k-th Order Deep Homomorphism Networks

Takanori Maehara
Roku, Inc.
tmaehara@roku.com

Abstract

We develop a pattern-algebraic framework for higher-order graph neural networks by extending Deep Homomorphism Networks (DHN) from rooted patterns to k-labelled patterns. The framework captures GNN expressivity along two independent axes: skeleton complexity k, which controls global structure, and pattern vocabulary $\mathcal{P}$, which captures local structure. We introduce *auxiliary gluing* ($\circledast$), the pattern-algebraic counterpart of the k-FWL sift. A pattern closure built with $\circledast$ and invariant under label permutations generates every pattern of its $(k+1)$-label idle lift with only k labels and therefore matches its expressive power. This yields an algebraic proof of $(k+1)$-WL $\equiv$ k-FWL. Neuralising these operations gives k-DHN and k-FDHN, which recover the distinguishing power of k-GNN, PPGN, δ-k-LWL, and spectral invariant GNNs, and capture node-marking subgraph GNNs, with expressivity characterised by homomorphism indistinguishability over their pattern closures. The framework also resolves open expressivity separation problems in the δ-k-LWL hierarchy at every $k \geq 2$, and gives a CFI-based proof of homomorphism-distinguishing closedness for the bounded-cycle cactus class associated with r-ℓWL.

1 Introduction

1.1 Background and Motivation

Graph neural networks (GNNs) have emerged as the dominant paradigm for learning on graph-structured data [67, 40, 62], with message-passing neural networks (MPNNs) [32] being the most widely adopted owing to their simplicity and scalability. A central concern in GNN design is expressive power. MPNNs are equivalent to the 1-dimensional Weisfeiler–Leman (1-WL) test [68, 51], which does not determine cycle counts in general [2], limiting their applicability to tasks where such substructures matter.

One line of work addresses this by operating on k-tuples of vertices. Such models realise k-WL (k-GNN [51]) or the strictly stronger k-FWL (PPGN-style updates [46]; Example 4.5); for instance, 2-FWL determines homomorphism counts of cycles of arbitrary length [26, 22], while 2-WL cannot. Despite this gap, both maintain $O(n^k)$ states. This cost is incurred regardless of which patterns are actually needed: detecting short cycles does not justify the cost of detecting arbitrarily long ones.

Another line of work enhances expressivity through targeted mechanisms: subgraph GNNs [7, 30], spectral features [21], and pattern aggregation [11]. Deep Homomorphism Networks (DHN) [45] exemplify the pattern aggregation approach, allowing users to specify rooted patterns (1-labelled graphs) to aggregate. This enables efficient detection of bounded-length cycles without the $O(n^2)$ cost of 2-tuple methods. However, its single-root formulation can express only globally tree-like arrangements of local patterns, formed by attaching each new pattern copy at a single vertex [45].

These observations suggest that GNN expressivity is governed by two independent axes: *skeleton complexity* k, controlling global structures, and *pattern vocabulary* $\mathcal{P}$, capturing local structures.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Adapalene (molecular graph)

(b) Full-wave bridge rectifier

Figure 1: **(a)** Adapalene [65]: ring backbone (blue, treewidth ≤ 2) via $k = 2$; adamantyl substituent (red, treewidth 3) via $\mathcal{P}$. **(b)** Bridge rectifier (K_4, treewidth 3) via $\mathcal{P}$.

For instance, molecular rings need only $k = 2$, while substituents such as the adamantyl cage (treewidth 3; Figure 1a) need a richer $\mathcal{P}$. Electronic circuits are series-parallel (treewidth ≤ 2 [25]) except for bridge subcircuits (Figure 1b), which need a richer $\mathcal{P}$. Goto-free programs have treewidth ≤ 3 [61, 41], but programs with goto need a richer $\mathcal{P}$. Such local structures, often called *network motifs* [48], are well studied in network science.

In each case, raising k to match the highest-treewidth component incurs $O(n^k)$ cost uniformly, even though only a few local structures exceed the baseline treewidth. Existing methods fix one axis; k-GNN fixes $\mathcal{P}$, while DHN fixes $k = 1$. This raises a natural question: can we design a model whose expressivity matches a given $(k, \mathcal{P})$, controlling each axis independently? We answer this by extending DHN from rooted to k-labelled patterns, developing a pattern-algebraic framework for higher-order GNNs.

1.2 Contributions

Unifying framework via pattern classes. Starting from the identity pattern, series composition ($\triangleright$) with $\mathcal{P}$ and gluing ($\odot$) generate the k-labelled *pattern closure* $\langle \mathcal{P} \rangle_{\triangleright,\odot}$ [17, 59]. Neuralising these operations gives k-DHN, with expressive power characterised by the homomorphism-distinguishing closure $\mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright,\odot})$ (Theorem 3.2): choosing k controls global structural complexity while $\mathcal{P}$ targets specific high-treewidth motifs without the uniform $O(n^{k'})$ cost of raising k to k'. Our examples recover the distinguishing power of k-GNN [51], PPGN [46], δ-k-LWL [50], spectral invariant GNNs [31], and node-marking subgraph GNNs [70, 30] (Examples 3.3, 4.5, 3.5, 3.6, 3.8; Figure 2). Closure inclusion compares expressivity; our Cai–Fürer–Immerman (CFI) criterion proves strict separations through finite $\mathbb{F}_2$ checks per pattern step (Appendix F). We prove k-WL $<$ δ-k-LWL $<$ SLFWL(k) $<$ k-FWL for all $k \geq 2$ (Appendices F.3.2, F.3, and F.5), where SLFWL denotes Subgraph-Local Folklore Weisfeiler–Leman; the last two comparisons resolve questions posed in [50, 71]. We further prove homomorphism-distinguishing closedness of the bounded-cycle cactus class associated with r-ℓWL [56] (Example 3.7, Appendix F.7).

The lifting theorem: why k-FWL $>$ k-WL. Both k-WL and k-FWL store $O(n^k)$ states, yet k-FWL is strictly stronger [13] because it synchronises substitutions across tuple positions (Section 2.2). We introduce *auxiliary gluing* ($\circledast$), which combines k patterns while creating an auxiliary vertex—the pattern-algebraic counterpart of the k-FWL sift. Our main result (Theorem 4.1) shows that a k-label closure built with $\circledast$ and invariant under label permutations contains every pattern of its own $(k{+}1)$-label *idle lift*, read at k of its labels. The two therefore have the same expressive power (Corollary 4.2). This yields an algebraic proof of the classical equivalence $(k{+}1)$-WL $\equiv$ k-FWL. Neuralising $\circledast$ alongside $\triangleright$ and $\odot$ gives k-FDHN, which generalises k-FWL; its expressivity is $\mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright,\odot,\circledast})$ (Theorem 4.4).

2 Preliminaries

2.1 Graphs and Homomorphisms

A *graph* $G = (V(G), E(G))$ consists of a vertex set $V(G)$ and an edge set $E(G) \subseteq \binom{V(G)}{2}$. We write $N(u)$ for the neighbourhood of u and $[\cdot]$ for the Iverson bracket. A *homomorphism* from a graph F to a graph G is a map $\varphi : V(F) \to V(G)$ such that $uv \in E(F)$ implies $\varphi(u)\varphi(v) \in E(G)$ [36]. We write $\mathrm{Hom}(F, G)$ for the set of homomorphisms from F to G and

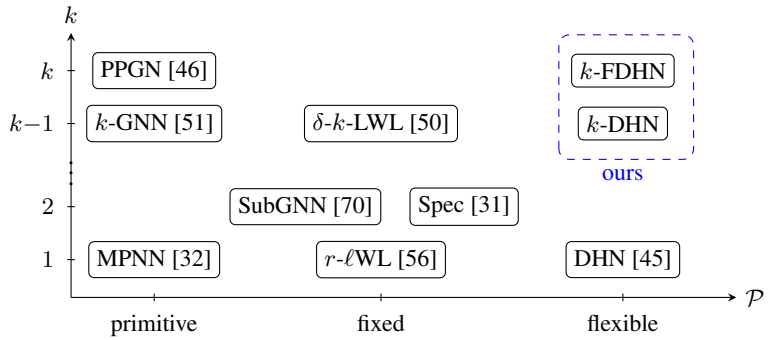


Figure 2: Landscape of GNN models by skeleton complexity (k) and pattern vocabulary ($\mathcal{P}$).

$\hom(F,G) = |\mathrm{Hom}(F,G)|$. Two graphs G and H are *homomorphism indistinguishable* over a class $\mathcal{F}$ of graphs, written $G \equiv_{\mathcal{F}} H$, if $\hom(F,G) = \hom(F,H)$ for all $F \in \mathcal{F}$. A class $\mathcal{F}$ is *homomorphism-distinguishing closed* if for every graph $F \notin \mathcal{F}$, there exist graphs G, H with $G \equiv_{\mathcal{F}} H$ but $\hom(F,G) \neq \hom(F,H)$ [58]. The *homomorphism-distinguishing closure* $\mathrm{cl}(\mathcal{F})$ is the minimal homomorphism-distinguishing closed class containing $\mathcal{F}$.

A *k-labelled graph* is a pair $(F, \boldsymbol{p})$ where F is a graph and $\boldsymbol{p} = (p_1, \dots, p_k) \in V(F)^k$ assigns k distinguished *labelled vertices*; the entries need not be distinct. A *labelled homomorphism* from $(F, \boldsymbol{p})$ to $(G, \boldsymbol{u})$ is a graph homomorphism $\varphi : F \to G$ satisfying $\varphi(p_i) = u_i$ for all $i \in [k]$. We write $\hom((F, \boldsymbol{p}), (G, \boldsymbol{u}))$ for the number of such labelled homomorphisms. We often write F for $(F, \boldsymbol{p})$ when labels are clear from context. For unlabelled hosts and expressivity comparisons, hom, $\equiv$, cl, class membership and inclusion refer to the underlying unlabelled graphs.

In machine learning applications, graphs typically carry *vertex features* $x \colon V(G) \to \mathcal{X}$. To handle them, we extend each pattern F to a *pattern with transformations* (F, μ), where $\mu = \{\mu_v\}_{v \in V(F)}$ is a collection of per-vertex transformations $\mu_v \colon \mathcal{X} \to \mathbb{R}$, and define the *generalised homomorphism number* [45]:

$$\hom((F,\mu),(G,x)) = \sum_{\varphi \in \mathrm{Hom}(F,G)} \prod_{v \in V(F)} \mu_v(x_{\varphi(v)}). \tag{1}$$

When each $\mu_v \equiv 1$, this reduces to the ordinary homomorphism number. Unless features are displayed explicitly, hom and cl refer to ordinary counts on unfeatured graphs.

2.2 Weisfeiler–Leman Test

The k-dimensional Weisfeiler–Leman algorithm refines colours of k-tuples, initialised by their atomic types (adjacency and coincidence). Two variants with different aggregation structures exist in the literature [49, 34], namely k-WL (oblivious; e.g., [33]) and k-FWL (folklore [13]), defined by:

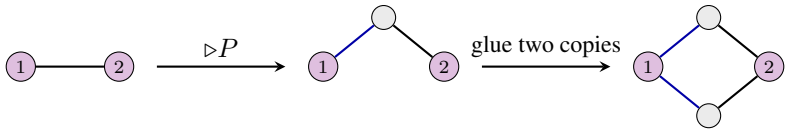
$$c^{\ell}_{\mathrm{WL}}(\boldsymbol{u}) = \textsc{hash}\Big(c^{\ell-1}(\boldsymbol{u}), \big\{\!\big\{c^{\ell-1}(\boldsymbol{u}[i \leftarrow w]) : w \in V\big\}\!\big\}_{i \in [k]}\Big), \tag{2}$$

$$c^{\ell}_{\mathrm{FWL}}(\boldsymbol{u}) = \textsc{hash}\Big(c^{\ell-1}(\boldsymbol{u}), \Big\{\!\Big\{ \underbrace{\big(c^{\ell-1}(\boldsymbol{u}[i \leftarrow w])\big)_{i \in [k]}}_{\mathrm{sift}(c^{\ell-1}, \boldsymbol{u}, w)} : w \in V \Big\}\!\Big\}\Big). \tag{3}$$

k-WL aggregates each position independently, while k-FWL collects colours into a tuple called the *sift*, capturing which colours co-occur at the same w. For $k \geq 2$, $(k{+}1)$-WL and k-FWL have the same distinguishing power [34]. Let $\mathcal{T}_k$ be the graphs of treewidth at most k (Appendix C). Then $G \equiv_{\mathcal{T}_k} H$ if and only if G and H are not distinguished by $(k{+}1)$-WL [26, 22, 34].

3 k-Deep Homomorphism Networks

The key insight of DHN [45] is that recursive formulas for homomorphism counts translate into a GNN layer. For a rooted pattern P, the *all-vertex rooted product* $P \cdot (F_v)_{v \in V(P)}$ attaches a copy



(a) F (b) $Q = F \triangleright P$ (c) $Q \odot Q$

Figure 3: Series composition and gluing at $k = 2$. Purple vertices carry the current labels; grey vertices are unlabelled. Blue edges are contributed by P.

of each rooted pattern F_v by its root at v, retaining the root of P. Its counts satisfy the chain rule of [45, Lemma 5.3]:

$$\hom\big(P \cdot (F_v)_{v \in V(P)}, (G, u)\big) = \sum_{\pi \in \mathrm{Hom}(P,(G,u))} \prod_{v \in V(P)} \hom(F_v, (G, \pi(v))). \tag{4}$$

Neuralising this recursion yields the DHN layer; when P is a single edge, it reduces to MPNN. To extend DHN to k-labelled patterns, we need analogous generation operations.

3.1 Series Composition and Gluing

We use the algebra of k-labelled graphs [17, 59]. A (k, k)-*bilabelled graph* is a tuple $(P, \boldsymbol{r}_{\mathrm{in}}, \boldsymbol{r}_{\mathrm{out}})$ where P is a graph, $\boldsymbol{r}_{\mathrm{in}} \in V(P)^k$ are the *in-labels*, and $\boldsymbol{r}_{\mathrm{out}} \in V(P)^k$ are the *out-labels*. *Series composition* $F \triangleright P$ identifies the labels $\boldsymbol{p}$ of a k-labelled graph $(F, \boldsymbol{p})$ with the out-labels $\boldsymbol{r}_{\mathrm{out}}$ of a (k, k)-bilabelled graph $(P, \boldsymbol{r}_{\mathrm{in}}, \boldsymbol{r}_{\mathrm{out}})$; the result has labels $\boldsymbol{r}_{\mathrm{in}}$. *Gluing* $F_1 \odot F_2$ identifies labels of two k-labelled graphs, with edge set $E(F_1) \cup E(F_2)$. These induce recursive formulas:

$$\hom(F \triangleright P, (G, \boldsymbol{u})) = \sum_{\varphi : P \to G : \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}} \hom(F, (G, \varphi(\boldsymbol{r}_{\mathrm{out}}))), \tag{5}$$

$$\hom(F_1 \odot F_2, (G, \boldsymbol{u})) = \hom(F_1, (G, \boldsymbol{u})) \cdot \hom(F_2, (G, \boldsymbol{u})). \tag{6}$$

All vertex-identification operations are partial: identifications that create loops are excluded from pattern closures, and their homomorphism counts are read as zero.

Let I denote the *identity pattern*: the k-labelled graph with k isolated vertices (the identity element of $\odot$). The *pattern closure* $\langle \mathcal{P} \rangle_{\triangleright, \odot}$ is the set of k-labelled graphs obtained by series composition with $\mathcal{P}$ and gluing, starting from I. Every pattern closure contains I, the value of the empty composition, whether or not $I \in \mathcal{P}$.

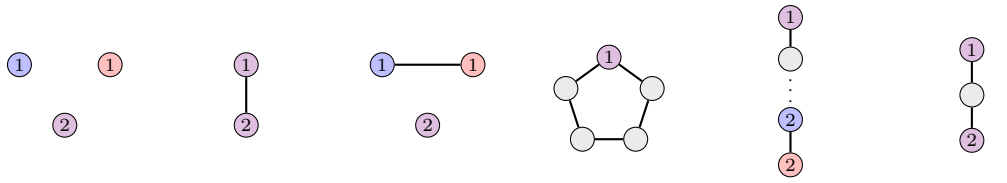
Figure 3 gives a concrete example at $k = 2$. Let F be an edge with its endpoints labelled 1 and 2. The bilabelled pattern P has an edge between its distinct in- and out-labels at position 1, and a shared isolated vertex at position 2. Composing with P gives the two-edge path $Q = F \triangleright P$, whose internal vertex is the former label 1. Gluing two copies of Q at their endpoints gives a four-cycle. For ordinary homomorphism counts at a host pair (u_1, u_2), series composition sums over the internal vertex, while gluing multiplies the two counts:

$$\hom(Q, (G, (u_1, u_2))) = \sum_{w \in N(u_1)} [w u_2 \in E(G)] = |N(u_1) \cap N(u_2)|, \tag{7}$$

$$\hom(Q \odot Q, (G, (u_1, u_2))) = |N(u_1) \cap N(u_2)|^2. \tag{8}$$

The treewidth of the generated graphs is bounded in terms of k and the base patterns (Proposition C.2).

Example 3.1. *Let* $\Sigma_k = \{J^i : i \in [k]\} \cup \{A^{ij} : 1 \le i < j \le k\}$ *be the* treewidth primitives*, where* J^i *is a* (k, k)*-bilabelled graph with* $k + 1$ *vertices and no edges, introducing a new vertex at position* i *(Figure 4a), and* A^{ij} *is a* (k, k)*-bilabelled graph with* k *vertices and one edge* ij *that adds an edge between positions* i *and* j *(Figure 4b). Every underlying graph in* $\langle \Sigma_k \rangle_{\triangleright, \odot}$ *has treewidth at most* $k - 1$*, and every graph of treewidth at most* $k - 1$ *occurs up to additional isolated vertices [59, Lemmas 6 and 9]. Since* $\hom(I, G) = |V(G)|^k$*, isolated-vertex factors cancel, giving* $\mathrm{cl}(\langle \Sigma_k \rangle_{\triangleright, \odot}) = \mathrm{cl}(\mathcal{T}_{k-1})$ *(Example C.3).*



(a) J^1 (b) A^{12} (c) J'^1 (d) C_5 (e) Subgraph (f) P_2

Figure 4: Bilabelled patterns used in this paper ($k = 2$ unless noted). Blue = in-label only ($\boldsymbol{r}_{\text{in}}$), red = out-label only ($\boldsymbol{r}_{\text{out}}$), purple = both, grey = unlabelled. **(a)** J^1: introduces a fresh vertex at position 1 (no edges). **(b)** A^{12}: adds an edge between positions 1 and 2 ($\boldsymbol{r}_{\text{in}} = \boldsymbol{r}_{\text{out}}$). **(c)** J'^1: introduces a fresh *neighbour* at position 1. **(d)** C_5 ($k = 1$): cycle with $\boldsymbol{r}_{\text{in}} = \boldsymbol{r}_{\text{out}}$ (leaf attachment). **(e)** Subgraph GNN: label 1 fixed, label 2 moves along a path. **(f)** Common neighbour: $\boldsymbol{r}_{\text{in}} = \boldsymbol{r}_{\text{out}} = (1, 2)$.

3.2 The k-DHN Layer

We neuralise series composition and gluing. We initialise $\mathbf{h}^{(0)}(\boldsymbol{u}) = \rho^{(0)}(x_{u_1}, \ldots, x_{u_k})$. Intuitively, $\mathbf{h}^{(\ell)}(\boldsymbol{u})$ encodes $\hom(F, (G, \boldsymbol{u}))$ for patterns F constructible in ℓ steps. Gluing is the Hadamard product; we neuralise it by passing $\mathbf{h}^{(\ell-1)}(\boldsymbol{u})$ through a neural network that can compute products of its encoded values. The k-DHN layer with pattern set $\mathcal{P}$ is:

$$\mathbf{h}^{(\ell)}(\boldsymbol{u}) = \rho\left(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}), \Big(\sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\text{in}}) = \boldsymbol{u}}} \mu_P\big(\mathbf{h}^{(\ell-1)}(\varphi(\boldsymbol{r}_{\text{out}})), \boldsymbol{x}_\varphi\big)\Big)_{P \in \mathcal{P}}\right), \tag{9}$$

where ρ is a combining function (e.g., a multilayer perceptron), μ_P is a per-pattern learnable transformation as in [45], and $\boldsymbol{x}_\varphi = (x_{\varphi(v)})_{v \in V(P)}$ denotes the vertex features along φ. For expressivity, we use the standard universal-function abstraction: μ_P and ρ can realise any function on the finite sets of inputs occurring on the compared finite graphs. The first argument handles gluing; the second handles series composition. With this definition, we obtain the following characterisation.

Theorem 3.2. *k-DHN with pattern set $\mathcal{P}$ distinguishes G from H if and only if there exists $F \in \mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright, \odot})$ such that $\hom(F, G) \neq \hom(F, H)$.*

Proof. By induction on the pattern construction (if direction) and on the number of layers (only-if direction); see Appendix E.1. □

The following examples relate k-DHN to existing architectures through specific choices of the pattern set $\mathcal{P}$.

Example 3.3 (k-WL)**.** *With $\mathcal{P} = \Sigma_k$ (Figure 4a,b), the unfeatured update is*

$$\mathbf{h}^{(\ell)}(\boldsymbol{u}) = \rho\Bigg(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}), \big([u_i u_j \in E]\mu_{A^{ij}}(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}))\big)_{i<j}, \Big\{\sum_{w \in V} \mu_{J^i}\big(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}[i \leftarrow w])\big)\Big\}_{i \in [k]}\Bigg). \tag{10}$$

For $k \geq 2$, its distinguishing power equals k-WL by Example C.3 and Theorem 3.2.

Example 3.4 (Relation to DHN)**.** *A DHN layer [45] sums, over each homomorphism $\varphi\colon P \to G$, a product of transformed states from all vertices $\varphi(v)$ of the rooted pattern. Each 1-DHN channel reads the state at one out-label. At $k = 1$, the multi-port extension in Appendix F.6 recovers DHN; DHN is at least as expressive as 1-DHN with the same patterns.*

Example 3.5 (δ-k-LWL)**.** *Let J'^i be the variant of J^i with an edge between in-label i and out-label i (Figure 4c). The* marked closure *$\langle \mathcal{P} \rangle^{\bullet}_{\triangleright, \odot}$ additionally allows labelled vertices to be identified, supplying the equality indicators $[u_i = u_j]$. With $\mathcal{P}_{k\text{-LWL}} = \{J'^i\} \cup \{A^{ij}\}$, series composition with J'^i restricts the aggregation to $w \in N(u_i)$, recovering δ-k-LWL [50] over the marked closure $\langle \mathcal{P}_{k\text{-LWL}} \rangle^{\bullet}_{\triangleright, \odot}$ (Lemma F.5). For all $k \geq 2$, we prove k-WL $<$ δ-k-LWL $<$ k-FWL (Theorems F.7 and F.9).*

Example 3.6 (Spectral invariant GNNs)**.** *Let A'^{ij}_m ($m \geq 1$, $i \neq j$) be the path of length m from position i to position j, with $\boldsymbol{r}_{\text{in}} = \boldsymbol{r}_{\text{out}}$. With $\mathcal{P} = \{J^i\} \cup \{A'^{ij}_m : m \geq 1\}$, 2-DHN recovers spectral invariant GNNs [31] (Lemma F.11). Appendix F.3.4 proves strict containment of the corresponding homomorphism-distinguishing closures, so spectral invariant GNNs are strictly weaker than δ-2-LWL.*

Example 3.7 (r-ℓWL)**.** *Let $C^{(d)}_m$ denote the cycle of length m, bilabelled with the in-label and the out-label at distance d along the cycle ($0 \leq d \leq \lfloor m/2 \rfloor$); for $d = 0$ the labels coincide and series composition attaches the cycle as a leaf (Figure 4d). With $\mathcal{P}_r = \{J'^1\} \cup \{C^{(d)}_m : 3 \leq m \leq r+2\}$, the pattern closure consists of connected cactus graphs in which every cycle has length at most $r+2$ and contains at most two cut vertices (Appendix F.7). On connected hosts, the atomic-type variant of r-ℓWL [56, Theorem 2 and Appendix G.3] determines bounded-cycle cactus counts, so it is at least as expressive as 1-DHN with $\mathcal{P}_r$. Our CFI criterion proves that the bounded-cycle cactus class is homomorphism-distinguishing closed (Theorem F.30); the multi-port extension attains its connected members (Proposition F.25).*

Example 3.8 (Subgraph GNNs)**.** *Node-marking subgraph GNNs [70, 7, 30] can be viewed as 2-DHN where the two labels play distinct roles: label 1 is the* root *(marked node), and label 2 is the* active node*. Node marking supplies $[u_1 = u_2]$ by identifying the two labels, giving the marked closure $\langle \mathcal{P} \rangle^{\bullet}_{\triangleright,\odot}$. DSS-GNN [7] corresponds to $\mathcal{P} = \{J'^2, J^1\}$: label 2 moves to a neighbour via J'^2 (message passing within each subgraph), and label 1 moves globally via J^1 (cross-subgraph aggregation). DS-GNN [7] drops J^1, fixing label 1 ($\boldsymbol{r}_{\text{in},1} = \boldsymbol{r}_{\text{out},1}$); NGNN [73] further restricts label 2 to within distance r of label 1 (Figure 4e) and matches the same subgraph WL test as DS-GNN under node marking [71]. On connected graphs, adding the global move J^2 leaves each variant's distinguishing power unchanged. This power equals that of the marked closure of the augmented pattern set (Lemma F.17, Corollary F.18). Each pattern admits a tree decomposition of width at most 2 with $\boldsymbol{r}_{\text{in}}$ and $\boldsymbol{r}_{\text{out}}$ as bags, and node marking preserves the bound, so the expressive power is bounded by 3-WL in all variants (Proposition C.2), giving a simple proof of [30, Corollary 7].*

Example 3.9 (Common-neighbour pattern)**.** *Let P_2 be the path-of-length-2 pattern (Figure 4f). The 2-DHN aggregation for P_2 sums over common neighbours of u_1 and u_2:*

$$\sum_{w \in N(u_1) \cap N(u_2)} \mu_{P_2}\big(\mathbf{h}^{(\ell-1)}(u_1, u_2), x_w\big). \tag{11}$$

When μ_{P_2} depends only on x_w, as in our implementation, this reduces to matrix multiplication (Appendix G).

