Random I-MLE	Select	Vertex	10	1	1.128 ±0.022 1.099 ±0.099
Random I-MLE	Delete	Edge	1	3	1.240 ±0.029 1.106 ±0.069
Random I-MLE	Delete	Edge	1	10	1.152 ±0.046 1.056 ±0.071
Random I-MLE	Delete	Edge	3	3	1.084 ±0.076 1.052 ±0.049
Random I-MLE	Delete	Edge	3	10	1.099 ±0.071 1.077 ±0.079
Random I-MLE	Delete	2-Ego	–	3	1.071 ±0.062 0.959 ±0.184

(b) Result for the ALCHEMY dataset.

Method					MAE ↓
Baseline					11.12 ±0.69
	OPERAT.	TYPE	#	# SUBG.	
Random I-MLE	Delete	Vertex	1	3	13.26 ±0.41 8.78 ±0.28
Random I-MLE	Delete	Vertex	1	10	12.11 ±0.21 8.87 ±0.12
Random I-MLE	Delete	Vertex	2	3	12.66 ±0.28 9.01 ±0.27
Random I-MLE	Delete	Vertex	5	3	10.29 ±0.30 9.22 ±0.06
Random I-MLE	Delete	Edge	1	3	11.66 ±0.63 10.80 ±0.31
Random I-MLE	Delete	Edge	2	3	10.79 ±0.64 10.56 ±0.44
Random I-MLE	Delete	Edge	5	3	9.15 ±0.12 9.08 ±0.28
Random I-MLE	Select	Vertex	5	3	11.48 ±0.60 9.22 ±0.14
Random I-MLE	Select	Edge	5	3	8.99 ±0.24 8.95 ±0.29
Random I-MLE	Delete	1-Ego	–	3	14.98 ±0.49 11.15 ±1.09
Random I-MLE	Select	5-Ego	–	3	14.97 ±0.23 13.83 ±1.06

GNNs regarding graph structure. All datasets, excluding EXP and OGBG-MOLESOL, are available from Morris et al. [2020a].[5]

Neural architectures and experimental protocol For all datasets and architectures, we used the competitive GIN layers [Xu et al., 2019] for the baselines and the downstream models. For data with (continuous) edge features, we used a 2-layer MLP to map them to the same number of components as the vertex features and combined them using summation. We describe the upstream and downstream models' architecture used for each dataset in the following. We stress here that we always used the same hyperparameters for the downstream model and the baselines.

Sampling subgraphs Since the number of unordered k-vertex subgraphs is considerably smaller than the number of ordered k-vertex subgraphs, we opted to consider unordered k-vertex subgraphs; see also Appendix F.1. Further, since vertex-subgraph k-OSANs, see Appendix E, are easier to implement efficiently and are closer to DS-GNNs variant of ESAN [Bevilacqua et al., 2021], we opted to use them for the empirical evaluation. In addition, we used a simple GNN architecture for the upstream model to compute initial features for the subgraphs for ease of implementation. We experimented with selecting and deleting a various number of vertices, edges, and subgraphs induced by k-hop neighborhoods (k-Ego) for all datasets; see Appendix C for details.

Upstream models For all datasets and experiments, we used a GCN model [Kipf and Welling, 2017] consisting of three GCN layers, with batch norm and ReLU activation after each layer. We set the hidden dimensions to that of the downstream model one. The model either outputs the vertex or edge embeddings according to the task. We computed edge embeddings based on the vertex features of the incident vertices after the last layers and the edge attributes provided by the dataset.

When sampling multiple subgraphs with I-MLE, they tend to have similar structures. In other words, I-MLE learns similar distributions in different channels of the neural network. This phenomenon is not in our favor, as we need to cover the original full graph as much as possible. To mitigate this issue, we propose an auxiliary loss for the diversity of subgraphs. We calculate the cosine similarity between the selected vertex or edge masks of every two subgraphs and try to minimize the average similarity value. We tune the weight for the auxiliary loss on the log scale, e.g., 0.1, 1, 10, and so on.

