Weisfeiler and Leman Go Relational

Anonymous Author(s)
Anonymous Affiliation
Anonymous Email

Abstract

Knowledge graphs, modeling multi-relational data, improve numerous applications such as question answering or graph logical reasoning. Many graph neural networks for such data emerged recently, often outperforming shallow architectures. However, the design of such multi-relational graph neural networks is ad-hoc, driven mainly by intuition and empirical insights. Up to now, their expressivity, their relation to each other, and their (practical) learning performance is poorly understood. Here, we initiate the study of deriving a more principled understanding of multi-relational graph neural networks. Namely, we investigate the limitations in the expressive power of the well-known Relational GCN and Compositional GCN architectures and shed some light on their practical learning undertaking. By aligning both architectures with a suitable version of the Weisfeiler-Leman test, we establish under which conditions both models have the same expressive power in distinguishing non-isomorphic (multi-relational) graphs or nodes with different structural roles. Further, by leveraging recent progress in designing expressive graph neural networks, we introduce the k-RN architecture that provably overcomes the expressiveness limitations of the above two architectures. Empirically, we confirm our theoretical findings in a node classification setting over small and large multi-relational graphs.

1 Introduction

Recently, GNNs [1, 2] emerged as the most prominent graph representation learning architecture. Notable instances of this architecture include, e.g., Duvenaud et al. [3], Hamilton et al. [4], and Veličković et al. [5], which can be subsumed under the message-passing framework introduced in Gilmer et al. [1]. In parallel, approaches based on spectral information were introduced in, e.g., Defferrard et al. [6], Bruna et al. [7], Kipf and Welling [8], and Monti et al. [9]—all of which descend from early work in Scarselli et al. [2], Baskin et al. [10], Kireev [11], Micheli and Sestito [12], Merkwirth and Lengauer [13], Micheli [14] and Sperduti and Starita [15].

By now, we have a deep understanding of the expressive power of GNNs [16]. To start with, connections between GNNs and Weisfeiler–Leman type algorithms have been shown. Specifically, Morris et al. [17] and Xu et al. [18] showed that the 1-WL limits the expressive power of any possible GNN architecture in terms of distinguishing non-isomorphic graphs. In turn, these results have been generalized to the k-WL, see, e.g., Morris et al. [17], Azizian and Lelarge [19], Geerts et al. [20], Geerts [21], Maron et al. [22], Morris et al. [23, 24], and connected to permutation-equivariant function approximation over graphs, see, e.g., Chen et al. [25], Geerts and Reutter [26], Maehara and NT [27]. Barceló et al. [28] further established an equivalence between the expressiveness of GNNs with readout functions and C^2, the 2-variable fragment of first-order logic with counting quantifiers.

Most previous works focus on graphs that admit labels on nodes but not edges. However, *knowledge* or *multi-relational graphs*, that admit labels on both nodes and edges play a crucial role in numerous applications, such as complex question answering in NLP [29] or visual question answering [30] in the intersection of NLP and vision. To extract the rich information encoded in the graph's multi-relational structure and its annotations, the knowledge graph community has proposed a large set of *relational* GNN architectures, e.g., [31–33] tailored toward knowledge or multi-relational graphs, targeting tasks such as node and link prediction [31, 33, 34]. Notably, Schlichtkrull et al. [31] proposed the first architecture, namely, R-GCN, being able to handle multi-relational data. Further, Vashishth et al. [32]

Submitted to the First Learning on Graphs Conference (LoG 2022). Do not distribute.

proposed an alternative GNN architecture, CompGCN, using less number of parameters and reported improved empirical performance. In the knowledge graph reasoning area, R-GCN and CompGCN, being strong baselines, spun off numerous improved GNNs for node classification and transductive link prediction tasks [35–37]. They also inspired architectures for more complex reasoning tasks such as inductive link prediction [34, 38–40] and query answering [41–43].

Although these approaches show meaningful empirical performance, their limitations in extracting relevant structural information, their learning performance, and their relation to each other are not understood well. For example, there is no understanding of these approaches' inherent limitations in distinguishing between knowledge graphs with different structural features, explicitly considering the unique properties of multi-relational graphs. Hence, a thorough theoretical investigation of multi-relational GNNs' expressive power and learning performance is yet to be established to become meaningful, vital components in today's knowledge graph reasoning pipeline.

Present work. Here, we initiate the study on deriving a principled understanding of the capabilities of GNNs for knowledge or multi-relational graphs. More concretely:

- We investigate the expressive power of two well-known GNNs for multi-relation data, *Relational GCNs* (R-GCN) [31] and *Compositional GCNs* (CompGCN) [32]. We quantify their limitations by relating them to a suitable version of the established Weisfeiler-Leman graph isomorphism test [44]. In particular, we show under which conditions the above two architectures possess the same expressive power in distinguishing non-isomorphic, multi-relational graphs or nodes with different structural features.
- To overcome both architectures' expressiveness limitations, we introduce the k-RN architecture, which provably overcomes their limitations and show that increasing k always leads to strictly more expressive architectures.
- Empirically, we confirm our theoretical findings on established small- and large-scale multi-relational node classification benchmarks.

See Appendix A.1 for an expanded discussion of related work.

2 Preliminaries

As usual, let $[n] = \{1, \dots, n\} \subset \mathbb{N}$ for $n \geq 1$, and let $\{\!\{ \dots \}\!\}$ denote a multiset.

A *(undirected) graph* G is a pair $(V(G), E(G))$ with a *finite* set of *vertices* $V(G)$ and a set of *edges* $E(G) \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$. For notational convenience, we usually denote an edge $\{u, v\}$ in $E(G)$ by (u, v) or (v, u). We assume the usual definition of *adjacency matrix* $\boldsymbol{A}$ of G. A *colored* or *labeled graph* G is a triple $(V(G), E(G), \ell)$ with a *coloring* or *label* function $\ell \colon V(G) \to \mathbb{N}$. Then $\ell(w)$ is a *color* or *label* of w, for w in $V(G)$. The *neighborhood* of v in $V(G)$ is denoted by $N(v) = \{u \in V(G) \mid (v, u) \in E(G)\}$.

An *(undirected) multi-relational graph* G is a tuple $(V(G), R_1(G), \dots, R_r(G))$ with a *finite* set of *vertices* $V(G)$ and *relations* $R_i \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$ for i in $[r]$. The *neighborhood* of v in $V(G)$ with respect to the relation R_i is denoted by $N_i(v) = \{u \in V(G) \mid (v, u) \in R_i\}$. We define *colored* (or *labeled*) multi-relational graphs in the expected way.

Two graphs G and H are *isomorphic* ($G \simeq H$) if there exists a bijection $\varphi \colon V(G) \to V(H)$ preserving the adjacency relation, i.e., (u, v) in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ in $E(H)$. We then call φ an *isomorphism* from G to H. If the graphs have vertex labels, the isomorphism is additionally required to match these labels. In the case of multi-relational graphs G and H, the bijection $\varphi \colon V(G) \to V(H)$ needs to preserve all relations, i.e., (u, v) is in $R_i(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $R_i(H)$ for each i in $[r]$. For labeled multi-relational graphs, the bijection needs to preserve the labels.

We define the atomic type $\mathsf{atp} \colon V(G)^k \to \mathbb{N}$ such that $\mathsf{atp}(\mathbf{v}) = \mathsf{atp}(\mathbf{w})$ for $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ if and only if the mapping $\varphi \colon V(G) \to V(G)$ where $v_i \mapsto w_i$ induces a *partial isomorphism*, i.e., $v_i = v_j \iff w_i = w_j$ and (v_i, v_j) in $E(G) \iff (\varphi(v_i), \varphi(v_j))$ in $E(G)$.

The Weisfeiler-Leman Algorithm. The 1-*dimensional Weisfeiler-Leman algorithm* (1-WL), or *color refinement*, is a simple heuristic for the graph isomorphism problem, originally proposed by Weisfeiler

and Leman [45].[1] Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling vertices. Given an initial coloring or labeling of the vertices of both graphs, e.g., their degree or application-specific information, in each iteration, two vertices with the same label get different labels if the number of identically labeled neighbors is not equal. If, after some iteration, the number of vertices annotated with a specific label is different in both graphs, the algorithm terminates and a stable coloring, inducing a vertex partition, is obtained. We can then conclude that the two graphs are not isomorphic. It is easy to see that the algorithm cannot distinguish all non-isomorphic graphs [47]. Nonetheless, it is a powerful heuristic that can successfully test isomorphism for a broad class of graphs [48–50].

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

$$C^{(t)}(v) \coloneqq \mathsf{RELABEL}\Big(\big(C^{(t-1)}(v), \{\!\{C^{(t-1)}(u) \mid u \in N(v)\}\!\}\big)\Big),$$

where RELABEL injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C^{(0)} \coloneqq \ell$. To test if two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of vertices colored c in $\mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. Moreover, if the number of colors between two iterations, t and $(t+1)$, does not change, i.e., the cardinalities of the images of $C^{(t)}$ and $C^{(t+1)}$ are equal, or, equivalently,

$$C^{(t)}(v) = C^{(t)}(w) \iff C^{(t+1)}(v) = C^{(t+1)}(w),$$

for all vertices v and w in $V(G)$, the algorithm terminates. For such t, we define the *stable coloring* $C^{\infty}(v) = C^{(t)}(v)$ for v in $V(G)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [51].

Due to the shortcomings of the 1-WL in distinguishing non-isomorphic graphs, several researchers, e.g., [52, 53], devised a more powerful generalization of the former, today known as the k-dimensional Weisfeiler-Leman algorithm (k-WL), see Appendix A.2 for details.

Graph Neural Networks. Intuitively, GNNs learn a vectorial representation, i.e., a d-dimensional vector, representing each vertex in a graph by aggregating information from neighboring vertices. Formally, let $G = (V(G), E(G), \ell)$ be a labeled graph with initial vertex features $(\boldsymbol{h}_v^{(0)})_{v \in V(G)}$ in $\mathbb{R}^d$ that are *consistent* with ℓ, that is, $\boldsymbol{h}_u^{(0)} = \boldsymbol{h}_v^{(0)}$ if and only if $\ell(u) = \ell(v)$, e.g., a one-hot encoding of the labelling ℓ. Alternatively, $(\boldsymbol{h}_v^{(0)})_{v \in V(G)}$ can be arbitrary vertex features annotating the vertices of G.