Remark 3.10 (Computational cost of pattern aggregation)**.** *Computing $\hom(P, G)$ is $\#\mathrm{W}[1]$-hard parameterised by $|V(P)|$ [29], but this hardness is driven by dense G. Real-world graphs are sparse: molecular graphs, protein contact networks, and social ego networks all have bounded degeneracy d [23], and enumerating dense patterns in such graphs is efficient (e.g., $O(d^{t-2}|E|)$ for K_t [16]). All patterns used in our experiments are guaranteed to admit $O(|E| \log |E|)$-time counting on bounded-degeneracy graphs [6]; see Appendix A.6.*

4 k-Folklore Deep Homomorphism Networks

k-DHN generalises k-WL. However, k-FWL-based models are often preferred in practice for their strictly higher expressive power. We develop k-FDHN, which generalises k-FWL.

4.1 Auxiliary Gluing

The primitive set Σ_k gives the expressive power of graphs of treewidth at most $k-1$. Yet k-FWL achieves treewidth k (bags of size $k+1$) with the same k labels.

We propose *auxiliary gluing*, an operation that combines k patterns using a temporary label $*$. After gluing, $*$ is forgotten; its vertex is unlabelled unless it also carries an ordinary label.

Let $k \geq 2$ and let $F_1, \ldots, F_k$ be k-labelled patterns. The *auxiliary gluing* $\circledast(F_1, \ldots, F_k)$ is defined as follows: (1) In F_i, temporarily rename label i as $*$. (2) Glue the operands at every common ordinary label and at $*$. (3) Forget the temporary label $*$, retaining any ordinary labels at its vertex. Figure 5 illustrates auxiliary gluing for $k = 3$.

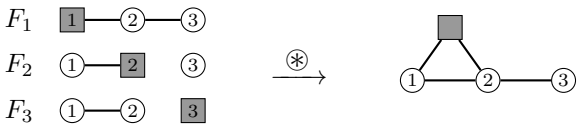
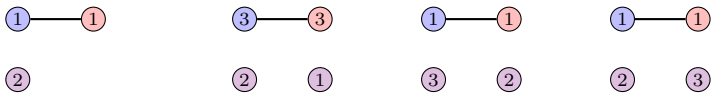


Figure 5: Auxiliary gluing example. Square nodes (■) indicate positions temporarily renamed $*$. After relabelling, F_1 contributes $\{*, 2\}, \{2, 3\}$; F_2 contributes $\{1, *\}$; and F_3 contributes $\{1, 2\}$. Gluing merges the three temporary labels into the square vertex, which is unlabelled in the result.



(a) J'^1 (b) $(J'^1)^{1+}$ (c) $(J'^1)^{2+}$ (d) $(J'^1)^{(3)+}$

Figure 6: The lift operator applied to J'^1 ($k = 2$). Panel (d) uses $P^{(k+1)+}$, which adds an isolated vertex at position $k{+}1$ and retains the original labels.

Auxiliary gluing satisfies

$$\hom(\circledast(F_1, \ldots, F_k), (G, \boldsymbol{u})) = \sum_{w \in V} \prod_{i=1}^{k} \hom(F_i, (G, \boldsymbol{u}[i \leftarrow w])), \tag{12}$$

whose right-hand side mirrors the sift aggregation of k-FWL.

The *i-th lift* P^{i+} of a bilabelled pattern adds a fresh label at position i, moving the old label i to position $k{+}1$ (Figure 6); the formal definition is in Appendix D. We now relate auxiliary gluing to the idle lift of a pattern closure. The *idle extension* of $(F, \boldsymbol{p})$ is $(F \dot\cup \{z\}, (\boldsymbol{p}, z))$ with z a new isolated vertex. The *idle lift* $\mathcal{L}_{k+1}$ of $\mathcal{L}_k = \langle \mathcal{P} \rangle_{\triangleright, \odot, \circledast}$ is the least class of $(k{+}1)$-labelled patterns containing the idle extension of every member of $\mathcal{L}_k$ and closed under gluing, permutation of the $k{+}1$ label positions, and series composition with J^i for $i \in [k{+}1]$ (J^{k+1} introduces a fresh vertex at position $k{+}1$). A class of k-labelled patterns is *position-symmetric* if it is closed under permuting its k label positions.

Our main result is the following lifting theorem.

Theorem 4.1. *Let $\mathcal{L}_k = \langle \mathcal{P} \rangle_{\triangleright, \odot, \circledast}$ and let $\mathcal{L}_{k+1}$ be its idle lift.*

1. *If $(F, \boldsymbol{p}) \in \mathcal{L}_k$, then its idle extension belongs to $\mathcal{L}_{k+1}$.*
2. *If $\mathcal{L}_k$ is position-symmetric and $(F, \boldsymbol{q}) \in \mathcal{L}_{k+1}$, then $(F, \boldsymbol{r}) \in \mathcal{L}_k$ for every k-subtuple $\boldsymbol{r}$ of $\boldsymbol{q}$.*

Proof. The first claim follows by definition and the second by induction on the construction; see Appendix D. □

Corollary 4.2. *If $\mathcal{L}_k$ is position-symmetric, then $\mathcal{L}_k$ and its idle lift $\mathcal{L}_{k+1}$ have the same expressive power: $G \equiv_{\mathcal{L}_k} H$ if and only if $G \equiv_{\mathcal{L}_{k+1}} H$ for all finite graphs G, H. Equivalently,* $\mathrm{cl}(\mathcal{L}_k) = \mathrm{cl}(\mathcal{L}_{k+1})$.

Proof. By Theorem 4.1 and cancellation of the isolated vertex; see Appendix D. □

The primitive sets Σ_k, $\mathcal{P}_{k\text{-LWL}}$ and the spectral pattern set are closed under relabelling (permuting positions maps each set to itself), so their pattern closures are position-symmetric (Appendix D).

Example 4.3. *For $k \geq 2$, the idle lift of $\langle \Sigma_k \rangle_{\triangleright, \odot, \circledast}$ is $\langle \Sigma_{k+1} \rangle_{\triangleright, \odot}$ (Appendix D). Thus the generated graphs have treewidth at most k, and every graph of treewidth at most k occurs up to additional isolated vertices. Equivalently,* $\mathrm{cl}(\langle \Sigma_k \rangle_{\triangleright, \odot, \circledast}) = \mathrm{cl}(\mathcal{T}_k)$.

4.2 The k-FDHN Layer

The k-FDHN layer extends k-DHN with an aggregation that neuralises auxiliary gluing. Since auxiliary gluing produces sums of products (12), we neuralise each factor independently with learnable $f_i\colon \mathbb{R}^d \to \mathbb{R}^d$ and Hadamard product $\circ$, following PPGN [46]. The k-FDHN layer combines gluing, series composition, and auxiliary gluing:

$$\mathbf{h}^{(\ell)}(\boldsymbol{u}) = \rho\Bigg(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}), \Bigg(\sum_{\substack{\varphi:P\to G\\ \varphi(\boldsymbol{r}_{\mathrm{in}})=\boldsymbol{u}}} \mu_P\big(\mathbf{h}^{(\ell-1)}(\varphi(\boldsymbol{r}_{\mathrm{out}})), \boldsymbol{x}_\varphi\big)\Bigg)_{P\in\mathcal{P}}, \sum_{w\in V} f_1\big(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}[1\leftarrow w])\big)\circ\cdots\circ f_k\big(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}[k\leftarrow w])\big)\Bigg). \tag{13}$$

The first two arguments are the same as in k-DHN (9); the third neuralises auxiliary gluing. With this definition, we obtain the following characterisation.

Theorem 4.4. *k-FDHN with pattern set $\mathcal{P}$ distinguishes G from H if and only if there exists $F \in \mathrm{cl}(\langle\mathcal{P}\rangle_{\triangleright,\odot,\circledast})$ such that $\hom(F,G) \neq \hom(F,H)$.*

Proof. The proof extends that of Theorem 3.2 by handling auxiliary gluing; see Appendix E.2. □

Example 4.5 (k-FWL)**.** *At $k = 2$, the auxiliary aggregation is PPGN's matrix product; at higher orders, it is the contraction proposed in [46]. For $\mathcal{P} = \Sigma_k$ and $k \geq 2$, Theorem 4.4 and Example 4.3 give k-FDHN the expressive power of k-FWL.*

Example 4.6 (SLFWL(k))**.** *Restricting the k-FWL sift to $\bigcup_i N[u_i]$, where $N[u] = N(u) \cup \{u\}$, extends the SLFWL(2) test of [71] to SLFWL(k). Its pattern class is the marked closure of $\{A^{ij}\}$ allowing auxiliary gluing when its auxiliary vertex coincides with or is adjacent to an output label (Lemma F.19). The graph S_k (Appendix F.3) is constructible under this restriction, so the CFI pair of S_k separates SLFWL(k) from δ-k-LWL. SLFWL(k) cannot distinguish the CFI pair of the treewidth-k graph D_k (Appendix F.5), which separates it from k-FWL. Together with Theorem F.7, this gives k-WL $<$ δ-k-LWL $<$ SLFWL(k) $<$ k-FWL for all $k \geq 2$; the last two comparisons resolve the questions in [50, 71] (Appendix F.5).*

5 Experiments

The preceding sections establish expressivity guarantees algebraically. Here we verify them computationally. Section 5.1 confirms that the predicted distinguishing power materialises on the BREC benchmark (deterministic refinement, no training), and Section 5.2 checks that the neural model can learn to exploit it on a synthetic planted-motif task. Because each $(k,\mathcal{P})$-configuration has a precise expressive power characterisation, the BREC results can be read as statements about any model whose expressive power matches the given $(k,\mathcal{P})$-closure. Code and logs are available at `https://gitlab.com/tmaehara/kdhn`.

5.1 Expressivity (BREC)

We test on BREC [64], a benchmark of 400 non-isomorphic graph pairs grouped into four categories: Basic, Regular, Extension, and CFI. We run the deterministic $\mathcal{P}$-based WL refinement, replacing all learnable components with hash functions, and report the number of pairs distinguished, so that the results reflect expressive power exactly, independent of optimisation. Table 1 shows overall results; a breakdown of the Regular category is in Appendix I. In the table, $+F$ denotes $\mathcal{P} = \Sigma_2 \cup \{F\}$. The width($\mathcal{P}$) column gives the maximum treewidth of the class each model generates, which bounds its expressive power (Section 2.2). The Precomp. column reports the time to enumerate homomorphisms for all patterns in $\mathcal{P}$; Refine reports the refinement time.

Skeleton axis. With $\mathcal{P} = \Sigma_2$, 2-DHN recovers 2-WL and 2-FDHN recovers 2-FWL (Examples 3.3 and 4.5). Since 2-WL $\equiv$ 1-WL [49], 2-DHN(Σ_2) distinguishes nothing (0/400), while 2-FDHN(Σ_2) matches 3-WL ($\equiv$ 2-FWL) at 270/400, confirming that auxiliary gluing ($\circledast$) upgrades expressive power from k-WL to k-FWL.

Table 1: BREC benchmark: pairs distinguished per category.

Model	$\mathcal{P}$	width ($\mathcal{P}$)	Basic (60)	Regular (140)	Extension (100)	CFI (100)	Total (400)	Precomp. (s)	Refine (s)
2-DHN	Σ_2	1	0	0	0	0	0	0.0	10.2
2-DHN	$+K_3$	2	59	49	60	0	168	0.1	10.4
2-DHN	$+K_4$	3	26	70	28	0	124	0.3	10.5
2-DHN	$+K_{2,2}$	3	60	100	91	0	251	2.0	11.0
2-DHN	$+K_{3,3}$	4	60	136	91	0	287	6.5	13.8
2-FDHN	Σ_2	2	60	50	100	60	270	0.0	8.7
2-FDHN	$+K_3$	2	60	50	100	60	270	0.1	8.8
2-FDHN	$+K_4$	3	60	100	100	60	320	0.3	9.0
2-FDHN	$+K_{2,2}$	3	60	100	100	60	320	2.0	9.9
2-FDHN	$+K_{3,3}$	4	60	136	100	60	356	6.5	12.7
3-FWL	—	3	60	140	100	80	380	—	1823.1

Lifting theorem. The lifting theorem (Theorem 4.1) implies that $\langle \Sigma_2 \rangle_{\triangleright,\odot,\circledast}$ has the same distinguishing power as the graphs of treewidth at most 2, while $\langle \Sigma_2 \rangle_{\triangleright,\odot}$ is limited to treewidth at most 1. The triangle K_3 has treewidth $2 = k$, so adding it should help 2-DHN (which lacks $\circledast$) but be redundant for 2-FDHN (whose pattern closure already contains K_3 via $\circledast$). Adding K_3 raises the number of pairs distinguished by 2-DHN from 0 to 168, while the count for 2-FDHN remains 270.

Pattern axis and closure inclusion. Patterns with treewidth $> k$ lie outside the reach of $\circledast$ and provide genuinely new expressivity for both DHN and FDHN. Since K_4 can be built from $K_{2,2}$ by adding edges via A^{ij}, we have $K_4 \in \langle \Sigma_2 \cup \{K_{2,2}\} \rangle_{\triangleright,\odot}$, so the theory predicts that $+K_4$ is weakly dominated by $+K_{2,2}$. For 2-DHN, this is confirmed: $124 < 251$. For 2-FDHN both score 320, suggesting that $\circledast$ closes the gap between K_4 and $K_{2,2}$ on these instances.

Cost–expressivity trade-off. 2-FDHN($+K_{3,3}$) reaches 356/400 with $k = 2$ ($O(n^2)$ state), taking 19.2 s including pattern precomputation. 3-FWL reaches 380, but requires $O(n^3)$ state and takes 1,823 s. Precomputation is fast despite worst-case intractability; $K_{3,3}$ has 6 vertices ($O(n^6)$ worst case) yet is enumerated in 6.5 s across all 800 BREC graphs, because dense patterns aggressively prune the search on sparse inputs (Remark 3.10). Two patterns of treewidth 3, on 8 and 10 vertices, separate the four remaining distance-regular pairs (Appendix I).

5.2 Learnability (Synthetic Graph Classification)

BREC measures expressive power via deterministic refinement; here we test whether the neural model can learn to exploit this expressivity from data, and how the pattern vocabulary affects what it can learn.

We construct binary classification tasks on planted-motif graphs. Fix a base distribution of random d-regular graphs on n vertices ($d = 5$, $n = 32$). A *planted-motif* graph is obtained by rewiring edges so that a copy of the target motif M appears as a subgraph; each dataset contains 500 graphs per class. We design three tasks that probe three regimes of the pattern closure: (i) planted K_4 vs. random, where the target is absent from $\langle \Sigma_2 \rangle_{\triangleright,\odot,\circledast}$ but correlated lower-order patterns are present; (ii) planted K_4 vs. planted diamond ($K_4^- = K_4 - e$), where the correlated patterns are present in both classes; (iii) planted $K_{3,3}$ vs. random, testing a second motif family. Both K_4 and $K_{3,3}$ have treewidth 3 and lie outside the homomorphism-distinguishing closure of the graphs of treewidth at most 2 [53], so neither the graph isomorphism network (GIN) [68] nor 2-FDHN(Σ_2) determines their homomorphism counts on all graphs. We evaluate 2-FDHN, with GIN as a baseline.

The $+K_4$ and $+K_{3,3}$ rows each solve their matching task and stay at chance on the other motif. The $+P_2$ row shows how pattern counts can support learning beyond direct coverage of the target motif. The path-of-length-2 pattern P_2 does not place K_4 or $K_{3,3}$ in the pattern closure $\langle \Sigma_2 \cup \{P_2\} \rangle_{\triangleright,\odot,\circledast}$, yet it achieves above-chance accuracy on all three tasks, because both motifs are rich in common-neighbour substructures whose counts covary with the planted motif. This surrogate

Table 2: Synthetic planted-motif accuracy (%, mean ± population standard deviation over three runs). P_2 is the path-of-length-2 pattern (Example 3.9).

Model	$\mathcal{P}$	K_4 vs. random	K_4 vs. K_4^-	$K_{3,3}$ vs. random
GIN	—	50.3±1.3	49.8±1.3	49.5±1.1
2-FDHN	Σ_2	49.3±1.2	49.2±1.0	49.2±0.8
	$+P_2$	92.7±7.2	83.0±10.3	83.5±8.7
	$+K_4$	98.7±0.2	91.2±0.6	50.8±1.3
	$+K_{3,3}$	49.7±1.3	50.2±0.8	99.3±0.6

signal is useful but fragile: accuracy is consistently lower and more variable than with the matching motif pattern, and degrades on K_4 vs. K_4^-, where the covarying substructures are present in both classes. The homomorphism-distinguishing closure determines exactly which distinctions the model can represent.

6 Conclusion

We developed a pattern-algebraic framework for higher-order GNNs by extending Deep Homomorphism Networks to k-labelled patterns. The framework decouples skeleton complexity (k) from pattern vocabulary ($\mathcal{P}$). It captures existing models (k-GNN, PPGN, δ-k-LWL, spectral invariant GNNs, node-marking subgraph GNNs), enabling systematic expressivity comparison via homomorphism-distinguishing closure inclusion. The lifting theorem relates auxiliary gluing to the idle lift of a position-symmetric pattern closure, providing an algebraic explanation of the k-WL/k-FWL gap and an algebraic proof of $(k+1)$-WL $\equiv$ k-FWL. Neuralising the framework gives k-DHN and k-FDHN, which generalise k-WL and k-FWL, respectively. The framework extends to non-adjacency and distinctness constraints (Appendix B), capturing subgraph aggregation methods as a special case.

Theorem 3.2 guarantees that k-DHN with pattern set $\mathcal{P}$ determines the homomorphism count of every pattern in $\mathrm{cl}(\langle\mathcal{P}\rangle_{\triangleright,\odot})$, yielding a model whose expressive power is exactly controlled by k and $\mathcal{P}$. We give a simple proof that node-marking subgraph GNNs are bounded by 3-WL [30, Corollary 7]. We prove k-WL $<$ δ-k-LWL $<$ SLFWL(k) $<$ k-FWL for all $k \geq 2$ and supply finite certificates of homomorphism-distinguishing closedness for minor-closed classes with known forbidden minors (Appendix F, Corollary F.33). We also give a CFI-based proof of homomorphism-distinguishing closedness of the bounded-cycle cactus class associated with r-ℓWL [56].

Experiments show that varying $\mathcal{P}$ while fixing k changes the detection capability as the homomorphism-distinguishing closure predicts. Lower-order patterns can still give a learnable surrogate signal for motifs outside the pattern closure when their counts covary with the target.

We leave open the following problem.

Problem 6.1. *When does* $\langle\mathcal{P}_1\rangle_{\triangleright,\odot} \subsetneq \langle\mathcal{P}_2\rangle_{\triangleright,\odot}$ *imply distinct distinguishing power?*

If $\langle\mathcal{P}_1\rangle_{\triangleright,\odot}$ is homomorphism-distinguishing closed, the inclusion implies strict expressive dominance.

On the practical side, the planted-motif experiments show that lower-order surrogate patterns can suffice for coarse discrimination tasks. This raises the question of whether standard benchmarks also fall into this regime, which would explain the limited gains of higher-order models [27].

Acknowledgments and Disclosure of Funding

This work received no third-party funding or support. The author declares no competing interests.

The author formulated the problem, theorem statements, and initial proofs. Discussions with Anthropic Claude Opus 4.5, 4.6, and 5.5 and OpenAI GPT-6 Astra helped improve the proofs, readability, and presentation. Discussions with the Opus models also supported development of the experimental code. The author assumes responsibility for all content.

References

[1] Isolde Adler, Eva Fluck, Tim Seppelt, and Gian Luca Spitzer. Going deep and going wide: Counting logic and homomorphism indistinguishability over graphs of bounded treedepth and treewidth. *Logical Methods in Computer Science*, 22(2):33:1–33:48, 2026.

[2] Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On Weisfeiler-Leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

[3] Linus Bao, Emily Jin, Michael Bronstein, Ismail Ilkan Ceylan, and Matthias Lanzinger. Homomorphism counts as structural encodings for graph learning. In *Proceedings of the International Conference on Learning Representations*, pages 96873–96901, 2025.

[4] Pablo Barceló, Floris Geerts, Juan L. Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In *Advances in Neural Information Processing Systems*, volume 34, pages 25280–25293, 2021.

[5] Suman K. Bera, Lior Gishboliner, Yevgeny Levanzov, C. Seshadhri, and Asaf Shapira. Counting subgraphs in degenerate graphs. *Journal of the ACM*, 69(3):1–21, 2022.

[6] Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Near-linear time homomorphism counting in bounded degeneracy graphs: The barrier of long induced cycles. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2315–2332, 2021.

[7] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *Proceedings of the International Conference on Learning Representations*, 2022.

[8] Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, pages 19–36, 1997.

[9] Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Liò, and Michael M. Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *Proceedings of the International Conference on Machine Learning*, pages 1026–1037, 2021.

[10] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, 2005.

[11] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023.

[12] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.

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

[14] Martin Carrasco, Olga Zaghen, Kavir Sumaraj, Erik Bekkers, and Bastian Rieck. Homomorphism distortion: A metric to distinguish them all and in the latent space bind them. arXiv:2511.03068, 2025.

[15] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems*, volume 33, pages 10383–10395, 2020.

[16] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.

[17] Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[18] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge University Press, 2012.

[19] Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the Annual ACM SIGACT Symposium on Theory of Computing*, pages 210–223, 2017.

[20] Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

[21] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29, pages 3837–3845, 2016.

[22] Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, pages 40:1–40:14, 2018.

[23] Erik D. Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D. Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *Journal of Computer and System Sciences*, 105:199–241, 2019.

[24] Josep Díaz, Maria Serna, and Dimitrios M Thilikos. Counting h-colorings of partial k-trees. *Theoretical Computer Science*, 281(1–2):291–309, 2002.

[25] Richard J. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965.

[26] Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, 2010.

[27] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[28] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *Proceedings of the International Conference on Learning Representations*, 2020.

[29] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, 33(4):892–922, 2004.

[30] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and extending subgraph GNNs by rethinking their symmetries. In *Advances in Neural Information Processing Systems*, volume 35, pages 31376–31390, 2022.

[31] Jingchu Gai, Yiheng Du, Bohang Zhang, Haggai Maron, and Liwei Wang. Homomorphism expressivity of spectral invariant graph neural networks. In *Proceedings of the International Conference on Learning Representations*, pages 43267–43308, 2025.

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

[33] Martin Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 63–72, 2000.

[34] Martin Grohe. The logic of graph neural networks. In *Proceedings of the Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–17, 2021.

[35] Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism tensors and linear equations. *Advances in Combinatorics*, 2025.

[36] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[37] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

[38] Andrea Jiménez, Benjamin Moore, Daniel A. Quiroz, and Youngho Yoo. Homomorphism counting for immersion-closed classes is not isomorphism. arXiv:2602.08738, 2026.

[39] Emily Jin, Michael M. Bronstein, Ismail Ilkan Ceylan, and Matthias Lanzinger. Homomorphism counts for graph neural networks: All about that basis. In *Proceedings of the International Conference on Machine Learning*, pages 22075–22098, 2024.

[40] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, volume Conference Track, 2017.

[41] Philipp Klaus Krause, Lukas Larisch, and Felix Salfelder. The tree-width of C. *Discrete Applied Mathematics*, 278:136–152, 2020.

[42] László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18(3–4):321–328, 1967.