Downstream and baseline models See Appendix A for a detailed description of the architecture used for the downstream and baseline models, and how we processed subgraphs. For processing the subgraphs, we performed similar steps like ESAN. We first applied intra-subgraph aggregation for the vertices within each subgraph and obtained graph embeddings for each subgraph. After that, we performed inter-subgraph mean

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

pooling to obtain a single embedding vector for the original graph. It is worth noting that ESAN does not exclude the vertices deleted during graph pooling but removes the adjacent edges of those nodes. In our experiments, we masked out the deleted or unselected nodes.

See Appendix C for further details on the experiments.

5.1 Results and discussion

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

A1 See Tables 1, 2a, 4 and 5 (in the appendix). On all five datasets, the subgraph-enhanced GNN models based on I-MLE beat the random baseline, excluding edge sampling configurations on the ALCHEMY and the QM9 dataset; see Tables 1b and 2a. For example, on the OGBG-MOLESOL the average gain over the random baseline is over 11%. Similar improvements can be observed over the other four datasets. Moreover, the results on the EXP dataset, see Appendix C, clearly indicate that the added expressivity of the (data-driven) subgraph-enhanced GNNs translates into improved predictive performance. The data-driven subgraph-enhanced GNNs improve the accuracy of the non-subgraph-enhanced GNN by almost 50% in all configurations while improving over the random subgraph-enhanced GNN baseline by almost 6%. The data-driven subgraph-enhanced GNNs also clearly improve over the (non-subgraph-enhanced) GNN baseline on four out of five datasets.

A2 See Tables 1 and 2a (in the appendix). Deleting or selecting subgraphs leads to a clear boost in predictive performance across datasets over the random baseline while also improving over the non-subgraph-enhanced GNN baseline. Further, on all datasets, learning to delete or select k-hop neighborhood subgraphs for $k \in \{2, 3\}$ leads to a clear boost over the random as well as non-subgraph-enhanced baselines. However, the number of deleted vertices seems to affect the predictive performance. For example, on the OGBG-MOLESOL dataset, going from deleting one 2-vertex subgraph to one 10-vertex subgraph leads to a drop in performance. Hence, the drop in performance of the latter is in contrast to our theoretical findings, i.e., larger subgraphs lead to improved expressivity, indicating that more work should be done to understand subgraph-enhanced GNNs' generalization ability. Interestingly, deleting edges did not perform as well as deleting vertices or other subgraphs. We speculate that a more powerful edge embedding method is needed here, which computes edge features directly instead of learning them from vertex features.

A3 See Table 2b (in the appendix). The I-MLE-based ESAN severely speeds up the computation time. That is, across all configurations, we achieve a significant speed-up. For some configurations, e.g., sampling three vertices, the I-MLE based ESAN is more than 3.5 times faster than the non-data-driven ESAN while taking about the same time as the simple random baseline. We stress here that the ESAN implementation provided by Bevilacqua et al. [2021] precomputes subgraphs in a preprocessing step, which is not possible when learning to sample subgraphs using ESAN. Regarding predictive performance, the I-MLE based ESAN is slightly behind the non-data-driven one, although always better than the non-subgraph enhanced GNN baseline; see Table 5 in Appendix C.

6 Conclusion

We introduced the k-OSAN framework to study the expressive power of recently introduced subgraph-enhanced GNNs. We showed that any such architecture is strictly less powerful than the $(k+1)$-WL while being incomparable to the k-WL in representing permutation-invariant functions over graphs. Further, to circumvent random or heuristic subgraph selection, we devised a data-driven variant of k-OSANs which learn to select subgraph for a given data distribution. Empirically, we verified that such data-driven subgraph selection is superior to previously used random sampling in predictive performance. Further, when compared to state-of-the-art models, we showed promising performance in terms of computation time while still providing good predictive performance. We believe that our paper provides a first step in unifying combinatorial insights on the expressive power of GNNs with data-driven insights.

