Homomorphism Counts for Graph Neural Networks: All About That Basis

Emily Jin [1]   **Michael Bronstein** [1]   **İsmail İlkan Ceylan** [1]   **Matthias Lanzinger** [2 1]

Abstract

A large body of work has investigated the properties of graph neural networks and identified several limitations, particularly pertaining to their expressive power. Their inability to count certain *patterns* (e.g., cycles) in a graph lies at the heart of such limitations, since many functions to be learned rely on the ability of counting such patterns. Two prominent paradigms aim to address this limitation by enriching the graph features with *subgraph* or *homomorphism* pattern counts. In this work, we show that both of these approaches are sub-optimal in a certain sense and argue for a more *fine-grained* approach, which incorporates the homomorphism counts of *all* structures in the "basis" of the target pattern. This yields strictly more expressive architectures without incurring any additional overhead in terms of computational complexity compared to existing approaches. We prove a series of theoretical results on node-level and graph-level *motif parameters* and empirically validate them on standard benchmark datasets.

1. Introduction

Graph neural networks (GNNs) (Scarselli et al., 2008; Gori et al., 2005) are a class of architectures for learning invariant functions on graphs. Their success in various domains (Shlomi et al., 2021; Duvenaud et al., 2015; Zitnik et al., 2018) has led to a number of architectures (Kipf & Welling, 2017; Xu et al., 2019; Velickovic et al., 2018; Hamilton et al., 2017). Most of the architectures used in practice fall under the framework of message passing neural networks (MPNNs) from Gilmer et al. (2017), where the core idea is to iteratively update each node's representation based on the messages received from the node's neighbors.

(a) $\mathsf{Sub}(\pentagon, G_1) = 2$ and $\mathsf{Hom}(\pentagon, G_1) = 120$.

(b) $\mathsf{Sub}(\pentagon, G_2) = 2$ and $\mathsf{Hom}(\pentagon, G_2) = 140$.

Figure 1. Two 1-WL indistinguishable graphs G_1 and G_2. These graphs have the same number of 5-cycles, but they can be distinguished by the homomorphism counts of 5-cycles. $\mathsf{Sub}(F, G)$ is the number of times F occurs as a subgraph in G, and $\mathsf{Hom}(F, G)$ is the number of homomorphisms from F to G.

Graph Motif Parameters for Counting Patterns. The expressive power of MPNNs is upper bounded by the *1-dimensional Weisfeiler Leman graph isomorphism test (1-WL)* (Xu et al., 2019; Morris et al., 2019). Hence, all known inexpressiveness results for 1-WL apply to MPNNs. In particular, this implies that MPNNs cannot express a wide variety of *graph motif parameters*, which include functions that count the occurrences of basic graph patterns, such as paths or cycles. Since these patterns are abundant in real-world domains (e.g., cycles in molecules), a large body of work proposes enhancing the input graphs with such pattern counts. One line of work enriches the graph features with *subgraph* counts (Bouritsas et al., 2023; Bevilacqua et al., 2022; Frasca et al., 2022), while another enriches the graph features with *homomorphism* counts (Barceló et al., 2021). Although both approaches increase the expressive power of standard MPNNs, it remains unclear how these approaches compare to one another in terms of expressiveness gained.

Example 1.1. Consider the graphs G_1 and G_2 from Figure 1, which are indistinguishable by 1-WL and both contain *two* 5-cycles as *subgraphs*: formally we write $\mathsf{Sub}(\pentagon, G_1) = \mathsf{Sub}(\pentagon, G_2) = 2$. Hence, adding subgraph counts for 5-cycles to 1-WL does not help in distinguishing these graphs, but adding homomorphism counts for 5-cycles does help in distinguishing these graphs since $\mathsf{Hom}(\pentagon, G_1) \neq \mathsf{Hom}(\pentagon, G_2)$. △

[1]Department of Computer Science, University of Oxford, Oxford, UK [2]Institute for Logic and Computation, TU Wien, Vienna, AT. Correspondence to: Emily Jin <emily.jin@cs.ox.ac.uk>.
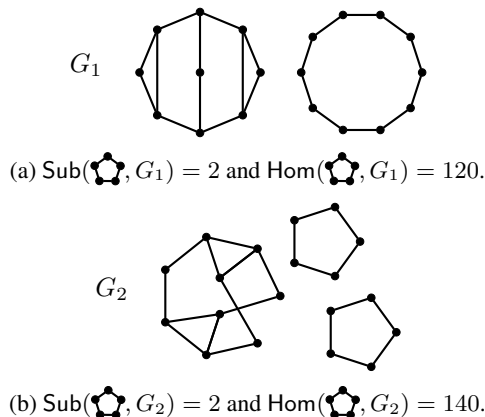
This example presents two 1-WL-indistinguishable graphs, where injecting (i.e. enriching with) homomorphism counts is a better strategy than injecting subgraph counts. How about the other direction? Are there graph pairs where injecting subgraph counts is the superior approach?

Example 1.2. Let H_1 be the first component of G_1 (left of Figure 1a) and H_2 be the first component of G_2 (left of Figure 1b). It is easy to verify that H_1 and H_2 cannot be distinguished by 1-WL, and we further observe that:

$$\mathsf{Hom}(\pentagon, H_1) = \mathsf{Hom}(\pentagon, H_2) = 120.$$

Thus, adding homomorphism counts for $\pentagon$ does not help in distinguishing these graphs. Conversely, we have $\mathsf{Sub}(\pentagon, H_1) = 2$ and $\mathsf{Sub}(\pentagon, H_2) = 0$, which suffices to distinguish the graphs. △

These examples show that these two approaches are incomparable in terms of the expressiveness gain that they yield. This brings us to the main quest of this paper: what information can we inject in GNNs that strictly subsumes both approaches? In this paper, we build on recent advances in graph theory that show that *graph motif parameters*, which encapsulate many functions of interest, can be expressed as a *linear combination of homomorphism counts*.

Example 1.3. Consider our running example from Figure 1. We can express $\mathsf{Sub}(\pentagon, G)$ as the linear sum:

$$\frac{1}{10}\mathsf{Hom}(\pentagon, G) - \frac{1}{2}\mathsf{Hom}(\text{Y}, G) + \frac{1}{2}\mathsf{Hom}(\triangle, G)$$

where the terms for $\pentagon$, Y, and $\triangle$ define the *homomorphism basis* of the target function $\mathsf{Sub}(\pentagon, \cdot)$. △

In a nutshell, the counts of the terms from the homomorphism basis are the key factors for expressiveness: we can access the homomorphism counts of all "base terms" that are *necessary and sufficient* to express the counts of the target pattern. In this sense, the target expressiveness gain is a very precise and a fine-grained one. Let us illustrate on our example how the basis terms are obtained.

Example 1.4. Consider the example from Figure 1. The graphs in the basis of the target pattern $\pentagon$ are those that are obtained by contracting non-adjacent vertices:
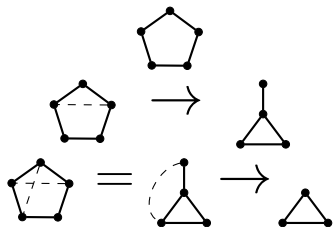
which results in a richer set of patterns in the basis. △
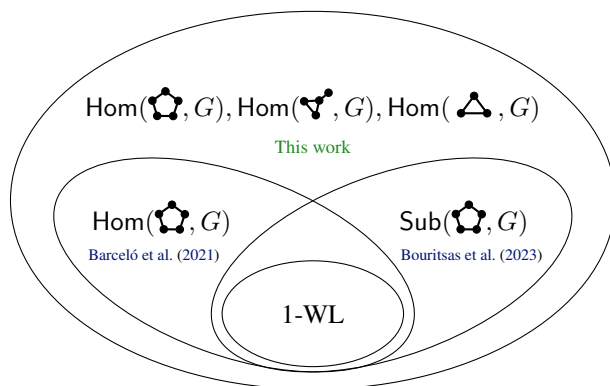

Figure 2. Expressiveness gain from injecting parameters. All inclusions are proper. All three features require (asymptotically) equivalent effort to calculate; they all can be computed in quadratic time (and no better).

Computing the homomorphism count for all basis terms of a target pattern is strictly more expressive than subgraph and homomorphism counting, as shown in Figure 2. A very appealing aspect of our approach is that it does not incur any additional overhead in terms of computational complexity compared to subgraph counting. In fact, the most efficient way of computing subgraph counts is by computing the linear combination of homomorphism counts.

Graph Motif Parameters for Beyond Counting Patterns. Although our exposition primarily focuses on subgraph and homomorphism counts for GNNs, our main objective is to consider the overarching class of graph motif parameters, which has broad implications beyond fixed pattern counting.

Example 1.5. We can express whether an atom (i.e., node) in a molecular dataset is part of an alcohol group with the following first-order formula:

$$\psi(x, y, z) := (\mathsf{bond}(x, y) \wedge \mathsf{bond}(y, z) \wedge \mathsf{O}(y) \wedge \mathsf{H}(z))$$

Similarly, we can express whether an atom is part of an acyl halide functional group as:

$$\tau(x, y, z, h) := (\mathsf{bond}(x, y) \wedge \mathsf{dbond}(y, z) \wedge \mathsf{bond}(y, h) \wedge \\ \mathsf{C}(y) \wedge \mathsf{O}(z) \wedge (\mathsf{F}(h) \vee \mathsf{Cl}(h) \vee \mathsf{Br}(h) \vee \mathsf{I}(h)))$$

where $\mathsf{O}, \mathsf{H}, \mathsf{C}, \mathsf{F}, \mathsf{Cl}, \mathsf{Br}$, and I represent corresponding elements in the periodic table. We can also disjunctively combine these properties to express that an atom is part of an alcohol or acyl halide functional group:

$$\phi(x, y, z, h) := \psi(x, y, z) \vee \tau(x, y, z, h)$$

The number of satisfying assignments of $\phi(x, y, z, h)$ is a graph motif parameter that can be expressed as a linear combination of homomorphism counts. This can be done at node-level and for specific variables, e.g., how often variable x is mapped to node v in a satisfying assignment. △

Example 1.5 captures a general property that goes beyond fixed pattern counting, yet our method can still precisely identify which homomorphism counts need to be injected in order to lift a model to the desired level of expressivity. By considering the broader framework of graph motif parameters, we provide an exact and principled way of achieving targeted expressivity gains for GNN architectures.

Contributions. Our main contributions can be summarized as follows:

- We study the expressivity of GNNs with respect to injecting graph motif parameters. We show that, in terms of expressiveness, it is highly beneficial to inject the homomorphism basis of a parameter instead of the parameter itself (Theorem 4.1), even for higher-order GNNs.

- This gain in expressiveness comes with additional benefits. In particular, we show how our method can naturally avoid redundant information (Section 4.3), and that it applies both on graph- and node-level (Section 4.4).

- Our approach provides a flexible framework for injecting information far beyond the fixed pattern counts considered in previous work, e.g., graphlet counts or logical properties (Section 4.5).

- We empirically validate the theoretical findings and demonstrate efficacy of the approach on a wide variety of benchmarks (Section 5). We observe state of the art results in expressivity and significant improvements in real-world regression and link-prediction tasks.

From a broader perspective, this work strengthens the connections between recent graph-theoretical results and graph machine learning, informing future work for the targeted design of GNN architectures suitable for a domain of interest.

2. Related Work

It is well-known that the expressive power of MPNNs is upper bounded by 1-WL in terms of graph distinguishability (Xu et al., 2019; Morris et al., 2019). Models such as graph isomorphism network (GIN) (Xu et al., 2019) can match this bound, whereas other popular architectures such as graph convolutional networks (GCNs) (Kipf & Welling, 2017), or graph attention networks (GATs) (Velickovic et al., 2018) cannot.

This expressiveness limitation has motivated a large body of work in the graph machine learning literature, including the study of *higher-order* GNN architectures (Morris et al., 2019; Maron et al., 2019a;b; Keriven & Peyré, 2019) which generally match the expressive power of the k-WL algorithm for some $k > 1$. Some works consider graphs with *unique node features* (Loukas, 2020), and others augment the node features with *random features* (Sato et al., 2021;

Abboud et al., 2021) to achieve higher expressive power. Expressive power of GNNs has also been evaluated in terms of their ability to detect certain graph components, such as biconnected components (Zhang et al., 2023b). Building on the classical literature of isomorphism testing and exploiting graph components (and associated graph decompositions) has led to efficient, isomorphism-complete learning algorithms for planar graphs (Dimitrov et al., 2023).

There are different means for studying expressive power, and our work's primary focus is on measuring expressive power based on the ability to capture graph motif parameters. MPNNs are very limited in terms of their expressive power when it comes to counting patterns and so any function of this form goes beyond the ability of these architectures (Chen et al., 2020). This led to a rich line of work of GNN architectures that inject such pattern counts explicitly (Bouritsas et al., 2023; Barceló et al., 2021; Bevilacqua et al., 2022; Frasca et al., 2022), some of which are known as subgraph GNNs. Zhang et al. (2023a) also recently provided an expressiveness hierarchy for subgraph GNNs via the so-called subgraph Weisfeiler-Leman tests.

Papp & Wattenhofer (2022) investigated the expressiveness of specific properties, such as counts for all k-vertex substructures. They presented some specific results on the expressiveness gain such as, showing that GNNs with all counts for k-vertex substructures can count k-cycles, but not $k - 1$ cycles. Our framework provides unified answers for these specific expressivity questions. Barceló et al. (2021) presented one of the early works highlighting the importance of using homomorphism counts with GNNs. Very recently, Welke et al. (2023) propose a randomised approach that uses homomorphism counts to distinguish graphs *in expectation*. The work of Zhang et al. (2024) is closely related to ours, where it has been shown that the expressiveness of GNN architectures under consideration can be characterized in terms of homomorphism counts. This further motivates our approach as will be discussed in Section 4.3.