[43] László Lovász. *Large Networks and Graph Limits*, volume 60. American Mathematical Society, 2012.

[44] Takanori Maehara and Hoang NT. A simple proof of the universality of invariant/equivariant graph neural networks. arXiv:1910.03802, 2019.

[45] Takanori Maehara and Hoang NT. Deep homomorphism networks. In *Advances in Neural Information Processing Systems*, volume 37, pages 56076–56107, 2024.

[46] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[47] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

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

[49] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten M. Borgwardt. Weisfeiler and Leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

[50] Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *Advances in Neural Information Processing Systems*, volume 33, pages 21824–21840, 2020.

[51] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[52] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity — Graphs, Structures, and Algorithms*. Algorithms and combinatorics. Springer, 2012.

[53] Daniel Neuen. Homomorphism-distinguishing closedness for graphs of bounded tree-width. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science*, pages 53:1–53:12, 2024.

[54] Daniel Neuen and Tim Seppelt. Distinguishing graphs by counting homomorphisms from sparse graphs. In *Proceedings of the Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 75:1–75:29, 2026.

[55] Hoang NT and Takanori Maehara. Graph homomorphism convolution. In *Proceedings of the International Conference on Machine Learning*, pages 7306–7316, 2020.

[56] Raffaele Paolino, Sohir Maskey, Pascal Welke, and Gitta Kutyniok. Weisfeiler and Leman go loopy: A new hierarchy for graph representational learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 120780–120831, 2024.

[57] Daniel Paul-Pena and C. Seshadhri. A dichotomy hierarchy for linear time subgraph counting in bounded degeneracy graphs. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 48–87, 2025.

[58] David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree. *Journal of Combinatorial Theory, Series B*, 181:1–61, 2026.

[59] Tim Seppelt. An algorithmic meta theorem for homomorphism indistinguishability. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, pages 82:1–82:19, 2024.

[60] Arie Soeteman and Balder ten Cate. Logical expressiveness of graph neural networks with hierarchical node individualization. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 96462–96497, 2025.

[61] Mikkel Thorup. All structured programs have small tree width and good register allocation. *Information and Computation*, 142(2):159–181, 1998.

[62] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*, volume Conference Track, 2018.

[63] Nikil Wale, Ian A. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[64] Yanbo Wang and Muhan Zhang. Towards better evaluation of GNN expressiveness with BREC dataset. arXiv:2304.07702, 2023.

[65] William K. Weigel, III, Hoang T. Dang, Abigail Feceu, and David B. C. Martin. Direct radical functionalization methods to access substituted adamantanes and diamondoids. *Organic & Biomolecular Chemistry*, 20(1):10–36, 2022.

[66] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[67] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*, 2019.

[69] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

[70] Jiaxuan You, Jonathan Michael Gomes Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10737–10745, 2021.

[71] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph GNNs via subgraph Weisfeiler–Lehman tests. In *Proceedings of the International Conference on Machine Learning*, volume 202, pages 41019–41077, 2023.

[72] Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond Weisfeiler-Lehman: A quantitative framework for GNN expressiveness. In *Proceedings of the International Conference on Learning Representations*, 2024.

[73] Muhan Zhang and Pan Li. Nested graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 15734–15747, 2021.

A Related Work

A.1 Higher-order GNNs and the WL hierarchy

The Weisfeiler–Leman (WL) hierarchy [66, 13] provides a standard yardstick for GNN expressivity. Xu et al. [68] and Morris et al. [51] independently showed that MPNNs [32] are bounded by 1-WL. To surpass this limit, Morris et al. [51] introduced k-GNN, neuralising the k-WL update on k-tuples. Maron et al. [46] proposed the second-order PPGN architecture and sketched its higher-order extension. Example 4.5 identifies its aggregation and derives k-FWL equivalence for our Σ_k instance. Morris et al. [50] developed δ-k-LWL, a sparse variant that restricts aggregation to neighbours. See Morris et al. [49] for a survey.

Our framework recovers these models' distinguishing power (Examples 3.3, 4.5, 3.5) by choosing the pattern set $\mathcal{P}$. Beyond recovery, the pattern-algebraic viewpoint explains why k-FWL is strictly stronger than k-WL (the gap corresponds to a single operation: auxiliary gluing, $\circledast$), and enables systematic expressivity comparison between models via their pattern classes.

A.2 Subgraph and pattern-based GNNs

Bevilacqua et al. [7] (ESAN) and Frasca et al. [30] run MPNNs on bags of subgraphs and analyse the resulting symmetry groups; the latter proves that node-marking subgraph GNNs are bounded by 3-WL. Our framework captures this class as 2-DHN with asymmetric labels (Example 3.8), giving a simple alternative proof of the 3-WL bound via Proposition C.2. You et al. [70] (ID-GNN) distinguish a centre node via heterogeneous message passing within its ego network. Soeteman et al. [60] propose Hierarchical Ego GNNs, which generalise subgraph GNNs via an individualisation-refinement hierarchy; their logical characterisation complements our algebraic one.

Bouritsas et al. [11] (GSN) augment node features with subgraph *isomorphism* counts. Appendix B extends our framework to non-adjacency and distinctness constraints, thereby capturing subgraph counts as constrained patterns with $\boldsymbol{r}_{\mathrm{in}} = \boldsymbol{r}_{\mathrm{out}}$. Chen et al. [15] characterise which substructures 1-WL can and cannot count. Bodnar et al. [9] lift graphs to simplicial complexes and define message passing on higher-dimensional cells.

A.3 Homomorphism-based GNNs

Homomorphism counts have been used in GNNs in three distinct roles: as precomputed features, as a layer-level computation mechanism, and as an expressivity measure.

Precomputed features. NT and Maehara [55] proposed Graph Homomorphism Convolution (GHC), using homomorphism counts from a fixed set of patterns as graph-level features. Barceló et al. [4] augment node features with local homomorphism counts; as noted in [45], this corresponds to using pattern homomorphisms only in the first layer and MPNN in subsequent layers. Bao et al. [3] (MoSE) use homomorphism counts as structural encodings fed into a transformer. Jin et al. [39] study the role of the homomorphism basis (from Möbius inversion [19]) in structural feature injection. Carrasco et al. [14] define a homomorphism-based distance metric between graphs.

Layer-level computation. Maehara and NT [45] introduced Deep Homomorphism Networks (DHN), which recursively generate rooted patterns via rooted product and aggregate homomorphism counts within the layer update. Our work extends DHN from rooted (1-labelled) to k-labelled patterns, introducing the algebraic operations ($\triangleright$, $\odot$, $\circledast$) that systematically generate k-labelled patterns with provable closure properties, and the lifting theorem that connects auxiliary gluing to the WL/FWL gap. The cycle-based r-ℓWL hierarchy [56, Appendix G.3] also has a homomorphism-count guarantee: on connected hosts, its atomic-type variant counts homomorphisms of cactus graphs with cycle length at most $r + 2$. The bounded-cycle pattern sets in Example 3.7 give the corresponding expressivity lower bound.

Expressivity measure. The idea that GNN expressivity can be characterised by which homomorphism counts a model preserves originates in the classical theorem of Lovász [42] and its refinements by Dvořák [26] and Dell et al. [22] (see also Appendix A.5). Maehara and NT [44] used this connection to prove universality of tensorised GNNs. Zhang et al. [72] systematised this as a uniform

comparison tool, terming it *homomorphism expressivity* and deriving characterisations for MPNNs, subgraph GNNs, and local k-GNNs. Gai et al. [31] extend this to spectral invariant GNNs; our framework recovers their model as 2-DHN with walk-counting patterns (Example 3.6). Our framework captures the architecture classes studied in [72] through suitable pattern sets (Examples 3.3–3.8): the homomorphism expressivity of k-DHN with pattern set $\mathcal{P}$ is $\mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright,\odot})$ (Theorem 3.2). The advantage of the pattern-algebraic approach is that it is constructive: given $\mathcal{P}$, the expressivity is determined by the closure, without case-by-case analysis.

A.4 Graph algebras

Courcelle [17, 18] introduced the algebra of k-sourced graphs, where graphs carry k distinguished vertices (sources) and are combined via operations that glue or compose along these sources. Seppelt [59] gave an algorithmic meta-theorem for homomorphism indistinguishability using the same algebra.

Our series composition ($\triangleright$) and gluing ($\odot$) correspond directly to the operations in this algebra, and our construction trees (Appendix C) are analogous to the parse trees of k-sourced graph expressions. We use the algebra to generate *pattern* graphs, which are then counted via homomorphisms into the input graph. Auxiliary gluing ($\circledast$) is, to our knowledge, new; for pattern closures invariant under label permutations, it attains with k labels the expressive power of the $(k{+}1)$-label idle lift (Theorem 4.1 and Corollary 4.2).

A.5 Homomorphism indistinguishability

Our expressivity characterisation uses homomorphism indistinguishability. Lovász [42] showed that graphs are determined up to isomorphism by their homomorphism counts from all graphs; see [43] for a modern treatment. Dvořák [26] and Dell, Grohe, and Rattan [22] established that k-WL indistinguishability corresponds to homomorphism indistinguishability over the class $\mathcal{T}_{k-1}$ of graphs of treewidth at most $k-1$. Grohe, Rattan, and Seppelt [35] developed homomorphism tensors, providing algebraic tools for studying indistinguishability.

Roberson [58] introduced the notion of homomorphism-distinguishing closure, which we use in our main theorems (Theorems 3.2 and 4.4). Homomorphism-distinguishing closedness has been established for several graph classes, including those of bounded degree [58], bounded treewidth [53], bounded treedepth [1], bounded genus [54], and immersion-closed classes [38]. Our Problem 6.1 asks when strict inclusion between the underlying pattern classes generated by k-DHN implies distinct distinguishing power. Homomorphism-distinguishing closedness of the smaller class suffices.

A.6 Enumeration complexity on sparse graphs

Remark 3.10 notes that computing $\hom(P, G)$ is $\#\mathrm{W}[1]$-hard parameterised by $|V(P)|$. Dynamic programming over tree decompositions [24] reduces the cost to $O(n^{\mathrm{tw}(P)+1})$, but the patterns of interest in the $k \geq 2$ setting have treewidth $\geq k+1 \geq 3$, limiting the practical benefit.

The worst-case hardness is driven by dense G, whereas real-world inputs are sparse. Chiba and Nishizeki [16] gave the first degeneracy-based enumeration algorithms, showing that clique listing in a graph with degeneracy d and m edges costs $O(d^{t-2}m)$. Bera, Pashanasangi, and Seshadhri [6] established a sharp dichotomy for homomorphism counting in bounded-degeneracy graphs: if the longest induced cycle of P has length ≤ 5, then $\hom(P, G)$ can be computed in $O(m \log m)$ time; if the longest induced cycle has length ≥ 6, no $o(m^{1+\gamma})$-time algorithm exists under standard fine-grained conjectures.

The patterns used in this paper fall well within the tractable regime. Complete graphs K_t have longest induced cycle length 3 (every induced subgraph on ≥ 3 vertices contains a triangle), and complete bipartite graphs $K_{s,t}$ have longest induced cycle length 4. Thus, for bounded-degeneracy inputs, all pattern aggregations in our experiments admit near-linear-time counting. This is consistent with the empirical observation (Table 1) that precomputation is fast: $K_{3,3}$ (6 vertices, $O(n^6)$ worst case) is counted across all 800 BREC graphs in 6.5 s.

More broadly, Bera et al. [5] extended the dichotomy to subgraph counting; Paul-Pena and Seshadhri [57] discovered an infinite hierarchy interpolating between bounded degeneracy and

bounded expansion [52], where the tractability threshold for patterns with longest induced cycle length c shifts as G becomes sparser.

B Local Constraints on Patterns

Throughout the paper, we use homomorphisms, which correspond to patterns with two types of pair constraints for a map $\varphi\colon V(F) \to V(G)$: *edge* ($\varphi(u)$ and $\varphi(v)$ must be adjacent) and *free* (no constraint). Subgraph features (injective homomorphisms) are also commonly used in practice [11]; these involve two additional constraints: *non-adjacency* ($\varphi(u)$ and $\varphi(v)$ must be non-adjacent) and *distinctness* ($\varphi(u) \neq \varphi(v)$). Here we show that the framework generalises to both constraints, and that each reduces to the standard framework via inclusion–exclusion.

We call the bilabelled pattern used in a series-composition step a *connector*. A *constrained pattern* augments a bilabelled pattern F with a set of *non-adjacency constraints* $E^- \subseteq \binom{V(F)}{2} \setminus E(F)$ and a set of *distinctness constraints* $E^{\neq} \subseteq \binom{V(F)}{2}$. A *constrained homomorphism* from $(F, E^-, E^{\neq})$ to a graph G is a map $\varphi\colon V(F) \to V(G)$ that preserves edges, maps each pair in E^- to a non-adjacent pair, and satisfies $\varphi(u) \neq \varphi(v)$ for all $uv \in E^{\neq}$; write $\hom_{E^-,E^{\neq}}(F, G)$ for their number. Taking $E^{\neq} = \binom{V(F)}{2}$ alone recovers injective homomorphisms. For an unlabelled pattern F, dividing their count by $|\operatorname{Aut}(F)|$ gives the number of subgraph copies. Adding $E^- = \binom{V(F)}{2} \setminus E(F)$ recovers induced-subgraph embeddings.

Series composition ($\triangleright$), gluing ($\odot$), and auxiliary gluing ($\circledast$) extend to constrained patterns by merging constraints under vertex identification. If a non-adjacency constraint falls on an edge, or a distinctness pair is identified, the composite has inconsistent constraints and is omitted from the generated class. Their constrained homomorphism sets are empty, so their terms in the counting identities vanish. Diagonal non-adjacency constraints created by identification are redundant and are discarded.

Proposition B.1. *Theorems 3.2 and 4.4 hold when layer aggregation ranges over constrained homomorphisms of each connector and constrained counts replace* hom. *Here* $\mathrm{cl}(\cdot)$ *is the homomorphism-distinguishing closure of the unlabelled constrained patterns, with isomorphisms required to preserve both constraint sets.*

Proof. A constrained homomorphism of a composite is exactly a compatible family of constrained homomorphisms of its factors, so the series, gluing, and auxiliary-gluing counting identities and the base value at I remain valid. □

Constraints are *local* to each base pattern. In a composition $F \triangleright P$, the constraints of F and P are enforced independently within each factor, but vertices not identified by gluing may still collide or become adjacent in the image. GSN [11] uses constrained patterns with $\boldsymbol{r}_{\mathrm{in}} = \boldsymbol{r}_{\mathrm{out}}$ and never composes them. It therefore detects no composed pattern.

Each non-adjacency constraint is removed by summing over whether the pair becomes an edge or free:

$$\hom_{E^-,E^{\neq}}(F, G) = \sum_{S \subseteq E^-} (-1)^{|S|} \hom_{\emptyset,E^{\neq}}(F + S, G), \tag{14}$$

where $F + S$ adds the pairs in S as edges. Each distinctness constraint is removed by summing over whether the pair is identified or not:

$$\hom_{\emptyset,E^{\neq}}(F, G) = \sum_{S \subseteq E^{\neq}} (-1)^{|S|} \hom(F/S, G), \tag{15}$$

where F/S contracts each pair in S, read under the convention of Section 3 that an identification merging the endpoints of an edge has no result and its count is zero. Therefore, each constrained count is a signed sum of standard homomorphism counts over the finite family $\{(F + S)/S' : S \subseteq E^-,\ S' \subseteq E^{\neq}\}$ of unconstrained patterns derived from F.

C Structural Characterisation of $\langle \mathcal{P} \rangle_{\triangleright,\odot}$

A *construction tree* for a k-labelled pattern $(F, \boldsymbol{p})$ is a rooted tree T in which:

- each node t carries a k-tuple $\boldsymbol{u}_t \in V(F)^k$ of vertices, its *current labels*;
- the root carries $\boldsymbol{u}_{\mathrm{root}} = \boldsymbol{p}$;
- a series node t has one child c; its edge carries a bilabelled pattern $(P, \boldsymbol{r}_{\mathrm{in}}, \boldsymbol{r}_{\mathrm{out}}) \in \mathcal{P}$ with $\boldsymbol{u}_t = \boldsymbol{r}_{\mathrm{in}}$ and $\boldsymbol{u}_c = \boldsymbol{r}_{\mathrm{out}}$ after the identifications made by series composition;
- a gluing node has two children carrying its label tuple, and its two edges carry no connector.

Starting with I at each leaf, applying the series and gluing operations towards the root builds $(F, \boldsymbol{p})$.

Every pattern in $\langle \mathcal{P} \rangle_{\triangleright,\odot}$ is represented by a construction tree: inductively, I gives a leaf, series composition adds a new root joined to the old root by its connector, and gluing adds a new root above the two operand roots with their common label tuple.

Lemma C.1. *For a construction tree T for a k-labelled pattern $(F, \boldsymbol{p})$ and any vertex $v \in V(F)$, the set $\{t \in V(T) : v \in \boldsymbol{u}_t\}$ is connected in T.*

Proof. We proceed by induction on the construction. For the base case I (identity pattern), each vertex appears in exactly one position at the root, so connectivity is trivial.

For series composition $F \triangleright P$: let v be a vertex in the result. If v comes from F, by induction $\{t : v \in \boldsymbol{u}_t\}$ is connected in the subtree of F; when $\boldsymbol{r}_{\mathrm{out}}$ repeats a vertex, the composition identifies the corresponding labels of F, and the set is the union of their connected sets, each of which meets that subtree's root. The new node (for P) contains v only if $v \in \boldsymbol{r}_{\mathrm{in}}$, and a vertex of F enters P only through the identification at $\boldsymbol{r}_{\mathrm{out}}$, so $v \in \boldsymbol{r}_{\mathrm{out}}$ as well. Thus the new node attaches to a node already containing v, preserving connectivity. If v is newly introduced by P, it appears at the new node when $v \in \boldsymbol{r}_{\mathrm{in}}$, and at no node otherwise.

For gluing $F_1 \odot F_2$: vertices at matching label positions are identified. By induction, each v forms a connected set in each subtree. Any vertices identified within one operand are labels, so their occurrence sets meet at that operand's root. If v occurs in both operands, it is a shared label, so both operand roots and the new gluing root contain it. If v occurs in only one operand, it appears at the new root only when that operand root contains it. Thus connectivity is preserved. □

A *tree decomposition* of a graph G is a tree T together with a *bag* $B_t \subseteq V(G)$ for each node $t \in V(T)$, satisfying:

(T1) every vertex $v \in V(G)$ belongs to at least one bag;

(T2) for every edge $uv \in E(G)$, some node t has $u, v \in B_t$;

(T3) for every vertex v, the set $\{t : v \in B_t\}$ induces a connected subtree of T (the running intersection property).

The *width* of a tree decomposition is $\max_t |B_t| - 1$, and the *treewidth* of G is the minimum width over all tree decompositions [8]. Lemma C.1 says the label sets $\boldsymbol{u}_t$ satisfy (T3).

Taking the sets $\boldsymbol{u}_t$ as bags does not in general give a tree decomposition of F. A vertex introduced by a pattern P at node t need not appear in $\boldsymbol{u}_t$, so (T1) can fail. An edge of P can join vertices outside $\boldsymbol{u}_t$, so (T2) can fail.

Call a tree decomposition of a bilabelled pattern $(P, \boldsymbol{r}_{\mathrm{in}}, \boldsymbol{r}_{\mathrm{out}})$ *bag-constrained* if the sets of vertices appearing in $\boldsymbol{r}_{\mathrm{in}}$ and $\boldsymbol{r}_{\mathrm{out}}$ are both bags, and call the least width of such a decomposition the *bag-constrained treewidth* of the pattern.

Proposition C.2. *Let w be at least $k-1$ and at least the bag-constrained treewidth of every pattern in $\mathcal{P}$. Then every $F \in \langle \mathcal{P} \rangle_{\triangleright,\odot}$ has treewidth at most w.*

Proof. Maintain a width-w decomposition with all current labels in one bag. For I, use the bag of its k vertices. Gluing connects the two label bags by a new tree edge; series composition connects the operand's label bag to the connector's out-label bag by a new tree edge and retains the connector's

in-label bag. Take the images of all bags under the identifications. Vertices merged within either operand lie in its joining bag, so their occurrence subtrees remain connected, and no bag grows. □

Example C.3 ($\Sigma_k = \{J^i, A^{ij}\}$: treewidth $k-1$)**.** *The pattern A^{ij} has k vertices and the single edge ij, with $\boldsymbol{r}_{\mathrm{in}} = \boldsymbol{r}_{\mathrm{out}} = (1, \ldots, k)$; the single bag $\{1, \ldots, k\}$ gives a tree decomposition of width $k-1$. The pattern J^i has $k+1$ vertices and no edges, with $\boldsymbol{r}_{\mathrm{in}} = (1, \ldots, k)$ and $\boldsymbol{r}_{\mathrm{out}} = (1, \ldots, i{-}1, i', i{+}1, \ldots, k)$; two bags $\boldsymbol{r}_{\mathrm{in}}$ and $\boldsymbol{r}_{\mathrm{out}}$ each of size k give width $k-1$. Both bag-constrained treewidths are $k-1$, so Proposition C.2 applies at $w = k-1$: $\langle \Sigma_k \rangle_{\triangleright,\odot}$ generates only graphs of treewidth $\leq k-1$. In fact, every graph of treewidth $\leq k-1$ with at least k vertices occurs in $\langle \Sigma_k \rangle_{\triangleright,\odot}$ [59, Lemmas 6 and 9]; smaller graphs occur after adding isolated vertices. For $k \geq 1$, equal counts over this generated class imply equal host orders because $\hom(I, G) = |V(G)|^k$. If $F \mathbin{\dot\cup} qK_1$ is generated, its count is $\hom(F, G)|V(G)|^q$; on nonempty hosts we cancel the common positive factor to recover the count of F. If G and H are both empty, all counts agree trivially. Thus the generated class and $\mathcal{T}_{k-1}$ induce the same indistinguishability relation and hence the same homomorphism-distinguishing closure. Hence a pattern of bag-constrained treewidth $k-1$ adds nothing to Σ_k.*

D Lifting Theorem

For a pattern $(P, \boldsymbol{r}_{\mathrm{in}}, \boldsymbol{r}_{\mathrm{out}}) \in \mathcal{P}$ and $i \in [k]$, the *i-th lift* P^{i+} is the $(k{+}1, k{+}1)$-bilabelled pattern $(P^{i+}, \boldsymbol{r}^{i+}_{\mathrm{in}}, \boldsymbol{r}^{i+}_{\mathrm{out}})$ where the underlying graph is P with one added isolated vertex v_*, and

$$\boldsymbol{r}^{i+}_{\mathrm{in}}[j] = \begin{cases} v_* & \text{if } j = i, \\ \boldsymbol{r}_{\mathrm{in}}[i] & \text{if } j = k{+}1, \\ \boldsymbol{r}_{\mathrm{in}}[j] & \text{otherwise,} \end{cases} \qquad \boldsymbol{r}^{i+}_{\mathrm{out}}[j] = \begin{cases} v_* & \text{if } j = i, \\ \boldsymbol{r}_{\mathrm{out}}[i] & \text{if } j = k{+}1, \\ \boldsymbol{r}_{\mathrm{out}}[j] & \text{otherwise.} \end{cases} \tag{16}$$

That is, P^{i+} relabels position i to $k{+}1$ and introduces an isolated vertex v_* at position i (Figure 6). We also write $P^{(k+1)+}$ for P with an isolated vertex added at position $k{+}1$. The *lift* of $\mathcal{P}$ is $\mathcal{P}^+ = \{P^{i+} : P \in \mathcal{P}, i \in [k{+}1]\} \cup \{J^{k+1}\}$, where J^{k+1} introduces a fresh vertex at position $k{+}1$.

Call position i of a labelled pattern *blank* if it names a vertex that carries no edge and that no other position names.

The *idle extension at position i* of a k-labelled pattern $(F, \boldsymbol{p})$, for $i \in [k{+}1]$, is the $(k{+}1)$-labelled pattern $(F \mathbin{\dot\cup} \{z\}, (p_1, \ldots, p_{i-1}, z, p_i, \ldots, p_k))$ with z a new isolated vertex. It is blank at position i, and at $i = k{+}1$ it is the idle extension of Section 4.1. The *idle lift* $\mathcal{L}_{k+1}$ of $\mathcal{L}_k = \langle \mathcal{P} \rangle_{\triangleright,\odot,\circledast}$ is the least class of $(k{+}1)$-labelled patterns containing the idle extension of every member of $\mathcal{L}_k$ and closed under gluing, permutation of the $k{+}1$ label positions, and series composition with J^i for $i \in [k{+}1]$ (J^{k+1} introduces a fresh vertex at position $k{+}1$). A pattern class is *position-symmetric* if it is closed under permuting its label positions.