A GNN architecture consists of a stack of neural network layers, i.e., a composition of permutation-invariant or -equivariant parameterized functions. Similarly to 1-WL, each layer aggregates local neighborhood information, i.e., the neighbors' features, around each vertex and then passes this aggregated information on to the next layer.

GNNs are often realized as follows [17]. In each layer, $t > 0$, we compute vertex features

$$\boldsymbol{h}_v^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_v^{(t-1)} \boldsymbol{W}_0^{(t)} + \sum_{w \in N(v)} \boldsymbol{h}_w^{(t-1)} \boldsymbol{W}_1^{(t)}\Big) \in \mathbb{R}^e, \tag{1}$$

for v in $V(G)$, where $\boldsymbol{W}_0^{(t)}$ and $\boldsymbol{W}_1^{(t)}$ are parameter matrices from $\mathbb{R}^{d \times e}$ and σ denotes an entry-wise non-linear function, e.g., a sigmoid or a ReLU function.[2] Following Gilmer et al. [1] and Scarselli et al. [2], in each layer, $t > 0$, we can generalize the above by computing a vertex feature

$$\boldsymbol{h}_v^{(t)} \coloneqq \mathsf{UPD}^{(t)}\Big(\boldsymbol{h}_v^{(t-1)}, \mathsf{AGG}^{(t)}\big(\{\!\{\boldsymbol{h}_w^{(t-1)} \mid w \in N(v)\}\!\}\big)\Big),$$

[1] Strictly speaking, 1-WL and color refinement are two different algorithms. That is, 1-WL considers neighbors and non-neighbors to update the coloring, resulting in a slightly higher expressive power when distinguishing vertices in a given graph, see Grohe [46] for details. For brevity, we consider both algorithms to be equivalent.

[2] For clarity of presentation, we omit biases.

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

$$\boldsymbol{h}_G \coloneqq \mathsf{READOUT}\big(\{\!\!\{\boldsymbol{h}_v^{(T)} \mid v \in V(G)\}\!\!\}\big),$$

to compute a single vectorial representation based on learned vertex features after iteration T. Again, $\mathsf{READOUT}$ may be a differentiable parameterized function. To adapt the parameters of the above three functions, they are optimized end-to-end, usually through a variant of stochastic gradient descent, e.g., [54], together with the parameters of a neural network used for classification or regression.

Graph neural networks for multi-relational graphs. In the following, we describe GNN layers for multi-relational graphs, namely R-GCN [31] and CompGCN [32]. Initial features are computed in the same way as in the previous subsection.

R-GCN. Let G be a labeled multi-relational graph. In essence, R-GCN generalizes Equation (1) by using an additional sum iterating over the different relations. That is, we compute a vertex feature

$$\boldsymbol{h}_{v,\mathsf{R\text{-}GCN}}^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_{v,\mathsf{R\text{-}GCN}}^{(t-1)}\boldsymbol{W}_0^{(t)} + \sum_{i\in[r]}\sum_{w\in N_i(v)} \boldsymbol{h}_{w,\mathsf{R\text{-}GCN}}^{(t-1)}\boldsymbol{W}_i^{(t)}\Big) \in \mathbb{R}^e, \tag{2}$$

for v in $V(G)$, where $\boldsymbol{W}_0^{(t)}$ and $\boldsymbol{W}_i^{(t)}$ for i in $[r]$ are parameter matrices from $\mathbb{R}^{d\times e}$, and σ denotes a entry-wise non-linear function. We note here that the original R-GCN layer defined in [31] uses a mean operation instead of a sum in the most inner sum of Equation (2). We investigate the empirical advantages of this two layer variation in Section 5.

CompGCN. Let G be a labeled multi-relational graph. A CompGCN layer generalizes Equation (1) by encoding relational information as edge features. That is, we compute a vertex feature

$$\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)}\boldsymbol{W}_0^{(t)} + \sum_{i\in[r]}\sum_{w\in N_i(v)} \phi\big(\boldsymbol{h}_{w,\mathsf{CompGCN}}^{(t-1)}, \boldsymbol{z}_i^{(t)}\big)\boldsymbol{W}_1^{(t)}\Big) \in \mathbb{R}^e, \tag{3}$$

for v in $V(G)$, where $\boldsymbol{W}_0^{(t)}$ and $\boldsymbol{W}_1^{(t)}$ are parameter matrices from $\mathbb{R}^{d\times e}$ and $\mathbb{R}^{c\times e}$, respectively, and $\boldsymbol{z}_i^{(t)}$ in $\mathbb{R}^b$ is the learned edge feature for the i-th relation at layer t. Further, $\phi\colon \mathbb{R}^d \times \mathbb{R}^b \to \mathbb{R}^c$ is a *composition map*, mapping two vectors onto a single vector in a non-parametric way, e.g., summation, point-wise multiplication, or concatenation. We note here that the original CompGCN layer defined in [32] uses an additional sum to differentiate between in-going and out-going edges and self loops, see Appendix E for details.

3 Relational Weisfeiler–Leman algorithm

In the following, to study the limitations in expressivity of the above two GNN layers, R-GCN and CompGCN, we define the *multi-relational* 1-*WL* (1-RWL). Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then the 1-RWL computes a vertex coloring $C_{\mathsf{R}}^{(t)}\colon V(G) \to \mathbb{N}$ for $t > 0$ by interpreting the different relations as edge types, i.e.,

$$C_{\mathsf{R}}^{(t)}(v) \coloneqq \mathsf{RELABEL}\Big(\big(C_{\mathsf{R}}^{(t-1)}(v), \{\!\!\{(C_{\mathsf{R}}^{(t-1)}(u), i) \mid i \in [r], u \in N_i(v)\}\!\!\}\big)\Big), \tag{4}$$

for v in $V(G)$. In iteration 0, the coloring $C_{\mathsf{R}}^{(0)} \coloneqq \ell$. In particular, two vertices v and w of the same color in iteration $(t-1)$ get different colors in iteration t if there is a relation R_i such that the number of neighbors in $N_i(v)$ and $N_i(w)$ colored with a certain color is different. We define the stable coloring C_{R}^{∞} in the expected way, analogously to the 1-WL.

Relationship between 1-RWL, R-GCN, and CompGCN Morris et al. [17], Xu et al. [18] established the exact relationship between the expressive power of 1-WL and GNNs. In particular, 1-WL upper bounds the capacity of any GNN architecture for distinguishing nodes in graphs. In turn, over every graph G there is a GNN architecture with the same expressive power as 1-WL for distinguishing nodes in G. In this section, we show that the same relationship can be established between multi-relational 1-WL, on the one hand, and the R-GCN and CompGCN architectures, on the other.

[3] Strictly speaking, Gilmer et al. [1] consider a slightly more general setting in which vertex features are computed by $\boldsymbol{h}_v^{(t+1)} \coloneqq \mathsf{UPD}^{(t+1)}\Big(\boldsymbol{h}_v^{(t)}, \mathsf{AGG}^{(t+1)}\big(\{\!\!\{(\boldsymbol{h}_v^{(t)}, \boldsymbol{h}_w^{(t)}, \ell(v,w)) \mid w \in N(v)\}\!\!\}\big)\Big)$.

Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph, and let

$$\mathbf{W}^{(t)}_{\text{R-GCN}} \;=\; \big(\boldsymbol{W}_0^{(t')}, \boldsymbol{W}_i^{(t')}\big)_{t' \leq t, i \in [r]}$$

denote the sequence of R-GCN parameters given by Equation (2) up to iteration t. Analogously, we denote by

$$\mathbf{W}^{(t)}_{\text{CompGCN}} \;=\; (\boldsymbol{W}_0^{(t')}, \boldsymbol{W}_1^{(t')}, \boldsymbol{z}_i^{(t')})_{t' \leq t, i \in [r]}$$

the sequence of CompGCN parameters given by Equation (3) up to iteration t. We first show that the multi-relational 1-WL upper bounds the expressivity of both the R-GCN and CompGCN layers in terms of their capacity to distinguish nodes in labeled multi-relational graphs.

Theorem 1. *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$ the following holds:*

- *For all choices of initial vertex features consistent with ℓ, sequences $\mathbf{W}^{(t)}_{\textit{R-GCN}}$ of R-GCN parameters, and nodes v and w in $V(G)$,*

$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \implies \boldsymbol{h}^{(t)}_{v,\textit{R-GCN}} = \boldsymbol{h}^{(t)}_{w,\textit{R-GCN}}.$$

- *For all choices of initial vertex features consistent with ℓ, sequences $\mathbf{W}^{(t)}_{\textit{CompGCN}}$ of CompGCN parameters, composition functions ϕ, and nodes v and w in $V(G)$,*

$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \implies \boldsymbol{h}^{(t)}_{v,\textit{CompGCN}} = \boldsymbol{h}^{(t)}_{w,\textit{CompGCN}}.$$

Noticeably, the converse also holds. That is, there is a sequence of parameter matrices $\mathbf{W}^{(t)}_{\text{R-GCN}}$ such that R-GCN has the same expressive power in terms of distinguishing nodes in graphs as the coloring $C^{(t)}_{\mathsf{R}}$. This equivalence holds provided the initial labels are encoded by linearly independent vertex features, e.g., using one-hot encodings. The result also holds for CompGCN as long as the composition map ϕ can express vector scaling, e.g., ϕ is point-wise multiplication or circular-correlation, two of the composition functions studied and implemented in the paper that introduced the CompGCN architecture [32].

Theorem 2. *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$ the following holds:*

- *There are initial vertex features and a sequence $\mathbf{W}^{(t)}_{\textit{R-GCN}}$ of parameters such that for all v and w in $V(G)$,*

$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \iff \boldsymbol{h}^{(t)}_{v,\textit{R-GCN}} = \boldsymbol{h}^{(t)}_{w,\textit{R-GCN}}.$$

- *There are initial vertex features, a sequence $\mathbf{W}^{(t)}_{\textit{CompGCN}}$ of parameters and a composition function ϕ such that for all v and w in $V(G)$,*

$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \iff \boldsymbol{h}^{(t)}_{v,\textit{CompGCN}} = \boldsymbol{h}^{(t)}_{w,\textit{CompGCN}}.$$

On the choice of the composition function for CompGCN architectures As Theorem 2 shows the expressive power of the 1-RWL is matched by that of the CompGCN architectures if we allow the latter to implement vector scaling in composition functions. However, not all composition maps that have been considered in relationship with CompGCN architectures admit such a possibility. Think, for instance, of natural composition maps such as point-wise summation or vector concatenation. Interestingly, we can show that CompGCN architectures equipped with these composition maps are provably weaker in terms of expressive power than the ones studied in the proof of Theorem 2, as they correlate with a weaker variant of 1-WL that we define next.

Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. The *weak multi-relational* 1-*WL* computes a vertex coloring $C^{(t)}_{\mathsf{WR}} \colon V(G) \to \mathbb{N}$ for $t > 0$ as follows:

$$C^{(t)}_{\mathsf{WR}}(v) \coloneqq \mathsf{RELABEL}\Big(\big(C^{(t-1)}_{\mathsf{WR}}(v), \{\!\{C^{(t-1)}_{\mathsf{WR}}(u) \mid i \in [r], u \in N_i(v)\}\!\}, |N_1(v)|, \ldots, |N_r(v)|\big)\Big)$$

for v in $V(G)$. In iteration 0, the coloring $C^{(0)}_{\mathsf{WR}} \coloneqq \ell$. During aggregation, the weak variant does not take information about the relations into account. The only information relative to the different relations is the number of neighbors associated with each of them. We define the stable coloring C^{∞}_{WR} analogously to the 1-WL. As it turns out, this variant is less powerful than the original one.

Proposition 3. *There exist a labeled, multi-relational graph $G = (V(G), R_1(G), R_2(G), \ell)$ and two nodes v and w in $V(G)$, such that $C^{(1)}_{R}(v) \neq C^{(1)}_{R}(w)$ but $C^{\infty}_{WR}(v) = C^{\infty}_{WR}(w)$.*

As shown next, the expressive power of CompGCN architectures that use point-wise summation or vector concatenation is captured by this weaker form of 1-RWL.

Theorem 4. *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$ the following holds:*

- *For all choices of initial vertex features consistent with ℓ, sequence $\mathbf{W}^{(t)}_{CompGCN}$ of CompGCN parameters, and nodes v and w in $V(G)$,*

$$C^{(t)}_{WR}(v) = C^{(t)}_{WR}(w) \implies \boldsymbol{h}^{(t)}_{v,CompGCN} = \boldsymbol{h}^{(t)}_{w,CompGCN},$$

  for either point-wise summation or concatenation as the composition map.

- *There exist initial vertex features and a sequence $\mathbf{W}^{(t)}_{CompGCN}$ of CompGCN parameters, such that for all nodes v and w in $V(G)$,*

$$C^{(t)}_{WR}(v) = C^{(t)}_{WR}(w) \iff \boldsymbol{h}^{(t)}_{v,CompGCN} = \boldsymbol{h}^{(t)}_{w,CompGCN},$$

  for either point-wise summation or concatenation as the composition map.

Together with Proposition 3 and Theorem 2, this result states that CompGCN architectures based on vector summation or concatenation are provably weaker in terms of their capacity to distinguish nodes in graphs than the ones that use vector scaling.

We have shown that R-GCN and CompGCN with point-wise multiplication have the same expressive power in terms of distinguishing non-isomorphic multi-relational graphs or distinguishing nodes in a multi-relational graph. As it turns out, these two architectures actually define the *same* functions. A similar result holds between CompGCN with vector summation/subtraction and concatenation. See Appendix B.2 for details.

4 Limitations and more expressive architectures

Theorem 1 shows that both R-GCN as well as CompGCN have severe limitations in distinguishing structurally different multi-relational graphs. Indeed the following results shows that there exist pairs of non-isomorphic, multi-relational graphs that neither R-GCN nor CompGCN can distinguish.

Proposition 5. *For all $r \geq 1$, there exists a pair of non-isomorphic graphs $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ and $H = (V(H), R_1(H), \ldots, R_r(H), \ell)$ that cannot be distinguished by R-GCN or CompGCN.*

We note here that the two graphs G and H from the above theorem can also be used to show that neither R-GCN nor CompGCN will be able to compute different features for nodes in G and H, making them indistinguishable. Hence, to overcome the limitations of the CompGCN and R-GCN, we introduce *local k-order relational networks* (k-RNs), leveraging recent progress in overcoming GNNs' inherent limitations in expressive power [16, 17, 23, 24]. To do so, we first extend the local k-dimensional Weisfeiler–Leman algorithm [23], see Appendix A.2, to multi-relational graphs.

Multi-relational local k-WL. Given a multi-relational graph $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$, we define the *multi-relational atomic type* $\mathsf{atp}_r \colon V(G)^k \to \mathbb{N}$ such that $\mathsf{atp}_r(\mathbf{v}) = \mathsf{atp}_r(\mathbf{w})$ for $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ if and only if the mapping $\varphi \colon V(G) \to V(G)$ where $v_g \mapsto w_g$ induces a partial isomorphism, preserving the relations, i.e., we have $v_p = v_q \iff w_p = w_q$ and $(v_p, v_q) \in R_i(G) \iff (\varphi(v_p), \varphi(v_q)) \in R_i(G)$ for i in $[r]$. The *multi-relational local k-WL* (k-RLWL) computes $C^{(t)}_{k,r} \colon V(G)^k \to \mathbb{N}$ for $t \geq 0$, where $C^{(0)}_{k,r} \coloneqq \mathsf{atp}_r(\mathbf{v})$, and refines a coloring $C^{(t)}_{k,r}$ (obtained after t iterations of the k-RLWL) via the *aggregation function*

$$\begin{aligned} M^{(t)}_r(\mathbf{v}) \coloneqq \big(&\{\!\{(C^{(t)}_{k,r}(\theta_1(\mathbf{v}, w)), i) \mid w \in N_i(v_1) \text{ and } i \in [r]\}\!\}, \ldots, \\ &\{\!\{(C^{(t)}_{k,r}(\theta_k(\mathbf{v}, w)), i) \mid w \in N_i(v_k) \text{ and } i \in [r]\}\!\}\big), \end{aligned} \tag{5}$$

where $\theta_j(\mathbf{v}, w) \coloneqq (v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k)$. That is, $\theta_j(\mathbf{v}, w)$ replaces the j-th component of the tuple $\mathbf{v}$ with the vertex w. Like the local k-WL, the algorithm considers only the local j-neighbors,

i.e., v_i and w must be adjacent, for each relation in each iteration and additionally differentiates between different relations. The coloring functions for the iterations of the multi-relational k-RLWL are then defined by

$$C^{(t+1)}_{k,r}(\mathbf{v}) \coloneqq (C^{(t)}_{k,r}(\mathbf{v}), M^{(t)}_r(\mathbf{v})).$$

In the following, we derive a neural architecture, the k-RN, that has the same expressive power as the k-RLWL in terms of distinguishing non-isomorphic multi-relational graphs.

The k-RN architecture. Given a labeled, multi-relational graph G, for each k-tuple $\mathbf{v}$ in $V(G)^k$, a k-RN architecture computes an initial feature $\boldsymbol{h}^{(0)}_v$ *consistent* with its multi-relational atomic type, e.g., a one-hot encoding of $\mathsf{atp}_r(\mathbf{v})$. In each layer, $t > 0$, a k-RN computes a k-tuple feature

$$\begin{aligned}\boldsymbol{h}^{(t)}_{\mathbf{v},k} \coloneqq \mathsf{UPD}^{(t)}\Big(\boldsymbol{h}^{(t-1)}_{\mathbf{v},k}, \mathsf{AGG}^{(t)}\big(\{\!\!\{\phi(\boldsymbol{h}^{(t-1)}_{\theta_1(\mathbf{v},w),k}, \boldsymbol{z}^{(t)}_i) \mid w \in N_i(v_1) \text{ and } i \in [r]\}\!\!\}, \dots, \\ \{\!\!\{\phi(\boldsymbol{h}^{(t-1)}_{\theta_k(\mathbf{v},w),k}, \boldsymbol{z}^{(t)}_i) \mid w \in N_i(v_k) \text{ and } i \in [r]\}\!\!\}\big)\Big) \in \mathbb{R}^e,\end{aligned} \tag{6}$$

where the functions $\mathsf{UPD}^{(t)}$ and $\mathsf{AGG}^{(t)}$ for $t > 0$ may be a differentiable parameterized functions, e.g., neural networks. Similarly to Equation (3), $\boldsymbol{z}^{(t)}_i$ in $\mathbb{R}^c$ is the learned edge feature for the ith relation at layer t and $\phi\colon \mathbb{R}^d \times \mathbb{R}^b \to \mathbb{R}^c$ is a composition map. In the case of graph-level tasks, e.g., graph classification, one uses

$$\boldsymbol{h}_G \coloneqq \mathsf{READOUT}\big(\{\!\!\{\boldsymbol{h}^{(T)}_{\mathbf{v}} \mid \mathbf{v} \in V(G)^k\}\!\!\}\big) \in \mathbb{R}^e, \tag{7}$$

to compute a single vectorial representation based on learned k-tuple features after iteration T. The following results shows that the k-RLWL upperbounds the expressivity of any k-RN in terms of distinguishing non-isomorphic graphs.

Proposition 6. *Let $G = (V(G), R_1(G), \dots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$, $r > 0$, $k \geq 1$, and all choices of $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, and all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)$,*

$$C^{(t)}_{k,r}(\mathbf{v}) = C^{(t)}_{k,r}(\mathbf{w}) \implies \boldsymbol{h}^{(t)}_{\mathbf{v},k} = \boldsymbol{h}^{(t)}_{\mathbf{w},k}.$$

Moreover, we can also show the converse, resulting in the following theorem.

Proposition 7. *Let $G = (V(G), R_1(G), \dots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$ and $k \geq 1$, there exists $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, such that for all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)$,*

$$C^{(t)}_{k,r}(\mathbf{v}) = C^{(t)}_{k,r}(\mathbf{w}) \iff \boldsymbol{h}^{(t)}_{\mathbf{v},k} = \boldsymbol{h}^{(t)}_{\mathbf{w},k}.$$

The following result implies that increasing k leads to a strict boost in terms of expressivity of the k-RLWL and k-RN architectures in terms of distinguishing non-isomorphic multi-relational graphs.