Other works have studied homomorphism counts for more traditional machine learning models, such as support vector machines (SVMs) and random forest classifiers (Nguyen & Maehara, 2020; Wolf et al., 2023). Grohe (2017) also investigated using homomorphism vectors to represent graphs.

3. Background

Graphs, Homomorphisms, and Invariants. A (undirected) graph G is a tuple (V, E) where V is a set of *vertices* and $E \subseteq V \times V$ is a symmetric *edge* relation. In some contexts we write $V(G)$ and $E(G)$ for the vertices and edges, respectively, of graph G. We say that G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The *induced subgraph* of G by $U \subseteq V(G)$ is the graph $G[U]$ with vertices

U and exactly those edges from G where both ends of the edge are in U. Let ρ be a partition of $V(G)$. The quotient graph G/ρ is the graph where the vertices in each block are contracted into a single vertex, i.e., the vertices of G/ρ are the blocks $B \in \rho$, and there is an edge (B_1, B_2) iff there is an edge (v, u) in G for any $v \in B_1, u \in B_2$.

A *homomorphism* from graph G to graph H is a function $h : V(G) \to V(H)$ such that for every $(v, w) \in E(G)$, $(h(v), h(w)) \in E(H)$. An *isomorphism* between graphs G and H, denoted $G \simeq H$, is a bijection $f : V(G) \to V(H)$ such that $(v, w) \in E(G)$ if and only if $(f(v), f(w)) \in E(H)$, for each $v, w \in V(G)$. We write $\mathsf{Hom}(G, H)$ for the number of homomorphisms from G to H, $\mathsf{Sub}(G, H)$ for the number of subgraphs H' of H such that $H' \simeq G$, and $\mathsf{IndSub}(G, H)$ for the number of sets $U \subseteq V(H)$ such that $H[U] \simeq G$, i.e., the number of times G is an (induced) subgraph of H. By $\mathsf{Hom}(G, \cdot)$, we denote the function that maps graph H to $\mathsf{Hom}(G, H)$, and analogously for Sub. For a set of graphs $\mathbb{G}$, we write $\mathsf{Hom}(\mathbb{G})$ to represent the set of functions $\{\mathsf{Hom}(G, \cdot) \mid G \in \mathbb{G}\}$.

A *graph invariant* ξ is a function on graphs such that for all graphs G, H, and all isomorphisms f between G and H, we have $\xi(G) = \xi(H)$. A graph invariant *distinguishes* two graphs G and H, denoted $G \not\equiv_\xi H$, if $\xi(G) \neq \xi(H)$, and conversely, we write $G \equiv_\xi H$ if $\xi(G) = \xi(H)$.

Graph Neural Networks. Let us consider featured graphs $G = (V, E, \zeta)$ where $\zeta : V(G) \to \mathbb{R}$, and so $\zeta(u)$ denotes the feature of a node $u \in V$. All notions introduced earlier extend to featured graphs in the usual way. We focus on *Message-Passing Neural Networks (MPNNs)* (Gilmer et al., 2017) that encapsulate the vast majority of GNNs. An MPNN updates the initial node representations $\boldsymbol{h}_v^{(0)} = \zeta(v)$ of each node v for $0 \leq \ell < L$ iterations based on its own state and the state of its neighbors $\mathcal{N}_v$ as:

$$\boldsymbol{h}_v^{\ell+1} = \text{UPD}^{(\ell)} \left(\boldsymbol{h}_v^\ell, \text{AGG}^{(\ell)} \left(\boldsymbol{h}_v^\ell, \{\!\{\boldsymbol{h}_u^\ell \mid u \in \mathcal{N}_v\}\!\} \right) \right),$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset and $\text{UPD}^{(\ell)}$ and $\text{AGG}^{(\ell)}$ are differentiable *update* and *aggregation* functions, respectively. We denote by $d^{(\ell)}$ the dimension of the node embeddings at iteration (layer) ℓ. The final representations $\boldsymbol{h}_v^{(L)}$ of each node v can be used for predicting node-level properties, or they can be pooled to form a graph embedding vector $\boldsymbol{z}_G$, which can be used for predicting graph-level properties.

Weisfeiler-Leman Hierarchy. We will refer to the 1-WL algorithm (Weisfeiler & Lehman, 1968), as well as to the usual k-WL hierarchy, which are iterative algorithms that yield graph invariants. We will always refer to the *folklore* version of these algorithms (k-WL), rather than the oblivious version (oblivious k-WL), where the only difference is in the dimension counts: k-WL is equivalent to oblivious $(k + 1)$-WL, for every $k > 1$ (Grohe, 2017; Grohe & Otto, 2015).

Graph Motif Parameters. A *graph parameter* is a function that maps graphs into $\mathbb{Q}$. A *graph motif parameter* (Curticapean et al., 2017) is a graph parameter that can be expressed as a finite linear combination of homomorphism counts into G. Formally, Γ is a graph motif parameter, if there are graphs $F_1, \ldots, F_\ell$ and a function α mapping them to $\mathbb{Q} \setminus \{0\}$, such that for every graph G:

$$\Gamma(G) = \sum_{i=1}^\ell \alpha(F_i) \cdot \mathsf{Hom}(F_i, G). \tag{1}$$

We say that a motif parameter Γ is *connected* if every graph in $\mathsf{Supp}(\Gamma)$ is connected. Informally we also refer to the graphs in the support as the *homomorphism basis* of Γ.

For any graph F, $\mathsf{Sub}(F, \cdot)$ is a graph motif parameter. Its support, commonly called $\mathsf{Spasm}(F)$, is the set of all quotient graphs of F (up to isomorphism) (Lovász, 2012). When the host graphs are assumed to be loop-free, the Spasm is implicitly understood to only contain the loop-free quotients. If F is connected, $\mathsf{Sub}(F, \cdot)$ is a connected graph motif parameter. As an example, recall the quotients of ⬠ as illustrated in Example 1.4[1].

Expressing a Graph Parameter. We say that a GNN architecture Ψ can *express* a graph parameter f if for every pair of graphs G, H, there exists a model parametrization Ψ_θ such that $G \equiv_{\Psi_\theta} H$ implies $f(G) = f(H)$. In other words, there can be no situation where two graphs are equivalent to the model Ψ_θ but f is different on the two graphs. We are interested in the question of how the expressiveness gained by different sets of additional features compare to one another. Thus, we extend our notion of expressiveness to the case where Ψ additionally has access to a set of additional feature maps $\mathbb{A}$, i.e., invariant functions that map the input graph to a rational number. We say that a GNN architecture Ψ with $\mathbb{A}$ can *express* a graph parameter f if for every pair of graphs G, H, there exists a model parametrization Ψ_θ such that if $G \equiv_{\Psi_\theta} H$ and $a(G) = a(H)$ for all $a \in \mathbb{A}$, then $f(G) = f(H)$. For sets of features $\mathbb{A}, \mathbb{A}'$ and GNN Ψ, we say that Ψ with $\mathbb{A}$ is *at least as expressive* as Ψ with $\mathbb{A}'$ if the former expresses all functions that are expressed by the latter. Similarly, Ψ with $\mathbb{A}'$ is *strictly more expressive* than Ψ with $\mathbb{A}$ if the former is at least as expressive and can express strictly more functions than the latter.

4. All About the Homomorphism Basis

We study the expressivity of GNNs with respect to graph motif parameters and propose an efficient method for injecting such parameters with provably stronger expressiveness in addition to a variety of further benefits.

---

[1] We omit the technical discussion of how precisely the coefficients are obtained; refer instead to Curticapean et al. (2017).

4.1. It Is More Expressive

Recall that $\mathbb{A} = \{\mathsf{Sub}(\pentagon, \cdot)\}$ and $\mathbb{A}' = \{\mathsf{Hom}(\pentagon, \cdot)\}$ are incomparable if added to 1-WL expressiveness (Example 1.1). This also holds in general for arbitrary patterns that have more than one acyclic graph in their basis (follows from Theorem 4.1). At the same time, 1-WL with either kind of additional feature can be more expressive than plain 1-WL, as previously observed by Bouritsas et al. (2023) and Barceló et al. (2020). We show that the homomorphism basis of $\mathsf{Sub}(\pentagon, \cdot)$ is more expressive than either approach. In fact, we prove a much more general and stronger statement. For (almost) every connected graph motif parameter Γ, providing the homomorphism basis $\mathbb{B}_\Gamma$ as additional features is *strictly* more expressive than injecting Γ.

Theorem 4.1. *Let Γ be a connected graph motif parameter and $k \geq 1$. Then k-WL with $\mathbb{B}_\Gamma$ is at least as expressive as k-WL with $\{\Gamma\}$. Moreover, if at least two functions in $\mathsf{Hom}(\mathsf{Supp}(\Gamma))$ cannot be expressed by k-WL with $\{\Gamma\}$, then Ψ with $\mathbb{B}_\Gamma$ is strictly more expressive.*[2]

We wish to emphasize that the gain in expressiveness observed in Theorem 4.1 is not only a matter of narrow theoretical increase. This is illustrated well by subgraph counting. Recall the basis of $\Gamma = \mathsf{Sub}(\pentagon, \cdot)$ from Example 1.4. There, 1-WL with $\mathbb{B}_\Gamma$ does not only distinguish $\mathsf{Sub}(\pentagon, \cdot)$ but also $\mathsf{Sub}(\checkmark, \cdot)$ and $\mathsf{Sub}(\triangle, \cdot)$. In general, we observe the following.

Proposition 4.2. *Let Ψ be a GNN, let G be a graph, and let $F \in \mathsf{Spasm}(G)$. Then $\mathsf{Sub}(F, \cdot)$ can be expressed by Ψ with $\mathbb{B}_{\mathsf{Sub}(G, \cdot)}$.*

4.2. It Is Computationally Efficient

In practice, the additional features given by $\mathbb{A}$ to a GNN must be computed externally for each input graph. Curticapean et al. (2017) have shown that, in slightly simplified terms, the most efficient way to compute $\Gamma(G)$ is to compute the terms of the homomorphism basis and then obtain $\Gamma(G)$ via Equation (1). Hence, providing a GNN with $\mathbb{B}_\Gamma$ is not just more expressive but also computationally no more effort than providing Γ itself. The precise technical statements are intricate as they require tools from parameterized complexity theory; we refer the interested reader to Appendix B for formal details.

Popular subgraph counting implementations often exhibit significantly worse complexity than required to compute $\mathsf{Hom}(\mathsf{Supp}(\Gamma))$. Typically, to compute $\mathsf{Sub}(F, G)$ they have running times exponential in $|V(G)|$ (cf., Ribeiro et al. (2022)). Earlier works on injecting structural information in GNNs focus on particular settings where this

worst-case behavior is not observed: sparse input graphs such as molecules and special patterns like cycles allow for good ad hoc algorithms for such graphs. For slightly more complex graphs or patterns, direct computation of Sub becomes intractable and computation via the homomorphism basis becomes much more efficient also in practice.

4.3. It Avoids Redundancy

While expressiveness is a natural criterion to consider with respect to injecting additional features, we also want to avoid redundant information. In this regard, the homomorphism basis is ideally suited as the expressiveness of GNNs with respect to homomorphism counts is well-understood. In practice, this means that instead of providing all of $\mathbb{B}_\Gamma$ as additional features to distinguish some target function Γ, we can often precisely determine which functions in $\mathbb{B}_\Gamma$ are not already distinguished by the GNN and only inject those.

Recent results provide a clear picture of when a GNN can distinguish a function $\mathsf{Hom}(F, \cdot)$. Neuen (2023) recently showed that $(k-1)$-WL cannot distinguish any function $\mathsf{Hom}(F, \cdot)$ when k is the *treewidth* of F. A classic result by Dvořák (2010) states that k-WL is enough to distinguish all such functions. Geerts & Reutter (2022) showed an upper bound by k-WL for pattern counting based on the treewidth of patterns in the Spasm. This has significant practical implications: for architectures such as GIN (Xu et al., 2019) that reach 1-WL expressiveness, injecting only the cyclic graphs of $\mathbb{B}_\Gamma$ gives equivalent expressiveness to injecting all of $\mathbb{B}_\Gamma$. On the other hand, architectures like GAT (Velickovic et al., 2018) and GCN (Kipf & Welling, 2017) are weaker than 1-WL and thus motivate providing all of $\mathbb{B}_\Gamma$ as additional features to the network. Recently, Zhang et al. (2024) gave precise definitions for the sets of functions $\mathsf{Hom}(F, \cdot)$ that can (and cannot) be expressed by Local k-GNNs, and Subgraph GNNs, thus showing precisely what part of $\mathbb{B}_\Gamma$ can already be expressed by the corresponding GNN architectures.

Furthermore, it is never necessary to consider homomorphism counts from disconnected graphs. Note that this is not trivial but particular to homomorphism counting, e.g., no similar simplification can be made for subgraph counting. See Section 4.5 below for an application of this observation.

Proposition 4.3. *Let Ψ be a GNN, let $\mathbb{G}$ be a set of graphs and let $\mathsf{CC}(\mathbb{G})$ be the set of the (maximal) connected components in $\mathbb{G}$. Then Ψ with $\mathsf{Hom}(\mathsf{CC}(\mathbb{G}))$ is at least as expressive as Ψ with $\mathsf{Hom}(\mathbb{G})$.*

## 4.4. It Works for Node-level Subgraph Information

Previous work has shown the efficacy of providing subgraph counts at node-level. We will use the term *anchored graph* to mean a graph G with a marked vertex $\circledast$. When G is an-

---

[2]We prove a significantly stronger statement showing that this applies to many common GNNs rather than just k-WL. See Appendix A.1 for details.

chored, we write $\mathsf{Sub}(G, H, v)$ for the number of subgraphs H' of H where $G \simeq H'$ such that the marked vertex maps to v in the isomorphism. In particular, Bouritsas et al. (2023) propose labeling each node $v \in H$ with $\mathsf{Sub}(G', H, v)$ for every way G' of marking a graph G.