Restatement (Theorem 4.1)**.** *Let $\mathcal{L}_k = \langle \mathcal{P} \rangle_{\triangleright,\odot,\circledast}$ and let $\mathcal{L}_{k+1}$ be its idle lift.*

1. *If $(F, \boldsymbol{p}) \in \mathcal{L}_k$, then its idle extension belongs to $\mathcal{L}_{k+1}$.*
2. *If $\mathcal{L}_k$ is position-symmetric and $(F, \boldsymbol{q}) \in \mathcal{L}_{k+1}$, then $(F, \boldsymbol{r}) \in \mathcal{L}_k$ for every k-subtuple $\boldsymbol{r}$ of $\boldsymbol{q}$.*

For a permutation σ of $[k]$, write $\sigma \cdot F$ for F with its label positions permuted by σ. Relabelling commutes with gluing and with series composition, and with auxiliary gluing once the operands are reordered: $\sigma \cdot \circledast(F_1, \ldots, F_k) = \circledast(\sigma \cdot F_{\sigma(1)}, \ldots, \sigma \cdot F_{\sigma(k)})$. Call $\mathcal{P}$ *closed under relabelling* if, for every $(P, \boldsymbol{r}_{\mathrm{in}}, \boldsymbol{r}_{\mathrm{out}}) \in \mathcal{P}$ and every permutation σ of $[k]$, the relabelled connector with tuples $(\boldsymbol{r}_{\mathrm{in}}[\sigma(j)])_{j \in [k]}$ and $(\boldsymbol{r}_{\mathrm{out}}[\sigma(j)])_{j \in [k]}$ lies in $\mathcal{P}$ up to isomorphism. If $\mathcal{P}$ is closed under relabelling, then $\mathcal{L}_k$ is position-symmetric, because each generating operation commutes with relabelling as above. The pattern sets this paper uses with auxiliary gluing are closed under relabelling: σ carries J^i to $J^{\sigma^{-1}(i)}$, J'^i to $J'^{\sigma^{-1}(i)}$, A^{ij} to $A^{\sigma^{-1}(i)\sigma^{-1}(j)}$, and A'^{ij}_m to $A'^{\sigma^{-1}(i)\sigma^{-1}(j)}_m$. Thus Σ_k, $\mathcal{P}_{k\text{-LWL}}$, and the spectral pattern set $\{J^i\} \cup \{A'^{ij}_m\}$ of Example 3.6 qualify.

The *release at position* i of a pattern blank at i is the k-labelled pattern obtained by deleting that vertex and keeping the remaining labels in order, so a pattern blank at i is the idle extension at i of its release there.

Lemma D.1. *For every k-labelled pattern F and every $i \in [k]$,*

$$F \triangleright J^i = \circledast(I, \dots, I, F, I, \dots, I), \tag{17}$$

where F occupies argument i and every other argument is I.

Proof. By definition of $\circledast$: each argument's designated position becomes the auxiliary vertex $*$, and gluing I at the other positions contributes no edges. □

By Lemma D.1, series composition with J^i is available in $\mathcal{L}_k$ for every $\mathcal{P}$, as is gluing with $\circledast(I, \dots, I)$, which adds an isolated unlabelled vertex.

Proof of Theorem 4.1. Item 1 holds by definition of the idle lift. For item 2, induct on the construction of $(F, \boldsymbol{q})$; permutations are handled by position symmetry.

Decomposition. Write F as a gluing $\bigodot_j Q_j$ of pieces Q_j, none of which is itself a gluing. Each Q_j is, up to a permutation of positions, either the idle extension of some $M \in \mathcal{L}_k$, or $G' \triangleright J^i$ for some G' built earlier. In either case, designate the position of the added isolated vertex after this permutation; this position is blank in the piece.

Releases. Releasing a piece of the first kind at its designated position gives M up to relabelling. Releasing a piece of the second kind, $G' \triangleright J^i$, at its designated position gives G' with label i omitted, which lies in $\mathcal{L}_k$ by induction.

Reassembly. Permute the omitted position to $k+1$ and call its vertex a. Group the pieces by their designated position. Let H_i be the gluing of the releases in group i, reordered so that a occupies position i for $i \le k$ and the remaining labels retain their positions; empty groups contribute I. Writing $\boldsymbol{r}$ for the retained k-subtuple of $\boldsymbol{q}$, we have

$$(F, \boldsymbol{r}) = \circledast(H_1, \dots, H_k) \odot H_{k+1}. \tag{18}$$

The right-hand side makes exactly the identifications of the original gluing, with a as the auxiliary vertex. Every H_i belongs to $\mathcal{L}_k$ by the two release cases above and closure under gluing, proving item 2. □

Proof of Corollary 4.2. By item 2 of Theorem 4.1, reading k labels of a member of $\mathcal{L}_{k+1}$ gives a member of $\mathcal{L}_k$ with the same underlying graph. Hence every underlying graph of $\mathcal{L}_{k+1}$ is one of $\mathcal{L}_k$, and $G \equiv_{\mathcal{L}_k} H$ implies $G \equiv_{\mathcal{L}_{k+1}} H$. Conversely, by item 1 the idle extension of every $F \in \mathcal{L}_k$ lies in $\mathcal{L}_{k+1}$; its underlying graph is $F \dot\cup \{z\}$ with z isolated, and $\hom(F \dot\cup \{z\}, G) = \hom(F, G)|V(G)|$. Since $I \in \mathcal{L}_k$, the idle extension of I has count $|V(G)|^{k+1}$, so $G \equiv_{\mathcal{L}_{k+1}} H$ gives $|V(G)| = |V(H)|$. On nonempty hosts the common factor cancels, so $\hom(F, G) = \hom(F, H)$; two empty hosts are isomorphic. For the closures, the first inclusion of underlying graphs gives $\mathrm{cl}(\mathcal{L}_{k+1}) \subseteq \mathrm{cl}(\mathcal{L}_k)$. Conversely, suppose $F \in \mathcal{L}_k$ were not in $\mathrm{cl}(\mathcal{L}_{k+1})$. Closedness gives G, H with $G \equiv_{\mathrm{cl}(\mathcal{L}_{k+1})} H$ and $\hom(F, G) \neq \hom(F, H)$, but $G \equiv_{\mathcal{L}_{k+1}} H$ by Lemma E.1, which contradicts the previous paragraph. Thus $\mathcal{L}_k \subseteq \mathrm{cl}(\mathcal{L}_{k+1})$, and minimality of the closure gives $\mathrm{cl}(\mathcal{L}_k) \subseteq \mathrm{cl}(\mathcal{L}_{k+1})$. □

Lemma D.2 (Release)**.** *Let $\mathcal{L}_k$ be position-symmetric, as in Theorem 4.1. Then for every $F \in \mathcal{L}_{k+1}$ and every position i at which F is blank, the release of F at i lies in $\mathcal{L}_k$.*

Proof of Lemma D.2. Induct on the construction, using position symmetry for permutations. Releasing an idle extension at any blank position gives its original pattern up to relabelling: if an old blank vertex is removed, the new isolated vertex replaces it. Release commutes with gluing at a blank position. For $G' \triangleright J^i$, release at the new blank position gives G' with label i omitted, so Theorem 4.1 applies. At another blank position, release commutes with series composition with J^i unless the two old positions name the same vertex. The commuting case follows by induction and Lemma D.1. In the coincident case, the result is G' with only the labels in a k-subtuple retained. Position symmetry restores the order of the retained labels. □

Corollary D.3. *Let $\mathcal{L}_k$ be position-symmetric, as in Theorem 4.1. Then for every $P \in \mathcal{P}$, every $i \in [k+1]$ and every $F \in \mathcal{L}_{k+1}$ blank at position i, the composite $F \triangleright P^{i+}$ belongs to $\mathcal{L}_{k+1}$ whenever it is defined.*

Proof of Corollary D.3. Let $F \in \mathcal{L}_{k+1}$ be blank at position i. Swap positions i and $k+1$, and let $F_\circ$ be the release at the now-last blank position. Then $F_\circ$ belongs to $\mathcal{L}_k$ by Lemma D.2, so the idle extension of $F_\circ \triangleright P$ belongs to $\mathcal{L}_{k+1}$. Undoing the swap gives $F \triangleright P^{i+}$. □

Derivation of Example 4.3. Since the lift Σ_k^+ of Appendix D equals Σ_{k+1}, every series composition and gluing step of $\langle \Sigma_k \rangle_{\triangleright,\odot}$ lifts to $\langle \Sigma_{k+1} \rangle_{\triangleright,\odot}$. For auxiliary gluing, swap positions i and $k+1$ in the idle extension of each operand F_i, glue the results, and compose with J^{k+1}. This produces the idle extension of $\circledast(F_1, \dots, F_k)$. Thus every generator of the idle lift is generated by Σ_{k+1}. Conversely, series composition with J^i is an idle-lift operation, and series composition with A^{ij} is gluing with a permuted idle extension of the k-labelled one-edge pattern $I \triangleright A^{12}$. Hence the idle lift is $\langle \Sigma_{k+1} \rangle_{\triangleright,\odot}$. By Theorem 4.1 and Example C.3, the underlying graphs of $\langle \Sigma_k \rangle_{\triangleright,\odot,\circledast}$ have treewidth at most k, and every such graph occurs up to additional isolated vertices. Consequently, $\mathrm{cl}(\langle \Sigma_k \rangle_{\triangleright,\odot,\circledast}) = \mathrm{cl}(\mathcal{T}_k)$. □

E Expressive Power of k-DHN and k-FDHN

Lemma E.1. *For any graph class $\mathcal{F}$, $\equiv_{\mathrm{cl}(\mathcal{F})} = \equiv_{\mathcal{F}}$.*

Proof. Let $\mathcal{C}$ consist of the graphs Q such that $G \equiv_{\mathcal{F}} H$ implies $\hom(Q, G) = \hom(Q, H)$ for every G, H. Then $\mathcal{F} \subseteq \mathcal{C}$, and $\equiv_{\mathcal{C}}$ equals $\equiv_{\mathcal{F}}$. For each $Q \notin \mathcal{C}$, its defining counterexample therefore witnesses closedness of $\mathcal{C}$. If $\mathcal{D}$ is a closed class containing $\mathcal{F}$, then for each $Q \notin \mathcal{D}$, a witness excluding Q from $\mathcal{D}$ also excludes it from $\mathcal{C}$, because $\equiv_{\mathcal{D}}$ implies $\equiv_{\mathcal{F}}$. Thus $\mathcal{C}$ is the least closed class containing $\mathcal{F}$, namely $\mathrm{cl}(\mathcal{F})$. □

E.1 Proof of Theorem 3.2

Restatement (Theorem 3.2)**.** *k-DHN with pattern set $\mathcal{P}$ distinguishes G from H if and only if there exists $F \in \mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright,\odot})$ such that $\hom(F, G) \neq \hom(F, H)$.*

We use the following closure property.

Lemma E.2 (Algebra closure)**.** *The class of functions $V(G)^k \to \mathbb{R}$ computable by k-DHN with pattern set $\mathcal{P}$ is closed under addition, scalar multiplication, and pointwise multiplication.*

Proof. Addition and pointwise multiplication: stack two models h_1, h_2 into $[h_1, h_2]$ by concatenating feature vectors and stacking the ρ functions, then apply a linear or bilinear map at the final layer, respectively. Scalar multiplication applies a linear map to one model's output. □

Proof of Theorem 3.2. For simplicity, we prove the case without node features ($\mathbf{h}^{(0)} = 1$, $\mu_P = \mathrm{id}$). The extension to graphs with node features is given in Proposition E.5.

Let $\mathcal{F} = \langle \mathcal{P} \rangle_{\triangleright,\odot}$.

Define the *$\mathcal{F}$-type* of $(G, \boldsymbol{u})$ as the function $\tau_G(\boldsymbol{u})\colon F \mapsto \hom(F, (G, \boldsymbol{u}))$ for $F \in \mathcal{F}$.

Completeness (if direction). We show that for every $F \in \mathcal{F}$, there exists a k-DHN computing $\boldsymbol{u} \mapsto \hom(F, (G, \boldsymbol{u}))$.

We proceed by induction on the construction of $F \in \mathcal{F}$:

- *Base* ($F = I$): $\hom(I, (G, \boldsymbol{u})) = 1 = \mathbf{h}^{(0)}(\boldsymbol{u})$.
- *Series composition* ($F = F' \triangleright P$ for $P \in \mathcal{P}$): by the chain rule (5),

$$\hom(F' \triangleright P, (G, \boldsymbol{u})) = \sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}}} \hom\big(F', (G, \varphi(\boldsymbol{r}_{\mathrm{out}}))\big). \tag{19}$$

By the induction hypothesis, some k-DHN with $\ell-1$ layers has a component of $\mathbf{h}^{(\ell-1)}(\boldsymbol{w})$ equal to $\hom(F', (G, \boldsymbol{w}))$. Let c index this component. One additional layer with pattern P gives:

$$(\mathrm{agg}_P(\boldsymbol{u}))_c = \sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}}} \mathbf{h}_c^{(\ell-1)}(\varphi(\boldsymbol{r}_{\mathrm{out}})) = \hom(F' \triangleright P, (G, \boldsymbol{u})). \tag{20}$$

- *Gluing* ($F = F_1 \odot F_2$): $\hom(F_1 \odot F_2, (G, \boldsymbol{u})) = \hom(F_1, (G, \boldsymbol{u})) \cdot \hom(F_2, (G, \boldsymbol{u}))$. Both factors are computable by the induction hypothesis; their product is computable by Lemma E.2.

A summation readout yields $\sum_{\boldsymbol{u}} \hom(F, (G, \boldsymbol{u})) = \hom(F, G)$, so the k-DHN just constructed distinguishes G from H whenever $\hom(F, G) \neq \hom(F, H)$. By Lemma E.1, $\mathrm{cl}(\mathcal{F})$ and $\mathcal{F}$ have the same distinguishing power. Thus, if some $F \in \mathrm{cl}(\mathcal{F})$ satisfies $\hom(F, G) \neq \hom(F, H)$, some $F' \in \mathcal{F}$ also satisfies $\hom(F', G) \neq \hom(F', H)$.

Soundness (only-if direction). Contrapositive: assume $\hom(F, G) = \hom(F, H)$ for all $F \in \mathcal{F}$. We show no k-DHN distinguishes G from H.

Claim E.3 (Graph-level sums of counts agree)**.** *For all $F_1, \ldots, F_m \in \mathcal{F}$ and all exponents $e_1, \ldots, e_m \geq 0$, the sum $\sum_{\boldsymbol{u}} \prod_i \hom(F_i, (G, \boldsymbol{u}))^{e_i}$ takes the same value on G as on H.*

Proof of Claim E.3. For any such $F_1, \ldots, F_m$ and $e_1, \ldots, e_m$:

$$\sum_{\boldsymbol{u} \in V(G)^k} \prod_{i=1}^{m} \hom\big(F_i, (G, \boldsymbol{u})\big)^{e_i} = \sum_{\boldsymbol{u}} \hom\Big(\bigodot_i F_i^{\odot e_i}, (G, \boldsymbol{u})\Big) = \hom\big(\bigodot_i F_i^{\odot e_i}, G\big), \tag{21}$$

where the first equality uses the gluing formula and the second sums over $\boldsymbol{u}$. If the gluing is defined, it belongs to $\mathcal{F}$, so the right-hand side agrees for G and H (undefined composites count zero; Section 3). □

Claim E.4 (Layer features depend only on the type)**.** *For every layer ℓ there is a function $\mathbf{f}^{(\ell)}$ of the $\mathcal{F}$-type, independent of the graph, such that $\mathbf{h}^{(\ell)}(\boldsymbol{u}) = \mathbf{f}^{(\ell)}(\tau_G(\boldsymbol{u}))$.*

Proof of Claim E.4. We prove the claim by induction on ℓ.

Base ($\ell = 0$): $\mathbf{h}^{(0)}(\boldsymbol{u}) = 1$ for all $\boldsymbol{u}$, which trivially depends only on the $\mathcal{F}$-type.

Inductive step: assume $\mathbf{h}^{(\ell-1)}(\boldsymbol{w}) = \mathbf{f}^{(\ell-1)}(\tau_G(\boldsymbol{w}))$ for a function $\mathbf{f}^{(\ell-1)}$ of the $\mathcal{F}$-type, independent of the graph. Only finitely many $\mathcal{F}$-types occur on G and H; choose $F_1, \ldots, F_m \in \mathcal{F}$ separating them. The indicator of each type class is then a finite product of affine combinations of the counts $\hom(F_i, (G, \cdot))$, and every component f_j of $\mathbf{f}^{(\ell-1)}$ is the corresponding combination of those indicators. Expanding these products and rewriting each product of counts as a single count by the gluing formula gives coefficients $\{c_{\mathbf{e}}\}$, the same for both graphs, with

$$f_j(\tau_G(\boldsymbol{w})) = \sum_{\mathbf{e}} c_{\mathbf{e}} \prod_{i=1}^{m} \hom(F_i, (G, \boldsymbol{w}))^{e_i} = \sum_{\mathbf{e}} c_{\mathbf{e}} \hom\Big(\bigodot_i F_i^{\odot e_i}, (G, \boldsymbol{w})\Big). \tag{22}$$

Here $\mathbf{e} = (e_1, \ldots, e_m)$ ranges over finitely many exponent vectors.

Now consider the aggregation:

$$\begin{aligned} \mathrm{agg}_P(\boldsymbol{u}) &= \sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}}} \mathbf{h}^{(\ell-1)}(\varphi(\boldsymbol{r}_{\mathrm{out}})) = \sum_{\varphi} \mathbf{f}^{(\ell-1)}(\tau_G(\varphi(\boldsymbol{r}_{\mathrm{out}}))) \\ &= \sum_{\mathbf{e}} c_{\mathbf{e}} \sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}}} \hom\Big(\bigodot_i F_i^{\odot e_i}, (G, \varphi(\boldsymbol{r}_{\mathrm{out}}))\Big) \\ &= \sum_{\mathbf{e}} c_{\mathbf{e}} \hom\Big(\big(\bigodot_i F_i^{\odot e_i}\big) \triangleright P, (G, \boldsymbol{u})\Big), \end{aligned} \tag{23}$$

where the last equality is the chain rule. Each defined composite $(\bigodot_i F_i^{\odot e_i}) \triangleright P$ belongs to $\mathcal{F}$. Thus $\mathrm{agg}_P(\boldsymbol{u})$ depends only on $\tau_G(\boldsymbol{u})$.

Since $\mathbf{h}^{(\ell-1)}(\boldsymbol{u})$ and all $\mathrm{agg}_P(\boldsymbol{u})$ depend only on $\tau_G(\boldsymbol{u})$, and ρ is a fixed function, $\mathbf{h}^{(\ell)}(\boldsymbol{u}) = \rho(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}), \{\mathrm{agg}_P(\boldsymbol{u})\}_{P\in\mathcal{P}})$ also depends only on $\tau_G(\boldsymbol{u})$. □

Conclusion. Let g be any function of the $\mathcal{F}$-type, independent of the graph. By (22), the indicator of $\{\boldsymbol{u} : g(\tau_G(\boldsymbol{u})) = a\}$ is a combination of counts $\hom(\bigodot_i F_i^{\odot e_i}, (G, \boldsymbol{u}))$; summing it over all tuples expresses the multiplicity of a in $\{\{g(\tau_G(\boldsymbol{u})) : \boldsymbol{u} \in V(G)^k\}\}$ as a combination of graph-level counts, which agrees for G and H by Claim E.3. That multiset is therefore the same for both. By Claim E.4 this applies to $\mathbf{h}^{(\ell)}$ at every layer ℓ, so any invariant readout yields the same graph-level output and no k-DHN distinguishes G from H. □

We now extend the characterisation to graphs with node features. The layer formulas (9) already include $\mu_P(\cdot, \boldsymbol{x}_\varphi)$, which transforms the vertex features $\boldsymbol{x}_\varphi = (x_{\varphi(v)})_{v\in V(P)}$ along each homomorphism φ. Write α_v for the transformations μ_v of Section 2. For a labelled pattern $(F, \boldsymbol{p})$, the labelled form of the generalised homomorphism number is

$$\hom((F,\alpha),(G,\boldsymbol{u},x)) = \sum_{\substack{\psi\colon F\to G\\ \psi(\boldsymbol{p})=\boldsymbol{u}}} \prod_{v\in V(F)} \alpha_v(x_{\psi(v)}). \tag{24}$$

Proposition E.5 (Extension to node features)**.** *For $k \geq 1$, a k-DHN with pattern set $\mathcal{P}$ distinguishes finite featured graphs (G, x) and (H, y) if and only if there are $F \in \langle\mathcal{P}\rangle_{\triangleright,\odot}$ and transformations $\alpha_v : \mathcal{X} \to \mathbb{R}$ with*

$$\hom((F,\alpha),(G,x)) \neq \hom((F,\alpha),(H,y)). \tag{25}$$

For $k \geq 2$, the same equivalence holds for k-FDHN with $F \in \langle\mathcal{P}\rangle_{\triangleright,\odot,\circledast}$. Here the feature space $\mathcal{X}$ and transformation functions are arbitrary.

The generated-class formulation follows [45, Theorem 5.2]. The accompanying Lean formalisation (supplementary material) proves both equivalences and their specialisation to ordinary networks when every vertex of both hosts has the same constant feature.

Proof. Completeness. Multiply the weights of all original vertices identified into each quotient vertex, counting each original vertex once. For weights α^F on F and α^P on P, let α be these quotient weights. Then

$$\begin{aligned}&\hom((F\triangleright P,\alpha),(G,\boldsymbol{u},x))\\ &\quad= \sum_{\substack{\varphi:P\to G\\ \varphi(\boldsymbol{r}_{\mathrm{in}})=\boldsymbol{u}}} \left(\prod_{v\in V(P)} \alpha_v^P(x_{\varphi(v)})\right) \hom((F,\alpha^F),(G,\varphi(\boldsymbol{r}_{\mathrm{out}}),x)).\end{aligned} \tag{26}$$

One layer realises this identity by multiplying the incoming count by the connector weights; gluing uses the corresponding product identity. Any desired quotient weights can be realised by assigning each weight to one original preimage and using unit weights on the others. The initial map computes the weighted identity pattern as a product of functions of the labelled features. For auxiliary gluing, assign the auxiliary vertex's weight to label i of an operand F_i: evaluating that operand at $\boldsymbol{u}[i \leftarrow w]$ supplies the factor depending on x_w through its state. Any incoming-label weights not assigned to operands are supplied by the initial feature map and retained in the state for the combining function.

Soundness. Define the featured type by its generalised counts. On the two finite featured hosts, expand each component of μ_P in products of indicators of the connector's vertex features. The coefficients are functions of the incoming count type. Finite interpolation expresses those coefficients as finite linear combinations of products of generalised counts. Gluing and the weighted chain rules therefore express each aggregation as a finite linear combination of generalised counts. The weighted auxiliary identity gives the same conclusion for $\circledast$. Summing indicators of final states over the labelled tuples then proves readout equality, as in the unfeatured proof. □

E.2 Proof of Theorem 4.4

Restatement (Theorem 4.4)**.** *k-FDHN with pattern set $\mathcal{P}$ distinguishes G from H if and only if there exists $F \in \mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright,\odot,\circledast})$ such that* $\hom(F,G) \neq \hom(F,H)$.

The proof extends that of Theorem 3.2 by handling the auxiliary gluing operation $\circledast$, through the auxiliary gluing formula (12).

Proof of Theorem 4.4. As in the proof of Theorem 3.2, we prove the unfeatured case, write $\mathcal{F} = \langle \mathcal{P} \rangle_{\triangleright,\odot,\circledast}$, and define the $\mathcal{F}$-type $\tau_G(\boldsymbol{u})$ as there.

Completeness (if direction). As in the proof of Theorem 3.2, it suffices to show that for every $F \in \mathcal{F}$, a k-FDHN computes $\boldsymbol{u} \mapsto \hom(F,(G,\boldsymbol{u}))$. By induction on the construction of F:

- *Base, series composition, gluing*: as in the proof of Theorem 3.2.
- *Auxiliary gluing* ($F = \circledast(F_1,\dots,F_k)$): by (12),

$$\hom(F,(G,\boldsymbol{u})) = \sum_{w \in V} \prod_{i=1}^{k} \hom(F_i,(G,\boldsymbol{u}[i \leftarrow w])). \tag{27}$$

  By induction and Lemma E.2, some k-FDHN computes $\mathbf{h}^{(\ell-1)}(\boldsymbol{v})$ whose components include $\hom(F_i,(G,\boldsymbol{v}))$ for each $i \in [k]$. Choose the layer functions to be the projections $f_i(\mathbf{z}) = \mathbf{z}_{c_i}$, where c_i is the coordinate encoding $\hom(F_i,(G,\cdot))$. Then:

$$\sum_{w \in V} \prod_{i=1}^{k} f_i\big(\mathbf{h}^{(\ell-1)}(\boldsymbol{u}[i \leftarrow w])\big) = \hom(F,(G,\boldsymbol{u})). \tag{28}$$

Soundness (only-if direction). Contrapositive: assume $\hom(F,G) = \hom(F,H)$ for all $F \in \mathcal{F}$.