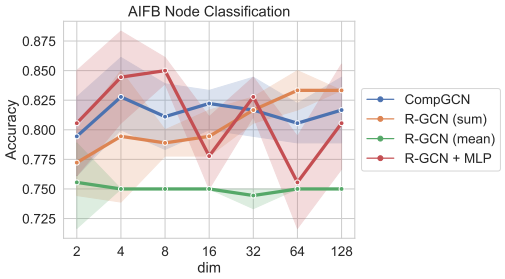
Proposition 8. *For $k \geq 2$ and $r \geq 1$, there exists a pair of non-isomorphic multi-relational graphs $G_r = (V(G_r), R_1(G_r), \dots, R_r(G_r), \ell)$ and $H = (V(H_r), R_1(H_r), \dots, R_r(H_r), \ell)$ such that:*

- *For all choices of $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, for $t > 0$, and $\mathsf{READOUT}$ the k-RN architecture will not distinguish the graphs G_r and H_r.*
- *There exists $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, for $t > 0$, and $\mathsf{READOUT}$ such that the $(k+1)$-RN will distinguish them.*

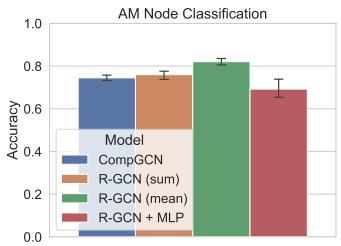
Moreover, the following results shows that for $k = 2$ the k-RN architecture is strictly more expressive than CompGCN and R-GCN in distinguishing non-isomophics graphs.

Corollary 9. *There exists a 2-RN architecture that is strictly more expressive than the CompGCN and the R-GCN architecture in terms of distinguishing non-isomorphic graphs.*

See Appendix C for discussion on scalability and node-level prediction with a k-RN architecture.



(a) AIFB results with varying input feature dimension.



(b) AM results with $dim = 4$.

Figure 1: Node classification performance of CompGCN and R-GCN on smaller (AIFB) and larger (AM) graphs. Initial vertex feature dimensions higher than 4 do not improve the accuracy.

5 Experimental study

Here, we investigate to what extend the above theoretical results hold for real-world data distribution. Specifically, we aim to answer the following questions.

Q1 Does the theoretical equivalence of R-GCN and CompGCN hold in practice?
Q2 Does the performance depend on the dimension of node features?
Q3 Does CompGCN benefit from normalization and learnable edge weights?
Q4 Does the theoretical difference in composition functions of CompGCN hold in practice?

Datasets. To answer Q1 to Q4, we investige R-GCN and CompGCN's empirical performance on the small-scale AIFB (6 000 nodes) and the large-scale AM (1.6 million nodes) [55] vertex classification benchmark dataset; see Appendix F for dataset statistics.

Featurization. Most relational GNNs for vertex- and link-level tasks assume that the initial vertex states come from a learnable vertex embedding matrix [56, 57]. However, this vertex feature initialization or featurization method makes the model inherently transductive, i.e., the model must be re-trained when adding new vertices. Moreover, such an initialization strategy is incompatible with our Weisfeiler-Leman-based theoretical results since a learnable vertex embedding matrix will result in most initial node features being pair-wise different. Here, however, being faithful to the Weisfeiler-Leman formulation, we initialize *all* vertex features with the *same* d-dimensional vector[4], namely, a standard basis vector of $\mathbb{R}^d$, e.g., $(1, 0, \ldots, 0)$ in $\mathbb{R}^d$. Relation-specific weight matrices in the case of R-GCN and edge features in the case of CompGCN are still learnable. We stress here that such a featurization strategy endows GNNs with inductive properties. Since we are using the same vertex feature initialization, we can perform inference on previously unseen vertices or graphs.

Implementation. We use the R-GCN and CompGCN implementation provided by PyG framework [59]. The source code of all methods and evaluation procedures is available at `https://www.github.com/ABC/XXX`.[5] For the smaller AIFB dataset, both models use two GNN layers. For the larger AM dataset, R-GCN saturates with three layers. Following the theory, we do not use any basis decomposition of relation weights in R-GCN. We list other hyperparameters in Appendix F. We report averaged results of five independent runs using different random seeds. We conducted all experiments in full-batch mode on a single GPU using a Tesla V100 32 GB or RTX 8000.

Discussion. Probing R-GCN with different aggregations and CompGCN on the smaller AIFB (Fig. 1a) and larger AM (Figure 1b) datasets, we largely confirm the theoretical hypothesis of their expressiveness equivalence (**Q1**) and observe similar performance of both GNNs. The higher variance on AIFB is due to the small test set size (36 nodes), i.e., one misclassified vertex drops accuracy by $\approx 3\%$.

To test if increasing the input vertex feature dimensions leads to more expressive GNN architectures (**Q2**), we vary the initial vertex feature dimension in $\{2, 4, 8, \ldots, 64, 128\}$ on the smaller AIFB dataset (Figure 1a) and do not observe any significant differences starting from $d = 4$ and above. Having

[4]We also probed a vector initialized with the Glorot and Bengio [58] strategy, showing similar results.
[5]Hidden for the anonymous review.

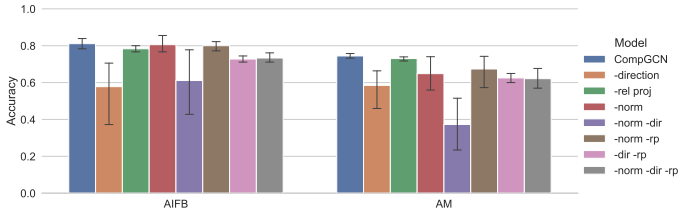


Figure 2: CompGCN ablations. Directionality (*-dir*) and normalization (*-norm*) are the most crucial components, i.e., their removal does lead to significant performance drops.

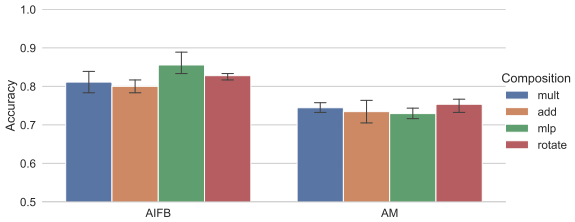


Figure 3: CompGCN with different composition functions. No significant differences.

identified that, we report the best results of compared models on the larger AM graph with the vertex feature dimension d in $\{4, 8\}$.

Following the theory where the `sum` aggregator is most expressive, we investigate this finding on the smaller AIFB dataset for both GNNs. R-GCN with `mean` aggregation shows slightly better results on the larger AM dataset, which we attribute to the unstable optimization process of the `sum` aggregator where nodes might have thousands of neighbors, leading to large losses and noisy gradients. We hypothesize that stabilizing the training process on larger graphs might improve performance.

Furthermore, we perform an ablation study (Figure 2) of main CompGCN components (**Q3**), i.e., direction-based weighting (over direct, inverse, and self-loop edges), relation projection update in each layer, and message normalization in the GCN style $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$; see also Appendices D and E.

The crucial components for the smaller and larger graphs are (1) three-way direction-based message passing and (2) normalization. Replacing message passing over three directions (and three weight matrices) with one weight matrix using a single adjacency leads to a significant drop in performance. Removing normalization increases variance in the larger graph. Finally, removing both directionality and normalization leads to significant degradation in predictive performance.

Studying composition functions (Figure 3), we do not find significant differences among non-parametric `mult`, `add`, `rotate` functions (**Q4**); see Appendix E. Performance of an MLP over a concatenation of node and edge features falls within confidence intervals of other compositions and does not exhibit a significant accuracy boost.

6 Conclusion

Here, we investigated the expressive power of two popular GNN architectures for knowledge or multi-relational graphs, namely, CompGCN and R-GCN. By deriving a variant of the 1-WL, we quantified their limits in distinguishing vertices in multi-relational graphs. Further, we investigated under which conditions, i.e., the choice of the composition function, CompGCN, reaches the same expressive power as R-GCN. To overcome the limitations of the two architectures, we derived the provably more powerful k-RN architecture. By increasing k, the k-RN architecture gets strictly more expressive. Empirically, we verified that our theoretical results translate largely into practice. Using CompGCN and R-GCN in a vertex classification setting over small and large multi-relational graphs shows that both architectures provide a similar performance level. We believe that our paper is the first step in a principled design of GNNs for knowledge or multi-relational graphs.

References

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

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

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

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

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

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

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

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

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

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

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

[12] A. Micheli and A. S. Sestito. A new neural network model for contextual processing of graphs. In *Italian Workshop on Neural Nets Neural Nets and International Workshop on Natural and Artificial Immune Systems*, volume 3931 of *Lecture Notes in Computer Science*, pages 10–17. Springer, 2005.

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

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

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

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

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

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

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

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

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

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

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

[24] C. Morris, G. Rattan, S. Kiefer, and S. Ravanbakhsh. Speqnets: Sparsity-aware permutation-equivariant graph networks. *CoRR*, abs/2203.13913, 2022.

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

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

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

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

[29] B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, and J. Sun. A survey on complex question answering over knowledge base: Recent advances and challenges. *CoRR*, abs/2007.13069, 2020.

[30] N. Huang, Y. R. Deshpande, Y. Liu, H. Alberts, K. Cho, C. Vania, and I. Calixto. Endowing language models with multimodal knowledge graph representations. *CoRR*, abs/2206.13163, 2022.

[31] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, 2018.

[32] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[33] Z. Ye, Y. Jaya Kumar, G. Ong Sing, F. Song, and J. Wang. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741, 2022.

[34] Z. Zhu, Z. Zhang, L.-P. A. C. Xhonneux, and J. Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *CoRR*, abs/2106.06935, 2021. URL `https://arxiv.org/abs/2106.06935`.

[35] M. Galkin, P. Trivedi, G. Maheshwari, R. Usbeck, and J. Lehmann. Message passing for hyper-relational knowledge graphs. In *Conference on Empirical Methods in Natural Language Processing*, pages 7346–7359, 2020.

[36] D. Yu, Y. Yang, R. Zhang, and Y. Wu. Generalized multi-relational graph convolution network. *CoRR*, abs/2006.07331, 2020.

[37] Z. Zhang, J. Wang, J. Ye, and F. Wu. Rethinking graph convolutional networks in knowledge graph completion. In *ACM Web Conference 2022*, page 798–807, 2022.

[38] K. Teru, E. Denis, and W. Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457, 2020.

[39] M. Ali, M. Berrendorf, M. Galkin, V. Thost, T. Ma, V. Tresp, and J. Lehmann. Improving inductive link prediction using hyper-relational facts. In *International Semantic Web Conference*, pages 74–92. Springer, 2021.