Local homomorphism counts of the graphs in the Spasm are not enough to determine local subgraph counts (see Appendix C). However, in the following we show how the framework we have used so far naturally extends to this setting. For an anchored graph G, the anchor of quotient G/ρ is the vertex for the block $B \in \rho$ that contains the anchor of G. Analogously to the standard case, $\mathsf{Spasm}^{\circledast}(G)$ is the set of all (loop-free) quotient graphs of G.

Example 4.4. The $\mathsf{Spasm}(\square)$ contains $\square$, $\nearrow$, and $\nearrow$, whereas $\mathsf{Spasm}^{\circledast}(\square)$ consists of the four graphs:
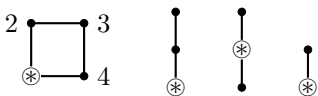
In $\mathsf{Spasm}^{\circledast}$, there are two anchored versions of $\nearrow$. The quotient that identifies vertices 2 and 4 has the anchor at the end of the path, while the quotient that identifies the $\circledast$ and 3 has its anchor on the middle vertex. $\triangle$

With this definition of $\mathsf{Spasm}^{\circledast}$ we can now show that homomorphism counts are indeed also suited to node-level tasks and the previous arguments apply similarly in this setting.

Theorem 4.5. *Let G be an anchored graph and let H be a graph and $v \in V(H)$. Then*

$$\mathsf{Sub}(G, H, v) = \sum_{F \in \mathsf{Spasm}^{\circledast}(G)} \alpha_F \cdot \mathsf{Hom}(F, H)[\circledast \mapsto v]$$

where $\mathsf{Hom}(F, H)[\circledast \mapsto v]$ is the number of homomorphisms that map the anchor of F to $v \in V(H)$.

It follows that the arguments of the previous sections apply analogously also to this setting. With respect to Section 4.3 particularly, Lanzinger & Barceló (2024) showed that $\mathsf{Hom}(F, H)[\circledast \mapsto v]$ can be computed from the node-level k-WL labeling for every graph F of treewidth k. Hence, in our example above, node-level homomorphism counts of $\square$ to 1-WL are sufficient additional information to determine node-level subgraph counts of the four cycle.

4.5. It Goes Beyond Pattern Counting

We have mainly focused on the setting where a GNN is provided additional information motivated by some target structure that is considered important to the task (e.g., cycles). However, the framework of graph motif parameters also motivates a natural alternative approach that is not dependent on such structure selection.

For some (decidable) graph property ϕ, let $\mathsf{IndSub}_k^{\phi}(G)$ be the function that maps G to the number of induced subgraphs in G with k vertices that satisfy ϕ. Through the choice of ϕ – e.g., connectedness, hamiltonicity, planarity – IndSub_k^{ϕ} can cover an immense variety of graph parameters. For example, $\mathsf{IndSub}(F, G) = \mathsf{IndSub}_{|V(F)|}^{\psi}(F)$ where ψ is the property of being isomorphic to F and when ϕ is connectedness, $\mathsf{IndSub}_k^{\phi}(G)$ is the number of k-graphlets in G, a popular metric in graph analysis (see Ribeiro et al. (2022)). It is known that all such functions IndSub_k^{ϕ} are indeed graph motif parameters (Roth & Schmitt, 2020).

For some settings where there is no obvious kind of structure to inject information for, this presents an attractive alternative. In fact it is possible to provide practical amounts of additional information to a GNN, such that it can distinguish *all* functions IndSub_k^{ϕ} for small k. In concrete terms, by inspection of the arguments by Roth & Schmitt (2020) in combination with Proposition 4.3, we observe the following.

Proposition 4.6. *Let Ψ be a GNN, let ϕ be a decidable graph property, and let $k \geq 1$. Let $\Omega_{\leq k}^{\mathrm{con}}$ be the set of connected graphs with at most k vertices. Then IndSub_k^{ϕ} can be expressed by Ψ with $\mathsf{Hom}(\Omega_{\leq k}^{con})$.*

The possible applications of our approach range even further. Counting the satisfying assignments for certain fragments of first-order logic in a graph is also a graph motif parameter (Dell et al., 2019). This presents the possibility of targetedly injecting information to express logic-specified properties.

5. Experiments

We empirically validate our results on a variety of real-world tasks and architectures. Specifically, we conduct experiments for molecular property prediction with ZINC (Section 5.1) and QM9 (Section 5.2), and for link prediction with COLLAB (Section 5.3). We complement our empirical analysis with a dedicated expressiveness experiment on the BREC benchmark (Section 5.4). Full experimental details are reported in Appendix D.

5.1. Graph Regression Experiments on ZINC

Experimental Setup. ZINC (Irwin et al., 2012; Dwivedi et al., 2023) is a graph regression task for predicting the constrained solubility of molecules. Following Dwivedi et al. (2023), we use a subset of the dataset that contains 12,000 graphs. Each graph represents an individual molecule, and the original node features denote the atom type of a given node. We select GAT, GCN, and GIN, using the same experimental protocol and hyperparameters from Dwivedi et al. (2023). We also select BasePlanE (Dimitrov et al., 2023), a recent architecture that is complete over planar graphs. For all models, we do not perform any additional model tuning.

Table 1. We report the mean absolute error (MAE) for graph regression on the ZINC-12K dataset (without edge features). Using the Spasm and Spasm$^\circledast$ counts yields very substantial improvements in performance of every model compared to their respective baselines and other approaches using $C_3, ..., C_8$.

	GAT	GCN	GIN	BasePlanE
Base	0.457 ± 0.004	0.417 ± 0.007	0.294 ± 0.012	0.124 ± 0.004
Sub	0.210 ± 0.006	0.206 ± 0.006	0.147 ± 0.006	0.108 ± 0.002
Hom	0.269 ± 0.033	0.254 ± 0.017	0.208 ± 0.025	0.106 ± 0.004
Spasm	0.155 ± 0.006	0.166 ± 0.003	0.158 ± 0.004	0.104 ± 0.005
Spasm$^\circledast$	$\mathbf{0.147} \pm 0.004$	$\mathbf{0.165} \pm 0.004$	$\mathbf{0.146} \pm 0.005$	0.100 ± 0.002

Our experiment compares the following configurations:

- Base: The base performance of the GAT, GCN, GIN, and BasePlanE model architectures.
- Sub: The baseline models enriched with *subgraph* counts of cycles $\{C_3, ..., C_8\}$.
- Hom: The baseline models enriched with *homomorphism* counts of cycles $\{C_3, ..., C_8\}$.
- Spasm: The baseline models enriched with homomorphism counts of Spasm$(C_7) \cup$ Spasm$(C_8)^3$.
- Spasm$^\circledast$: The baseline models enriched with homomorphism counts of Spasm$^\circledast(C_7) \cup$ Spasm$^\circledast(C_8)$.

In all cases we inject the counts at node level, making Sub and Hom correspond directly to the approaches of Bouritsas et al. (2023) and Barceló et al. (2021), respectively. This allows us to precisely compare the effect of the homomorphism basis to these previous works. For Spasm and Spasm$^\circledast$, we can obtain the global and local subgraph counts, respectively, so we include those counts as well. In all cases, the raw counts are encoded using a 2-layer MLP. Following the analysis of Section 4.3, we exclude counts for the acyclic graphs of the basis for GIN and BasePlanE experiments but add them for GAT and GCN.

Results. Our results are summarized in Table 1. In all models, adding the Spasm$^\circledast$ counts as additional node features yields the best performance. Despite ZINC being a graph-level task, we observe that precise node-level subgraph information is critical for top performance as Spasm$^\circledast$ outperforms Spasm. Additionally, in GAT and GCN, we clearly see the improvements in expressiveness from Theorem 4.1 in effect as Spasm already clearly outperforms Sub. This is further confirmed by the effectively equivalent performance of GAT/Spasm$^\circledast$, GIN/Spasm$^\circledast$, and GIN/Sub. That is, their performance is intuitively the performance of 1-WL together with node-level subgraph counts (up to C_8). Any additional expressiveness provided by Spasm$^\circledast$ is seemingly not relevant to the ZINC task. Since GIN is as expressive as 1-WL, little is gained, whereas GAT manages

---

^{3}Note that Spasm$(C_7) \cup$ Spasm(C_8) also contain the basis graphs for $C_3, C_4, C_5,$ and C_6 (recall Proposition 4.2).

to match GIN through the increased expressiveness from the homomorphism basis. Moreover, the gap between Spasm and Spasm$^\circledast$ illustrates the discussion in Section 4.4 and in particular the effect of Spasm$^\circledast$ as predicted in Theorem 4.5.

For BasePlanE, we observe that using the homomorphism basis instead of the parameter itself results in improved performance. This is insightful: even though BasePlanE can distinguish planar graphs (and all ZINC graphs are planar), distinguishing graphs is a weaker notion than capturing graph motif parameters, so injecting these parameters leads to empirical improvements even for stronger models specifically designed for planar graphs (see also Appendix D.3).

Using the Entire Basis. To study the effect of the homomorphism basis in more detail, we perform experiments where we inject increasingly more (in order of the number of vertices, lowest first) elements of the basis as features. To avoid the effect of the complex interplay of odd/even cycles on the benchmark, we use only Spasm(C_8). Figure 3 clearly shows that the performance of every model improves as more of the basis is included. Moreover, just as in Table 1, GAT and GIN converge in performance, again confirming that their expressiveness (as relevant to this task) matches with this additional information.
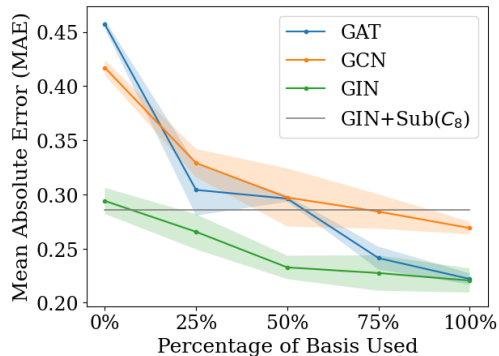

Figure 3. We report the effect of using incrementally more of Spasm(C_8) as node features for each model. The performance of GIN with Sub$(C_8, \cdot)$ at node-level is also included for reference.

Table 2. We report the mean absolute error (MAE) for graph regression on QM9 for all 13 proprieties. Using the homomorphism counts yields significant improvements over the baseline model.
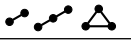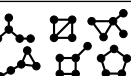
PROPERTY	R-GCN	+ $\mathsf{Hom}(\Omega^{con}_{\leq 5} \cup \{C_6\})$
MU	3.21 ±0.06	**2.29** ±0.03
ALPHA	4.22 ±0.45	**1.77** ±0.05
HOMO	1.45 ±0.01	**1.30** ±0.03
LUMO	1.62 ±0.04	**1.41** ±0.02
GAP	2.42 ±0.14	**2.00** ±0.04
R2	16.38 ±0.49	**10.29** ±0.35
ZPVE	17.40 ±3.56	**3.03** ±0.38
U0	7.82 ±0.80	**1.09** ±0.18
U	8.24 ±1.25	**1.21** ±0.17
H	9.05 ±1.21	**1.22** ±0.14
G	7.00 ±1.51	**1.14** ±0.13
Cv	3.93 ±0.48	**1.46** ±0.08
OMEGA	1.02 ±0.05	**0.81** ±0.02

Table 3. We report the Hits@50 metric for link prediction on COL-LAB. We use the homomorphism basis of n-vertex paths P_n and cliques K_n. The first column shows the additional graphs in the Spasm of each pattern. Since $\mathsf{Spasm}(P_\ell) \subseteq \mathsf{Spasm}(P_{\ell+1})$ we show only the additional basis elements in the respective rows.

ADD. BASIS	PATTERN	GCN	GAT
—		46.13%±2.10	48.27%±1.05
	K_3	49.41%±0.42	49.43%±0.73
	K_4	47.76%±0.53	48.35%±0.85
	K_5	48.01%±0.87	49.75%±0.16
	P_4	49.59%±0.23	50.76%±0.51
	P_5	49.60%±0.29	51.55%±0.94
	P_6	**50.35%**±0.21	**51.62%**±0.66

5.2. Graph Regression Experiments on QM9

Experimental Setup. QM9 is real-world molecular dataset consisting of 130,000 graphs (Wu et al., 2018; Brockschmidt, 2020) for graph regression over 13 different target properties. We select R-GCN (Schlichtkrull et al., 2018) as our base model due to its comparatively stable behavior in the original benchmark (Brockschmidt, 2020). Contrasting Section 5.1, we inject only graph-level homomorphism counts (rather than node-level). The additional features are injected only once as additional inputs to the final MLP. The regression tasks in QM9 target a variety of quantum chemical properties of the molecule and it is unclear which specific pattern information might be helpful. We thus follow the ideas from Section 4.5 and inject counts for $\Omega^{con}_{\leq 5}$ (connected graphs with at most 5 vertices) and the 6-cycle C_6. This set contains all of $\mathsf{Spasm}(C_6)$. Baseline results for R-GCN are from Brockschmidt (2020).

Results. Table 2 shows that using the homomorphism counts for $\Omega^{con}_{\leq 5}$ and C_6 significantly improves the performance of the base R-GCN model on all properties, even without additional model tuning. Alon & Yahav (2021) showed that the performance of R-GCN on this task can also be boosted with a fully-adjacent (FA) layer. We report performance increases also in combination with FA in Appendix D.4 (see Table 9). These observations demonstrate the well-foundedness of our approach as it confirms the practical relevance of Section 4.5 and Proposition 4.6. Moreover, we see that the addition of graph level homomorphism counts alone are sufficient for substantial performance improvements. This is particularly noteworthy as it simplifies integration with more complex models, where node-level injection of many features might affect hyperparameters significantly. To the best of our knowledge, this is also the first study demonstrating the potential of injecting graph-level homomorphism information to a GNN.