Claim E.3 holds here verbatim: its argument uses gluing only, and $\mathcal{F}$ is closed under $\odot$.

Claim E.6 (Layer features depend only on the type, with auxiliary gluing)**.** *For every layer ℓ of a k-FDHN there is a function $\mathbf{f}^{(\ell)}$, independent of the graph, such that $\mathbf{h}^{(\ell)}(\boldsymbol{u}) = \mathbf{f}^{(\ell)}(\tau_G(\boldsymbol{u}))$.*

Proof of Claim E.6. The series composition and gluing arguments carry over from Claim E.4. We show the auxiliary gluing aggregation also depends only on the $\mathcal{F}$-type.

We induct on ℓ. By the induction hypothesis, $\mathbf{h}^{(\ell-1)}(\boldsymbol{v}) = \mathbf{f}^{(\ell-1)}(\tau_G(\boldsymbol{v}))$. Let Θ be the finite set of $\mathcal{F}$-types appearing in G or H. The layer functions $f_1,\dots,f_k\colon \mathbb{R}^d \to \mathbb{R}^d$ of the auxiliary gluing aggregation (Section 4), where d is the feature dimension, are fixed, so we may define $g\colon \Theta^k \to \mathbb{R}^d$ using an entrywise product:

$$g(\tau_1,\dots,\tau_k) = \prod_{i=1}^{k} f_i(\mathbf{f}^{(\ell-1)}(\tau_i)). \tag{29}$$

We evaluate g at $\tau_i = \tau_G(\boldsymbol{w}_i)$ with $\boldsymbol{w}_i = \boldsymbol{u}[i \leftarrow w]$. Choose $F_1,\dots,F_m \in \mathcal{F}$ that separate all types in Θ. Apply (22) separately to each component of $f_i(\mathbf{f}^{(\ell-1)}(\tau_i))$, for $i \in [k]$, and multiply the resulting expansions. This expresses every component g_j as

$$g_j(\tau_1,\dots,\tau_k) = \sum_{\mathbf{e}} c_{\mathbf{e}} \prod_{i=1}^{k} \hom\Big(\bigodot\nolimits_s F_s^{\odot e_{is}},(G,\boldsymbol{w}_i)\Big), \tag{30}$$

where $\mathbf{e} = (e_{is})_{i\in[k],s\in[m]}$. Discard terms in which any operand gluing is undefined, since their counts vanish. For each remaining exponent array, write $G_i^{\mathbf{e}} = \bigodot_s F_s^{\odot e_{is}} \in \mathcal{F}$. The auxiliary

gluing aggregation becomes:

$$\begin{aligned}
&\sum_{w\in V} g\big(\tau_G(\boldsymbol{u}[1\leftarrow w]),\dots,\tau_G(\boldsymbol{u}[k\leftarrow w])\big)\\
&=\sum_{\mathbf{e}} c_{\mathbf{e}} \sum_{w\in V}\prod_{i=1}^{k}\hom\big(G_i^{\mathbf{e}},(G,\boldsymbol{u}[i\leftarrow w])\big) \qquad (31)\\
&=\sum_{\mathbf{e}} c_{\mathbf{e}}\,\hom\big(\circledast(G_1^{\mathbf{e}},\dots,G_k^{\mathbf{e}}),(G,\boldsymbol{u})\big),
\end{aligned}$$

where the last equality is the auxiliary gluing formula (12); an undefined auxiliary composite contributes zero and its term is omitted. Each defined $\circledast(G_1^{\mathbf{e}},\dots,G_k^{\mathbf{e}})$ belongs to $\mathcal{F}$. Thus the auxiliary gluing aggregation depends only on $\tau_G(\boldsymbol{u})$.

The three inputs to ρ are the previous features $\mathbf{h}^{(\ell-1)}(\boldsymbol{u})$, the series composition aggregations, and the auxiliary gluing aggregation. Each depends only on $\tau_G(\boldsymbol{u})$, hence so does $\mathbf{h}^{(\ell)}(\boldsymbol{u})$. □

Conclusion. By Claims E.3 and E.6, as in the proof of Theorem 3.2, the multiset $\{\{\mathbf{h}^{(\ell)}(\boldsymbol{u}) : \boldsymbol{u}\in V(G)^k\}\}$ is the same for G and H at every layer, so no k-FDHN distinguishes G from H. □

F CFI Indistinguishability for Pattern Closures

To prove that two pattern closures have different distinguishing power, we use the ***CFI construction*** [13]. Let G be a connected graph and write $E_G(v)$ for the set of edges incident to v. The ***CFI graph*** $\mathrm{CFI}(G)$ has vertex set

$$V(\mathrm{CFI}(G)) = \{(v,\mathbf{a}) : v\in V(G), \mathbf{a}\in\{0,1\}^{E_G(v)}, \sum_e a_e \equiv 0 \pmod 2\}. \qquad (32)$$

Two vertices $(u,\mathbf{a})$ and $(v,\mathbf{b})$ are adjacent if and only if $e=uv\in E(G)$ and $a_e=b_e$ (*matching adjacency*). The *projection* maps $(v,\mathbf{a})$ to v. For $U\subseteq V(G)$, write G_U for the CFI graph in which gadgets at vertices in U use odd parity and all others use even parity; thus $G_\emptyset=\mathrm{CFI}(G)$. For a vertex $x\in V(G)$, called the *twist vertex*, $G_{\{x\}}$ replaces the even-parity condition at x with odd parity ($\sum_e a_e\equiv 1$). The singleton twists at any two vertices x,y are isomorphic. Indeed, for an edge $e=uv$, flipping the e-coordinate in every fibre vector at both u and v toggles their parity requirements. This involution preserves adjacency: both compared bits flip on e, and the compared bits on every other edge are unchanged. Applying these maps along an x–y path toggles only the endpoint parities, giving $G_{\{x\}}\cong G_{\{y\}}$; for $x=y$, use the identity. Thus we write $\mathrm{CFI}^{\times}(G)$ when the choice of twist vertex is immaterial, with no degree restriction on G.

Lemma F.1 ([58, Lemma 3.14])**.** *Every connected graph G satisfies* $\hom(G,\mathrm{CFI}(G)) \neq \hom(G,\mathrm{CFI}^{\times}(G))$.

A connected graph G is ***CFI-indistinguishable over*** a class $\mathcal{F}$ of patterns if $\hom(F,\mathrm{CFI}(G)) = \hom(F,\mathrm{CFI}^{\times}(G))$ for every $F\in\mathcal{F}$, where a graph is read as a pattern with no labels.

The separation strategy is as follows: find a connected graph $Q\in\langle\mathcal{P}'\rangle_{\triangleright,\odot}\setminus\langle\mathcal{P}\rangle_{\triangleright,\odot}$ and show $\hom(F,\mathrm{CFI}(Q)) = \hom(F,\mathrm{CFI}^{\times}(Q))$ for every $F\in\langle\mathcal{P}\rangle_{\triangleright,\odot}$. Then $\langle\mathcal{P}'\rangle_{\triangleright,\odot}$ distinguishes the pair (by Lemma F.1) while $\langle\mathcal{P}\rangle_{\triangleright,\odot}$ does not. We develop a per-pattern criterion based on the $\mathbb{F}_2$ linear system of Roberson [58]. It reduces the equality for the whole class to a finite family of conditions, one per step of the construction.

In Appendices F.1 and F.2, fix any graph G, an integer $k\geq 1$, and a vertex $\hat{u}\in V(G)$. We work with the fixed-twist pair $G_\emptyset$, $G_{\{\hat{u}\}}$; connectivity is needed when moving the twist in (P1) of Lemma F.4 and when identifying $G_{\{\hat{u}\}}$ with $\mathrm{CFI}^{\times}(G)$.

F.1 The $\mathbb{F}_2$ system

Given a graph F and a homomorphism $\psi\colon F\to G$, define the linear system $\mathcal{S}(F,\psi,U)$ over $\mathbb{F}_2$:

- *Variables:* $x_e^a \in \mathbb{F}_2$ for each $a \in V(F)$, $e \in E_G(\psi(a))$.
- *Parity:* (P_a): $\sum_{e \in E_G(\psi(a))} x_e^a = [\psi(a) \in U]$, for each $a \in V(F)$.
- *Matching:* (M_{ab}): $x_e^a + x_e^b = 0$ where $e = \psi(a)\psi(b)$, for each $ab \in E(F)$.

Write M_ψ for the coefficient matrix (the left-hand side of the system; this does not depend on U) and b_U for the right-hand side vector (which depends on U through the parity equations). The system is $M_\psi x = b_U$.

For $\psi \in \mathrm{Hom}(F, G)$, write $\mathrm{Hom}_\psi(F, G_U) := \{\tilde{\psi} \in \mathrm{Hom}(F, G_U) : \tilde{\psi} \text{ projects to } \psi\}$. Solutions of $\mathcal{S}(F, \psi, U)$ biject with $\mathrm{Hom}_\psi(F, G_U)$ [58, Lemma 3.4].

Lemma F.2. *Let F be a graph and $\psi \in \mathrm{Hom}(F, G)$.*

(F1) *$x = 0$ solves $\mathcal{S}(F, \psi, \emptyset)$, so $|\mathrm{Hom}_\psi(F, G_\emptyset)| = |\ker(M_\psi)| \geq 1$.*

(F2) *The solutions of $\mathcal{S}(F, \psi, U)$ form either the empty set or a coset of $\ker(M_\psi)$, so $|\mathrm{Hom}_\psi(F, G_U)| \in \{0, |\ker(M_\psi)|\}$.*

(F3) *$\hom(F, G_\emptyset) = \hom(F, G_{\{\hat{u}\}})$ if and only if $\mathcal{S}(F, \psi, \{\hat{u}\})$ is solvable for every $\psi \in \mathrm{Hom}(F, G)$.*

Proof. (F1) and (F2) are immediate from linearity. For (F3), partition the homomorphisms by their projections: $\hom(F, G_U) = \sum_{\psi \in \mathrm{Hom}(F,G)} |\mathrm{Hom}_\psi(F, G_U)|$. By (F1), the $U = \emptyset$ sum is $\sum_\psi |\ker(M_\psi)|$ with each term ≥ 1. By (F2), the $U = \{\hat{u}\}$ sum has each term in $\{0, |\ker(M_\psi)|\}$. The sums agree if and only if no term is 0. □

F.2 The per-pattern criterion

For $i \neq j$, let M^{ij} denote the *coincidence connector*: the (k, k)-bilabelled graph on $k - 1$ isolated vertices. Its in- and out-label tuples are equal. Positions i and j sit on one vertex, and every other position sits on its own vertex.

Let $\boldsymbol{v} \in V(G)^k$ be a placement of the labels. A *boundary vector* at $\boldsymbol{v}$ is an assignment to the coordinates (i, e) with $i \in [k]$ and $e \in E_G(v_i)$, and a *situation* at $\boldsymbol{v}$ is a set of boundary vectors at $\boldsymbol{v}$. For a pattern F with labels $\boldsymbol{p}$ and a homomorphism $\psi \in \mathrm{Hom}(F, G)$, the *boundary state* $S(F, \psi)$ is the situation at $\psi(\boldsymbol{p})$ consisting of the restrictions to the label coordinates of the solutions of $\mathcal{S}(F, \psi, \{\hat{u}\})$.

For a set $\mathcal{P}$ of (k, k)-bilabelled patterns, let $\mathrm{Sat}(\mathcal{P})$ be the least family of situations satisfying the following rules. We call its members *saturated* situations.

(i) At every placement $\boldsymbol{v} \in V(G)^k$, the family contains the situation of I: the boundary vectors satisfying the parity equation $\sum_{e \in E_G(v_i)} x_e^i = [v_i = \hat{u}]$ at each label.

(ii) For every $C \in \mathrm{Sat}(\mathcal{P})$ at placement $\boldsymbol{v}$, every $P \in \mathcal{P}$, and every $\psi \in \mathrm{Hom}(P, G)$ with $\psi(\boldsymbol{r}_{\mathrm{out}}) = \boldsymbol{v}$, consider the solutions of $\mathcal{S}(P, \psi, \{\hat{u}\})$ whose out-label restriction lies in C. Their in-label restrictions form a situation in $\mathrm{Sat}(\mathcal{P})$ at $\psi(\boldsymbol{r}_{\mathrm{in}})$.

(iii) The family is closed under intersection of two situations at the same placement.

Theorem F.3 (Per-pattern criterion)**.** *Let $\mathcal{P}$ be a set of (k, k)-bilabelled patterns. Then*

$$\hom(F, G_\emptyset) = \hom(F, G_{\{\hat{u}\}}) \quad \textit{for every } F \in \langle \mathcal{P} \rangle_{\triangleright, \odot} \tag{33}$$

if and only if no situation in $\mathrm{Sat}(\mathcal{P})$ is empty.

Proof. For F in the closure and $\psi \in \mathrm{Hom}(F, G)$, the boundary state $S(F, \psi)$ is nonempty exactly when $\mathcal{S}(F, \psi, \{\hat{u}\})$ is solvable, so by Lemma F.2(F3) the count identity in (33) holds for F exactly when $S(F, \psi)$ is nonempty for every ψ.

We show by induction on the construction of F that $S(F, \psi) \in \mathrm{Sat}(\mathcal{P})$ for every ψ. For I there are no matching equations, so $S(I, \psi)$ is the situation of (i). For $F \triangleright P$, a solution of the composite system is a solution of $\mathcal{S}(P, \psi|_P, \{\hat{u}\})$ together with a solution of $\mathcal{S}(F, \psi|_F, \{\hat{u}\})$ agreeing on the

coordinates of $\boldsymbol{r}_{\mathrm{out}}$, which are the labels of F; so $S(F \triangleright P, \psi)$ is the series step of (ii) applied to $S(F, \psi|_F)$. For $F' \odot F''$, a solution of the glued system is a pair of solutions of the two operand systems agreeing on the label coordinates, so $S(F' \odot F'', \psi) = S(F', \psi|_{F'}) \cap S(F'', \psi|_{F''})$, which is (iii).

If no situation in $\mathrm{Sat}(\mathcal{P})$ is empty, every $S(F, \psi)$ is nonempty and the count identity holds for every F. Conversely, read the three rules forwards to realise each situation as $S(F, \psi)$. At a series or intersection step, the operand homomorphisms agree on the identified labels, so they induce a homomorphism of the quotient into G. No edge can collapse under those identifications, since its image would be a loop in G; hence the composite is defined and belongs to the closure. An empty situation therefore gives an F whose count identity fails. □

The family $\mathrm{Sat}(\mathcal{P})$ is finite and determined by $\mathcal{P}$, k, G and $\hat{u}$: a situation at $\boldsymbol{v}$ is a set of vectors over the coordinates (i, e), and there are at most $k \cdot d_{\max}(G)$ such coordinates, writing $d_{\max}(G)$ for the largest degree of G; there are finitely many placements.

Let G have twist vertex $\hat{u}$, and fix a set of vertices of G, called *safe*. An edge vector $T \in \mathbb{F}_2^{E(G)}$ restricts, at a placement $\boldsymbol{v}$, to the boundary vector $T|_{\boldsymbol{v}}$ with coordinates $(i, e) \mapsto T_e$ for $e \in E_G(v_i)$. The vertices in $\{v_1, \ldots, v_k\}$ are *occupied* at this placement. Call a situation C at $\boldsymbol{v}$ *pushed* if $T|_{\boldsymbol{v}} \in C$ for every safe $c \notin \{v_1, \ldots, v_k\}$ and every T whose set of odd-degree vertices is $\{\hat{u}\} \triangle \{c\}$: the twist can be pushed onto any unoccupied safe vertex.

Lemma F.4 (The pushed invariant)**.** *For a graph G with twist vertex $\hat{u}$ and a fixed safe set, the following hold.*

- (P1) *A pushed situation at $\boldsymbol{v}$ is nonempty whenever some safe vertex unoccupied at $\boldsymbol{v}$ lies in the component of $\hat{u}$, in particular whenever G is connected and some safe vertex is unoccupied.*
- (P2) *The situation of I is pushed at every placement.*
- (P3) *The intersection of two pushed situations at one placement is pushed.*
- (P4) *The series step with A^{ij} carries a pushed situation to a pushed situation.*
- (P5) *So does the series step with the coincidence connector M^{ij}.*

Proof. For (P1), a path from $\hat{u}$ to the unoccupied safe vertex c has odd-degree set $\{\hat{u}\} \triangle \{c\}$; use the empty path when $c = \hat{u}$. For (P2), the parity at each occupied vertex $v_i \neq c$ is $[v_i = \hat{u}]$. Property (P3) follows from the universal quantifier in the definition. For (P4) and (P5), restrictions of a global edge vector agree on every shared edge coordinate and at coincident labels, so they satisfy the additional connector constraints. □

F.3 Analysis of δ-k-LWL

The marked identity $I \triangleright M^{ij}$ counts $[u_i = u_j]$; we write $I^\bullet := I \triangleright M^{12}$ at $k = 2$. The *marked closure* $\langle \mathcal{P} \rangle^\bullet_{\triangleright, \odot}$ is generated by $\mathcal{P}^\bullet := \mathcal{P} \cup \{M^{ij} : i \neq j\}$ under series composition and gluing. Theorem 3.2 applies to this marked closure with pattern set $\mathcal{P}^\bullet$, since each M^{ij} is a (k, k)-bilabelled pattern. The same marking convention applies when auxiliary gluing is included among the operations.

Let J'^i be the variant of J^i with an edge between in-label i and out-label i. With $\mathcal{P}_{k\text{-LWL}} = \{J'^i\} \cup \{A^{ij}\}$, we analyse the closure $\langle \mathcal{P}_{k\text{-LWL}} \rangle_{\triangleright, \odot}$.

F.3.1 Two-way simulation: δ-k-LWL and k-DHN($\mathcal{P}_{k\text{-LWL}}$)

Following Morris et al. [50], δ-k-LWL colours k-tuples $\boldsymbol{u} \in V(G)^k$ and refines the colouring by aggregating only over local neighbours:

$$c^{(\ell)}(\boldsymbol{u}) = \mathrm{HASH}\Big(c^{(\ell-1)}(\boldsymbol{u}), \Big\{\Big\{ c^{(\ell-1)}(\boldsymbol{u}[i \leftarrow w]) : w \in N(u_i) \Big\}\Big\}_{i \in [k]}\Big). \tag{34}$$

Following Morris et al., the initial colour $c^{(0)}(\boldsymbol{u})$ is the *atomic type* of the tuple: its adjacency pattern $([u_i u_j \in E(G)])_{i<j}$ together with its coincidence pattern $([u_i = u_j])_{i<j}$. The algorithm distinguishes G from H if the colour histograms $\big\{\big\{ c^{(\ell)}(\boldsymbol{u}) : \boldsymbol{u} \in V(G)^k \big\}\big\}$ differ for some ℓ.

Lemma F.5 (Two-way simulation for δ-k-LWL)**.** *For every k, δ-k-LWL and k-DHN over the marked closure of $\mathcal{P}_{k\text{-LWL}}$ have the same distinguishing power: δ-k-LWL distinguishes G from H if and only if $\hom(F, G) \neq \hom(F, H)$ for some $F \in \mathrm{cl}(\langle \mathcal{P}_{k\text{-LWL}} \rangle^{\bullet}_{\triangleright,\odot})$.*

Proof. Write $\mathcal{F} = \langle \mathcal{P}_{k\text{-LWL}} \rangle^{\bullet}_{\triangleright,\odot}$. The connectors A^{ij} and M^{ij} multiply counts by the adjacency and coincidence indicators of the atomic type, while J'^{i} sums over replacements by neighbours of u_i. With the product identity for $\odot$, these three connector facts show inductively that every labelled pattern count is determined by a sufficiently refined colour. Thus equal colour histograms give equal graph-level counts.

Conversely, construct a k-DHN whose tuple states encode each refinement colour on the two finite graphs. The A^{ij} and M^{ij} channels first recover the atomic type. At each subsequent layer, choose the transformations to return the indicators of the finitely many current colours; the J'^{i} channels sum these over $N(u_i)$. An injective update of the previous state and these multiplicities encodes the next colour in (34), and an indicator readout recovers its histogram. Thus a colour-histogram distinction gives a network distinction, and Theorem 3.2 gives a separating count in $\mathrm{cl}(\mathcal{F})$. □

F.3.2 k-WL $< \delta$-k-LWL

Lemma F.6 (Equitable colour-class sizes)**.** *Let two finite connected graphs be coloured from a common colour set. Suppose there are common transition counts $\mathrm{t}(a, b)$ such that, in either graph, every vertex of colour a has exactly $\mathrm{t}(a, b)$ neighbours of colour b. If the graphs share a colour, they have the same colour support and proportional colour-class sizes. If, in addition, they have equal vertex counts or one colour class of the same positive size in both graphs, then all corresponding colour-class sizes are equal.*

Proof. Each colour support is exactly the colours reachable from the shared colour through positive transitions, so the supports agree. Write m_a, n_a for the corresponding colour-class sizes. Double-counting edges between colour classes gives $m_a \mathrm{t}(a, b) = m_b \mathrm{t}(b, a)$ and the same identity for n. The ratios m_a/n_a agree along every edge of the connected colour quotient, so the class sizes are proportional. Thus equal vertex counts, or one colour class of the same positive size in both graphs, make all class sizes equal. □

Theorem F.7. *For every $k \geq 2$, k-WL is strictly weaker than atomic δ-k-LWL on finite simple graphs, including disconnected graphs.*

Proof. Stable colouring. We first prove the containment. Fix two hosts G, H indistinguishable by atomic δ-k-LWL. Run the refinement with common colour encodings on the tagged union of their tuple spaces, and let c be a colouring at a stable round. The atomic type is determined by c, and stability gives common transition counts

$$t_i(a, b) = \left|\{w \in N(u_i) : c(\boldsymbol{u}[i \leftarrow w]) = b\}\right|, \quad \text{when } c(\boldsymbol{u}) = a. \tag{35}$$

Indeed, the next colour records these counts, and a stable partition makes that colour a function of the preceding one.

Component columns. Let C, D be connected components of either host, and suppose $\boldsymbol{u} \in V(C)^k$, $\boldsymbol{v} \in V(D)^k$ have $c(\boldsymbol{u}) = c(\boldsymbol{v})$. Fix a coordinate i. Give each $w \in V(C)$ the colour $c(\boldsymbol{u}[i \leftarrow w])$ and each $z \in V(D)$ the colour $c(\boldsymbol{v}[i \leftarrow z])$. We call each of these coloured components a *column*. These are equitable colourings with common transition counts t_i and a shared colour. Choose $j \neq i$, using $k \geq 2$. Equality of coordinates i, j holds at exactly one vertex of each column, namely u_j or v_j. Since equality is recorded by the atomic type, the colour at u_j has a singleton class; the common support forces that colour to occur at v_j, also in a singleton class. By Lemma F.6, this fixes the proportionality factor, so

$$\{\{c(\boldsymbol{u}[i \leftarrow w]) : w \in V(C)\}\} = \{\{c(\boldsymbol{v}[i \leftarrow z]) : z \in V(D)\}\}, |V(C)| = |V(D)|. \tag{36}$$

Run oblivious k-WL on each component separately. Induction on the rounds shows that each round's tuple colour is one common function of c across all components. The initial atomic type is retained, and each update follows by mapping the equal multisets in (36) through the preceding round's function.

Component histograms. A shared tuple colour also determines the entire component histogram. Join two tuples of $V(C)^k$ when they differ in one coordinate by an edge of C. This Cartesian power is connected, because coordinates can be moved one at a time along paths. Its colouring c is equitable with transition counts $\sum_i t_i(a,b)$; different coordinates give different neighbours. The corresponding graph on $V(D)^k$ shares a colour and has the same order by (36). Lemma F.6 therefore gives equal c-histograms, hence equal oblivious k-WL histograms at every round. In particular, equal colours of diagonal tuples $(x,\dots,x)$ determine equal component orders and indistinguishable components.

Assembly. Let F be a nonempty connected pattern with $\mathrm{tw}(F) \le k-1$. For each occurring diagonal colour a, choose a representative vertex x_a with component C_a and set $q_F(a) = \hom(F, C_a)/|V(C_a)|$. This is well-defined: the preceding argument fixes the component order and its k-WL equivalence class, which determines $\hom(F, C_a)$ by the characterisation in Section 2.2. Write C_x for the component containing x, and $d_G(a), d_H(a)$ for the numbers of vertices in the respective hosts whose diagonal tuples have colour a. Every homomorphism from connected F has its image in one component, so

$$\hom(F,G) = \sum_{C \in \mathrm{Comp}(G)} \hom(F,C) = \sum_{x \in V(G)} \frac{\hom(F,C_x)}{|V(C_x)|} = \sum_a d_G(a) q_F(a). \tag{37}$$

Diagonal tuples are recognisable from their atomic type, so equal full colour histograms give $d_G = d_H$ and thus equal F-counts. Homomorphism counts multiply over the connected components of a pattern, proving equality for every pattern of treewidth at most $k-1$. Empty hosts contribute empty sums for nonempty connected patterns; the empty pattern has count one. The characterisation in Section 2.2 gives k-WL indistinguishability, establishing the containment on arbitrary hosts.