Acknowledgments and Disclosure of Funding

CM is partially funded a DFG Emmy Noether grant (468502433) and RWTH Junior Principal Investigator Fellowship under the Excellence Strategy of the Federal Government and the Länder. GR is funded by the DFG Research Grants Program–RA 3242/1-1–411032549. MN acknowledges funding by the German Research Foundation under Germany's Excellence Strategy–EXC 2075.

https://doi.org/10.52202/068431-1529

References

R. Abboud, İ. İ. Ceylan, M. Grohe, and T. Lukasiewicz. The surprising power of graph neural networks with random node initialization. *CoRR*, abs/2010.01179, 2020.

S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29, 2019.

R. Addanki, P. W. Battaglia, D. Budden, A. Deac, J. Godwin, T. Keck, W. L. S. Li, A. Sanchez-Gonzalez, J. Stott, S. Thakoor, and P. Velickovic. Large-scale graph representation learning with very deep gnns and self-supervision. *CoRR*, abs/2107.09422, 2021.

U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications. *CoRR*, abs/2006.05205, 2020.

B. M. Anderson, T. Hy, and R. Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519, 2019.

W. Azizian and M. Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. *CoRR*, abs/2006.15646, 2020.

L. Babai. Lectures on graph isomorphism. University of Toronto, Department of Computer Science. Mimeographed lecture notes, October 1979, 1979.

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

L. Babai and L. Kucera. Canonical labelling of graphs in linear average time. In *Symposium on Foundations of Computer Science*, pages 39–46, 1979.

M. Balcilar, P. Héroux, B. Gaüzère, P. Vasseur, S. Adam, and P. Honeine. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, pages 599–608, 2021.

A.-L. Barabasi and Z. N. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. L. Reutter, and J. P. Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020.

P. Barceló, F. Geerts, J. L. Reutter, and M. Ryschkov. Graph neural networks with local graph parameters. *CoRR*, abs/2106.06707, 2021.

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

D. Beaini, S. Passaro, V. Létourneau, W. L. Hamilton, G. Corso, and P. Liò. Directional graph networks. *CoRR*, abs/2010.02863, 2020.

B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron. Equivariant subgraph aggregation networks. *CoRR*, abs/2110.02910, 2021.

F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pages 874–883, 2020.

C. Bodnar, F. Frasca, N. Otter, Y. G. Wang, P. Liò, G. Montúfar, and M. M. Bronstein. Weisfeiler and Lehman go cellular: CW networks. *CoRR*, abs/2106.12575, 2021a.

C. Bodnar, F. Frasca, Y. Wang, N. Otter, G. F. Montufar, P. Lió, and M. Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pages 1026–1037, 2021b.

C. Bodnar, F. D. Giovanni, B. P. Chamberlain, P. Liò, and M. M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *CoRR*, abs/2202.04579, 2022.

https://doi.org/10.52202/068431-1529

K. M. Borgwardt, M. E. Ghisu, F. Llinares-López, L. O'Bray, and B. Rieck. Graph kernels: State-of-the-art and future challenges. *Foundations and Trends in Machine Learning*, 13(5-6), 2020.

G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *CoRR*, abs/2006.09252, 2020.

G. Bouritsas, F. Frasca, S. P. Zafeiriou, and M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

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

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

C. Cangea, P. Velickovic, N. Jovanovic, T. Kipf, and P. Liò. Towards sparse hierarchical graph classifiers. *CoRR*, abs/1811.01287, 2018.

I. Chami, Z. Ying, C. Ré, and J. Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 4869–4880, 2019.

I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *CoRR*, abs/2005.03675, 2020.