5.3. Link Prediction on COLLAB

So far, we have demonstrated multiple ways in which our approach is effective on graph-level regression tasks on small graphs. In this experiment, we contrast this by performing a link prediction task on a graph where direct subgraph counting is intractable for most patterns.

Experimental Setup. We evaluate the link prediction task for the COLLAB dataset from Open Graph Benchmark (Hu et al., 2020). The dataset contains a single undirected graph with over 235,000 nodes – each with 128 features – and roughly 1.3 million edges, which represents a collaboration network. Previous works (Barceló et al., 2021) enriched the node features by the number of cliques (i.e., K_3, K_4, K_5) the node is part of. We select GAT and GCN as our baseline models and enrich their node features with the node-level homomorphism counts of the Spasm of different-sized cliques and paths. The Spasm of a clique contains only itself, and the clique experiments thus match the experiments conducted by Barceló et al. (2021). To control for high counts values that occur in large graphs, we encode them via positional encoding (Vaswani et al., 2017). We provide a more in-depth analysis of the importance of using the positional encoding later on.

Results. From Table 3 we see that our approach of injecting the basis homomorphism counts for paths outperforms the previously studied injection of clique counts in both models. This further demonstrates the targeted gain of expressiveness, irrespective of task, that can be realized through our approach. We note that while short paths are seemingly simple structures, actually counting them is not expressible in 1-WL as illustrated by the cyclic graphs in the basis (and thus also in the tested, weaker models). Despite the large graph size, we compute all necessary node-level homomorphisms counts in roughly 3 hours (see Appendix D.1).
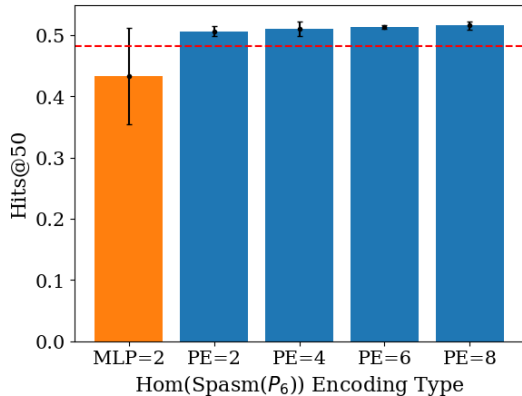
Figure 4. We report the effect of using different encoding methods and dimensions for the $\mathsf{Spasm}(P_6)$ homomorphism count features with GAT on COLLAB. The dashed red line indicates the performance of GAT without additional features. Using a positional encoding (PE) boosts performance of the baseline model, whereas a standard MLP encoder degrades performance.

Ablation with Positional Encoding. To study the impact of normalizing homomorphims counts with a sinusoidal positional encoding (Vaswani et al., 2017), we perform an ablation with GAT on the COLLAB dataset. We compare the performance of using an MLP encoder to using a positional encoder for the $\mathsf{Hom}(\mathsf{Spasm}(P_6))$ features. From Figure 4, we see that including counts via an MLP encoder actually degrades performance of the baseline model and introduces high variance. This is likely because homomorphism count values can grow extremely large in larger graphs, causing the model to treat the extra information as noise. Conversely, using a positional encoding allows us to normalize the raw homomorphism count values in a way that preserves the relative distances between different patterns across different graphs. However, adjusting the positional encoding dimension presented minimal effects in our ablation.

5.4. Expressiveness Evaluation on BREC

BREC is a recent expressiveness benchmark containing 400 pairs of non-isomorphic graphs (Wang & Zhang, 2023). The benchmark tests how capable models are at distinguishing the graphs in these pairs. The pairs are split into four categories: Basic, Regular, Extension and CFI graphs.

Experimental Setup. We select GIN and PPGN (Maron et al., 2019a) as the two models to extend with node-level homomorphism counts. We again opt against choosing a particular pattern and instead inject the homomorphism counts of $\Omega_{\leq 5}^{\mathsf{con}}$ as node features in both cases. For GIN + $\mathsf{Hom}(\overline{\Omega_{\leq 5}^{\mathsf{con}}})$, we use the same experimental setup as was used for GSN in (Wang & Zhang, 2023), replacing subgraph counts directly with homomorphism counts. For PPGN + $\mathsf{Hom}(\Omega_{\leq 5}^{\mathsf{con}})$ we normalize features in accordance with Maron et al. (2019a).

Table 4. Summary of BREC expressiveness experiments. Percentages refer to the percentage of graph pairs in each category that were correctly distinguished by the model.

	Bas.	Reg.	Ext.	CFI	Total
2-WL[4]	100%	35.7%	100%	60%	67.5%
PPGN	100%	35.7%	100%	23%	58.2%
GSN	100%	70.7%	95%	0%	63.5%
I^2-GNN	100%	71.4%	100%	21%	70.2%
GIN+Hom($\Omega_{<5}^{\mathsf{con}}$)	100%	85.7%	96%	0%	69%
PPGN+Hom($\overline{\Omega}_{\leq 5}^{\mathsf{con}}$)	100%	85.7%	100%	25%	**76.25%**

Results. A summary of results is presented in Table 4. It compares our results to the previous state of the art I^2-GNN, and the closely related models PPGN and GSN. We achieve state of the art results using PPGN + $\mathsf{Hom}(\Omega_{\leq 5}^{\mathsf{con}})$. Notably, in the category of regular graphs, where PPGN was initially lacking, the inclusion of our approach enables the model to distinguish over twice as many graphs. This experimentally illustrates our analysis in Section 4.3 in the context of higher-order models: $\Omega_{\leq 5}^{\mathsf{con}}$ contains graphs with treewidth greater than 2 (in particular 4 graphs containing a 4-clique). These homomorphism counts thus improve the 2-WL[4] expressiveness of PPGN. GIN + $\mathsf{Hom}(\Omega_{\leq 5}^{\mathsf{con}})$ also performs well by ranking 4th overall, outperforming many higher-order models (see Appendix D.6 for full results).

6. Discussion and Future Work

We have proposed the injection of the homomorphism basis of graph parameters into GNNs as a flexible framework for increasing expressiveness in a targeted and efficient manner. We experimentally validate that this increase in expressiveness is reflected in significant performance gains in real-world GNN tasks. This lays the foundation for a wide-range of further applications: both node- and graph-level homomorphism counts have proven to boost performance in a variety of tasks and architectures. Our experiments have primarily focused on standard architectures but motivate combination with complex top performing models to further boost their performance. Moreover, the use of homomorphism bases induced by logical formulas (Section 4.5) has been beyond the scope of this paper but presents numerous interesting possibilities.

The broad applicability of our approach brings with it a number of technical question that require more detailed analysis. In large graphs, homomorphism counts can span a large range of values, opening up questions of how to best encode them to realize the theoretical expressiveness gain. Additionally, the choice of basis to inject depends on the desired expressivity and further work is needed to explore the effects of different bases in common GNN tasks.

---

[4]Recall that we always refer to the folklore version of k-WL.

Acknowledgements

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Abboud, R., Dimitrov, R., and Ceylan, İ. İ. Shortest path networks for graph property prediction. In *LoG*, 2022.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *ICLR*, 2021.

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

Barceló, P., Geerts, F., Reutter, J. L., and Ryschkov, M. Graph neural networks with local graph parameters. In *NeurIPS*, 2021.

Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M. M., and Maron, H. Equivariant subgraph aggregation networks. In *ICLR*, 2022.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):657–668, 2023.

Bressan, M., Lanzinger, M., and Roth, M. The complexity of pattern counting in directed graphs, parameterised by the outdegree. In *STOC*, 2023.

Brockschmidt, M. GNN-FiLM: Graph neural networks with feature-wise linear modulation. In *ICML*, 2020.

Chen, H. and Mengel, S. Counting answers to existential positive queries: A complexity classification. In *PODS*, 2016.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *NeurIPS*, 2020.

Curticapean, R., Dell, H., and Marx, D. Homomorphisms are a good basis for counting small subgraphs. In *STOC*, 2017.

Dell, H., Roth, M., and Wellnitz, P. Counting answers to existential questions. In *ICALP*, 2019.

Díaz, J., Serna, M. J., and Thilikos, D. M. Counting h-colorings of partial k-trees. *Theor. Comput. Sci.*, 281 (1-2):291–309, 2002.

Dimitrov, R., Zhao, Z., Abboud, R., and İsmail İlkan Ceylan. PlanE: representation learning over planar graphs. In *NeurIPS*, 2023.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015.

Dvorák, Z. On recognizing graphs by numbers of homomorphisms. *J. Graph Theory*, 64(4):330–342, 2010.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Frasca, F., Bevilacqua, B., Bronstein, M. M., and Maron, H. Understanding and extending subgraph GNNs by rethinking their symmetries. In *NeurIPS*, 2022.

Geerts, F. and Reutter, J. L. Expressiveness and approximation properties of graph neural networks. *ICLR*, 2022.

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

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *IJCNN*, 2005.

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

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

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

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

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Keriven, N. and Peyré, G. Universal invariant and equivariant graph neural networks. In *NeurIPS*, 2019.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Lanzinger, M. and Barceló, P. On the power of the weisfeiler-leman test for graph motif parameters. In *ICLR*, 2024.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *ICLR*, 2020.

Lovász, L. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

Lovász, L. *Large Networks and Graph Limits*, volume 60 of *Colloquium Publications*. American Mathematical Society, 2012.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *NeurIPS*, 2019a.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In *ICML*, 2019b.

Marx, D. Can you beat treewidth? *Theory Comput.*, 6(1):85–112, 2010.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

Neuen, D. Homomorphism-distinguishing closedness for graphs of bounded tree-width. *arXiv preprint arXiv:2304.07011*, 2023.

Nguyen, H. and Maehara, T. Graph homomorphism convolution. In *ICML*, 2020.

Papp, P. A. and Wattenhofer, R. A theoretical comparison of graph neural network extensions. In *ICML*, 2022.

Ribeiro, P., Paredes, P., Silva, M. E. P., Aparício, D., and Silva, F. M. A. A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets. *ACM Comput. Surv.*, 54(2):28:1–28:36, 2022.

Roth, M. and Schmitt, J. Counting induced subgraphs: A topological approach to #W[1]-hardness. *Algorithmica*, 82(8):2267–2291, 2020.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *SDM*, 2021.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

Shlomi, J., Battaglia, P., and Vlimant, J.-R. Graph neural networks in particle physics. In *MLST*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Wang, Y. and Zhang, M. Towards better evaluation of gnn expressiveness with brec dataset. *arXiv preprint arXiv:2304.07702*, 2023.

Weisfeiler, B. and Lehman, A. A. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

Welke, P., Thiessen, M., Jogl, F., and Gärtner, T. Expectation-complete graph representations with homomorphisms. In *ICML*, 2023.

Wolf, H., Oeljeklaus, L., Kühner, P., and Grohe, M. Structural node embeddings with homomorphism counts. *arXiv preprint arXiv:2308.15283*, 2023.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Zhang, B., Feng, G., Du, Y., He, D., and Wang, L. A complete expressiveness hierarchy for subgraph GNNs via subgraph Weisfeiler-Lehman tests. In *ICML*, 2023a.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the expressive power of Gnns via graph biconnectivity. In *ICLR*, 2023b.

Zhang, B., Gai, J., Du, Y., Ye, Q., He, D., and Wang, L. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness. In *ICLR*, 2024.

Zitnik, M., Agrawal, M., and Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

A. Technical Details

Proof of Proposition 4.2. Since $\mathsf{Sub}(G, \cdot)$ is a graph motif parameter with support equal to $\mathsf{Spasm}(G)$, we have that $\mathsf{Sub}(G, \cdot)$ is determined by $\mathsf{Hom}(\mathsf{Spasm}(G))$ since it functionally depends on the terms of $\mathsf{Hom}(\mathsf{Spasm}(G))$.

Observe that if $F \in \mathsf{Spasm}(G)$, then $\mathsf{Spasm}(F) \subseteq \mathsf{Spasm}(G)$. In particular, $H \in \mathsf{Spasm}(F)$ implies that $H \simeq F/\rho$ for some partition ρ of $V(F)$. Furthermore, $F \simeq G/\rho'$ where ρ' is some partition of $V(G)$ since $F \in \mathsf{Spasm}(G)$. Thus, ρ is a partition of blocks of ρ', and hence there is also a partition ρ'', obtained by applying the two partitions one after another, such that $H \simeq G/\rho''$. $\qquad \square$

Proof of Proposition 4.6. Investigation of the proof of Roth & Schmitt (2020, Theorem 12) reveals that $\mathsf{Supp}(\mathsf{IndSub}_k^\phi) \subseteq \Omega_k$. Thus, in relation to our setting this means that a GNN Ψ with $\mathsf{Hom}(\Omega_k)$ expresses IndSub_k^ϕ. We then apply Proposition 4.3 to observe the final statement. $\qquad \square$

A.1. Theorem 4.1

We will prove a more general version of the theorem in the main body that holds not just for the steps of the k-WL hierarchy, but for all GNN models that follow a certain natural behaviour with respect to homomorphisms. It is well known that $G \equiv_{k-\mathrm{WL}} H$ if and only if $\mathsf{Hom}(F, G) = \mathsf{Hom}(F, H)$ for all graphs F of treewidth at most k (Dvořák, 2010). Furthermore, for every graph F with treewidth greater than k, there are G, H with $G \equiv_{k-\mathrm{WL}} H$ where $\mathsf{Hom}(F, G) \neq \mathsf{Hom}(F, H)$, i.e., the class of graphs with treewidth at most k is the maximal class for which homomorphism vector equivalence is equivalent to k-WL equivalence (Neuen, 2023). This means that there is a concrete set of graphs whose homomorphism counts precisely determine the expressivity of the model. Zhang et al. (2024) recently showed that this property also holds true for other commonly studied models that do not precisely match the expressiveness of the k-WL hierarchy, e.g., Subgraph GNNs and Local k-GNNs.