Strictness. The closure $\langle \mathcal{P}_{k\text{-LWL}} \rangle_{\triangleright,\odot}$ contains the $(k+1)$-clique K_{k+1}: starting from I, apply A^{ij} for all $i<j$ to form K_k; then J'^1 introduces a fresh vertex v with edge $\{1,v\}$; applying A^{1j} for $j=2,\dots,k$ completes K_{k+1}. Since $\mathrm{tw}(K_{k+1}) = k$, δ-k-LWL can count $\hom(K_{k+1},\cdot)$, which k-WL cannot by bounded-treewidth closedness [26, 22, 53]. □

F.3.3 δ-k-LWL $<$ k-FWL

Every pattern in $\langle \mathcal{P}_{k\text{-LWL}} \rangle_{\triangleright,\odot}$ has treewidth $\le k$. The pattern A^{ij} has width $k-1$, its k labels forming a single bag. The pattern J'^i has width k: its single bag consists of the k in-labels together with the fresh vertex i', and that bag contains $\boldsymbol{r}_{\mathrm{in}}$, $\boldsymbol{r}_{\mathrm{out}}$ and the edge $\{i,i'\}$. The bound follows from Proposition C.2.

The graph S_k is obtained from K_{k+1} by adding, for each k-subset U of $V(K_{k+1})$, a new vertex adjacent to every vertex of U; at $k=2$ it is the Hajós graph [12] (Figure 7). We call the $k+1$ vertices inherited from K_{k+1} the *core vertices* of S_k and the $k+1$ added vertices its *pendant vertices* (each adjacent to exactly k core vertices). The graph is connected, since every pendant has a neighbour in the core clique. Its treewidth is $\mathrm{tw}(S_k) = k$. The core clique gives the lower bound. For the upper bound, use the core clique as one bag and attach, for each pendant, a bag containing that pendant and its k neighbours.

Proposition F.8. *For all $k \ge 2$, $S_k \notin \langle \mathcal{P}_{k\text{-LWL}} \rangle_{\triangleright,\odot}$.*

Proof. Suppose a construction tree builds S_k. Maintain an assignment of positions to vertices in which the current label at position i has assignment i. Assign each vertex of I its label position. Inductively, A^{ij} preserves these assignments, J'^i assigns position i to its fresh vertex, and gluing identifies only labels with the same assignment. Thus current labels remain distinct, and running intersection (Lemma C.1) makes these assignments agree throughout each vertex's occurrence subtree. Two of the $k+1$ core vertices therefore occupy the same position, say p. Their edge must arise from a J'^p-step, since A^{ij} joins different positions.

Cut the tree edge of this step. The two core vertices occur on opposite sides, while every common neighbour must occur on both sides: its incident edges are introduced within those sides, since the cut step introduces only the edge between the two core vertices. By running intersection, all common neighbours occupy the $k-1$ unchanged positions at the cut. But the two core vertices have $k-1$ other core vertices and $k-1$ pendant vertices as common neighbours, contradicting $2(k-1) > k-1$ for $k \ge 2$. □

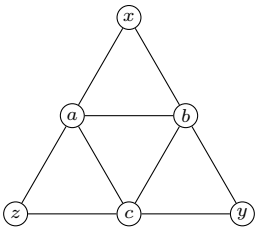


Figure 7: The graph S_2 (Hajós graph; also Figure 8 of Zhang et al. [71]).

Theorem F.9. *For all $k \geq 2$, k-FWL distinguishes $\mathrm{CFI}(S_k)$ from $\mathrm{CFI}^{\times}(S_k)$ and δ-k-LWL does not; δ-k-LWL is therefore strictly weaker than k-FWL. Moreover, $\hom(F, \mathrm{CFI}(S_k)) = \hom(F, \mathrm{CFI}^{\times}(S_k))$ for every $F \in \langle \mathcal{P}_{k\text{-LWL}} \rangle^{\bullet}_{\triangleright,\odot}$.*

Proof. Since S_k is connected and has treewidth k, k-FWL distinguishes the pair by Lemma F.1 and Section 2.2. For the other direction, apply Lemma F.4 with the $k+1$ core vertices as the safe set. Some safe vertex is always unoccupied, so pushed situations are nonempty; only preservation by J'^i remains to be checked.

Suppose J'^i moves position i from v_i to a neighbour w along $f = v_i w$. For a safe c' unoccupied at the new placement, take an edge vector T' pushing the twist to c'. If $c' \neq v_i$, its restriction already belongs to the input situation and satisfies the connector constraints. If $c' = v_i$, choose an unoccupied core c and a path Q from v_i to c avoiding f and every other occupied vertex. Then $T = T' + Q$ pushes the twist to c and agrees with T' on the unchanged labels and on f, as required.

Such a path exists and has at most two edges. If an unoccupied core differs from w, the direct core edge serves. Otherwise w is the only unoccupied core, so all labels occupy the other cores and every pendant is free; a pendant adjacent to both v_i and w supplies the two-edge detour. Thus every saturated situation is pushed. Theorem F.3, applied to $\mathcal{P}^{\bullet}_{k\text{-LWL}}$, gives equality of counts over the marked closure, and Lemma F.5 gives the claimed indistinguishability. □

Morris et al. [50] established δ-k-LWL $\leq$ k-FWL. Zhang et al. [71] settled the $k = 2$ case; the above theorem extends the separation to all $k \geq 2$.

Remark F.10. *Zhang et al. [71] construct the same base graph S_2 (their Figure 8) for the $k = 2$ separation by directly analysing the aggregation neighbourhood of δ-2-LWL on CFI graphs. Our approach arrives at S_k algebraically: Proposition F.8 shows that $S_k \notin \langle \mathcal{P}_{k\text{-LWL}} \rangle_{\triangleright,\odot}$ via the construction-tree formalism, which also yields the generalisation to all k.*

F.3.4 Spectral invariant GNNs and δ-2-LWL

We show that the spectral invariant GNN of Gai et al. [31] is strictly weaker than δ-2-LWL.

A graph is a *parallel edge* with endpoints u, v, $u \neq v$, if its edge set can be partitioned into internally disjoint simple u–v paths that together cover all its vertices. A graph F is a *parallel tree* if there exists a tree T such that F is obtained from T by replacing each edge $uv \in E(T)$ with a parallel edge having endpoints u, v [31, Definitions 3.1–3.2].

Let A'^{ij}_m ($m \geq 1$, $i \neq j$) denote the (k,k)-bilabelled path of length m from position i to position j with fresh internal vertices, with $\boldsymbol{r}_{\mathrm{in}} = \boldsymbol{r}_{\mathrm{out}}$; we use $k = 2$, where the only such pattern is A'^{12}_m. Series composition with A'^{ij}_m counts walks of length m between positions i and j, since homomorphisms of a path coincide with walks. We define $\mathcal{P}_{\mathrm{Spec}} = \{J^i\} \cup \{A'^{ij}_m : m \geq 1\}$.

Following Gai et al. [31], let $\Lambda(G)$ denote the set of distinct eigenvalues of the adjacency matrix A_G, and for $\lambda \in \Lambda(G)$ let P_λ denote the orthogonal projection onto the corresponding eigenspace. The *spectral invariant* of a pair $(u, v) \in V(G)^2$ is the multiset $\mathcal{S}_G(u, v) = \{\{(\lambda, P_\lambda(u, v)) : \lambda \in \Lambda(G)\}\}$, and the spectral invariant GNN is abstracted by the *spectral refinement* [31, Section 2.1]

$$c^{(\ell)}(u) = \textsc{hash}\Big(c^{(\ell-1)}(u), \Big\{\Big\{\big(c^{(\ell-1)}(v), \mathcal{S}_G(u, v)\big) : v \in V(G)\Big\}\Big\}\Big), \tag{38}$$

with a constant initial colour $c^{(0)}$; the model distinguishes G from H if the colour histograms $\{\{c^{(\ell)}(u) : u \in V(G)\}\}$ differ at some round ℓ.

Lemma F.11 (Two-way simulation for spectral invariant GNNs)**.** *Spectral invariant GNNs and* 2*-DHN with pattern set $\mathcal{P}_{\mathrm{Spec}}$ have the same distinguishing power: a spectral invariant GNN distinguishes G from H if and only if* $\hom(F, G) \neq \hom(F, H)$ *for some* $F \in \mathrm{cl}(\langle \mathcal{P}_{\mathrm{Spec}} \rangle_{\triangleright,\odot})$.

Proof. Put $\omega_G(u, v) = (A_G^m(u, v))_{m \geq 0}$, and let $c_W^{(\ell)}$ be the refinement in (38) with $\mathcal{S}_G$ replaced by ω_G. The two refinements have the same distinguishing power. Indeed, $A_G^m = \sum_{\lambda \in \Lambda(G)} \lambda^m P_\lambda$ recovers every walk coordinate from the spectral data. Conversely, equal walk-colour histograms determine equal traces of all powers: the coordinate $\omega_G(u, v)_0 = [u = v]$ singles out the diagonal entry in each round-one multiset. Polynomial interpolation on $\Lambda(G) \cup \Lambda(H)$ then gives equal eigenvalue multiplicities from these traces, hence a common spectrum. Each P_λ is then a polynomial in the adjacency matrix, so the walk vectors determine the spectral data and subsequent spectral colours.

For every generated pattern F, its rooted count is a function of $c_W^{(d)}(u)$, $c_W^{(d)}(v)$ and $\omega_G(u, v)$ for some d. This follows by induction on the construction: I has count one, gluing multiplies counts, a path connector multiplies by a walk coordinate, and J^i sums over one endpoint using the other endpoint's next colour. Summing over both endpoints therefore recovers $\hom(F, G)$ from the round-$(d + 1)$ histogram.

Conversely, on two fixed hosts, a 2-DHN with $\mathcal{P}_{\mathrm{Spec}}$ computes both endpoint colours by induction on the round. Only finitely many colours and walk vectors occur, so finitely many walk coordinates with $m \geq 1$ distinguish the vectors with each fixed zeroth coordinate. Path channels read these coordinates, and J^i channels sum the indicators of their joint values with the endpoint colour. To recover the diagonal and off-diagonal multiplicities, the same channels compute the diagonal walk vector using $A_G(u, u) = 0$ and $A_G^m(u, u) = \sum_v A_G(u, v) A_G^{m-1}(u, v)$ for $m \geq 2$. The diagonal record is thus known from the retained endpoint colour and these diagonal coordinates; subtracting its single occurrence from the corresponding indicator sum gives the off-diagonal multiplicity. This reconstructs the multiset for the next walk colour. The readout over pairs contains $|V(G)|$ copies of the vertex-colour histogram, and the readout's cardinality determines $|V(G)|^2$. Cancelling the host-order factor recovers that histogram on nonempty hosts; empty hosts are immediate. Theorem 3.2 converts the network distinction into a generated-pattern count distinction, and homomorphism-distinguishing closure preserves this equivalence. □

Proposition F.12. *Some pattern of $\langle \mathcal{P}_{\text{2-LWL}} \rangle_{\triangleright,\odot}$ separates a pair of graphs that no pattern of $\langle \mathcal{P}_{\mathrm{Spec}} \rangle_{\triangleright,\odot}$ separates. Hence δ-2-LWL distinguishes a pair of graphs that spectral invariant GNNs do not, which reads $\mathcal{F}^{\delta\text{-2-LWL}} \not\subseteq \mathcal{F}^{\mathrm{Spec}}$ in the homomorphism-expressivity notation of Gai et al. [31].*

Proof. Let B be K_4^- (K_4 minus an edge) with a pendant edge at one of its two degree-2 vertices. Its three degree-3 vertices form a triangle; take these as the safe set of Lemma F.4 at $k = 2$. Two labels always miss a safe vertex, so pushed situations are nonempty. The identity situation and intersection preserve pushedness by (P2) and (P3).

For an $A_m'^{ij}$-step, take any edge vector T pushing the twist to a safe vertex c unoccupied by the two labels. Use the coordinates of T at every vertex of the path connector. At each internal vertex mapped to c, flip the coordinate of an incident edge of c unused by its two path neighbours. Such an edge exists because c has degree 3. These corrections restore parity while preserving the matching equations and both label restrictions.

For a J^i-step, take any edge vector T' pushing the twist to a safe vertex c' unoccupied at the new placement. If c' was also unoccupied before the step, use T' at both placements. Otherwise c' is the vacated vertex; choose a safe vertex c unoccupied before the step and a path Q from c' to c in the safe triangle that avoids the other label. Use $T = T' + \mathbf{1}_{E(Q)}$ at the old placement and T' at the new vertex. The two vectors agree at the unchanged label, and J^i carries no edge constraint. So no pattern of $\langle \mathcal{P}_{\mathrm{Spec}} \rangle_{\triangleright,\odot}$ separates $\mathrm{CFI}(B)$ from $\mathrm{CFI}^{\times}(B)$, while $B = I \triangleright J'^1 \triangleright A^{12} \triangleright J'^1 \triangleright A^{12} \triangleright J'^2 \triangleright A^{12}$ lies in $\langle \mathcal{P}_{\text{2-LWL}} \rangle_{\triangleright,\odot}$ and separates them by Lemma F.1. The statement about the two models follows by Lemma F.5 and by Lemma F.11. □

Remark F.13. *Corollary 3.8 of [31] reaches the same conclusion but with an incorrect proof: it uses S_2 as a separator, claiming $S_2 \in \mathcal{F}^{\delta\text{-2-LWL}}$, whereas $S_2 \notin \mathcal{F}^{\delta\text{-2-LWL}}$ (Theorem F.9 with Lemma F.1).*

Corollary F.14. *The closures satisfy* $\mathrm{cl}(\langle \mathcal{P}_{\mathrm{Spec}} \rangle_{\triangleright,\odot}) \subseteq \mathrm{cl}(\langle \mathcal{P}_{\text{2-LWL}} \rangle^{\bullet}_{\triangleright,\odot})$.

Proof. The containment direction of [31, Corollary 3.8], using [72, Theorems 3.4 and 3.9], states that spectral invariant GNNs are at most as expressive as atomic δ-2-LWL. If two graphs agree on all counts from the marked local closure, Lemma F.5 makes them indistinguishable by δ-2-LWL, hence by spectral invariant GNNs. Lemma F.11 then gives equality of all spectral-pattern counts, which is the claimed closure inclusion. □

With Proposition F.12, spectral invariant GNNs are strictly weaker than δ-2-LWL.

Proposition F.15. *Every pattern in $\langle \mathcal{P}_{\mathrm{Spec}} \rangle_{\triangleright,\odot}$ (with $k = 2$) has as underlying graph a disjoint union of parallel trees; conversely, for every disjoint union F of parallel trees, the closure contains a pattern whose underlying graph is F together with additional isolated vertices. Consequently (Lemma F.11), 2-DHN with pattern set $\mathcal{P}_{\mathrm{Spec}}$ and spectral invariant GNNs have the same distinguishing power.*

Proof. Call the forest before its edges are replaced by parallel edges the *skeleton*. A disjoint union of parallel trees is exactly a graph decomposed into an acyclic skeleton and one parallel edge for each skeleton edge. Indeed, group the replacement paths by their endpoint pair; conversely, replace each edge of every skeleton component by its associated parallel edge, retaining isolated skeleton vertices.

For the forward direction, maintain a decomposition into a forest skeleton and its parallel edges in which the two labels are distinct skeleton vertices, either in different components or at the endpoints of one skeleton edge. It holds for I. A J^i-step moves a label to a fresh isolated vertex. An A'^{ij}_m-step joins two components or adds a path to their labelled skeleton edge. Gluing identifies the labelled skeleton vertices, merging their skeleton edges when both operands have one. In either operation, retain just one copy of any duplicated length-one path, since the two copies become the same edge. The skeleton remains a forest, so the invariant is preserved.

Conversely, take a disjoint union with two distinct labelled skeleton vertices, either adjacent in the skeleton or in different components. We construct it by induction on $|V| + |E|$. If the labels are skeleton-adjacent, remove one path of their parallel edge, including its internal vertices. The remaining skeleton is a forest, with the labels adjacent or separated depending on whether another path remains; induction followed by the corresponding A'^{12}_m-step restores the graph. If the labels are separated and both have skeleton neighbours, split off one branch at the first label, retaining both labels in both parts and making the second label isolated in the branch part. Each part has fewer vertices, the labelled pairs remain separated, and gluing their inductively constructed patterns restores the graph. If one labelled vertex is isolated, delete that vertex and move its label to a skeleton neighbour of the other labelled vertex, if one exists, or to a skeleton vertex of another component. Induction followed by J^i restores the isolated vertex. When there is no such vertex the graph is I. Finally, any union with at least two vertices admits such a label placement: choose the ends of a skeleton edge, or any two vertices if it is edgeless. Adding isolated vertices handles unions of order zero or one.

These isolated vertices do not affect distinguishing power: I determines the host order, so their common factor $|V(G)|^j$, for j added isolated vertices, cancels on equally sized nonempty hosts; empty hosts are immediate. The network consequence follows from Lemma F.11 and Theorem 3.2. □

F.4 Analysis of node-marking subgraph GNNs

Zhang et al. [71] formalise node-based subgraph GNNs as *subgraph WL* tests: colourings $c^{(\ell)}(u, v)$ of pairs $(u, v) \in V(G)^2$, where u indexes the subgraph (the root) and v the active node. Under the canonical *node-marking* policy, every subgraph is the original graph G and the initial colour is the marking indicator

$$c^{(0)}(u, v) = [u = v]. \tag{39}$$

One round refines $c^{(\ell-1)}$ by hashing $c^{(\ell-1)}(u,v)$ together with a fixed selection of *atomic aggregations* [71, Definition 3.1]:

- the *point terms* $c^{(\ell-1)}(u,u)$, $c^{(\ell-1)}(v,v)$ and $c^{(\ell-1)}(v,u)$;
- the *global terms* $\{\{c^{(\ell-1)}(u,w) : w \in V(G)\}\}$ and $\{\{c^{(\ell-1)}(w,v) : w \in V(G)\}\}$;
- the *local terms* $\{\{c^{(\ell-1)}(u,w) : w \in N(v)\}\}$ and $\{\{c^{(\ell-1)}(w,v) : w \in N(u)\}\}$.

We call an aggregation *root-side* if its values are taken at pairs whose first coordinate is u: the point term $c^{(\ell-1)}(u,u)$, the global term $\{\{c^{(\ell-1)}(u,w) : w \in V(G)\}\}$, and the local term over $w \in N(v)$. The graph invariant is the *vertex–subgraph pooling* $\{\{\{\{c^{(\ell)}(u,v) : v \in V(G)\}\} : u \in V(G)\}\}$; the test distinguishes G from H if their vertex–subgraph poolings differ at some round.

Each atomic aggregation corresponds to a bilabelled pattern. For the point terms, let Y^{ij} ($i,j \in \{1,2\}$) denote the $(2,2)$-bilabelled graph on two isolated vertices a_1, a_2 with $\boldsymbol{r}_{\text{in}} = (a_1, a_2)$ and $\boldsymbol{r}_{\text{out}} = (a_i, a_j)$; the chain rule (5) gives $\hom(F \triangleright Y^{11}, (G,(u,v))) = \hom(F,(G,(u,u)))$, and similarly Y^{22} and Y^{21} realise the other two point terms. The global terms are series composition with J^2 for $\{\{c^{(\ell-1)}(u,w) : w \in V(G)\}\}$ and with J^1 for $\{\{c^{(\ell-1)}(w,v) : w \in V(G)\}\}$. The local terms are series composition with J'^2 for $\{\{c^{(\ell-1)}(u,w) : w \in N(v)\}\}$ and with J'^1 for $\{\{c^{(\ell-1)}(w,v) : w \in N(u)\}\}$.

The marking indicator (39) also has a pattern-algebraic counterpart. It is composition with the coincidence connector M^{12} defined in Appendix F.2.

The marked closure subsumes the edge pattern: $\hom(I^\bullet \triangleright J'^2, (G,(u,v))) = [uv \in E(G)]$, so gluing $I^\bullet \triangleright J'^2$ emulates series composition with A^{12}.

Lemma F.16 (Adding the root-side global term does not change distinguishing power)**.** *Let $\mathcal{A}$ be a selection of atomic aggregations and let $\mathcal{A}^+$ be $\mathcal{A}$ together with the root-side global term. If either $\mathcal{A}$ consists of root-side aggregations only, or the compared graphs are connected and $\mathcal{A}$ contains the local term over $w \in N(v)$, then the node-marking subgraph WL tests with selections $\mathcal{A}$ and $\mathcal{A}^+$, both with vertex–subgraph pooling, distinguish the same pairs of graphs.*

Proof. In the root-side case, write $R^{(\ell)}(u) = \left\{\!\!\left\{ c^{(\ell)}_{\mathcal{A}}(u,w) : w \in V(G) \right\}\!\!\right\}$ for the *row multiset* at u. Inductively, $c^{(\ell)}_{\mathcal{A}^+}(u,v)$ is determined by $c^{(\ell)}_{\mathcal{A}}(u,v)$ and $R^{(\ell-1)}(u)$. Every aggregated value has the same root u, so the additional row multiset is shared; predecessor colours are retained. The row multiset $R^{(\ell)}(u)$ therefore determines the entire augmented row multiset, hence the augmented vertex–subgraph pooling. The reverse determination follows because adding an aggregation refines the colours. The connected-host case follows from [71, Theorem 4.4], whose proof (Appendix E.3 and Lemma E.6 there) assumes connected hosts. □

Lemma F.17 (Two-way simulation for node-marking subgraph GNNs)**.** *Let $\mathcal{A}$ be a selection of atomic aggregations and let $\mathrm{SWL}_{\mathcal{A}}$ denote the node-marking subgraph WL with selection $\mathcal{A}$ and vertex–subgraph pooling. Let $\mathcal{P}_{\mathcal{A}} \subseteq \{Y^{11}, Y^{22}, Y^{21}, J^2, J^1, J'^2, J'^1\}$ consist of the patterns corresponding to the aggregations in $\mathcal{A}$, together with J^2. Let G and H be graphs such that at least one of the following holds:*

(i) the root-side global term belongs to $\mathcal{A}$;

(ii) $\mathcal{A}$ consists of root-side aggregations only;

(iii) G and H are connected and $\mathcal{A}$ contains the local term over $w \in N(v)$.

Then $\mathrm{SWL}_{\mathcal{A}}$ distinguishes G from H if and only if $\hom(F,G) \neq \hom(F,H)$ for some $F \in \mathrm{cl}(\langle \mathcal{P}_{\mathcal{A}} \rangle^\bullet_{\triangleright,\odot})$.

Proof. By Lemma F.16, we may include the root-side global term. Write $\mathcal{F}^\bullet = \langle \mathcal{P}_{\mathcal{A}} \rangle^\bullet_{\triangleright,\odot}$. The $\mathcal{F}^\bullet$-*type* of a pair (u,v) is the vector of its labelled counts $\big(\hom(F,(G,(u,v)))\big)_{F \in \mathcal{F}^\bullet}$. The connector identities above match marking, point evaluation, and local or global summation to the corresponding colour operations. As in Lemma F.5, induction on the construction makes each labelled count

a function of a sufficiently refined colour, while finite interpolation recovers each colour from the $\mathcal{F}^\bullet$-type. Equal vertex–subgraph poolings thus imply equal graph-level counts.

For the converse, we must recover the nested pooling. Summing an interpolated colour-class indicator over v is series composition with J^2. For each resulting 2-labelled pattern R, the count is independent of the second host coordinate; write its common value as $\hom(R, (G, (u, \cdot)))$. These rooted counts determine the inner multiset at u. Interpolate again to express the indicator of each occurring inner multiset as a combination of products of these rooted counts. Gluing realises those products, and summing over u gives

$$|V(G)| \sum_u \hom(R, (G, (u, \cdot))) = \hom(R, G). \tag{40}$$

The graph order is determined by $\hom(I, G) = |V(G)|^2$; the empty graph is immediate. Thus the graph-level counts determine the outer multiset as well. □

Corollary F.18. *Under the node-marking policy and vertex–subgraph pooling:*

(a) for connected G and H, the subgraph WL with the local term over $w \in N(v)$ and the global term $\{\{c^{(\ell-1)}(w, v) : w \in V(G)\}\}$ (GSWL, matching DSS-GNN and SUN [71, Proposition 8.1]) distinguishes G from H if and only if some $F \in \mathrm{cl}(\langle\{J'^2, J^1, J^2\}\rangle^\bullet_{\triangleright,\odot})$ separates their homomorphism counts;

(b) the subgraph WL with only the local term over $w \in N(v)$ (SWL(VS), matching DS-GNN, NGNN, and ID-GNN [71, Proposition 8.1]) corresponds in the same sense to $\mathrm{cl}(\langle\{J'^2, J^2\}\rangle^\bullet_{\triangleright,\odot})$;

(c) every pattern in $\{J'^2, J^1, J^2, M^{12}\}$ admits a bag-constrained tree decomposition of width at most 2. By Proposition C.2, both closures contain only graphs of treewidth at most 2, so both tests are bounded by 3-WL on arbitrary graphs [26, 22]. The named architectures inherit the bound through the equivalences of [71].