[40] Y. Zhang and Q. Yao. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM Web Conference 2022*, pages 912–924, 2022.

[41] D. Daza and M. Cochez. Message passing for query answering over knowledge graphs. *CoRR*, abs/2002.02406, 2020.

[42] D. Alivanistos, M. Berrendorf, M. Cochez, and M. Galkin. Query embedding on hyper-relational knowledge graphs. In *International Conference on Learning Representations*, 2022.

[43] Z. Zhu, M. Galkin, Z. Zhang, and J. Tang. Neural-symbolic models for logical queries on knowledge graphs. In *International Conference on Machine Learning*, 2022.

[44] C. Morris, G. Rattan, S. Kiefer, and S. Ravanbakhsh. SpeqNets: Sparsity-aware permutation-equivariant graph networks. In *International Conference on Machine Learning*, pages 16017–16042, 2022.

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

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

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

[48] V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On the power of color refinement. In *International Symposium on Fundamentals of Computation Theory*, pages 339–350, 2015.

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

[50] S. Kiefer, P. Schweitzer, and E. Selman. Graphs identified by logics with counting. In *International Symposium on Mathematical Foundations of Computer Science*, pages 319–330, 2015.

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

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

[53] N. Immerman and E. Lander. *Describing Graphs: A First-Order Approach to Graph Canonization*, pages 59–81. Springer, 1990.

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

[55] P. Ristoski, G. K. D. de Vries, and H. Paulheim. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International semantic web conference*, pages 186–194, 2016.

[56] M. Wang, L. Qiu, and X. Wang. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3):485, 2021.

[57] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, and J. Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[58] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

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

[60] V. Degraeve, G. Vandewiele, F. Ongenae, and S. Van Hoecke. R-GCN: the R could stand for random. *CoRR*, abs/2203.02424, 2022.

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

[62] C. Qian, G. Rattan, F. Geerts, C. Morris, and Mathias Niepert. Ordered subgraph aggregation networks. *CoRR*, abs/2206.11168, 2022.

[63] Z. Sun, Z.-H. Deng, J.-Y. Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

A Appendix

A.1 Related work

In the following, we expand on relevant related work.

GNNs. Recently, GNNs [1, 2] emerged as the most prominent graph representation learning architecture. Notable instances of this architecture include, e.g., Duvenaud et al. [3], Hamilton et al. [4] and Veličković et al. [5], which can be subsumed under the message-passing framework introduced in Gilmer et al. [1]. In parallel, approaches based on spectral information were introduced in, e.g., Defferrard et al. [6], Bruna et al. [7], Kipf and Welling [8] and Monti et al. [9]—all of which descend from early work in Scarselli et al. [2], Baskin et al. [10], Kireev [11], Micheli and Sestito [12], Merkwirth and Lengauer [13], Micheli [14] and Sperduti and Starita [15].

Limits of GNNs and more expressive architectures. Recently, connections between GNNs and Weisfeiler–Leman type algorithms have been shown [17, 18]. Specifically, Morris et al. [17] and Xu et al. [18] showed that the 1-WL limits the expressive power of any possible GNN architecture in terms of distinguishing non-isomorphic graphs. In turn, these results have been generalized to the k-WL, see, e.g., Morris et al. [17], Azizian and Lelarge [19], Geerts et al. [20], Geerts [21], Maron et al. [22], Morris et al. [23, 24], and connected to permutation-equivariant function approximation over graphs, see, e.g., Chen et al. [25], Geerts and Reutter [26], Maehara and NT [27]. Barceló et al. [28] further established an equivalence between the expressiveness of GNNs with readout functions and C^2, the 2-variable fragment of first-order logic extended by counting quantifiers.

Relational GNNs. Relational GNNs enjoy a profound usage in many areas of machine learning, such as complex question answering in NLP [29] or visual question answering [30] in the intersection of NLP and vision. Notably, Schlichtkrull et al. [31] proposed the first architecture, namely, R-GCN, being able to handle multi-relational data. Further, Vashishth et al. [32] proposed an alternative GNN architecture, namely, CompGCN, using less number of parameters and reporting improved empirical performance. In the knowledge graph reasoning area, R-GCN and CompGCN, being strong baselines, spun off numerous improved GNNs for node classification and transductive link prediction tasks [35–37]. Furthermore, they inspired architectures for more complex reasoning tasks such as inductive link prediction [34, 38–40] and logical query answering [41–43].

Despite various applications, there has not been any theoretical work shedding light on multi-relational GNNs' expressive power and learning performance. Some recent empirical results highlight interesting properties of relational GNNs, e.g., a randomly initialized and untrained R-GCN still demonstrates non-trivial performance [60], or that random perturbation of the relations does not lead to performance drops for CompGCN [37].

A.2 The Weisfeiler–Leman algorithm

In the following, we briefly describe Weisfeiler-Leman-type algorithms, starting with the 1-*dimensional Weisfeiler-Leman algorithm* (1-WL).

The 1-WL. The 1-WL, or *color refinement*, is a simple heuristic for the graph isomorphism problem, originally proposed by Weisfeiler and Leman [45].[6] Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling vertices. Given an initial coloring or labeling of the vertices of both graphs, e.g., their degree or application-specific information, in each iteration, two vertices with the same label get different labels if the number of identically labeled neighbors is not equal. If, after some iteration, the number of vertices annotated with a specific label is different in both graphs, the algorithm terminates and a stable coloring (partition) is obtained. We can then conclude that the two graphs are not isomorphic. It is easy to see that the algorithm cannot distinguish all non-isomorphic graphs [47]. Nonetheless, it is a powerful heuristic that can successfully test isomorphism for a broad class of graphs [48–50].

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

[6] Strictly speaking, 1-WL and color refinement are two different algorithms. That is, 1-WL considers neighbors and non-neighbors to update the coloring, resulting in a slightly higher expressive power when distinguishing vertices in a given graph, see Grohe [46] for details. For brevity, we consider both algorithms to be equivalent.

$t > 0$, we set

$$C^{(t)}(v) \coloneqq \mathsf{RELABEL}\Big(\big(C^{(t-1)}(v), \{\!\{C^{(t-1)}(u) \mid u \in N(v)\}\!\}\big)\Big),$$

where RELABEL injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C^{(0)} \coloneqq \ell$. To test if two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of vertices colored c in $\mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. Moreover, if the number of colors between two iterations, t and $(t+1)$, does not change, i.e., the cardinalities of the images of $C^{(t)}$ and $C^{(t+1)}$ are equal, or, equivalently,

$$C^{(t)}(v) = C^{(t)}(w) \iff C^{(t+1)}(v) = C^{(t+1)}(w),$$

for all vertices v and w in $V(G)$, the algorithm terminates. For such t, we define the *stable coloring* $C^{\infty}(v) = C^{(t)}(v)$ for v in $V(G)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [51].

Due to the shortcomings of the 1-WL or color refinement in distinguishing non-isomorphic graphs, several researchers, e.g., [52, 53], devised a more powerful generalization of the former, today known as the k-dimensional Weisfeiler-Leman algorithm (k-WL).[7]

Oblivious k-WL. Intuitively, to surpass the limitations of the 1-WL, the k-WL colors ordered subgraphs instead of a single vertex. More precisely, given a graph G, it colors the tuples from $V(G)^k$ for $k \geq 2$ instead of the vertices. By defining a neighborhood between these tuples, we can define a coloring similar to the 1-WL. Formally, let G be a labeled graph, and let $k \geq 2$. In each iteration $t \geq 0$, the algorithm, similarly to the 1-WL, computes a *coloring* $C_k^{(t)} \colon V(G)^k \to \mathbb{N}$. In the first iteration, $t = 0$, the tuples $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ get the same color if they have the same atomic type, i.e., $C_k^{(0)}(\mathbf{v}) \coloneqq \mathsf{atp}(\mathbf{v})$. Now, for $t \geq 0$, $C_k^{(t+1)}$ is defined by

$$C_{(t+1)}^{k}(\mathbf{v}) \coloneqq \mathsf{RELABEL}\Big(\big(C_k^{(t)}(\mathbf{v}), M^{(t)}(\mathbf{v})\big)\Big),$$

with $M^{(t)}(\mathbf{v})$ the tuple

$$M^{(t)}(\mathbf{v}) \coloneqq \big(\{\!\{C_k^{(t)}(\theta_1(\mathbf{v}, w)) \mid w \in V(G)\}\!\}, \ldots, \{\!\{C_k^{(t)}(\theta_k(\mathbf{v}, w)) \mid w \in V(G)\}\!\}\big). \tag{8}$$

We also call $M^{(t)}$ an *aggregation function*. Here

$$\theta_j(\mathbf{v}, w) \coloneqq (v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k).$$

That is, $\theta_j(\mathbf{v}, w)$ replaces the j-th component of the tuple $\mathbf{v}$ with the vertex w. Hence, two tuples $\mathbf{v}$ and $\mathbf{w}$ with the same color in iteration t get different colors in iteration $(t+1)$ if there exists a $j \in [k]$ such that the number of j-neighbors of $\mathbf{v}$ and $\mathbf{w}$, respectively, colored with a certain color is different.

Hence, two tuples are *adjacent* or *j-neighbors* if they are different in the jth component (or equal, in the case of self-loops). Again, we run the algorithm until convergence, i.e.,

$$C_k^{(t)}(\mathbf{v}) = C_k^{(t)}(\mathbf{w}) \iff C_k^{(t+1)}(\mathbf{v}) = C_k^{(t+1)}(\mathbf{w}),$$

for all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)^k$ holds, and call the partition of $V(G)^k$ induced by $C_k^{(t)}$ the stable partition. For such t, we define $C_k^{\infty}(\mathbf{v}) \coloneqq C_k^{(t)}(\mathbf{v})$ for $\mathbf{v}$ in $V(G)^k$.

To test whether two graphs G and H are non-isomorphic, we run the k-WL in "parallel" on both graphs. Then, if the two graphs have a different number of vertices colored c in $\mathbb{N}$, the k-WL *distinguishes* the graphs as non-isomorphic. By increasing k, the algorithm becomes more powerful in distinguishing non-isomorphic graphs, i.e., for each $k \geq 1$, there are non-isomorphic graphs distinguished by $(k+1)$-WL but not by k-WL [47].