Definition A.1. Let G, H be graphs and let $\mathcal{F}$ be a set of graphs. We say that G and H are *homomorphism indistinguishable* under $\mathcal{F}$, and we write $G \equiv_{\mathcal{F}} H$, if $\mathsf{Hom}(F, G) = \mathsf{Hom}(F, H)$ for all $F \in \mathcal{F}$.

Definition A.2. We say that a class of graphs $\mathcal{F}$ is *closed under restriction to connected components* if for every graph $F \in \mathcal{F}$, every *maximal* connected component of F is in also in $\mathcal{F}$.

Definition A.3. Let Ψ be a GNN model. We say that Ψ is *strongly homomorphism expressive* if there is a set of graphs $\mathcal{F}$ such that:

 (i) for every pair of graphs G, H, it holds that $G \equiv_{\Psi} H$ if and only if $G \equiv_{\mathcal{F}} H$,

 (ii) $\mathcal{F}$ is closed under restriction to connected components,

 (iii) and $\mathcal{F}$ is the maximal set satisfying the first property.

We refer to $\mathcal{F}$ as the homomorphism expressiveness of Ψ.

As mentioned above, any GNN model with expressiveness equal to k-WL is strongly homomorphism expressive with $\mathcal{F}$ being the set of treewidth k graphs. Zhang et al. (2024) showed that Local k-GNNs, local k-FGNNs, and Subgraph GNNs are also strongly homomorphism expressive.

In this light it also becomes clear when the expressiveness of strongly homomorphism expressive Ψ with $\mathsf{Hom}(\mathsf{Spasm}(G))$ is strictly more expressive than Ψ with $\mathsf{Hom}(G, \cdot)$: exactly when there is some $F \in \mathsf{Spasm}(G)$ that is not in the homomorphism expressiveness of Ψ.

The theorem we will prove in the rest of this section is the following stronger version of Theorem 4.1.

Theorem A.4. *Let Γ be a connected graph motif parameter and let Ψ be a strongly homomorphism expressive GNN model. Then Ψ with $\mathbb{B}_\Gamma$ is at least as expressive as Ψ with $\{\Gamma\}$. Moreover, if at least two functions in $\mathsf{Hom}(\mathsf{Supp}(\Gamma))$ are not expressed by Ψ with $\{\Gamma\}$, then Ψ with $\mathbb{B}_\Gamma$ is strictly more expressive.*

Observe that the condition for the strictly more expressive case cannot be improved. If only one term $\mathsf{Hom}(F^*, \cdot)$ is not expressed by Ψ, then for every G, H with $\Gamma(G) = \Gamma(H)$ and $G \equiv_\Psi H$ we have from the former that

$$\sum_{F \in \mathsf{Supp}(\Gamma)} \alpha(F)\mathsf{Hom}(F, G) = \sum_{F \in \mathsf{Supp}(\Gamma)} \alpha(F)\mathsf{Hom}(F, H).$$

Since all terms in the sum, except for F^* are expressed by Ψ, those terms must be equal on both sides. Hence we are left with $\mathsf{Hom}(F^*, G) = \mathsf{Hom}(F^*, H)$ whenever $\Gamma(G) = \Gamma(H)$.

The technical challenge in proving Theorem A.4 comes from the analysis of how equality under Ψ interacts with equality under Γ. The definition of strongly homomorphism expressive GNNs guarantees, that for every graph F for which $\mathsf{Hom}(F, \cdot)$ is not expressed by Ψ, there are graphs $G \equiv_\Psi H$ that differ on $\mathsf{Hom}(F, \cdot)$. On a technical level, the existence of such graphs needs to imply that there are graphs that are equivalent under both Ψ and Γ. Our proof shows how to construct such a graph whenever Γ is connected. Our argument requires connectedness of the graph motif parameter in order to control Γ in the construction. In particular, when Γ is connected, it is additive over disjoint union, which ultimately allows us to construct graphs through these operations in a way where Γ is controlled.

We will first observe a number important properties of strongly homomorphism expressive GNNs based on well known properties of Hom. To this end we first define two operations on graphs. For graphs G, H we write $G + H$ for the disjoint union of G and H, i.e., the graph that is obtained by relabeling H to share no vertices with G and then combining their vertices and edges into a single graph. Moreover, we write $G \times H$ for their *categorical product*, which is the graph P with $V(P) = V(G) \times V(H)$ and $((a, u), (b, v)) \in E(P)$ if and only if $(a, b) \in E(G)$ and $(u, v) \in E(H)$. For positive integer c we write cG for the c-fold disjoint union of G with itself, and similarly G^c for the c-fold product.

Proposition A.5 (Lovász (1967)). *For graphs F, G, H, the following statements hold:*

- $\mathsf{Hom}(F + G, H) = \mathsf{Hom}(F, H)\mathsf{Hom}(G, H)$,

- $\mathsf{Hom}(F, G + H) = \mathsf{Hom}(F, G) + \mathsf{Hom}(F, H)$ *if F is connected,*

- $\mathsf{Hom}(F, G \times H) = \mathsf{Hom}(F, G)\mathsf{Hom}(F, H)$.

From the first point of Proposition A.5 it follows that for any disconnected graph G, the homomorphisms from G into any other graph are determined fully by the homomorphism counts from the connected components of G. Proposition 4.3 then follows immediately.

Lemma A.6. *Let Ψ be a strongly homomorphism expressive GNN. Let G, G', H, H' be graphs such that $G \equiv_\Psi G'$ and $H \equiv_\Psi H'$. The following two statements hold:*

1. $G + H \equiv_\Psi G' + H'$

2. $G \times H \equiv_\Psi G' \times H'$

Proof. Let $\mathcal{F}$ be the homomorphism expressiveness of Ψ. Let F be a connected graph in $\mathcal{F}$. We have

$$\mathsf{Hom}(F, G + H) = \mathsf{Hom}(F, G) + \mathsf{Hom}(F, H) = \mathsf{Hom}(F, G') + \mathsf{Hom}(F, H') = \mathsf{Hom}(F, G' + H')$$

where the outer equalities are by Proposition A.5 and the middle equality is by the assumption that $G \equiv_\Psi G'$ and $H \equiv_\Psi H'$. The argument for the second statement is analogous.

This completes the argument for connected F. Since $\mathcal{F}$ is closed under restriction to connected components, every graph $F' \in \mathcal{F}$ is a disjoint union of connected graphs in F. By the first point of Proposition A.5, the equality on all the connected components of F' immediately extends to F'. $\qquad\square$

Proof of Theorem A.4. Let A, B be two graphs in $\mathsf{Supp}(\Gamma)$ that are not expressed by Ψ with Γ. Our plan will be to show that there are graphs G, H that are equivalent under Ψ and have $\Gamma(G) = \Gamma(H)$, but can be distinguished by either $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$.

Claim 1. Assuming Ψ with Γ does not express at least one of $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$, then there are graphs G_A, H_A, G_B, H_B that satisfy the following properties:

- $G_A \equiv_\Psi H_A$ and $G_B \equiv_\Psi H_B$

- $\mathsf{Hom}(A, G_A) \neq \mathsf{Hom}(A, H_A)$ and $\mathsf{Hom}(B, G_B) \neq \mathsf{Hom}(B, H_B)$

- $\max(\mathsf{Hom}(A, G_A), \mathsf{Hom}(A, H_A)) > \max(\mathsf{Hom}(B, G_A), \mathsf{Hom}(B, H_A))$

Proof of claim: The first two items follow immediately by assumption that Ψ cannot express $\mathsf{Hom}(A, \cdot)$ and $\mathsf{Hom}(B, \cdot)$. For the last point, assume the two maxima are the same, say $\mathsf{Hom}(A, G_A) = \mathsf{Hom}(B, G_A) > \mathsf{Hom}(B, H_A)$. Recall that it is known that there always exists a graph X on which $\mathsf{Hom}(A, X) \neq \mathsf{Hom}(B, X)$ ((Lovász, 1967)). Assume $\mathsf{Hom}(A, X) > \mathsf{Hom}(B, X)$ without loss of generality. We have $G_A + X \equiv_\Psi H_A + X$ by Lemma A.6. That is we have $\mathsf{Hom}(A, G_A + X) = \mathsf{Hom}(A, G_A) + \mathsf{Hom}(A, X) > \mathsf{Hom}(B, G_A) + \mathsf{Hom}(B, X) = \mathsf{Hom}(B, G_A + X)$. Similarly, $\mathsf{Hom}(A, H_A + X) = \mathsf{Hom}(A, H_A) + \mathsf{Hom}(A, X) < \mathsf{Hom}(A, G_A) + \mathsf{Hom}(A, X)$. So replacing G_A, H_A with $G_A + X, H_A + X$ satisfies the statement. ◁

If $\Gamma(G_A) = \Gamma(H_A)$ or $\Gamma(G_B) = \Gamma(H_B)$, the theorem would be proven: if Γ would agree on one of the pairs, Ψ with Γ would not express the respective homomorphism count function. Assuming that Γ is indeed the same on these two pairs of graphs, we show that we can always construct graphs G, H that are still equivalent under Ψ and disagree on either $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$, but also have $\Gamma(G) = \Gamma(H)$. We make a key observation on Γ first. Recall that Γ is a *connected* graph motif parameter, thus from Proposition A.5 together with Equation (1), we observe that $\Gamma(G + H) = \Gamma(G) + \Gamma(H)$.

For positive integer i, we define $\delta_i = \Gamma(G_A^i) - \Gamma(H_A^i)$ and similarly $\eta_i = \Gamma(H_B^i) - \Gamma(G_B^i)$ (recall, G^i is the i-fold categorical product). We will only care about η_0, which we therefore refer to as simply η. Assume, w.l.o.g., $\eta > 0$, otherwise switch G_B, H_B accordingly.

For i where $\delta_i > 0$ define $G_i := \eta G_A^i + \delta_i G_B$ an $H_i := \eta H_A^i + \delta_i H_B$.

Claim 2. For every positive integer i where $\delta_i > 0$ we have $\Gamma(G_i) = \Gamma(H_i)$ and $G_i \equiv_\Psi H_i$.

Proof of claim: Recall from above that $\Gamma(cG) = c\Gamma(G)$. For the equality under Γ we then observe that

$$\begin{aligned}
\Gamma(G_i) - \Gamma(H_i) &= \\
\eta\Gamma(G_A^i) + \delta_i\Gamma(G_B) - \eta\Gamma(H_A^i) - \delta_i\Gamma(H_B) &= \\
\delta_i(\Gamma(G_B) - \Gamma(H_B)) + \eta(\Gamma(G_A^i) - \Gamma(H_A^i)) &= \\
\delta_i(-\eta) + \delta_i\eta &= 0
\end{aligned}$$

Since G_i and H_i are constructed from disjoint unions and products of graphs that are equivalent under Ψ, we also have $G_i \equiv_\Psi H_i$ by Lemma A.6. ◁

There are arbitrarily many i such that δ_i has positive sign (if the sign is negative for all δ_i, switch G_A, H_A). As above, we are done if $\mathsf{Hom}(A, G_i) \neq \mathsf{Hom}(A, H_i)$ for any i where $\delta_i > 0$ since we have $G_i \equiv_\Psi H_i$ and $\Gamma(G_i) = \Gamma(H_i)$ (and the same for B). Let us thus assume that this is not the case. For each of i we then get the following equations:

$$\eta\mathsf{Hom}(A, G_A^i) + \delta_i\mathsf{Hom}(A, G_B) = \mathsf{Hom}(A, G_i) = \mathsf{Hom}(A, H_i) = \eta\mathsf{Hom}(A, H_A^i) + \delta_i\mathsf{Hom}(A, H_B) \quad (2)$$

$$\eta\mathsf{Hom}(B, G_A^i) + \delta_i\mathsf{Hom}(B, G_B) = \mathsf{Hom}(B, G_i) = \mathsf{Hom}(B, H_i) = \eta\mathsf{Hom}(B, H_A^i) + \delta_i\mathsf{Hom}(B, H_B) \quad (3)$$

First, suppose that $\mathsf{Hom}(B, G_A^i) = \mathsf{Hom}(B, H_A^i)$. Simplifying Equation (3) gives $\mathsf{Hom}(B, G_B) = \mathsf{Hom}(B, H_B)$ which contradicts our choice of B, H_B, G_B. In the other case we have $\mathsf{Hom}(B, G_A^i) \neq \mathsf{Hom}(B, H_A^i)$. Solving both equations for $\frac{\eta}{\delta_i}$ (recall $\delta_i \neq 0$), identifying and rewriting gives

$$\frac{\mathsf{Hom}(A, H_B) - \mathsf{Hom}(A, G_B)}{\mathsf{Hom}(B, H_B) - \mathsf{Hom}(B, G_B)} = \frac{\mathsf{Hom}(A, G_A^i) - \mathsf{Hom}(A, H_A^i)}{\mathsf{Hom}(B, G_A^i) - \mathsf{Hom}(B, H_A^i)} \quad (4)$$

Observe that the left side of Equation (4) is independent of i, and in fact is the same for arbitrarily other integers j. That is, the ratio on the right must be equal regardless of choice of integers. Recalling that $\mathsf{Hom}(F, G^c) = \mathsf{Hom}(F, G)^c$ from Proposition A.5 we see

$$\frac{\mathsf{Hom}(A, G_A)^i - \mathsf{Hom}(A, H_A)^i}{\mathsf{Hom}(B, G_A)^i - \mathsf{Hom}(B, H_A)^i} = \frac{\mathsf{Hom}(A, G_A)^j - \mathsf{Hom}(A, H_A)^j}{\mathsf{Hom}(B, G_A)^j - \mathsf{Hom}(B, H_A)^j} \quad (5)$$

which contradicts our initial choice of graphs where we had $\max(\mathsf{Hom}(A, G_A), \mathsf{Hom}(A, H_A)) > \max(\mathsf{Hom}(B, G_A), \mathsf{Hom}(B, H_A))$. That is, the ratio on the right-hand side of Equation (4) cannot stay constant. The equation was derived from the assumption that $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$ both (individually) are the same for G_i and

H_i. In consequence, there must be an i for which $\Gamma(G_i) = \Gamma(H_i)$ and $G_i \equiv_\Psi H_i$ but $\mathsf{Hom}(A, G_i) \neq \mathsf{Hom}(A, H_i)$ or $\mathsf{Hom}(B, G_i) \neq \mathsf{Hom}(B, H_i)$. Say Hom is different for A, then Ψ with Γ does not express $\mathsf{Hom}(A, \cdot)$.