G. Chen, P. Chen, C. Hsieh, C. Lee, B. Liao, R. Liao, W. Liu, J. Qiu, Q. Sun, J. Tang, R. S. Zemel, and S. Zhang. Alchemy: A quantum chemistry dataset for benchmarking AI models. *CoRR*, abs/1906.09427, 2019a.

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

G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Velickovic. Principal neighbourhood aggregation for graph nets. *CoRR*, abs/2004.05718, 2020a.

G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, 2020b.

L. Cotta, C. Morris, and B. Ribeiro. Reconstruction for powerful graph representations. In *Advances in Neural Information Processing Systems*, 2021.

G. Dasoulas, L. D. Santos, K. Scaman, and A. Virmaux. Coloring graph neural networks for node disambiguation. In *International Joint Conference on Artificial Intelligence*, pages 2126–2132, 2020.

M. Defferrard, B. X., and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

J. Domke. Implicit differentiation by perturbation. In *Advances in Neural Information Processing Systems*, pages 523–531, 2010.

D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *CoRR*, abs/2003.00982, 2020.

D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

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

https://doi.org/10.52202/068431-1529

M. Fey, J.-G. Yuen, and F. Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

M. Fey, J. E. Lenssen, F. Weichert, and J. Leskovec. GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*, pages 3294–3304, 2021.

D. Flam-Shepherd, T. Wu, P. Friederich, and A. Aspuru-Guzik. Neural message passing on high order paths. *CoRR*, abs/2002.10413, 2020.

F. Frasca, B. Bevilacqua, M. M. Bronstein, and H. Maron. Understanding and extending subgraph GNNs by rethinking their symmetries. *CoRR*, abs/2206.11140, 2022.

M. Fürer. Weisfeiler-Lehman refinement requires at least a linear number of iterations. In *International Colloquium on Automata, Languages and Programming*, pages 322–333, 2001.

H. Gao and S. Ji. Graph U-Nets. In *International Conference on Machine Learning*, volume 97, pages 2083–2092, 2019.

F. Geerts. The expressive power of kth-order invariant graph networks. *CoRR*, abs/2007.12035, 2020.

F. Geerts and J. L. Reutter. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022.

F. Geerts, F. Mazowiecki, and G. A. Pérez. Let's agree to degree: Comparing graph convolutional networks in the message-passing framework. *CoRR*, abs/2004.02593, 2020.

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

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *ICLR*, 2018.

D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi. Understanding pooling in graph neural networks. *CoRR*, abs/2110.05292, 2021.

M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

M. Grohe. The logic of graph neural networks. In *ACM/IEEE Symposium on Logic in Computer Science*, pages 1–17, 2021.

A. Grover, E. Wang, A. Zweig, and S. Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

M. Horn, E. D. Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. M. Borgwardt. Topological graph neural networks. *CoRR*, abs/2102.07835, 2021.

W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020.

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

S. G. J. Klicpera, J. Groß. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

W. Jin, C. W. Coley, R. Barzilay, and T. S. Jaakkola. Predicting organic reaction outcomes with Weisfeiler-Lehman network. In *Advances in Neural Information Processing Systems*, pages 2604–2613, 2017.

https://doi.org/10.52202/068431-1529

Y. Jin, G. Song, and C. Shi. Gralsp: Graph neural networks with local structural patterns. *CoRR*, abs/1911.07675, 2019.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 2021.

C. Kim, A. Sabharwal, and S. Ermon. Exact sampling with integer linear programs and random perturbations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

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

J. Klicpera, F. Becker, and S. Günnemann. Gemnet: Universal directional graph neural networks for molecules. *CoRR*, abs/2106.08903, 2021.

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

G. Li, M. Müller, B. Ghanem, and V. Koltun. Training graph neural networks with 1000 layers. In M. Meila and T. Zhang, editors, *International Conference on Machine Learning*, pages 6437–6449, 2021.

P. Li, Y. Wang, H. Wang, and J. Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 2020.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*, 2017.

T. Maehara and H. NT. A simple proof of the universality of invariant/equivariant graph neural networks. *CoRR*, abs/1910.03802, 2019.