[7] There exists two definitions of the k-WL, the so-called oblivious k-WL and the folklore or non-oblivious k-WL, see Grohe [46]. There is a subtle difference in how they aggregate neighborhood information. Within the graph learning community, it is customary to abbreviate the oblivious k-WL as k-WL, a convention that we follow in this paper.

Local δ-k-dimensional Weisfeiler–Leman algorithm. Morris et al. [23] introduced a more efficient variant of the k-WL, the *local δ-k-dimensional Weisfeiler–Leman algorithm* (δ-k-LWL). In contrast to the k-WL, the δ-k-LWL considers only a subset of the entire neighborhood of a vertex tuple. Let the tuple $\mathbf{w} = \theta_j(\mathbf{v}, w)$ be a j-neighbor of $\mathbf{v}$. We say that $\mathbf{w}$ is a *local* j-neighbor of $\mathbf{v}$ if w is adjacent to the replaced vertex v_j. Otherwise, the tuple $\mathbf{w}$ is a *global* j-neighbor of $\mathbf{v}$. The δ-k-LWL considers only local neighbors during the neighborhood aggregation process, and discards any information about the global neighbors. Formally, the δ-k-LWL refines a coloring $C^{(t)}_{k,\delta}$ (obtained after t rounds of the δ-k-LWL) via the aggregation function

$$M^{(t)}_{\delta}(\mathbf{v}) \coloneqq \big(\{\!\{C^{(t)}_{k,\delta}(\theta_1(\mathbf{v}, w)) \mid w \in N(v_1)\}\!\}, \ldots, \{\!\{C^{(t)}_{k,\delta}(\theta_k(\mathbf{v}, w)) \mid w \in N(v_k)\}\!\}\big),$$

hence considering only the local j-neighbors of the tuple $\mathbf{v}$ in each iteration. The coloring functions for the iterations of the δ-k-LWL are then defined by

$$C^{(t+1)}_{k,\delta}(\mathbf{v}) \coloneqq \mathsf{RELABEL}\Big(\big(C^{(t)}_{k,\delta}(\mathbf{v}), M^{(t)}_{\delta}(\mathbf{v})\big)\Big).$$

Note that the 1-WL is equivalent to the δ-1-LWL. Morris et al. [23] showed that, for each k, the δ-k-LWL can distinguish graphs that the k-WL cannot and derived a variation of the former that is strictly more powerful than the k-WL.

B Missing proofs in Section 3

Theorem 10 (Theorem 1 in the main text)**.** *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$ the following hold:*

- *For all choices of initial vertex features consistent with ℓ, sequences $\mathbf{W}^{(t)}_{\textsf{R-GCN}}$ of R-GCN parameters, and nodes v and w in $V(G)$,*

$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \implies \boldsymbol{h}^{(t)}_{v,\textsf{R-GCN}} = \boldsymbol{h}^{(t)}_{w,\textsf{R-GCN}}.$$

- *For all choices of initial vertex features consistent with ℓ, sequences $\mathbf{W}^{(t)}_{\textsf{CompGCN}}$ of CompGCN parameters, composition functions ϕ, and nodes v and w in $V(G)$,*

$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \implies \boldsymbol{h}^{(t)}_{v,\textsf{CompGCN}} = \boldsymbol{h}^{(t)}_{w,\textsf{CompGCN}}.$$

Proof. We only prove it for CompGCN as the proof for R-GCN is analogous. Fix initial vertex features $(\boldsymbol{h}^{(0)}_v)_{v \in V(G)}$ for G consistent with ℓ, a sequence $\mathbf{W}^{(t)}_{\textsf{CompGCN}}$ of parameters, a composition function ϕ, and two nodes v and w in $V(G)$. We prove the result by induction on $t \geq 0$. For $t = 0$, the statement follows immediately from the fact the initial features $(\boldsymbol{h}^{(0)}_v)_{v \in V(G)}$ are consistent with ℓ. Assume now that $C^{(t)}_{\mathsf{R}}(v) = C^{(t)}_{\mathsf{R}}(w)$, for $t > 0$. Hence, by Equation (4), it must be the case that

- $C^{(t-1)}_{\mathsf{R}}(v) = C^{(t-1)}_{\mathsf{R}}(w)$, and
- $\{\!\{C^{(t-1)}_{\mathsf{R}}(u) \mid u \in N_i(v)\}\!\} = \{\!\{C^{(t-1)}_{\mathsf{R}}(u) \mid u \in N_i(w)\}\!\}$, for each $i \in [r]$.

Then, by induction hypothesis, it holds that:

- $\boldsymbol{h}^{(t-1)}_{v,\textsf{CompGCN}} = \boldsymbol{h}^{(t-1)}_{w,\textsf{CompGCN}}$, and
- $\{\!\{\boldsymbol{h}^{(t-1)}_{u,\textsf{CompGCN}} \mid u \in N_i(v)\}\!\} = \{\!\{\boldsymbol{h}^{(t-1)}_{u,\textsf{CompGCN}} \mid u \in N_i(w)\}\!\}$, for each i in $[r]$.

From these two we conclude by applying Equation (3) that $\boldsymbol{h}^{(t)}_{v,\textsf{CompGCN}} = \boldsymbol{h}^{(t)}_{w,\textsf{CompGCN}}$. This is because we have that $\boldsymbol{h}^{(t-1)}_{v,\textsf{CompGCN}} \boldsymbol{W}^{(t)}_0 = \boldsymbol{h}^{(t-1)}_{w,\textsf{CompGCN}} \boldsymbol{W}^{(t)}_0$ and

$$\sum_{u \in N_i(v)} \phi\big(\boldsymbol{h}^{(t-1)}_{u,\textsf{CompGCN}}, \boldsymbol{z}^{(t)}_i\big) \boldsymbol{W}^{(t)}_1 = \sum_{u \in N_i(w)} \phi\big(\boldsymbol{h}^{(t-1)}_{u,\textsf{CompGCN}}, \boldsymbol{z}^{(t)}_i\big) \boldsymbol{W}^{(t)}_1,$$

for each $i \in [r]$. □

Theorem 11 (Theorem 2 in the main text)**.** *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. For all $t \geq 0$:*

- *There exist initial vertex features and a sequence $\mathbf{W}^{(t)}_{\textsf{R-GCN}}$ of parameters such that for all v and w in $V(G)$,*
$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \iff \boldsymbol{h}^{(t)}_{v,\textsf{R-GCN}} = \boldsymbol{h}^{(t)}_{w,\textsf{R-GCN}}.$$
- *There exist initial vertex features, a sequence $\mathbf{W}^{(t)}_{\textsf{CompGCN}}$ of parameters and a composition function ϕ such that for all v and w in $V(G)$,*
$$C^{(t)}_{R}(v) = C^{(t)}_{R}(w) \iff \boldsymbol{h}^{(t)}_{v,\textsf{CompGCN}} = \boldsymbol{h}^{(t)}_{w,\textsf{CompGCN}}.$$

Proof. We focus on the case of CompGCN when the composition map ϕ is vector scaling, that is, $\phi(\boldsymbol{h}, \alpha) = \alpha\boldsymbol{h}$, for $\boldsymbol{h}$ in $\mathbb{R}^d$ and α in $\mathbb{R}$. As we explain later, this implies the cases of R-GCN, CompGCN with point-wise multiplication, and also CompGCN with circular-correlation.

The proof is a refinement of [17, Theorem 2] for multi-relational graphs. For a matrix $\boldsymbol{B}$, we denote by $\boldsymbol{B}_i$ its i-th row. Let $n = |V(G)|$ and without loss of generality assume $V(G) = [n]$. We represent vertex features for G as a matrix $\boldsymbol{F}$ in $\mathbb{R}^{n\times d}$, where $\boldsymbol{F}_v$ corresponds to the vertex feature of v. By slightly abusing notation, we view vertex features as a coloring for G. In particular, we denote by $\mathit{\Gamma}_G(\boldsymbol{F})$ the application of one step of the 1-RWL on G. That is, $\mathit{\Gamma}_G(\boldsymbol{F})$ is a coloring $C\colon V(G) \to \mathbb{N}$ such that for each v in $V(G)$,

$$C(v) \coloneqq \mathsf{RELABEL}\Big(\big(C_{\boldsymbol{F}}(v), \{\!\{(C_{\boldsymbol{F}}(u), i) \mid i \in [r], u \in N_i(v)\}\!\}\big)\Big),$$

where $C_{\boldsymbol{F}}$ is the coloring corresponding to the matrix $\boldsymbol{F}$. On the other hand, the update rule of CompGCN can be written as follows:

$$\boldsymbol{F}' = \sigma(\boldsymbol{F}\boldsymbol{W}_0 + \sum_{i\in[r]} \alpha_i \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{W}_1 + b\boldsymbol{J}),$$

where $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ are the parameter matrices, α_i are the scaling factors, $\boldsymbol{A}_i$ is the adjacency matrix for the relation $R_i(G)$, and $\boldsymbol{J}$ is the all-one matrix of appropriate dimensions, representing the biases. Here we choose σ to be the sign function sign and the bias b to be $b = -1$. Using the same argument as in [17, Corollary 16], we can replace σ by the ReLU function.

We need the following lemma shown in [17, Lemma 9].

Lemma 12 ([17])**.** *Let $\boldsymbol{B}$ in $\mathbb{N}^{s\times t}$ be a matrix such that all the rows are pairwise distinct. Then there is a matrix $\boldsymbol{X}$ in $\mathbb{R}^{t\times s}$ such that the matrix $\mathsf{sign}(\boldsymbol{B}\boldsymbol{X} - \boldsymbol{J})$ in $\{-1, 1\}^{s\times s}$ is non-singular.*

Following [17], we say that a matrix is *row-independent modulo equality* if the set of all rows appearing in the matrix is linearly independent. For two colorings C and C' of G, we write $C \equiv C'$ if the colorings define the same partition on $V(G)$. The key lemma of the proof is the following:

Lemma 13. *Let $\boldsymbol{F}$ in $\mathbb{R}^{n\times d}$ be row-independent modulo equality. Then there are matrices $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ in $\mathbb{R}^{d\times e}$ and scaling factors α_i in $\mathbb{R}$, for i in $[r]$, such that the matrix*

$$\boldsymbol{F}' = \mathsf{sign}(\boldsymbol{F}\boldsymbol{W}_0 + \sum_{i\in[r]} \alpha_i \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{W}_1 - \boldsymbol{J})$$

is row-independent modulo equality and $\boldsymbol{F}' \equiv \mathit{\Gamma}_G(\boldsymbol{F})$.