Proof. Parts (a) and (b) follow from Lemma F.17, using hypotheses (iii) and (ii), respectively, and the architecture identifications in [71, Propositions 3.2 and 8.1]. For (c), adding the root-side global term strengthens either test and satisfies hypothesis (i) on arbitrary graphs. The connectors J'^i have a bag containing their three vertices; J^i has its two label bags, and M^{12} has one vertex. Proposition C.2 therefore bounds the generated treewidth by two, giving the 3-WL bound [26, 22]. □

Part (c) recovers the 3-WL bound of [30, Corollary 7] for these pattern classes on arbitrary graphs.

F.5 Analysis of SLFWL(k)

We analyse the *Subgraph-Local Folklore Weisfeiler–Leman* algorithm, SLFWL(k), which restricts the k-FWL sift to the closed-neighbourhood union $\bigcup_i N[u_i]$, where $N[u] = N(u) \cup \{u\}$. In the pattern-algebraic framework, this corresponds to a *restricted auxiliary gluing* $\circledast_{\mathrm{loc}}$, where the auxiliary vertex $*$ must coincide with or be adjacent to a labelled vertex. We write $\langle\mathcal{P}\rangle_{\triangleright,\odot,\circledast_{\mathrm{loc}}}$ for the resulting pattern class.

We show that δ-k-LWL $<$ SLFWL(k) $<$ k-FWL for all $k \geq 2$.

F.5.1 Two-way simulation: SLFWL(k) and the restricted auxiliary gluing

We extend the closed-neighbourhood SLFWL(2) test of [71, Definition 5.1] to $k \geq 2$. SLFWL(k) colours k-tuples $\boldsymbol{u} \in V(G)^k$ by

$$c^{(\ell)}(\boldsymbol{u}) = \textsc{hash}\Big(c^{(\ell-1)}(\boldsymbol{u}), \Big\{\Big\{\big(c^{(\ell-1)}(\boldsymbol{u}[i \leftarrow w])\big)_{i\in[k]} : w \in \textstyle\bigcup_i N[u_i]\Big\}\Big\}\Big), \tag{41}$$

with initial colours given by the atomic type, as in Appendix F.3.1. For $k \geq 2$, SLFWL(k) simulates δ-k-LWL round by round. To recover the local multiset for position i, choose $j \neq i$. Component j encodes the atomic type of $\boldsymbol{u}[j \leftarrow w]$, including whether $w \in N(u_i)$. Filter the sift multiset by this test and project component i; applying the previous round's simulation recovers precisely

the local-neighbour multiset for position i. The initial colours agree, so induction gives δ-k-LWL $\leq$ SLFWL(k), including their graph-level histograms. Likewise, for $k \geq 2$, k-FWL simulates SLFWL(k) by retaining sift tuples for which at least one adjacency or coincidence test succeeds. Thus SLFWL(k) $\leq$ k-FWL.

Write $\circledast_{\mathrm{loc}}$ for auxiliary gluing restricted to composites whose auxiliary vertex coincides with or is adjacent to an output label. An adjacency or coincidence between positions i and $j \neq i$ in operand F_i suffices, since position i becomes the auxiliary vertex and position j becomes output label j. Every homomorphism of such a composite sends the auxiliary vertex into $\bigcup_m N[u_m]$, so the omitted terms of (12) vanish:

$$\hom(\circledast(F_1, \dots, F_k), (G, \boldsymbol{u})) = \sum_{w \in \bigcup_m N[u_m]} \prod_{i=1}^{k} \hom(F_i, (G, \boldsymbol{u}[i \leftarrow w])). \tag{42}$$

Lemma F.19 (Characterisation of SLFWL(k))**.** *Let $k \geq 2$ and write $\mathcal{F}^* = \langle \{A^{ij}\} \rangle^{\bullet}_{\triangleright, \odot, \circledast_{\mathrm{loc}}}$, the marked closure generated without the introduction patterns J^i.*

(i) If the SLFWL(k) colour histograms of G and H agree at every round, then $\hom(F, G) = \hom(F, H)$ for every $F \in \mathcal{F}^*$.

(ii) Conversely, if $\hom(F, G) = \hom(F, H)$ for every $F \in \mathcal{F}^*$, then SLFWL(k) does not distinguish G from H.

Thus SLFWL(k) has the distinguishing power of $\mathrm{cl}(\mathcal{F}^*)$.

Proof. For (i), the count induction of Lemma F.5 applies, with (42) handling the restricted auxiliary step. For (ii), use the finite interpolation and histogram argument of Theorem 3.2; the additional task is to express restricted sift multiplicities in the $\mathcal{F}^*$-type. After interpolating each component colour, it suffices to recover sums $\sum_{w \in \bigcup_j N[u_j]} \prod_i \hom(H_i, (G, \boldsymbol{u}[i \leftarrow w]))$. Induct on the set of positions in the union, for every operand family in $\mathcal{F}^*$, starting with the empty sum. For $U = \bigcup_{j \in S} N[u_j]$, put $a_m(w) = [wu_m \in E] + [w = u_m] = [w \in N[u_m]]$ and use

$$[w \in U \cup N[u_m]] = [w \in U] + a_m(w) - [w \in U] a_m(w). \tag{43}$$

Choose $i \neq m$, possible since $k \geq 2$. Absorb the adjacency factor by gluing $I \triangleright A^{im}$ into H_i, and the equality factor by gluing the marked identity $I \triangleright M^{im}$ into H_i. Each resulting middle sum is an admissible auxiliary-gluing count: its auxiliary vertex is adjacent to or coincides with label m. The first and last sums follow by induction with the original operands and the two modified families, respectively; undefined composites contribute zero. This completes the interpolation step. □

F.5.2 Separating δ-k-LWL from SLFWL(k)

The graph S_k from Appendix F.3 belongs to $\langle \{A^{ij}\} \rangle_{\triangleright, \odot, \circledast_{\mathrm{loc}}}$ but not to $\langle \mathcal{P}_{k\text{-LWL}} \rangle_{\triangleright, \odot}$. Combined with Theorem F.9, this separates δ-k-LWL from SLFWL(k).

Proposition F.20. *For all $k \geq 2$, $S_k \in \langle \{A^{ij}\} \rangle_{\triangleright, \odot, \circledast_{\mathrm{loc}}}$, hence in $\langle \Sigma_k \rangle_{\triangleright, \odot, \circledast_{\mathrm{loc}}}$.*

Proof. First, a new unlabelled vertex adjacent to all k labels can be attached by restricted auxiliary gluing. For each $j \in [k]$, choose $i(j) \neq j$ and place $A^{i(j)j}$ in operand $i(j)$; gluing all assigned edges within each operand gives an auxiliary vertex adjacent to every label. Call this star gluing.

Write $v_0, \dots, v_k$ for the core vertices of S_k (Appendix F.3), with $v_1, \dots, v_k$ as labels. Start with K_k on these labels, formed by gluing the edges A^{ij} for all $i \neq j$, and attach one pendant by star gluing. For each j, construct G_j with position j adjacent to every other position and with one star-glued pendant. Glue $\circledast_{\mathrm{loc}}(G_1, \dots, G_k)$ onto the initial pattern. Its auxiliary vertex is the remaining core v_0, adjacent to all labels; the pendant of G_j is adjacent to v_0 and all v_i except v_j. Thus the result is S_k, and both auxiliary steps are permitted by $\circledast_{\mathrm{loc}}$ (adjacency suffices). □

By Lemma F.19(i), SLFWL(k) therefore distinguishes any G, H with $\hom(S_k, G) \neq \hom(S_k, H)$.

Corollary F.21. *For all $k \geq 2$, δ-k-LWL is strictly weaker than SLFWL(k).*

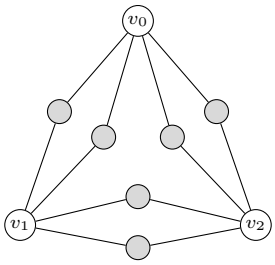


Figure 8: The doubled subdivision D_2. Core vertices (white) form an independent set; each original edge of K_3 is replaced by two parallel paths of length 2 through intermediate vertices (grey).

Proof. By Proposition F.20, $S_k \in \mathcal{F}^*$, so SLFWL(k) can count $\hom(S_k, \cdot)$ by Lemma F.19(i). By Lemma F.1, SLFWL(k) distinguishes $\mathrm{CFI}(S_k)$ from $\mathrm{CFI}^\times(S_k)$. By Theorem F.9, δ-k-LWL does not. Together with δ-k-LWL $\leq$ SLFWL(k) from Appendix F.5.1, this gives strictness. □

F.5.3 Separating SLFWL(k) from k-FWL

The *doubled subdivision of* K_{k+1}, denoted D_k, is obtained by replacing each edge ij of K_{k+1} with two internally disjoint paths of length 2 from i to j. The $k+1$ original vertices of K_{k+1} are called *core vertices*; the $2\binom{k+1}{2}$ new vertices are called *intermediate vertices*. Figure 8 shows D_2.

The core vertices form an independent set in D_k. The graph is connected: every intermediate vertex reaches a core vertex, and each pair of core vertices is joined by a two-edge path. For $k \geq 2$, take a central bag of all cores and attach, for each intermediate vertex m on a doubled edge v_iv_j, the bag $\{v_i, m, v_j\}$. This decomposition has width at most k. Contracting one edge incident to each intermediate vertex onto a core endpoint gives K_{k+1}, so minor-monotonicity gives $\mathrm{tw}(D_k) \geq k$ [8]. Combining the bounds gives $\mathrm{tw}(D_k) = k$ for $k \geq 2$.

Theorem F.22. *For all $k \geq 2$, $\hom(F, \mathrm{CFI}(D_k)) = \hom(F, \mathrm{CFI}^\times(D_k))$ for every $F \in \langle \Sigma_k \rangle^\bullet_{\triangleright,\odot,\circledast_{\mathrm{loc}}}$. Consequently, SLFWL($k$) does not distinguish $\mathrm{CFI}(D_k)$ from $\mathrm{CFI}^\times(D_k)$.*

Proof. Extend the per-pattern criterion to auxiliary gluing: an output boundary vector s is feasible if some auxiliary assignment t makes $s[i \leftarrow t]$ feasible in operand i for every i. This is exactly agreement on shared pattern vertices, so the induction of Theorem F.3 still applies. Use the core vertices of D_k as the safe set in Lemma F.4. Some core is always unoccupied. Only J^i and $\circledast_{\mathrm{loc}}$ need checking.

For J^i, an edge vector T pushing the twist to c' already extends through the input unless $c' = v_i$, the vacated core. In that case reroute to a free core along a two-edge path avoiding all other occupied vertices. If p of the other positions occupy cores, there are at least $k - p$ free cores, hence $2(k - p)$ candidate intermediate vertices, against at most $k - 1 - p$ occupied intermediates. Adding this path changes only the released boundary coordinates, so the required output vector is obtained.

For auxiliary gluing with auxiliary image w, an edge vector T pushing to c' restricts to every operand if $w \neq c'$. If w is an output label, then $w \neq c'$ because the free core c' is unoccupied; the same vector therefore works without rerouting. If $w = c'$, the coincidence case is impossible, so the restriction forces adjacency to a label and hence places some label on an intermediate vertex. Thus at most $p \leq k - 1$ positions occupy cores, leaving at least $k - p$ other free cores and $2(k - p)$ candidate intermediates, against at most $k - p$ occupied intermediates. Choose an unoccupied two-edge path from w to a free core and add it to T. The resulting vector restricts to every operand with one common auxiliary assignment and agrees with T at every output label.

All saturated situations are pushed and nonempty, so the counts agree. Lemma F.19(ii) gives the SLFWL consequence. □

Theorem F.23. *For all $k \geq 2$, k-FWL distinguishes $\mathrm{CFI}(D_k)$ from $\mathrm{CFI}^\times(D_k)$ and SLFWL(k) does not; SLFWL(k) is therefore strictly weaker than k-FWL.*

Proof. Since D_k is connected and $\mathrm{tw}(D_k) = k$, k-FWL distinguishes $\mathrm{CFI}(D_k)$ from $\mathrm{CFI}^{\times}(D_k)$ by Lemma F.1 with Section 2.2. By Theorem F.22, SLFWL(k) does not. Together with SLFWL(k) $\leq k$-FWL from Appendix F.5.1, this gives strictness. □

F.6 Connectors: joint attachment and DHN

The series composition of Section 3 attaches the previous pattern at the single out-tuple $\boldsymbol{r}_{\mathrm{out}}$, in line with the k-sourced graph algebra [17, 59]. The DHN layer [45] multiplies transformed states over *all* vertices of a rooted pattern within a single homomorphism, attaching the previous pattern at every vertex of that pattern at once; we call this *joint attachment*.

A *t-port k-connector* is a tuple $(P, \boldsymbol{r}_{\mathrm{in}}, (\boldsymbol{r}^1_{\mathrm{out}}, \dots, \boldsymbol{r}^t_{\mathrm{out}}))$, where P is a graph, $\boldsymbol{r}_{\mathrm{in}} \in V(P)^k$, and the *port tuples* $\boldsymbol{r}^j_{\mathrm{out}}$ lie in $V(P)^k$ for $j \in [t]$. Vertices may repeat within the input tuple and within each port tuple.

We write *k-connector* when the number of ports is unspecified. A one-port k-connector is precisely a (k,k)-bilabelled graph, with its port tuple serving as the out-labels. Given k-labelled patterns $F_1, \dots, F_t$, the *polyadic series composition* $\triangleright^*(F_1, \dots, F_t; P)$ is the k-labelled graph obtained by identifying the labels of F_j with the port tuple $\boldsymbol{r}^j_{\mathrm{out}}$ for each $j \in [t]$, with label tuple $\boldsymbol{r}_{\mathrm{in}}$. The polyadic chain rule then reads

$$\hom\big(\triangleright^*(F_1, \dots, F_t; P), (G, \boldsymbol{u})\big) = \sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}}} \prod_{j=1}^{t} \hom\big(F_j, (G, \varphi(\boldsymbol{r}^j_{\mathrm{out}}))\big). \tag{44}$$

Take pairwise disjoint copies of $F_1, \dots, F_t$ before identifying their labels with the port tuples of P. Once a homomorphism of P is fixed, compatible homomorphisms on the F_j can be chosen independently. When repeated label entries induce identifications inside P, the terms of the sum whose φ does not respect them vanish, since the corresponding factor is zero. The *multi-port closure* $\langle \mathcal{P} \rangle_{\triangleright^*, \odot}$ of a set $\mathcal{P}$ of k-connectors is the set of k-labelled graphs generated from I by polyadic series composition and gluing. Let $P \in \mathcal{P}$ be a t-port k-connector, where t may depend on P. The *multi-port k-DHN layer* replaces the pattern aggregation of (9) by

$$\mathrm{agg}_P(\boldsymbol{u}) = \sum_{\substack{\varphi: P \to G \\ \varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}}} \mu_P\Big(\big(\mathbf{h}^{(\ell-1)}(\varphi(\boldsymbol{r}^j_{\mathrm{out}}))\big)_{j=1}^{t}, \boldsymbol{x}_\varphi \Big). \tag{45}$$

Theorem F.24 (Multi-port generated-class characterisation)**.** *Let $\mathcal{P}$ be a set of k-connectors and let G and H be graphs. Multi-port k-DHN with pattern set $\mathcal{P}$ distinguishes G from H if and only if some pattern in $\langle \mathcal{P} \rangle_{\triangleright^*, \odot}$ separates them. Equivalently, the model distinguishes G from H if and only if* $\hom(F, G) \neq \hom(F, H)$ *for some* $F \in \mathrm{cl}(\langle \mathcal{P} \rangle_{\triangleright^*, \odot})$.

Proof. The proof of Theorem 3.2 applies with the following changes. For completeness, compute the operand counts and let μ_P multiply the selected coordinate from each port; the polyadic chain rule (44) gives the composite count. For soundness, interpolate the joint function of the port types on the two finite graphs. Each monomial groups into one glued count per port, so the chain rule turns its sum over connector homomorphisms into a polyadic composite count. Undefined composites contribute zero under the existing convention. The same type and readout arguments conclude the proof. □

For featured hosts (G, x) and (H, y), the generated-class equivalence uses generalised counts as in Proposition E.5: distinction is equivalent to unequal counts for some $F \in \langle \mathcal{P} \rangle_{\triangleright^*, \odot}$ and per-vertex transformations α. Multiplying weights over quotient fibres in (44) gives the weighted chain rule. In the completeness direction, we assign each composite vertex's weight to one preimage. In the soundness direction, we combine the feature-indicator expansion of Proposition E.5 with the joint port-type interpolation above.

At $k = 1$, annotate *every* vertex of a rooted pattern as a port. The multi-port layer then reads

$$\mathbf{h}^{(\ell)}(u) = \rho\Big(\mathbf{h}^{(\ell-1)}(u), \Big\{ \sum_{\substack{\varphi: P \to G \\ \varphi(r) = u}} \mu_P\big((\mathbf{h}^{(\ell-1)}(\varphi(v)))_{v \in V(P)}, \boldsymbol{x}_\varphi \big) : P \in \mathcal{P} \Big\} \Big). \tag{46}$$

This is the DHN layer of [45] up to the form of μ_P: DHN sums, over homomorphisms, the product $\prod_v \mu_{P,v}(\mathbf{h}^{(\ell-1)}(\varphi(v)))$ of per-vertex transformed states. The two forms have the same distinguishing power. In one direction, a product of per-vertex transformations is a joint function of the port states. In the other, any joint function on the finitely many state tuples occurring on two compared graphs is a finite sum of such products, realised across channels. The all-vertex product in (4) is the composite used in [45, Lemma 5.3]. Its single-attachment special case is the binary rooted product of that paper: one operand is attached at a chosen vertex of the other operand, and the latter's root is retained. Polyadic composition realises these attachments, so the multi-port closure contains the class of iterated rooted products of $\mathcal{P}$ and is additionally closed under gluing at the root. Theorem F.24 thus yields a homomorphism characterisation of DHN within the present framework.

By convention we write C_2 for the edge K_2, so that $C_2, C_3, \ldots$ form a single family of cycles.

For $s \geq 2$, the *bounded-cycle cactus class* $\mathcal{M}_s$ consists of disjoint unions of graphs generated from $K_1, C_2, \ldots, C_s$ by 1-sums, where a 1-sum identifies one vertex from each of two disjoint graphs. Equivalently, these are graphs whose blocks are single vertices, edges, or cycles of length at most s: 1-sums preserve the nontrivial blocks of their operands, and splitting at cut vertices decomposes each connected graph into its blocks.

Proposition F.25 (DHN reaches the bounded-cycle cactus class)**.** *Let $\mathcal{P}_r^{\mathrm{DHN}}$ consist of the rooted cycles $C_2, C_3, \ldots, C_{r+2}$, each with all vertices as ports. The underlying graphs of $\langle \mathcal{P}_r^{\mathrm{DHN}} \rangle_{\triangleright^*,\odot}$ are exactly the connected members of $\mathcal{M}_{r+2}$.*

Proof. ($\subseteq$) Induct on the construction. The identity is K_1. Polyadic composition with C_m, $2 \leq m \leq r+2$, attaches each operand to the cycle at its port by a 1-sum; gluing is a 1-sum at the roots. Thus every generated graph is a connected member of $\mathcal{M}_{r+2}$.

($\supseteq$) First, the generated class is closed under attaching a rooted generated graph at any vertex of another generated graph, retaining the latter's root. Induct on the construction of the latter graph. For the identity, attachment is gluing. At a gluing step, perform the attachment in an operand containing that vertex, then glue. At a polyadic cycle step, every cycle vertex is a port, so every vertex belongs to an operand; perform the attachment in an operand containing it, then repeat the polyadic composition. These operations produce precisely the required 1-sum.

Now induct on the 1-sum construction of a connected member of $\mathcal{M}_{r+2}$, allowing any chosen root. The singleton is I, and each C_m is obtained by composing with I at every port; vertex transitivity permits any root. For a 1-sum, take the operand containing the chosen root as the base and root the other at the identified vertex. The attachment property completes the induction. □

F.7 Analysis of r-ℓWL: the bounded-cycle cactus class $\mathcal{M}_s$

For r-ℓWL, the relevant class is $\mathcal{M}_{r+2}$ (defined in Appendix F.6).

On connected hosts, the atomic-type variant in [56, Appendix G.3, Equation (7)] determines homomorphism counts of connected members of $\mathcal{M}_{r+2}$ [56, Theorem 2]. The theorem is stated for general hosts, but its proof [56, Appendix G] assumes connectedness; we invoke that connected-host case. The atomic type of [56] records adjacency; adding coincidence as in our convention only refines the colours. Counts multiply over the components of the pattern, so this also gives the counts of every disjoint union in $\mathcal{M}_{r+2}$, with hosts still connected. The single-port closures of cycle patterns are proper subclasses of $\mathcal{M}_{r+2}$ (Example 3.7), while the multi-port closure of Proposition F.25 attains exactly its connected members.

Let $C_m^{(d)}$ be the cycle of length m, with its in-label and out-label at distance d, where $0 \leq d \leq \lfloor m/2 \rfloor$. We restate the single-port claim of Example 3.7, with J'^1 as in Appendix F.3.

Proposition F.26 (Single-port cycle patterns)**.** *With $\mathcal{P}_r = \{J'^1\} \cup \{C_m^{(d)} : 3 \leq m \leq r+2\}$, the pattern closure consists of connected cactus graphs in which every cycle has length at most $r+2$ and contains at most two cut vertices.*

Proof. ($\subseteq$) Each cycle pattern exposes only its in- and out-label, and gluing at a single label cannot close a cycle. Thus every generated graph has the stated properties.

($\supseteq$) Induct on the number of vertices, maintaining that every cycle contains at most two vertices that are cut vertices or the chosen root. Initially choose a cut vertex as root when one exists. The singleton is the identity pattern.

Cut root. Delete the root; take one resulting component as the first piece and the rest as the second, restore the root to both, and glue.

Leaf root. Peel the root by J'^1.

Cycle case. A root that is neither a leaf nor a cut vertex has two neighbours joined by a path avoiding it. This closes a cycle that $C_m^{(d)}$ peels. The rest of the graph hangs at the one further vertex of that cycle which the cut-vertex condition allows to carry anything. □

Proposition F.27 (Forbidden minors)**.** *For $s \geq 2$, $\mathcal{M}_s$ is minor-closed, and a graph lies in $\mathcal{M}_s$ if and only if it has neither C_{s+1} nor the diamond $K_4^- := K_4 - e$ as a minor. For $s \geq 4$ both are minor-minimal, so they are the forbidden minors of $\mathcal{M}_s$; for $s \leq 3$ the diamond has C_{s+1} as a minor, so the forbidden minor is C_{s+1} alone.*

Proof. A 2-connected minor of a 1-sum is a minor of one operand: at most one branch set contains the identified vertex, and deleting that branch set leaves the other branch sets connected within that operand. Neither C_{s+1} nor the diamond is a minor of K_1 or of a cycle of length at most s. Induction on 1-sums therefore excludes both minors from every member of $\mathcal{M}_s$.

Conversely, let F be connected and contain neither minor; induct on $|V(F)|$. The singleton is immediate. Otherwise take a longest simple path $v_0, \ldots, v_q$ and let j be the largest index with v_j adjacent to v_0. Every neighbour of v_0 lies on the path, since another neighbour would extend it. The segment $v_0, \ldots, v_j$ is an edge if $j = 1$, and closes a cycle otherwise. This edge or cycle has at most s vertices, since a longer cycle contracts to C_{s+1}. The cycle has no chord, and no component outside it meets two of its vertices: either would give three internally disjoint paths between two vertices and hence a diamond minor. For $j = 1$, the endpoint v_0 has no neighbour outside the selected edge, so the same attachment property holds. Consequently F is the selected edge or cycle with smaller connected induced graphs attached at its vertices by 1-sums. Each attached graph avoids the two minors and belongs to the class by induction. Apply this argument to each component for a disconnected F.

The characterisation implies minor-closure by transitivity of the minor relation. The cycle is minor-minimal for all $s \geq 2$. Every proper minor of the diamond is a cactus with cycles of length at most four, so it is also minimal when $s \geq 4$; for $s \leq 3$, it contains C_{s+1} as a minor. □

We use the following criterion from [58] to establish homomorphism-distinguishing closedness.