H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. *CoRR*, abs/1905.11136, 2019.

H. Maron, O. Litany, G. Chechik, and E. Fetaya. On learning sets of symmetric elements. In *International Conference on Machine Learning*, pages 6734–6744, 2020.

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

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

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

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

C. Morris, K. Kersting, and P. Mutzel. Glocalized Weisfeiler-Lehman kernels: Global-local feature maps of graphs. In *IEEE International Conference on Data Mining*, pages 327–336, 2017.

https://doi.org/10.52202/068431-1529

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

C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020a.

C. Morris, G. Rattan, and P. Mutzel. Weisfeiler and Leman go sparse: Towards higher-order graph embeddings. In *Advances in Neural Information Processing Systems*, 2020b.

C. Morris, Y. L., H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt. Weisfeiler and Leman go machine learning: The story so far. *CoRR*, abs/2112.09992, 2021.

C. Morris, G. Rattan, S. Kiefer, and S. Ravanbakhsh. SpeqNets: Sparsity-aware permutation-equivariant graph networks. In *International Conference on Machine Learning*, 2022.

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

M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

M. Niepert, P. Minervini, and L. Franceschi. Implicit MLE: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34:14567–14579, 2021.

H. NT and T. Maehara. Graph homomorphism convolution. *CoRR*, abs/2005.01214, 2020.

G. Papandreou and A. L. Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, pages 193–200, 2011.

P. A. Papp and R. Wattenhofer. A theoretical comparison of graph neural network extensions. *CoRR*, abs/2201.12884, 2022.

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

M. B. Paulus, D. Choi, D. Tarlow, A. Krause, and C. J. Maddison. Gradient estimation with stochastic softmax tricks. *arXiv preprint arXiv:2006.08063*, 2020.

R. Ramakrishnan, O. Dral, P., M. Rupp, and O. A. von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014. Nature.

M. Rolinek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius. Optimizing rank-based metrics with blackbox differentiation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7620–7630, 2020.

Y. Rong, W. Huang, T. Xu, and J. Huang. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks. *CoRR*, abs/2002.03155, 2020.

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

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

A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(2):714–35, 1997. IEEE.

J. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. Donghia, C. MacNair, S. French, L. Carfrae, Z. Bloom-Ackerman, V. Tran, A. Chiappino-Pepe, A. Badran, I. Andrews, E. Chory, G. Church, E. Brown, T. Jaakkola, R. Barzilay, and J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180:688–702.e13, 02 2020.

R. Talak, S. Hu, L. Peng, and L. Carlone. Neural trees for learning on graphs. *CoRR*, abs/2105.07264, 2021.

E. H. Thiede, W. Zhou, and R. Kondor. Autobahn: Automorphism-based graph neural nets. *CoRR*, abs/2103.01710, 2021.

J. Tönshoff, M. Ritzert, H. Wolf, and M. Grohe. Graph learning with 1D convolutions on random walks. *CoRR*, abs/2102.08786, 2021.

G. Tucker, A. Mnih, C. J. Maddison, D. Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

C. Vignac, A. Loukas, and P. Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *Advances in Neural Information Processing Systems*, 2020.

O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations*, 2016.

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

Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9:513–530, 2018.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462, 2018.

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

R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning*, pages 7134–7143, 2019.

J. You, J. Gomes-Selman, R. Ying, and J. Leskovec. Identity-aware graph neural networks. *CoRR*, abs/2101.10320, 2021.

M. Zhang and P. Li. Nested graph neural networks. *CoRR*, abs/2110.13197, 2021.

M. Zhang, Z. Cui, M. Neumann, and C. Yixin. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*, pages 4428–4435, 2018.

L. Zhao, W. Jin, L. Akoglu, and N. Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. *CoRR*, abs/2110.03753, 2021.

J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

https://doi.org/10.52202/068431-1529