Proof. Let q be the number of distinct rows in $\boldsymbol{F}$ and let $\widetilde{\boldsymbol{F}}$ in $\mathbb{R}^{q\times d}$ be the matrix whose rows are the distinct rows of $\boldsymbol{F}$ in an arbitrary but fixed order. We denote by $Q_1, \ldots, Q_q$ the associated *color classes*, that is, a vertex v in $[n]$ is in Q_j if and only if $\boldsymbol{F}_v = \widetilde{\boldsymbol{F}}_j$. By construction, the rows of $\widetilde{\boldsymbol{F}}$ are linearly independent, and hence there is a matrix $\boldsymbol{M}$ in $\mathbb{R}^{d\times q}$ such that $\widetilde{\boldsymbol{F}}\boldsymbol{M}$ in $\mathbb{R}^{q\times q}$ is the identity matrix. It follows that the matrix $\boldsymbol{F}\boldsymbol{M}$ in $\mathbb{R}^{n\times q}$ has entries:

$$(\boldsymbol{F}\boldsymbol{M})_{vj} = \begin{cases} 1 & \text{if } v \in Q_j \\ 0 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{D}$ in $\mathbb{N}^{n\times q(r+1)}$ be the matrix with entries:

$$\boldsymbol{D}_{vh} = \begin{cases} |N_i(v) \cap Q_j| & \text{if } h = iq + j \text{ for } i \in [r],\, j \in [q] \\ 1 & \text{if } h \in [q] \text{ and } v \in Q_h \\ 0 & \text{otherwise.} \end{cases}$$

So the v-th row of $\boldsymbol{D}$ is the concatenation of a one-hot vector encoding of the color of v and a vector encoding for the multiset of the colors in $N_i(v)$, for each i in $[r]$. We have

$$\mathit{\Gamma}_G(\boldsymbol{F}) \equiv \boldsymbol{D}$$

if we view $\boldsymbol{D}$ as a coloring of G. We can also see $\boldsymbol{D}$ as a block matrix $\boldsymbol{D} = [\boldsymbol{N}_0\, \boldsymbol{N}_1 \cdots \boldsymbol{N}_r]$, where $\boldsymbol{N}_0 = \boldsymbol{F}\boldsymbol{M}$ in $\mathbb{N}^{n\times q}$ and $\boldsymbol{N}_i = \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M}$ in $\mathbb{N}^{n\times q}$ for each i in $[r]$. Since $0 \leq \boldsymbol{D}_{vh} \leq n-1$, for all v in $[n]$, h in $[q(r+1)]$, we have

$$\boldsymbol{D} \equiv \boldsymbol{E}$$

where

$$\boldsymbol{E} = \boldsymbol{F}\boldsymbol{M} + \sum_{i\in[r]} n^i \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M}.$$

Indeed, $\boldsymbol{E}_{vj}$ is simply the n-base representation of the vector $(\boldsymbol{D}_{vj}, \boldsymbol{D}_{v(qj)}, \ldots, \boldsymbol{D}_{v(rqj)})$, and hence $\boldsymbol{E}_v = \boldsymbol{E}_w$ if and only if $\boldsymbol{D}_v = \boldsymbol{D}_w$.

Let p be the number of distinct rows in $\boldsymbol{E}$ and let $\widetilde{\boldsymbol{E}}$ in $\mathbb{N}^{p\times q}$ be the matrix whose rows are the distinct rows of $\boldsymbol{E}$ in an arbitrary but fixed order. We can apply Lemma 12 to $\widetilde{\boldsymbol{E}}$ and obtain a matrix $\boldsymbol{X}$ in $\mathbb{R}^{q\times p}$ such that $\mathsf{sign}(\widetilde{\boldsymbol{E}}\boldsymbol{X} - \boldsymbol{J})$ in $\mathbb{R}^{p\times p}$ is non-singular. In particular, $\mathsf{sign}(\boldsymbol{E}\boldsymbol{X} - \boldsymbol{J})$ is row-independent modulo equality and $\mathsf{sign}(\boldsymbol{E}\boldsymbol{X} - \boldsymbol{J}) \equiv \boldsymbol{E} \equiv \mathit{\Gamma}_G(\boldsymbol{F})$. Let $\boldsymbol{W}_0 = \boldsymbol{W}_1 = \boldsymbol{M}\boldsymbol{X}$ in $\mathbb{R}^{d\times p}$ and $\alpha_i = n^i$ for i in $[r]$. We have

$$\begin{aligned} \boldsymbol{F}' &= \mathsf{sign}(\boldsymbol{F}\boldsymbol{W}_0 + \sum_{i\in[r]} \alpha_i \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{W}_1 - \boldsymbol{J}) \\ &= \mathsf{sign}(\boldsymbol{F}\boldsymbol{M}\boldsymbol{X} + \sum_{i\in[r]} \alpha_i \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M}\boldsymbol{X} - \boldsymbol{J}) \\ &= \mathsf{sign}(\boldsymbol{E}\boldsymbol{X} - \boldsymbol{J}). \end{aligned}$$

Hence $\boldsymbol{F}'$ is row-independent modulo equality and $\boldsymbol{F}' = \mathsf{sign}(\boldsymbol{E}\boldsymbol{X} - \boldsymbol{J}) \equiv \mathit{\Gamma}_G(\boldsymbol{F})$. □

Now the theorem follows directly from Lemma 13. We start with initial vertex features $(\boldsymbol{h}_v^{(0)})_{v\in V(G)}$ consistent with ℓ such that different features are linearly independent. Hence the matrix $\boldsymbol{F}^{(0)}$ representing the initial features is row-independent modulo equality and we can apply iteratively Lemma 13 to obtain the required sequence $\mathbf{W}^{(t)}_{\mathsf{CompGCN}}$ such that $C^{(t)}_{\mathsf{R}} \equiv \boldsymbol{F}^{(t)}$, where $\boldsymbol{F}^{(t)}$ is the matrix representing the vertex features $(\boldsymbol{h}^{(t)}_{v,\mathsf{CompGCN}})_{v\in V(G)}$. In particular, $C^{(t)}_{\mathsf{R}}(v) = C^{(t)}_{\mathsf{R}}(w) \Leftrightarrow \boldsymbol{h}^{(t)}_{v,\mathsf{CompGCN}} = \boldsymbol{h}^{(t)}_{w,\mathsf{CompGCN}}$, for all v and w in $V(G)$.

Remark 14. Note that the dimensions $d \times e$ of the parameter matrices at layer t correspond to the number of distinct colors before (q) and after (p) the application of the layer.

The case of CompGCN with point-wise multiplication holds since we can simulate vector scaling as $\alpha\boldsymbol{h} = \boldsymbol{h} * (\alpha, \ldots, \alpha)$, where $*$ denotes point-wise multiplication. Similarly, the case of R-GCN follows as we can simulate vector scaling by setting $\boldsymbol{W}_i = \alpha_i\boldsymbol{W}_1$, for each i in $[r]$.

Finally, we show that the result also holds for CompGCN with circular-correlation. This composition map is defined as follows[8]:

$$(\boldsymbol{h} \star \boldsymbol{z})_i = \sum_{j=1}^{d} \boldsymbol{h}_j \boldsymbol{z}_{((i+j-2) \bmod d)+1},$$

[8]For 0-indexed vectors, this is simply $(\boldsymbol{h} \star \boldsymbol{z})_i = \sum_{j=0}^{d-1} \boldsymbol{h}_j \boldsymbol{z}_{(i+j) \bmod d}$ for $0 \leq i \leq d-1$.

where $\boldsymbol{h}, \boldsymbol{z}$ in $\mathbb{R}^d$, $\boldsymbol{h} \star \boldsymbol{z}$ in $\mathbb{R}^d$ and i in $[d]$. We can easily simulate one layer of CompGCN with vector scaling using two layers of CompGCN with circular-correlation. Indeed, for a layer of the form

$$\boldsymbol{h}_v = \sigma\Big(\boldsymbol{g}_v \boldsymbol{W}_0 + \sum_{i \in [r]} \sum_{w \in N_i(v)} \alpha_i \boldsymbol{g}_w \boldsymbol{W}_1 + \boldsymbol{b}\Big),$$

where $\boldsymbol{g}_u$ in $\mathbb{R}^d$, for all u in $V(G)$, we first use a layer of the form

$$\widetilde{\boldsymbol{h}}_v = \boldsymbol{g}_v \boldsymbol{P},$$

where $\boldsymbol{P}$ in $\mathbb{R}^{d \times d}$ reverts the vertex features, that is, all the entries are zero except for $\boldsymbol{P}_{(n-i+1)i} = 1$ for all i in $[d]$, followed by a layer

$$\boldsymbol{h}_v = \sigma\Big(\widetilde{\boldsymbol{h}}_v \boldsymbol{P} \boldsymbol{W}_0 + \sum_{i \in [r]} \sum_{w \in N_i(v)} (\widetilde{\boldsymbol{h}}_v \star (0, \ldots, 0, \alpha_i)) \boldsymbol{W}_1 + \boldsymbol{b}\Big).$$

□

B.1 On the choice of the composition function for R-GCN architectures

Proposition 15 (Proposition 3 in the main text)**.** *There exist a labeled, multi-relational graph $G = (V(G), R_1(G), R_2(G), \ell)$ and two nodes v and w in $V(G)$, such that $C^{(1)}_{R}(v) \neq C^{(1)}_{R}(w)$ but $C^{\infty}_{WR}(v) = C^{\infty}_{WR}(w)$.*

Proof. We have $V(G) = \{v, w, u_1, u_2\}$, $R_1(G) = \{(v, u_1), (w, u_2)\}$, $R_2(G) = \{(v, u_2), (w, u_1)\}$, $\ell(v) = \ell(w) = 0$, $\ell(u_1) = 1$ and $\ell(u_2) = 2$. Hence,

$$C^{(1)}_{\mathsf{R}}(v) = \mathsf{RELABEL}\Big(\big(0, \{\!\{(1,1),(2,2)\}\!\}\big)\Big) \qquad C^{(1)}_{\mathsf{R}}(w) = \mathsf{RELABEL}\Big(\big(0, \{\!\{(2,1),(1,2)\}\!\}\big)\Big),$$

that is, $C^{(1)}_{\mathsf{R}}(v) \neq C^{(1)}_{\mathsf{R}}(w)$. On the other hand,

$$C^{(1)}_{\mathsf{WR}}(v) = \mathsf{RELABEL}\Big(\big(0, \{\!\{1,2\}\!\}, 1, 1\big)\Big) \qquad C^{(1)}_{\mathsf{WR}}(w) = \mathsf{RELABEL}\Big(\big(0, \{\!\{1,2\}\!\}, 1, 1\big)\Big)$$

and then $C^{\infty}_{\mathsf{WR}}(v) = C^{\infty}_{\mathsf{WR}}(w)$. □

As shown next, the expressive power of CompGCN architectures that use point-wise summation/substraction or vector concatenation is captured by this weaker form of multi-relational 1-WL.