$\square$

A.2. Theorem 4.5

In addition to the counting functions introduced in the main body, we will require the counts for two more special kinds of homomorphisms. To that end we write $\mathsf{Inj}(G, H)$ for the number of *injective* homomorphisms from G to H. Moreover, an *automorphism* of G is a homomorphism from G to G, and we write $\mathsf{Aut}(G)$ for the number of automorphisms of G. In particular, we will care about anchored graphs G and the number of automorphisms that map the anchor to itself, which we denote as $\mathsf{Aut}^\circledast(G)$. We write $\mathsf{Sub}(G, H)[\circledast = v]$ for the number subgraphs of G that are isomorphic to anchored graph G when v in H is set as the anchor of H. This is equivalent to $\mathsf{Sub}(G, H, v)$ as defined in the main body of the text but aligns better with the proof.

Finally, we will need to reason about the lattice of vertex partitions of a graph. Let ρ, ρ' be partitions of $V(G)$. We say that ρ is a coarsening of ρ', or $\rho \geq \rho'$ if for every block $B' \in \rho'$ there is a block $B \in \rho$ such that $B' \subseteq B$.

Proof of Theorem 4.5. With the appropriate setup we can follow the same steps as the analogous graph level proofs by Curticapean et al. (2017). Recall that we want to show that

$$\mathsf{Sub}(G, H)[\circledast = v] = \sum_{F \in \mathsf{Spasm}^\circledast(G)} \alpha_F \cdot \mathsf{Hom}(F, H)[\circledast \mapsto v]$$

We start from two basic observations. First

$$\mathsf{Hom}(G, H)[\circledast \mapsto v] = \sum_\rho \mathsf{Inj}(G/\rho, H)[\circledast \mapsto v] \qquad (6)$$

where the sum ranges over all partitions of $V(G)$. The second is that

$$\mathsf{Inj}(G, H)[\circledast \mapsto v] = \mathsf{Aut}^\circledast(G)\mathsf{Sub}(G, H)[\circledast = v] \qquad (7)$$

where $\mathsf{Aut}^\circledast$ is the number of automorphisms of G that map the anchor to itself. Observe that $\mathsf{Hom}(G, H)[\circledast \mapsto v]$ is the upward zeta transformation of $\mathsf{Inj}(G/\rho, H)[\circledast \mapsto v]$ on the partition lattice from the trivial partition $\bot$ where every block is a singleton.

Möbius transformation of Equation (6) then gives us the expression

$$\mathsf{Inj}(G/\rho, H)[\circledast \mapsto v] = \sum_{\rho' \geq \rho} (-1)^{|\rho| - |\rho'|} \left(\prod_{B \in \rho'} (\lambda(\rho, \rho', B) - 1)! \right) \mathsf{Hom}(G/\rho', H)[\circledast \mapsto v] \qquad (8)$$

which, fixing $\rho = \bot$ is

$$\mathsf{Inj}(G, H)[\circledast \mapsto v] = \sum_{\rho' \geq \rho} (-1)^{|V(G)| - |\rho'|} \left(\prod_{B \in \rho'} (|B| - 1)! \right) \mathsf{Hom}(G/\rho', H)[\circledast \mapsto v] \qquad (9)$$

Note that by definition, G/ρ' is isomorphic to a term in $\mathsf{Spasm}^\circledast(G)$ for every partition ρ'. Collecting the terms for every graph in $\mathsf{Spasm}^\circledast(G)$ we then get

$$\mathsf{Inj}(G, H)[\circledast \mapsto v] = \sum_{F \in \mathsf{Spasm}^\circledast(G)} (-1)^{|V(G)| - |V(F)|} \mathsf{Hom}(F, H)[\circledast \mapsto v] \left(\prod_{\substack{B \in \rho' \\ G/\rho' \simeq F}} (|B| - 1)! \right) \qquad (10)$$

Combining this with Equation (7) we get desired expression for $\mathsf{Sub}(G, H)[\circledast = v]$. In particular, the coefficient α_F is

$$\alpha_F = \frac{(-1)^{|V(G)|-|V(F)|}}{\mathsf{Aut}^{\circledast}(G)} \left(\prod_{\substack{B \in \rho' \\ G/\rho' \simeq F}} (|B| - 1)! \right) \tag{11}$$

$\square$

B. Regarding Complexity

In this section we summarize key results from computational complexity theory that pertain to the argument of Section 4.2. For the sake of a simpler and more streamlined presentation we will first focus on the special case of subgraph counting. Afterwards, we discuss how these results apply in general for graph motif parameters and how this relates to practical algorithms. To begin, we thus consider the following algorithmic problem.

#SUB

Input   Graphs G, H
Output $\mathsf{Sub}(G, H)$

The problem is well known to be #P-hard (intuitively, the counting analogue of NP-hardness) and therefore considered to generally be intractable. It is then natural to consider for which pattern graphs – G in the definition of #SUB above – there might be specialized, more efficient algorithms. This particular question is primarily studied in the context of *parameterized complexity theory* and in particular the problem parameterized by the pattern graph G, defined as follows.

P-#SUB

Input   Graphs G, H
Parameter G
Output $\mathsf{Sub}(G, H)$

In the parameterized setting, the classical requirement for tractability is relaxed. For graph G we write $|G|$ for $|V(G)| + |E(G)|$[5]. A *fixed-parameter tractable* (or fpt) algorithm for P-#SUB is an algorithm whose runtime is bounded by $f(|G|)|H|^{O(1)}$ where f is a computable function. That is, we allow some possible superpolynomial effort in G, as long as this yields an algorithm that is polynomial in the usually much bigger host graph H.

In general, it is easy to see that P-#SUB is hard for the complexity class #W[1] (counting the number of k-cliques in a graph is the canonical #W[1]-complete problem when parameterized by k). This class can be understood as the parameterized counting complexity equivalent of NP and it is a standard assumption in parameterized complexity that hardness for #W[1] implies that there is no fpt algorithm for the problem. This means that there is no general algorithm that can efficiently compute Sub. However, recent results of Curticapean et al. (2017) have greatly clarified the picture for which specific pattern graphs G an fpt algorithm is possible. Namely, the complexity of P-#SUB depends precisely on the treewidth of the graphs in $\mathsf{Spasm}(G)$.

In particular, there is an algorithm that runs in time $f(|G|)O(|V(H)|^{k+1})$ where k is the maximal treewidth of graphs in the Spasm. The algorithm is conceptually simple once we know that $\mathsf{Sub}(G, \cdot)$ is a graph motif parameter.

1. First compute $\mathsf{Spasm}(G)$ and the corresponding coefficients. This clearly is independent of the graph H and takes $g(|G|)$ time. This can be seen as a preprocessing step from a practical point of view.

2. For each $F \in \mathsf{Spasm}(G)$, compute $\mathsf{Hom}(F, H)$ and then compute $\mathsf{Sub}(G, \cdot)$ according to Equation (1). Computing the number of homomorphisms $\mathsf{Hom}(F, H)$ using a tree decomposition is well understood and requires $O(|H|^k)$ time, where k is the treewidth of F (Díaz et al., 2002).

---

[5]For the sake of complexity analysis this is the size of a standard representation of a graph in a Turing Machine up to a log factor.

Note that this algorithm is universal for any graph motif parameter. The only part that changes, is how to compute the basis and the coefficients in the first step. This of course entirely depends on the kind of graph motif parameter, but by definition of a graph motif parameter, the basis does not depend on the input graph (H in this subgraph counting setting).

However, not only are homomorphism counts an efficient way to compute subgraph counts. In a sense, there can be no more efficient way to compute subgraph counts. Formally speaking, both sides of Equation (1) are fpt-interreducible to each other. More precise analysis reveals that these reductions can be performed very efficiently, and in fact so efficiently that we obtain very precise lower bounds for these problems.

Theorem B.1 (Simplified, Curticapean et al. (2017))**.** *Let G be a graph and k be the maximum treewidth in* Spasm(G). *Then there is no $f(|G|)|H|^{o(k/\log k)}$ time algorithm that computes* Sub$(G, \cdot)$ *for input H if #ETH holds.*[6]

Note that it is conjectured that the bound of $o(k/\log k)$ can be improved to $o(k)$ (Marx, 2010). Under this conjecture, this shows that there is no way to really significantly improve on the computation of Sub(G, H) via the homomorphism basis as was described above.

Intuitively, this then means that graph motif parameters cannot be computed more efficiently than their basis terms. Thus supporting the claim from Section 4.2, that computing a graph motif parameter directly cannot be significantly less computational effort than computing the homomorphism counts for the basis graphs.

Beyond #Sub First, we emphasize that the original result for Theorem B.1 by Curticapean et al. (2017) is not restricted to subgraph counting but holds for arbitrary graph motif parameters (the parameter then becomes the encoding size of the basis). The same principle generalizes even further and is notably not limited to plain graphs. When we have directed graphs (and thus also directed patterns), the picture changes slightly but still no improvement over counting homomorphisms in the basis is possible (Bressan et al., 2023). Analogous statements hold with node and/or edge-labels, and general relational structures (Chen & Mengel, 2016).

Practical Considerations In the context of this paper it is of particular note that the fpt algorithm that is obtained from using the homomorphism basis is practical. To the best of our knowledge, there is no general implementation for #Sub that does not use the homomorphism basis and avoids exponential runtime in either $|V(G)|$ or $|E(G)|$. The reason for this exponential runtime is that they enumerate through the possible subgraphs, solving the counting problem by actually counting the subgraphs at enormous cost.

In contrast, the fpt algorithm consists of two steps, both of which are feasible for realistic pattern sizes. First, the Spasm and the coefficients need to be computed. Even naive algorithms that enumerate all possible partitions are efficient enough up to patterns of size 12 without specialized hardware (from preliminary experiments). In a second step, the algorithm requires the computation of Hom(F, H) for each $F \in$ Spasm(G). While this may superficially seem similar to #Sub, counting homomorphisms is much easier as they can be counted via dynamic programming on tree decompositions. This process requires time $O(|V(H)|^{k+1})$ where k is the treewidth of F. In practice this is magnitudes easier on most patterns, e.g., compare the exponents mentioned above for the 7-cycle C_7 to the observation that every graph in the basis of C_7 has treewidth 2.

Of further practical relevance is that not only is counting homomorphisms easier than naive subgraph counting algorithms, but also much more flexible for large tasks. Recall that we need to compute Hom(F, H) for each $F \in$ Spasm(G). However, these computations are entirely independent and can all be performed in parallel without any deeper algorithmic considerations. We demonstrate the efficacy of counting homomorphisms for GNN tasks by computing all the homomorphism counts for our experiments in significantly less time than is required for the training of the respective models. See Table 5 for details.

C. More on Node-Level Counting

In Section 4.4 we show how homomorphism counts can also determine node level subgraph counts for each orbit of the pattern graph. We note that the Spasm itself is not enough to determine such node-level subgraph counts via Equation (1).

We demonstrate this claim by example. In Figure 5 we show why node-level homomorphism for the part of Spasm$^{\circledast}$ without the second version of the 3 vertex path are insufficient. Figure 5a and Figure 5b illustrate the homomorphisms to two graphs H_1 and H_2 (gray arrows show mappings for a second homomorphism, all cases have only 1 or two homomorphisms),

---

[6] #ETH is the counting version of the Exponential Time Hypothesis, a standard assumption in parameterized complexity.

(a) Homomorphisms from $\mathsf{Spasm}(\square)$ to H_1 with the anchor mapping to the blue vertex.