Lemma F.28 (Minor-based criterion for homomorphism-distinguishing closedness)**.** *Let $\mathcal{F}$ be a class of graphs such that*

(i) *$\mathcal{F}$ is closed under minors;*

(ii) *$\mathcal{F}$ is closed under disjoint unions;*

(iii) *$\mathcal{F}$ is closed under restriction to connected components;*

(iv) *$\mathcal{F}$ contains the graph with no vertices;*

(v) *there are connected graphs $H_1, \ldots, H_t$ such that every connected graph outside $\mathcal{F}$ has some H_i as a minor, and $\hom(F, \mathrm{CFI}(H_i)) = \hom(F, \mathrm{CFI}^\times(H_i))$ for every $F \in \mathcal{F}$ and every i.*

Then $\mathcal{F}$ is homomorphism-distinguishing closed.

Proof. Let $G \notin \mathcal{F}$ be connected; we show $G \notin \mathrm{cl}(\mathcal{F})$. By (v), some H_i is a minor of G. We claim $\mathrm{CFI}(G) \equiv_{\mathcal{F}} \mathrm{CFI}^\times(G)$. Suppose not: some $F \in \mathcal{F}$ satisfies $\hom(F, \mathrm{CFI}(G)) \neq \hom(F, \mathrm{CFI}^\times(G))$. Then F admits a weak oddomorphism to G [58, Theorem 3.13]. Since H_i is a connected minor of the connected graph G, transporting that weak oddomorphism along the

minor [58, Lemma 5.6] yields a minor F' of F admitting an oddomorphism to H_i. As H_i is connected, [58, Theorem 3.13] gives $\hom(F', \mathrm{CFI}(H_i)) \neq \hom(F', \mathrm{CFI}^{\times}(H_i))$. By (i), $F' \in \mathcal{F}$, contradicting (v).

Hence $\mathrm{CFI}(G) \equiv_{\mathcal{F}} \mathrm{CFI}^{\times}(G)$, while $\hom(G, \mathrm{CFI}(G)) \neq \hom(G, \mathrm{CFI}^{\times}(G))$ by Lemma F.1. So $G \notin \mathrm{cl}(\mathcal{F})$. Since G was an arbitrary connected graph outside $\mathcal{F}$, every connected member of $\mathrm{cl}(\mathcal{F})$ lies in $\mathcal{F}$. By (iii) and [58, Lemma 2.2], every component of a graph in $\mathrm{cl}(\mathcal{F})$ also lies in $\mathrm{cl}(\mathcal{F})$. The connected case just proved puts each such component in $\mathcal{F}$. The graph is the disjoint union of those components, so it lies in $\mathcal{F}$ by (ii); the graph with no components is the graph with no vertices, which lies in $\mathcal{F}$ by (iv). Hence $\mathrm{cl}(\mathcal{F}) = \mathcal{F}$. □

Lemma F.29 (Both forbidden minors are CFI-indistinguishable over the bounded-cycle cactus class)**.** *Let $s \geq 2$ and let H be C_{s+1} or K_4^-. Then $\hom(F, \mathrm{CFI}(H)) = \hom(F, \mathrm{CFI}^{\times}(H))$ for every $F \in \mathcal{M}_s$.*

Proof. By multiplicativity and Proposition F.25, it suffices to treat the multi-port closure generating connected members of $\mathcal{M}_s$. Extend the per-pattern criterion to a cycle connector whose port images form a closed walk $v_1, \ldots, v_t$ in H, with $2 \leq t \leq s$ (the case $t = 2$ is an edge). Its output at port o consists of vectors b_o extendible to vectors b_j in the operand states that agree on each connector edge. The same solution-gluing argument as in Theorem F.3 proves this rule.

Use every vertex as safe when $H = C_{s+1}$, and the two degree-three vertices when $H = K_4^-$. There is always a safe vertex outside the single output label, so pushed situations are nonempty. Identity and gluing preserve them by Lemma F.4. For the connector step, fix a safe $c \neq v_o$ and an edge vector T pushing the twist to c. We construct compatible operand vectors with output $T|_{v_o}$.

Over C_{s+1}, some vertex c' is missed by the $t \leq s$ ports. Choose a path Q from c to c' in $C_{s+1} - v_o$ and use $(T + Q)|_{v_j}$ at every port. All operands now push to the unoccupied c', and the output is unchanged.

Over the diamond, let p, q be its degree-three vertices and, by symmetry, take $c = p$. At ports with $v_j \neq p$, use $T|_{v_j}$, including the output port. At a port with $v_j = p$, the at most two ports adjacent to it miss some neighbour x of p. Add the path pq if $x = q$, and pxq otherwise, to push the twist to q. Its only edge at p goes to x, so it preserves both connector-edge coordinates. No adjacent ports both map to p, hence these choices are compatible. Thus the connector rule preserves pushedness, proving the count identity. □

Theorem F.30 (Homomorphism-distinguishing closedness of the bounded-cycle cactus class)**.** *For every $r \geq 0$, the class $\mathcal{M}_{r+2}$ is homomorphism-distinguishing closed.*

Proof. Put $s = r + 2$. By Proposition F.27, $\mathcal{M}_s$ is minor-closed, and every graph outside it has C_{s+1} or K_4^- as a minor; both are connected. The generators are connected and 1-sums preserve connectedness, so the components of a member of $\mathcal{M}_s$ are exactly its summands. Hence $\mathcal{M}_s$ is closed under disjoint unions and under restriction to components, and the empty union gives the graph with no vertices. By Lemma F.29, both C_{s+1} and K_4^- are CFI-indistinguishable over $\mathcal{M}_s$. So hypotheses (i) to (v) of Lemma F.28 hold with $H_1 = C_{s+1}$ and $H_2 = K_4^-$, and the lemma gives the claim. □

Remark F.31. *Theorem F.30 also follows from published closedness results. For $s \geq 3$, $\mathcal{M}_s$ is the intersection of the cactus class [54, Theorem 4.16] and the class of graphs of circumference at most s [58, Corollary 6.7]; intersections preserve homomorphism-distinguishing closedness [58, Lemma 6.1]. For $s = 2$, $\mathcal{M}_2$ is the forest class [58, Corollary 6.8].*

F.8 Deciding the criterion for a finite pattern set

Proposition F.32. *For a finite set $\mathcal{P}$ of (k,k)-bilabelled patterns and a finite set $\mathcal{H}$ of connected graphs, whether every member of $\mathcal{H}$ is CFI-indistinguishable over $\langle \mathcal{P} \rangle_{\triangleright,\odot}$ is decidable.*

Proof. Fix $H \in \mathcal{H}$ and a twist vertex. By Theorem F.3, H is CFI-indistinguishable over $\langle \mathcal{P} \rangle_{\triangleright,\odot}$ if and only if every situation in $\mathrm{Sat}(\mathcal{P})$ is nonempty. A situation is a set of boundary vectors at a placement, so there are finitely many situations. The family $\mathrm{Sat}(\mathcal{P})$ is the least family closed

under finitely many rules, each of which is computable from H and $\mathcal{P}$. It is therefore computed by saturation, and nonemptiness of its members is checked directly. □

The criterion of Lemma F.28 is *exact* for a graph class $\mathcal{F}$ if homomorphism-distinguishing closedness of $\mathcal{F}$ implies that all its forbidden minors are CFI-indistinguishable over $\mathcal{F}$.

Corollary F.33. *Let $\mathcal{P}$ be finite and let $\mathcal{F}$ be a minor-closed, union-closed graph class containing the graph with no vertices, and suppose a finite list of its minimal forbidden minors is given. Suppose that, for all finite graphs G, H, $G \equiv_{\mathcal{F}} H$ if and only if $G \equiv_{\langle \mathcal{P} \rangle_{\triangleright,\odot}} H$. There is a finite test whose positive outcome certifies homomorphism-distinguishing closedness of $\mathcal{F}$. If the criterion of Lemma F.28 is exact for $\mathcal{F}$, this test decides its homomorphism-distinguishing closedness.*

Proof. Each minimal forbidden minor is nonempty because $\mathcal{F}$ contains the graph with no vertices. If it were disconnected, each component would be a proper minor and hence belong to $\mathcal{F}$ by minimality. Union closure would then put the forbidden minor in $\mathcal{F}$, a contradiction. Apply Proposition F.32 to the supplied list and accept if every forbidden minor is CFI-indistinguishable over $\langle \mathcal{P} \rangle_{\triangleright,\odot}$. By the assumed equivalence, acceptance is equivalent to CFI-indistinguishability over $\mathcal{F}$, so Lemma F.28 gives closedness. Under exactness, rejection implies non-closedness. □

G Implementation Techniques

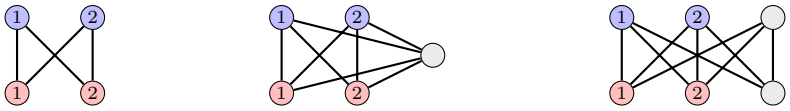
Canonical patterns under symmetry. Patterns related by automorphisms of the underlying graph share the same aggregation result. It therefore suffices to store one canonical representative per isomorphism class and weight each representative by its orbit size, reducing the number of distinct aggregation channels.

Precomputation of homomorphism mappings. The aggregation in (9) sums over homomorphisms $\varphi : P \to G$ with $\varphi(\boldsymbol{r}_{\mathrm{in}}) = \boldsymbol{u}$. Since the same pattern set $\mathcal{P}$ is used at every layer, these homomorphism mappings depend only on G and $\mathcal{P}$. They can therefore be enumerated once as a preprocessing step and reused across all layers; the cost per pattern is $O(\hom(P, G))$. In contrast to DHN [45], where pattern enumeration cost is a primary concern for maintaining $O(n)$ complexity, the $O(n^k)$ state cost of k-DHN already dominates for $k \geq 2$, so complex patterns can be employed more freely. In practice, the patterns worth adding to $\mathcal{P}$ are dense, because sparse patterns of small treewidth are already generated from Σ_k. Dense patterns also prune the homomorphism search aggressively, because real-world graphs are typically sparse.

Designing bilabelled patterns. Suppose a graph P (e.g., K_4) is known to be useful for the target domain. To incorporate P into $\mathcal{P}$, one must choose in-labels $\boldsymbol{r}_{\mathrm{in}}$ and out-labels $\boldsymbol{r}_{\mathrm{out}}$ from $V(P)$. Two natural choices arise: *local* ($\boldsymbol{r}_{\mathrm{in}} = \boldsymbol{r}_{\mathrm{out}}$, labels unchanged) and *exchange* ($\boldsymbol{r}_{\mathrm{in}} \neq \boldsymbol{r}_{\mathrm{out}}$, labels transition through P). A local pattern acts as a multiplicative check: it counts how many ways P can be completed around the current labels, but does not propagate information to new vertices. An exchange pattern aggregates features from tuples that co-participate in a copy of P, enabling information flow through P-substructures. We recommend exchange patterns: they subsume local patterns (which can be recovered via gluing with the identity) and generate richer pattern classes.

Complete and complete bipartite patterns. The most natural candidates for $\mathcal{P}$ beyond Σ_k are complete graphs K_t ($t \geq k{+}1$) and complete bipartite graphs $K_{s,t}$ ($\min(s,t) \geq k$), since both have treewidth at least k and thus lie outside $\langle \Sigma_k \rangle_{\triangleright,\odot}$. Their high symmetry keeps the number of canonical representatives small. For K_t with $k = 2$ and $t \geq 4$: choose $\boldsymbol{r}_{\mathrm{in}} = (a, b)$ and $\boldsymbol{r}_{\mathrm{out}} = (c, d)$ with a, b, c, d distinct; the remaining $t-4$ vertices are free. For K_3 with $k = 2$, the in- and out-labels share one vertex: choose $\boldsymbol{r}_{\mathrm{in}} = (a, b)$ and $\boldsymbol{r}_{\mathrm{out}} = (b, c)$ with a, b, c distinct. For $K_{s,t}$ with parts S, T and $k = 2$: place both in-labels on one side and both out-labels on the other, $\boldsymbol{r}_{\mathrm{in}} = (a, b) \in S^2$, $\boldsymbol{r}_{\mathrm{out}} = (c, d) \in T^2$ with $a \neq b$ and $c \neq d$; the pattern transitions labels across the bipartition. Figure 9 illustrates the labelling.

Initial feature encoding. We focus on the $k = 2$ case, where the state is an $n \times n \times d$ tensor $\mathbf{H}^{(\ell)}$; the techniques extend naturally to general k. The initial state $\mathbf{h}^{(0)}(u, v)$ must encode whether $(u, v) \in E(G)$. A natural approach is to include edge existence as a binary feature and apply a



(a) $K_{2,2}$ (b) $K_{2,2,1}$ (c) $K_{3,3}$

Figure 9: Example exchange patterns ($k = 2$). Blue = in-labels ($\boldsymbol{r}_{\text{in}}$), red = out-labels ($\boldsymbol{r}_{\text{out}}$), grey = free vertices. In each, $\boldsymbol{r}_{\text{in}}$ and $\boldsymbol{r}_{\text{out}}$ lie on opposite sides.

shared multilayer perceptron (MLP): $\mathbf{h}^{(0)}(u, v) = \text{MLP}([x_u; x_v; \mathbf{1}[(u, v) \in E]])$. We found the shared encoding difficult to train, so we embed each vertex as $\boldsymbol{e}_u = \sigma(W_0 x_u + b_0)$ and use separate affine maps for edges and non-edges:

$$\mathbf{h}^{(0)}(u, v) = \begin{cases} \boldsymbol{e}_u & \text{if } u = v, \\ \sigma(W_E[\boldsymbol{e}_u; \boldsymbol{e}_v] + b_E) & \text{if } (u, v) \in E(G), \\ \sigma(W_{\bar{E}}[\boldsymbol{e}_u; \boldsymbol{e}_v] + b_{\bar{E}}) & \text{otherwise,} \end{cases} \tag{47}$$

where $[\cdot\,;\cdot]$ denotes concatenation and σ is the GELU activation.

Readout. For graph-level tasks, we pool the state $\mathbf{H}^{(L)} \in \mathbb{R}^{n \times n \times d}$ into a fixed-size vector. Following Maron et al. [47, 46], we apply max pooling separately to the diagonal entries $\{(u, u)\}$ and the off-diagonal entries $\{(u, v) : u \neq v\}$, and concatenate the results. This diagonal/off-diagonal decomposition is a natural invariant of 2-tuple representations: diagonal entries encode per-vertex information while off-diagonal entries encode pairwise interactions.

Auxiliary gluing as batched matrix multiplication. For $k = 2$, the auxiliary gluing aggregation $\sum_w f_1(\mathbf{h}(u, w)) \circ f_2(\mathbf{h}(w, v))$ is a matrix product over the shared index w. For each channel $c \in [d]$, it computes $\mathbf{Z}_c = \mathbf{F}_c^{(1)} \cdot \mathbf{F}_c^{(2)}$ where $\mathbf{F}^{(1)} = f_1(\mathbf{H}), \mathbf{F}^{(2)} = f_2(\mathbf{H}) \in \mathbb{R}^{n \times n \times d}$. This requires $O(n^2 d^2)$ work for applying f_1, f_2 and $O(n^3 d)$ work for the batched matrix multiplication, which is efficient in both forward and backward passes (the gradient is two additional matrix multiplications).

Common-neighbour aggregation as matrix multiplication. Our implementation of the P_2 aggregation (11) takes $\mu_{P_2}(\mathbf{h}, x_w) = \mathbf{W}_\ell \boldsymbol{e}_w$, where $\boldsymbol{e}_w = \mathbf{h}^{(0)}(w, w) \in \mathbb{R}^d$ is the initial diagonal state, a learnable embedding of x_w shared by all layers and $\mathbf{W}_\ell$ is a learnable matrix of layer ℓ. Let $\mathbf{A}_G$ be the adjacency matrix of G, and for each channel $c \in [d]$ let $\mathbf{D}_c$ be the diagonal matrix whose (w, w) entry is $(\boldsymbol{e}_w)_c$. The aggregation equals $\mathbf{W}_\ell \sum_{w \in N(u_1) \cap N(u_2)} \boldsymbol{e}_w$, and channel c of this sum is the (u_1, u_2) entry of $\mathbf{A}_G \mathbf{D}_c \mathbf{A}_G$. We compute these d products once per forward pass and reuse them in every layer, applying only $\mathbf{W}_\ell$ per layer.

H Experimental Setting

All code is implemented in Rust using the Burn deep learning framework.[1]

BREC. All BREC experiments use deterministic WL refinement with no learned parameters. We measure the timings in Table 1 with an AWS Batch job on an EC2 c8a.xlarge instance (4 vCPUs, 7,500 MiB memory limit), submitted through Argo Workflows.

For the order-3 comparison, we use an AWS EC2 c7i.2xlarge instance (16 GB RAM), as 3-FDHN($\Sigma_3 \cup \{K_{3,3}\}$) exhausts memory on c7i.large. 3-FWL, 3-FDHN(Σ_3), and 3-FDHN($\Sigma_3 \cup \{K_{3,3}\}$) each distinguish the same 380 of 400 BREC pairs, including 80 of 100 CFI pairs. Their total runtimes are 2,573, 2,569, and 3,594 seconds, respectively. The $K_{3,3}$-augmented runtime includes pattern precomputation and is approximately 1.4 times the 3-FWL runtime.

Training (synthetic and TU). We run all training experiments on an AWS EC2 g4dn.xlarge instance (NVIDIA T4 GPU). For the synthetic tasks, each run uses 800 training graphs and 200 test

[1] `https://burn.dev`

Table 3: Licences for existing datasets. For the TU datasets, no explicit dataset licence was found in the linked distributions.

Dataset	Distribution	Licence
BREC [64]	Release: `brec_v3`	MIT
MUTAG [20]	TU archive	Not specified
PTC_MR [37]	TU archive	Not specified
NCI1 [63]	TU archive	Not specified
PROTEINS [10]	TU archive	Not specified
IMDB-BINARY [69]	TU archive	Not specified
IMDB-MULTI [69]	TU archive	Not specified
COLLAB [69]	TU archive	Not specified

Table 4: Full BREC results: pairs distinguished per subcategory. Regular comprises Simple regular, 4-Vertex and Dist. Reg.

Model	$\mathcal{P}$	Basic (60)	Simple regular (100)	4-Vertex (20)	Dist. Reg. (20)	Extension (100)	CFI (100)	Total (400)
2-DHN	Σ_2	0	0	0	0	0	0	0
2-DHN	$+K_3$	59	49	0	0	60	0	168
2-DHN	$+K_4$	26	70	0	0	28	0	124
2-DHN	$+K_{2,2}$	60	100	0	0	91	0	251
2-DHN	$+K_{3,3}$	60	100	20	16	91	0	287
2-FDHN	Σ_2	60	50	0	0	100	60	270
2-FDHN	$+K_3$	60	50	0	0	100	60	270
2-FDHN	$+K_4$	60	100	0	0	100	60	320
2-FDHN	$+K_{2,2}$	60	100	0	0	100	60	320
2-FDHN	$+K_{3,3}$	60	100	20	16	100	60	356

graphs, with split and shuffle seeds 100, 200, and 300 for the three runs. On these synthetic tasks, all models use Adam with learning rate 0.001, hidden dimension 32, two layers, batch size 128, and 100 epochs. Model weights use the backend's random initialisation. Table 2 reports test accuracy at epoch 100 as the mean and population standard deviation (divisor three) across runs. The TU training protocol is given in Appendix J.

Table 3 lists the external datasets used in the experiments and their licence information.

I BREC Regular Category Breakdown

Table 4 shows the full BREC results with all six subcategories.

The best configuration 2-FDHN($+K_{3,3}$) leaves 4 distance-regular pairs and 40 CFI pairs unsolved. Two bipartite patterns of treewidth 3 separate the four distance-regular pairs. Pair indices are zero-based BREC indices, and each pattern is given as an edge list in which ij denotes the edge $\{i, j\}$.

- Pairs 389, 398 and 399 (8-vertex cube): $\{05, 06, 07, 14, 16, 17, 24, 25, 27, 34, 35, 36\}$. The respective homomorphism-count pairs are $(3{,}974{,}658, 3{,}974{,}514)$, $(264{,}192, 262{,}656)$ and $(35{,}686{,}400, 35{,}684{,}864)$.
- Pair 380 (10-vertex subdivision of K_4): $\{05, 07, 09, 16, 17, 18, 24, 28, 29, 34, 35, 36\}$. The homomorphism counts are $5{,}762{,}232$ and $5{,}774{,}328$.

J Evaluation on TU Benchmark Datasets

We evaluate 2-FDHN on seven TU benchmark datasets.

Setup. Datasets (number of graphs in parentheses): MUTAG [20] (188), PTC_MR [37] (344), NCI1 [63] (4,110), PROTEINS [10] (1,113), IMDB-BINARY [69] (1,000), IMDB-MULTI [69] (1,500), COLLAB [69] (5,000). We use 10-fold cross-validation: for each outer fold f, fold

Table 5: Graph classification accuracy (%) on TU benchmarks. Our results report mean ± population standard deviation (divisor 10) over 10 folds. Bold marks the highest reported mean in each column. The 2-FDHN variants use filtered PROTEINS and COLLAB datasets containing graphs with at most 200 vertices.

Method	**MUTAG**	**PTC_MR**	**NCI1**	**PROT.**	**IMDB-B**	**IMDB-M**	**COLLAB**
GIN (ours)	76.4±7.2	53.3±9.5	75.3±1.8	70.8±3.5	70.2±5.1	45.2±3.6	70.4±1.9
2-FDHN	81.8±5.4	57.9±6.3	74.0±2.2	74.0±3.3	69.7±1.9	45.7±3.0	73.6±2.2
2-FDHN($+P_2$)	80.3±7.1	$\mathbf{62.2 \pm 8.0}$	77.7±2.1	73.4±2.6	70.7±3.7	45.7±3.4	74.6±2.3
2-FDHN($+K_4$)	$\mathbf{82.7 \pm 5.9}$	58.1±7.9	77.2±2.3	74.2±3.6	$\mathbf{71.1 \pm 4.8}$	$\mathbf{47.0 \pm 4.1}$	73.1±2.5
Baseline [28]	—	—	69.8±2.2	$\mathbf{75.8 \pm 3.7}$	50.7±2.4	36.1±3.0	55.0±1.9
GIN [28]	—	—	$\mathbf{80.0 \pm 1.4}$	73.3±4.0	66.8±3.9	42.2±4.6	$\mathbf{75.9 \pm 1.9}$

$(f+1) \bmod 10$ serves as validation and the remaining 8 folds as training data. All our models use 100 epochs, hidden dimension 32, 2 layers and learning rate 0.001. GIN uses batch size 128; 2-FDHN uses 128 on MUTAG, PTC_MR and NCI1 and 32 otherwise; 2-FDHN($+P_2$) uses 128 on MUTAG and 32 otherwise; 2-FDHN($+K_4$) uses 32. The 2-FDHN variants retain graphs with at most 200 vertices, giving 1,101 PROTEINS and 4,715 COLLAB graphs; GIN uses the full datasets. We select the epoch with the highest validation accuracy and report *test* accuracy on fold f. GIN is re-run in our codebase.

Results. Table 5 reports test accuracy for GIN, 2-FDHN, and the pattern-augmented 2-FDHN($+P_2$) and 2-FDHN($+K_4$), where P_2 is the path-of-length-2 pattern (Example 3.9), alongside published baselines from Errica et al. [28]. On TU, $+K_4$ uses the local labelling of K_4: the number of K_4s containing each pair is added as a pair feature. These TU experiments provide a general graph-classification check and are not designed to isolate the benefit of additional expressivity.

The planted-motif benchmarks (Table 2) present the complementary picture: when the classification signal requires detecting a specific substructure in d-regular graphs that 1-WL provably cannot distinguish, pattern choice becomes decisive.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1.2 state the contributions, and the theorems and experiments in the paper support them.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses theoretical assumptions, computational costs, and experimental scope in Sections 3–5 and Appendix J.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Assumptions are stated explicitly, and complete proofs are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 5 describes all experiments, and Appendices G–J give the pattern sets, datasets, and training settings. The code is publicly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, experiment logs, data preparation scripts, and reproduction instructions are publicly available at `https://gitlab.com/tmaehara/kdhn`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: BREC involves no training. Appendices H and J give the training settings for the synthetic and TU experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: BREC results are deterministic. The synthetic and TU results report mean $\pm$ population standard deviation over runs and folds (Appendices H and J).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Table 1 and Appendix H report the hardware and running times.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This is a theoretical and empirical study on graph neural network expressivity. It does not involve human subjects, personal data, or dual-use concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work establishes foundational theoretical results on the expressive power of graph neural networks. We do not foresee direct negative societal impacts.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release pretrained models or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Table 3 in Appendix H records the dataset sources and licence information. Explicit licences were not found in the TU distributions.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public code repository includes a README with dependencies, data preparation, experiment commands, and reference logs. The code is released under the MIT License.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLM use in preparing this work is disclosed in the acknowledgments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.