(b) Homomorphisms from $\mathsf{Spasm}(\square)$ to H_2 with the anchor mapping to the blue vertex.
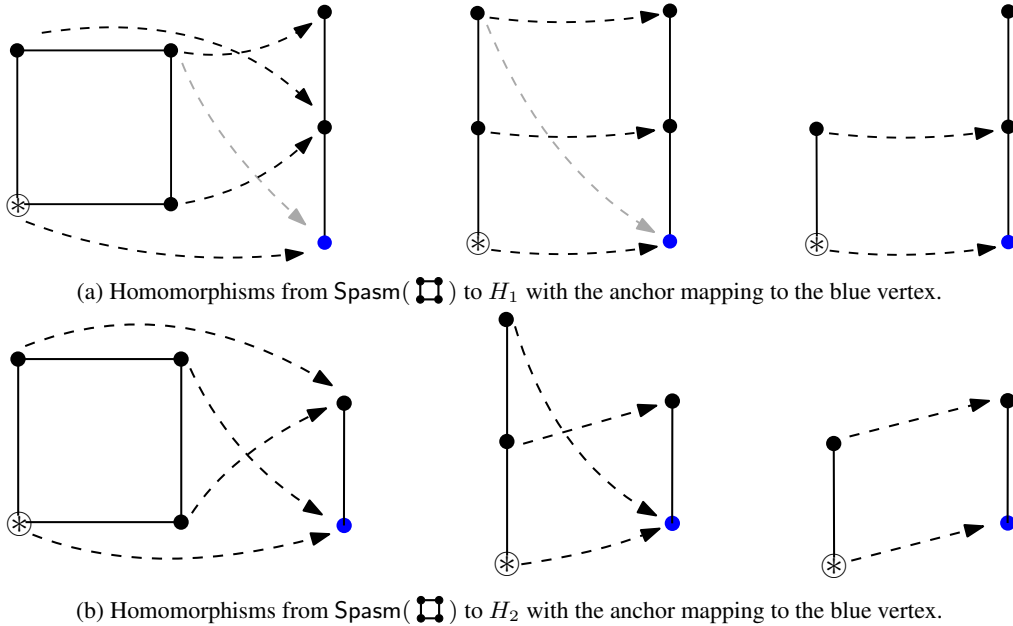
Figure 5.

both clearly without any $\square$ subgraphs. From left to right there are 2, 2, and 1 homomorphisms into H_1 and 2, 1, and 1 homomorphisms into H_2. Thus, to express the subgraph count as a linear combination of homomorphism counts would require constants $\alpha_1, \alpha_2, \alpha_3$ such that

$$\alpha_1 \cdot 2 + \alpha_2 \cdot 2 + \alpha_3 \cdot 1 = \alpha_1 \cdot 2 + \alpha_2 \cdot 1 + \alpha_3 \cdot 1$$

Clearly this implies $\alpha_2 = 0$ and we see that these graphs are insufficient to express the number of 4-cycles a vertex is part of.

Similarly, it is insufficient to only count the paths from the 3-vertex path anchored in the middle vertex, or even being provided only the sum of both 3-vertex path counts. The counterexample for both these cases is the n-vertex star graph: the path anchored at the middle vertex has $(n-1)^2$ homomorphisms to the center vertex, whereas the number of homomorphisms from all three other patterns is $n-1$.

To supplement the discussion of Appendix B we note that it is straightforward to reduce graph-level subgraph counting to node-level subgraph counting. Suppose we want to count $\mathsf{Sub}(G, H)$. Create the anchored graph G' by adding a new vertex v' to G that is connected to every other vertex. Additionally, set v' as the anchor of G'. Similarly, create H' by adding a new vertex u' to H that is connected to all other vertices. It is then easy to see that $\mathsf{Sub}(G, H) = \mathsf{Sub}(G', H', u')$. Furthermore, we can observe that since new vertex v' is connected to every other vertex, it can never be contacted with any other vertex to create a loop-free quotient. That is, every graph $F \in \mathsf{Spasm}(G')$ is a graph in $\mathsf{Spasm}(G)$ with an additional vertex that is connected to all other vertices. This means that the maximal treewidth in $\mathsf{Spasm}(G')$ is at most 1 higher than $\mathsf{Spasm}(G)$. Hence, we have a linear reduction from graph-level subgraph counting reduces to node-level subgraph counting, that increases the treewidth of the pattern at most by 1, i.e., node-level subgraph counting also critically depends on the treewidth of the Spasm.

D. Additional Experimental Details

Here, we provide all the experimental details and hyperparameters for the results presented in Section 5. The code and instructions to reproduce our results can be found at this GitHub repository: https://github.com/ejin700/hombasis-gnn.

Compute Resources. All homomorphism counting and experiments for ZINC, COLLAB, and BREC were run on a cluster with NVIDIA A10 GPUs (24 GB). Each node had 64 cores of Intel(R) Xeon(R) Gold 6326 CPU at 2.90GHz and 500GB of RAM. QM9 experiments were run on a cluster with NVIDIA A100 GPUs. All experiments used 1 GPU at a time.

Table 5. Time to compute all the homomorphism counts used in each experiment in Section 5 reported as elapsed real time (wall time).

DATASET	#GRAPHS	AVG. #NODES	AVG. #EDGES	BASIS	BASIS SIZE	#CORES	TIME
ZINC	12,000	$\sim$23.2	$\sim$49.8	$\mathsf{Spasm}(C_7)$	12	48	1M24S
				$\mathsf{Spasm}(C_8)$	35	48	4M45S
				$\mathsf{Spasm}^{\circledR}(C_7, C_8)$	118	48	19M10S
QM9	130,831	$\sim$18.0	$\sim$37.3	$\Omega_{\leq 5}^{\mathsf{con}}$	30	48	42M15S
				$\widetilde{C_6}$	1	48	2M2S
COLLAB	1	235,868	967,632	K_3	1	20	3M48S
				K_4	1	20	5M30S
				K_5	1	20	6M29S
				$\mathsf{Spasm}(P_4)$	4	20	4M16S
				$\mathsf{Spasm}(P_5)$	8	20	39M23S
				$\mathsf{Spasm}(P_6)$	15	20	3H6M
BREC	51,200	$\sim$34.7	$\sim$286.9	$\Omega_{\leq 5}^{\mathsf{con}}$	30	48	53M53S

D.1. Runtime for Counting Homomorphisms

We report the time required to compute all the homomorphism counts that were used for each experiment in Table 5. We also provide additional details regarding the size of the graphs in each dataset and the amount of compute used to generate these counts. From Table 5, we can clearly see that the time required to generate these homomorphism counts is almost negligible, especially since this counting step need only be performed once for a given parameter and dataset. In other words, once the counts have been obtained, they can be added to any existing model or architecture.

The computational efficiency of performing this homomorphism counting is especially apparent for QM9, where we are able to generate all $\mathsf{Hom}(\Omega_{\leq 5}^{\mathsf{con}})$ counts for over 130,000 individual graphs in roughly 40 minutes. This averages out to be less than 0.02 seconds per graph to count the homomorphisms for all 5-vertex connected components. For comparison, training a single run of R-GCN on QM9 to predict a single target property took several hours (more details on this in Appendix D.4).

For ogbl-COLLAB, we only perform counting on the training graph, which is a subset of the entire dataset. This is because for link prediction, a subset of the edges are removed from training and used for validation and testing. Due to the memory constraints of our compute resources, we use a slightly modified counting method for counting the number of 5-clique homomorphisms in COLLAB, which we provide in our codebase. Finally, we note that we do not perform counting for $\mathsf{Hom}(K_5)$ on the 4-vertex graphs in BREC because the graphs are constructed in a way that makes counting them infeasible.

D.2. Generating Node-level Features using Homomorphism Counts

In multiple of the experiments we inject homomorphism counts of graphs in $\mathsf{Spasm}(G)$ at node-level. In the terminology of Section 4.4, for every graph $F \in \mathsf{Spasm}(G)$, we pick an arbitrary vertex as its anchor to create F'. For each vertex v of input graph H we then compute the vector

$$\mathbf{f}_v := (\mathsf{Hom}(F', H)[\circledast \rightarrow v])_{F \in \mathsf{Spasm}(G)}$$

for some fixed order on the graphs of $\mathsf{Spasm}(G)$. We then associate each $\mathbf{f}_v$ to node v as additional features. The details in which this vector is added depend on the experiment and is described in the respective experiment section.

The motivation for using this approach is that it splits the number of total homomorphisms up "disjointly", i.e., no homomorphism is counted twice at different vertices. In particular this means that

$$\sum_{v \in V(H)} \mathbf{f}_v = (\mathsf{Hom}(F, H))_{F \in \mathsf{Spasm}(G)}. \tag{12}$$

That is, the global counts are simply a sum over the individual feature vectors, without the need to correct for potential double counting of homomorphisms. This thus presents a compromise of preserving the global counts as truthfully as possible, while still providing some node-level information. Although as discussed in Section 4.4 and Appendix C, precise node-level information requires more fine-grained homomorphism information.

Table 6. Hyperparameters for ZINC experiments.

MODEL	LAYERS	HIDDEN DIM	BATCH SIZE	MAX EPOCHS	#HEADS	READOUT
GAT	4	18	128	1000	8	MEAN
	16	22	128	1000	8	MEAN
GCN	4	125	128	1000	N/A	MEAN
	16	172	128	1000	N/A	MEAN
GIN	4	110	128	1000	N/A	SUM
	16	124	128	1000	N/A	SUM
BASEPLANE	3	128	256	500	N/A	SUM

D.3. ZINC Experimental Details

For the graph regression task on the ZINC dataset, the objective is to predict the constrained solubility of molecules. We use the same hyperparameters and evaluation protocol as in Dwivedi et al. (2023). As such, we use a subset of the ZINC dataset that contains 12,000 individual graphs, which are split into 10,000 graphs for training, 1,000 graphs for validation, and 1,000 graphs for testing. For each graph, the original vertex features denote the type of heavy atom for a given node, and the edge features denote the type of bond between two nodes. Dwivedi et al. (2023) do not use edge features when performing model benchmarking, so for comparability, we adhere to their protocol and also omit them from our experiments in Section 5.1. All results are reported as the average of 4 runs at different seeds.

D.3.1. MODEL SELECTION AND TRAINING

For ZINC, we select GAT (Velickovic et al., 2018), GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019), and BasePlanE (Dimitrov et al., 2023) as our baseline models. This is because GAT, GCN, and GIN are all standard architectures, and we want to evaluate how effective the homomorphism basis counts are at elevating the performance and expressivity of these models. We select BasePlanE because it is a recent strong architecture that is maximally expressive over planar graphs (all graphs in ZINC are planar), and it has been show to perform very well on several molecular benchmarks. Both GIN and BasePlanE theoretically reach 1-WL expressiveness, so we are able to demonstrate the fine-grained expressiveness of our approach by adding only the higher-order basis terms to the model. Note that our reported results for GIN are slightly stronger than those presented in the original benchmark because we use a more standard implementation of GIN from the PyTorch Geometric library.

In terms of training, we use the same hyperparameters for GAT, GCN, and GIN as reported by Dwivedi et al. (2023), and the same hyperparameters for BasePlanE as reported by Dimitrov et al. (2023). We also list these hyperparameters in Table 6. Following the setup from Dwivedi et al. (2023), we test GAT, GCN, and GIN with both 4 and 16 layers. The full results for ZINC are in Table 7. For GAT, GCN, and GIN, the training is done using the following configuration: $optimizer = Adam$, $initial\_lr = 0.001$, $lr\_reduce\_factor = 0.5$, $minimum\_lr = 1e - 5$, $patience = 10$. For BasePlanE, the training is done with same configuration as listed, but with a $patience = 30$ instead of 10.

Each model is evaluated using the mean average error (MAE), which is defined as:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{13}$$

where y_i is the predicted value, x_i is the true value, and n is the total number of data points.

In addition to providing the full extended results in Table 7, we also provide a comparison of our BasePlanE results to other leading models on the ZINC12k dataset (without edge features) in Table 8. From this comparison, we can see that with the addition of the Spasm and Spasm$^{\circledR}$ homomorphism counts for $C_3, ..., C_8$, we are able to achieve extremely competitive results. This further shows that our method does not only work on weak baseline models, but it can also boost the performance of more complex GNN architectures.

Table 7. Extended MAE results on ZINC12k graph regression (without edge features) with $C_3, ..., C_8$.

MODEL	LAYERS	BASE	Sub	Hom	Spasm	Spasm$^\circledR$
GAT	4	0.457±0.004	0.210±0.006	0.269±0.033	0.155±0.006	0.147±0.004
	16	0.380±0.009	0.201±0.004	0.229±0.008	0.155±0.008	0.152±0.007
GCN	4	0.417±0.007	0.206±0.006	0.254±0.017	0.166±0.003	0.165±0.004
	16	0.282±0.007	0.198±0.003	0.235±0.005	0.167±0.007	0.166±0.005
GIN	4	0.294±0.012	0.147±0.006	0.208±0.025	0.158±0.004	0.146±0.005
	16	0.246±0.019	0.143±0.002	0.197±0.015	0.157±0.005	0.143±0.004
BASEPLANE	3	0.124±0.004	0.108±0.002	0.106±0.004	0.104±0.005	0.100±0.002

Table 8. Comparison of BasePlanE results with other leading models on the ZINC12k dataset (without edge features).

MODEL	MAE
GCN	0.278 ± 0.003
GIN(-E)	0.387 ± 0.015
PNA	0.320 ± 0.032
GSN	0.140 ± 0.006
CIN	0.115 ± 0.003
BASEPLANE	0.124 ± 0.004
BASEPLANE + Sub	0.108 ± 0.002
BASEPLANE + Hom	0.106 ± 0.004
BASEPLANE + Spasm	0.104 ± 0.005
BASEPLANE + Spasm$^\circledR$	0.100 ± 0.002

D.3.2. SELECTING THE GRAPH MOTIF PARAMETERS

We select cycles as our patterns of interest because it is well established that rings are an important substructure in molecules. We use cycles up to length 8 for our experiments in Table 1 because Bouritsas et al. (2023) found that it yielded the best performance for their GSN-v model on the ZINC dataset. Note that we take the counts for $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ since this contains all the component elements that make up the basis for $C_3 - C_8$. Additionally, as mentioned in Section 4.2, we can use these Spasm and Spasm$^\circledR$ counts to quickly calculate graph-level or node-level subgraph counts, respectively. Therefore, we also include these subgraph counts as additional features since this information directly follows from computing the homomorphism counts of the entire basis of interest. In each model, the count features are encoded using a 2-layer MLP and concatenated to the initial node feature embeddings.

This 2-layer MLP that we use to encode the count features differs from the approach taken by Barceló et al. (2021), who use the z-score of the log-normalized count values. However, we found that this normalization method actually degraded the performance of using Sub counts. Therefore, we decided to pass the additional count features into an MLP, which would yield more fair and comparable results.

D.4. QM9 Experimental Details

For the graph regression task on the QM9 dataset, the objective is to predict 13 different target properties for the set of molecules (Wu et al., 2018). The entire dataset contains 130,831 graphs, which are split into 110,831 for training, 10,000 for validation, and 10,000 for testing. The original node features denote atom type as well as other chemical information about the atom. Similarly to ZINC, the edge features denote the type of bond between two nodes.

We select R-GCN (Schlichtkrull et al., 2018) as our base model, report the baseline results from Brockschmidt (2020), and adapted the codebase from Abboud et al. (2022) to perform our experiments. Our implementation of R-GCN with the addition of the homomorphism counts from $\Omega_{\leq 5}^{\mathrm{con}} \cup C_6$ uses the following hyperparameters: $batch\_size = 128$, $hidden\_dim = 128$, $layers = 8$, $lr = 0.001$, $aggregation = mean$, $epochs = 300$, and $optimizer = Adam$. We also use both batch and layer normalization and include residual connections at a frequency of every 2 layers. Our results in Table 2 are reported as an average across 5 different runs. On average, the model training time for R-GCN on one property took roughly 5 hours to complete, which is significantly longer than the time it took to compute all of the homomorphism feature counts for the entire QM9 dataset (see Table 5). This further highlights the negligible compute cost to generate the homomorphism counts.

Table 9. We report the mean absolute error (MAE) for graph regression on QM9. Using the homomorphism counts yields significant improvements over the baseline models (both with and without FA).

PROPERTY	R-GCN	+ Hom($\Omega^{\mathsf{con}}_{\leq 5} \cup C_6$)	+ FA	+ FA/Hom
MU	3.21 ±0.06	2.29 ±0.03	2.91 ±0.07	2.15 ±0.02
ALPHA	4.22 ±0.45	1.77 ±0.05	2.14 ±0.08	1.70 ±0.05
HOMO	1.45 ±0.01	1.30 ±0.03	1.37 ±1.41	1.22 ±0.02
LUMO	1.62 ±0.04	1.41 ±0.02	1.41 ±0.01	1.27 ±0.00
GAP	2.42 ±0.14	2.00 ±0.04	2.03 ±0.03	1.81 ±0.03
R2	16.38 ±0.49	10.29 ±0.35	13.55 ±0.50	9.99 ±0.33
ZPVE	17.40 ±3.56	3.03 ±0.38	5.81 ±0.61	2.86 ±0.31
U0	7.82 ±0.80	1.09 ±0.18	1.75 ±0.18	1.03 ±0.05
U	8.24 ±1.25	1.21 ±0.17	1.88 ±0.22	1.07 ±0.04
H	9.05 ±1.21	1.22 ±0.14	1.85 ±0.18	1.14 ±0.12
G	7.00 ±1.51	1.14 ±0.13	1.76 ±0.15	1.19 ±0.18
Cv	3.93 ±0.48	1.46 ±0.08	1.90 ±0.07	1.46 ±0.07
OMEGA	1.02 ±0.05	0.81 ±0.02	0.75 ±0.04	0.71 ±0.02

Alon & Yahav (2021) presented a slightly modified version of R-GCN that uses a fully-adjacent (FA) layer at the very end of the model. They show that this helps counteract the signal loss due to over-squashing effects. We also test with this model configuration and find that using the homomorphism basis counts continue to improve the performance of the R-GCN+FA model (see Table 9). For this experiment, we align with the R-GCN+FA hyperparameter setup which uses: $batch\_size = 128$, $hidden\_dim = 128$, $layers = 8$, $lr = 0.000572$, $aggregation = sum$, $epochs = 400$, and $optimizer = RMSProps$. We use batch and layer normalization, as well as residual connections every 2 layers.

We notice that for all but one property, the R-GCN + Hom($\Omega^{\mathsf{con}}_{\leq 5}, C_6$) model outperforms the R-GCN + FA model. This suggests that for tasks where the molecular structure is important, the graph-level homomorphism counts are still enough to provide the model with useful information, despite potential signal loss from oversquashing. Therefore, our method can be seen as complementary to using a final FA layer to help further boost the performance of an originally weak model.

D.4.1. SELECTING THE GRAPH MOTIF PARAMETERS

Because there are multiple different target tasks for the QM9 dataset, we follow Proposition 4.6 and decide to take the homomorphism counts for all connected components that contain up to 5 vertices, Hom($\Omega^{\mathsf{con}}_{\leq 5}$). In addition to this set, we also include Hom(C_6) because it is well established that 6-cycles are prevalent within molecules. Also, this ensures that Spasm(C_6) is contained within this set of features. Since we use R-GCN as our baseline model, we include all homomorphism counts for all components in this set, including the 1-WL ones. For each graph, all counts are encoded as graph-level features. Specifically, we concatenate the raw homomorphism counts at the very end after the message passing layers to the final feature representation vector for the graph before the final MLP decoder.

D.5. COLLAB Experimental Details

COLLAB is a collaboration network graph from Open Graph Benchmark that presents a link prediction task (Hu et al., 2020). In total, the entire graph contains 1,285,465 edges, but 23,367 of them are used for validation, and 294,466 edges are used for testing. Nodes represent scientists, and edges represent collaborations between two scientists. The nodes also contain 128-dimensional node features that correspond to the average word embeddings for each scientist's papers. Edge features include the year and the edge weight, representing the number of co-authored papers published that year.
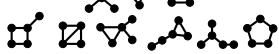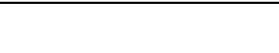
D.5.1. MODEL SELECTION AND TRAINING

We select GAT, GCN, and GraphSAGE (Hamilton et al., 2017) as our default baseline models. For GCN and GraphSAGE, we use the exact same model and hyperparameters as the default example provided by Hu et al. (2020). For GAT, we construct a model that closely aligns with the overall framework of the GCN and GraphSAGE implementations. These model details are outlined in Table 10, and our full results are presented in Table 11. All results are reported as an average of 4 runs at different seeds.

Table 10. Hyperparameters for COLLAB experiments.

MODEL	LAYERS	HIDDEN DIM	BATCH SIZE	LEARNING RATE	EPOCHS	PE DIM	#HEADS
GAT	3	256	65536	0.001	400	8	8
GCN	3	256	65536	0.001	400	8	N/A
GRAPHSAGE	3	256	65536	0.001	400	6	N/A

Table 11. Extended Hits@50 results for link prediction over the COLLAB dataset.

ADD. BASIS	Γ	GCN	GRAPHSAGE	GAT
	—	$46.13\%_{\pm 2.10}$	$48.85\%_{\pm 0.43}$	$48.27\%_{\pm 1.05}$
	K_3	$49.41\%_{\pm 0.42}$	$49.55\%_{\pm 0.66}$	$49.43\%_{\pm 0.73}$
	K_4	$47.76\%_{\pm 0.53}$	$49.29\%_{\pm 0.30}$	$48.35\%_{\pm 0.85}$
	K_5	$48.01\%_{\pm 0.87}$	$49.61\%_{\pm 0.24}$	$49.75\%_{\pm 0.16}$
	P_4	$49.59\%_{\pm 0.23}$	$50.01\%_{\pm 0.57}$	$50.76\%_{\pm 0.51}$
	P_5	$49.60\%_{\pm 0.29}$	$49.50\%_{\pm 0.47}$	$51.55\%_{\pm 0.94}$
	P_6	$50.35\%_{\pm 0.21}$	$50.18\%_{\pm 0.14}$	$51.62\%_{\pm 0.66}$

D.5.2. EVALUATION

Models are evaluated using the Hits@50 metric, which is a rank-based metric. Each test triplet $t_i = (v_{head}, r, v_{tail}) \in T_{test}$, is ranked against a set of corrupted facts that are generated by modifying the head and tail entity of the triplet of interest. Using these corrupted triplets, the model assigns a score to each triplet and sorts them in descending order by their predicted scores. The final $rank(t_i)$ of the "true" test triplet is computed as its index in the resulting sorted list. Repeating this process for all triplets in the test set T_{test} will result in a set of individual rank scores $Q = \{rank(t_i), ..., rank(t_n)\}$, that is used to compute the overall performance metric.

The Hits@k metric calculates how many true facts are ranked in the top k positions against their respective negatives, and divides that value by the number of triplets in the test set. For example, if a test set contains two true facts, and one of them ranks first amongst its negatives, while the other one ranks second amongst its negatives, the Hits@1 value would equal $1/2 = 0.5$, whereas the Hits@3 value would equal $2/2 = 1$. The formal definition of Hits@k is given in Equation 14:

$$Hits@k := \frac{|\{rank(t_i) \in Q \mid rank(t_i) \leq k\}|}{|Q|} \tag{14}$$

D.5.3. SELECTING THE GRAPH MOTIF PARAMETERS

Traditional substructure-style approaches (Barceló et al., 2021) have used cliques as patterns of interest for the COLLAB dataset. However, because COLLAB presents a link prediction task that may require multi-hop reasoning, paths could also be a useful pattern. Paths are interesting because counting the number of n-vertex paths in a graph cannot be done by a 1-WL model due to the cyclic components in the homomorphism basis. Also, standard subgraph counting techniques (Ribeiro et al., 2022) are mostly intractable for graphs as large as COLLAB, whereas our method is practically feasible.

Due to the large graph size, we normalize our raw homomorphism counts using the sine and cosine positional encoding technique from Vaswani et al. (2017). We experiment with positional encoding dimensions of size 4, 6, and 8. Specific hyperparameter details are presented in Table 10.

D.6. BREC Experimental Details

BREC is a new synthetic expressiveness dataset, where the task is to distinguish 400 pairs of non-isomorphic graphs (Wang & Zhang, 2023). The graphs span four different primary categories: basic, regular, extension, and CFI. There are 60 pairs of basic graphs, 140 pairs of regular graphs, 100 pairs of extension graphs, and 100 pairs of CFI graphs. The regular graphs are further subdivided into 50 pairs of simple regular graphs, 50 pairs of strongly regular graphs, 20 pairs of 4-vertex condition graphs, and 20 pairs of distance regular graphs. None of the graphs in this dataset contain any initial node or edge features.

Table 12. Full results for the BREC expressiveness experiment, comparing the performance of all models tested by Wang & Zhang (2023).

MODEL	BASIC (60)		REGULAR (140)		EXTENSION (100)		CFI (100)		TOTAL (400)	
	NUM	ACC	NUM	ACC	NUM	ACC	NUM	ACC	NUM	ACC
2-WL	60	100%	50	35.7%	100	100%	60	60%	270	67.5%
SPD-WL	16	26.7%	14	11.7%	41	41%	12	12%	83	20.8%
S_3	52	86.7%	48	34.3%	5	5%	0	0%	105	26.2%
S_4	60	100%	99	70.7%	84	84%	0	0%	243	60.8%
N_1	60	100%	99	85%	93	93%	0	0%	252	63%
N_2	60	100%	138	98.6%	100	100%	0	0%	298	74.5%
M_1	60	100%	50	35.7%	100	100%	41	41%	251	62.8%
NGNN	59	98.3%	48	34.3%	59	59%	0	0%	166	41.5%
DE+NGNN	60	100%	50	35.7%	100	100%	21	21%	231	57.8%
DS-GNN	58	96.7%	48	34.3%	100	100%	16	16%	222	55.5%
DSS-GNN	58	96.7%	48	34.3%	100	100%	15	15%	221	55.2%
SUN	60	100%	50	35.7%	100	100%	13	13%	223	55.8%
SSWL_P	60	100%	50	35.7%	100	100%	38	38%	248	62%
GNN-AK	60	100%	50	35.7%	97	97%	15	15%	222	55.5%
KP-GNN	60	100%	106	75.7%	98	98%	11	11%	275	68.8%
I^2-GNN	60	100%	100	71.4%	100	100%	21	21%	281	70.2%
PPGN	60	100%	50	35.7%	100	100%	23	23%	233	58.2%
δ-κ-LGNN	60	100%	50	35.7%	100	100%	6	6%	216	54%
KC-SETGNN	60	100%	50	35.7%	100	100%	1	1%	211	52.8%
GSN	60	100%	99	70.7%	95	95%	0	0%	254	63.5%
DROPGNN	52	86.7%	41	29.3%	82	82%	2	2%	177	44.2%
OSAN	56	93.3%	8	5.7%	79	79%	5	5%	148	37%
GRAPHORMER	16	26.7%	12	10%	41	41%	10	10%	79	19.8%
GIN + Hom($\Omega^{con}_{\leq 5}$) (OURS)	60	100%	120	85.7%	96	96%	0	0%	276	69%
PPGN + Hom($\bar{\Omega}^{con}_{\leq 5}$) (OURS)	60	100%	120	85.7%	100	100%	25	25%	305	76.25%

We present our results on BREC in the full context of results from Wang & Zhang (2023) in Table 12. We note that the 2-WL row in Wang & Zhang (2023) is labeled as "3-WL" in the original due to a mismatch in which style of k-WL is considered in the paper. Since we use k-WL to refer to the folklore version of k-WL, we have adapted the label from their table accordingly.

We follow the evaluation protocol of Wang & Zhang (2023) and report the best results from 10 seeds to get an upper bound on the expressiveness of each model. The effect of this protocol on our results is minimal. For the top performing PPGN+Hom($\Omega^{con}_{\leq k}$) we only observe minimal variance over changing seeds. The number of solved CFI graph pairs varied between 22% and 25%, and we observed no change in the other categories.

For GIN+Hom($\Omega^{con}_{\leq k}$), we use the exact same model construction and hyperparameters as was used to benchmark GSN on this dataset. Specifically, we used: $layers = 4$, $lr = $ 1e-4, $weight_decay = $ 1e-5, $batch\_size = 16$, $output\_dimension = 64$, $epochs = 20$. For PPGN+Hom($\Omega^{con}_{\leq k}$), we slightly tweak the hyperparameters of the model used in the BREC benchmark by changing the dimensions of the blocks from 32 to 48 in order to accommodate our extra features. Further details, model configurations, and setups can be found in our github repository.