Theorem 16 (Theorem 4 in the main text)**.** *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then:*

- *For all $t \geq 0$, choices of initial vertex features consistent with ℓ, sequence $\mathbf{W}^{(t)}_{CompGCN}$ of CompGCN parameters, and nodes v, w in $V(G)$,*

$$C^{(t)}_{WR}(v) = C^{(t)}_{WR}(w) \implies \boldsymbol{h}^{(t)}_{v,CompGCN} = \boldsymbol{h}^{(t)}_{w,CompGCN},$$

  for either point-wise summation/substraction or concatenation as the composition map.

- *For all $t \geq 0$, there exist initial vertex features and a sequence $\mathbf{W}^{(t)}_{CompGCN}$ of CompGCN parameters, such that for all nodes v, w in $V(G)$,*

$$C^{(t)}_{WR}(v) = C^{(t)}_{WR}(w) \iff \boldsymbol{h}^{(t)}_{v,CompGCN} = \boldsymbol{h}^{(t)}_{w,CompGCN},$$

  for either point-wise summation/substraction or concatenation as the composition map.

Proof. We start with the first item. We focus first on the case of CompGCN with vector concatenation. Note that if $\boldsymbol{h}$ in $\mathbb{R}^d$, $\boldsymbol{z}$ in $\mathbb{R}^b$ and $\boldsymbol{W}$ in $\mathbb{R}^{(d+b) \times e}$, then we have

$$(\boldsymbol{h}, \boldsymbol{z})\boldsymbol{W} = \boldsymbol{h}\boldsymbol{X} + \boldsymbol{z}\boldsymbol{Y},$$

where $\boldsymbol{X}$ in $\mathbb{R}^{d\times e}$ is the matrix given by the first d rows of $\boldsymbol{W}$, while $\boldsymbol{Y}$ in $\mathbb{R}^{b\times e}$ is the matrix given by the last b rows of $\boldsymbol{W}$. In particular, we can write

$$\begin{aligned}\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t)} &= \sigma\Big(\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)}\boldsymbol{W}_0^{(t)} + \sum_{i\in[r]}\sum_{u\in N_i(v)}(\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)}, \boldsymbol{z}_i^{(t)})\boldsymbol{W}_1^{(t)}\Big)\\ &= \sigma\Big(\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)}\boldsymbol{W}_0^{(t)} + \sum_{i\in[r]}\sum_{u\in N_i(v)}\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)}\boldsymbol{X}_1^{(t)} + \boldsymbol{z}_i^{(t)}\boldsymbol{Y}_1^{(t)}\Big)\\ &= \sigma\Big(\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)}\boldsymbol{W}_0^{(t)} + \sum_{i\in[r]}\sum_{u\in N_i(v)}\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)}\boldsymbol{X}_1^{(t)} + \sum_{i\in[r]}|N_i(v)|\boldsymbol{z}_i^{(t)}\boldsymbol{Y}_1^{(t)}\Big).\end{aligned}$$

Fix initial vertex features $(\boldsymbol{h}_v^{(0)})_{v\in V(G)}$ for G consistent with ℓ, a sequence $\mathbf{W}_{\mathsf{CompGCN}}^{(t)}$ of parameters and two nodes v and w in $V(G)$. We proceed by induction on $t \geq 0$. For $t = 0$ we are done as the features $(\boldsymbol{h}_v^{(0)})_{v\in V(G)}$ are consistent with ℓ. Assume now that $C_{\mathsf{WR}}^{(t)}(v) = C_{\mathsf{WR}}^{(t)}(w)$, for $t > 0$. Then, by Section 3, we have that

- $C_{\mathsf{WR}}^{(t-1)}(v) = C_{\mathsf{WR}}^{(t-1)}(w)$,
- $\{\!\{C_{\mathsf{WR}}^{(t-1)}(u) \mid i \in [r], u \in N_i(v)\}\!\} = \{\!\{C_{\mathsf{WR}}^{(t-1)}(u) \mid i \in [r], u \in N_i(w)\}\!\}$,
- $|N_i(v)| = |N_i(w)|$ for each $i \in [r]$.

Then, by induction hypothesis, it holds that:

- $\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)} = \boldsymbol{h}_{w,\mathsf{CompGCN}}^{(t-1)}$, and
- $\{\!\{\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)} \mid i \in [r], u \in N_i(v)\}\!\} = \{\!\{\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)} \mid i \in [r], u \in N_i(w)\}\!\}$.

Then we have

- $\sum_{i\in[r]}|N_i(v)|\boldsymbol{z}_i^{(t)} = \sum_{i\in[r]}|N_i(w)|\boldsymbol{z}_i^{(t)}$, and
- $\sum_{i\in[r]}\sum_{u\in N_i(v)}\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)} = \sum_{i\in[r]}\sum_{u\in N_i(w)}\boldsymbol{h}_{u,\mathsf{CompGCN}}^{(t-1)}$.

We conclude that $\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t)} = \boldsymbol{h}_{w,\mathsf{CompGCN}}^{(t)}$.

Note that the update rule for the case of point-wise summation/substraction is the same except that now $\boldsymbol{X}_1^{(t)} = \boldsymbol{Y}_1^{(t)}$. Hence exactly the same argument applies.

We now turn to the second item. We follow the same strategy and terminology as in the proof of Theorem 11. In this case, given a vertex feature matrix $\boldsymbol{F}$ in $\mathbb{R}^{n\times d}$, we denote by $\hat{I}_G(\boldsymbol{F})$ the application of one step of the weak 1-RWL. Hence, $\hat{I}_G(\boldsymbol{F})$ is a coloring $C\colon V(G) \to \mathbb{N}$ such that for each v in $V(G)$,

$$C(v) = \mathsf{RELABEL}\Big(\big(C_{\boldsymbol{F}}(v), \{\!\{C_{\boldsymbol{F}}(u) \mid i \in [r], u \in N_i(v)\}\!\}, |N_1(v)|, \ldots, |N_r(v)|\big)\Big),$$

where $C_{\boldsymbol{F}}$ is the coloring corresponding to the matrix $\boldsymbol{F}$. In this case, the update rule for $\mathsf{CompGCN}$ with vector concatenation can be written as follows:

$$\boldsymbol{F}' = \sigma(\boldsymbol{F}\boldsymbol{W}_0 + \sum_{i\in[r]}\boldsymbol{A}_i\boldsymbol{F}\boldsymbol{X}_1 + \sum_{i\in[r]}\boldsymbol{A}_i\boldsymbol{Z}_i\boldsymbol{Y}_1 + b\boldsymbol{J}),$$

where $\boldsymbol{W}_0$ in $\mathbb{R}^{d\times e}$ and $\boldsymbol{W}_1 = \begin{bmatrix}\boldsymbol{X}_1\\ \boldsymbol{Y}_1\end{bmatrix} \in \mathbb{R}^{(d+b)\times e}$, for $\boldsymbol{X}_1 \in \mathbb{R}^{d\times e}$, $\boldsymbol{Y}_1 \in \mathbb{R}^{b\times e}$, are the parameter matrices, $\boldsymbol{Z}_i \in \mathbb{R}^{n\times b}$ is the matrix where each row is a copy of the edge feature $\boldsymbol{z}_i \in \mathbb{R}^b$ associated with the relation $R_i(G)$, $\boldsymbol{A}_i$ is the adjacency matrix for the relation $R_i(G)$, and $\boldsymbol{J}$ is the all-one matrix of appropriate dimensions. We have the following:

Lemma 17. *Let $\boldsymbol{F}$ in $\mathbb{R}^{n\times d}$ be row-independent modulo equality. Then there are matrices $\boldsymbol{W}_0$ in $\mathbb{R}^{d\times e}$, $\boldsymbol{X}_1 \in \mathbb{R}^{d\times e}$, $\boldsymbol{Y}_1 \in \mathbb{R}^{b\times e}$ and vectors $\boldsymbol{z}_i \in \mathbb{R}^b$, for i in $[r]$ such that the matrix*

$$\boldsymbol{F}' = \mathsf{sign}(\boldsymbol{F}\boldsymbol{W}_0 + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{X}_1 + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{Z}_i\boldsymbol{Y}_1 - \boldsymbol{J})$$

is row-independent modulo equality and $\boldsymbol{F}' \equiv \hat{\Gamma}_G(\boldsymbol{F})$.

Proof. Let q be the number of distinct rows in $\boldsymbol{F}$ and let $\widetilde{\boldsymbol{F}}$ in $\mathbb{R}^{q\times d}$ be the matrix whose rows are the distinct rows of $\boldsymbol{F}$ in an arbitrary but fixed order. We denote by $Q_1, \dots, Q_q$ the associated *color classes*, that is, a vertex v in $[n]$ is in Q_j if and only if $\boldsymbol{F}_v = \widetilde{\boldsymbol{F}}_j$. By construction, the rows of $\widetilde{\boldsymbol{F}}$ are linearly independent, and hence there is a matrix $\boldsymbol{M}$ in $\mathbb{R}^{d\times q}$ such that $\widetilde{\boldsymbol{F}}\boldsymbol{M}$ in $\mathbb{R}^{q\times q}$ is the identity matrix. It follows that the matrix $\boldsymbol{F}\boldsymbol{M}$ in $\mathbb{R}^{n\times q}$ has entries:

$$(\boldsymbol{F}\boldsymbol{M})_{vj} = \begin{cases} 1 & \text{if } v \in Q_j \\ 0 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{M}_0, \boldsymbol{M}_1$ in $\mathbb{N}^{d\times(2q+r)}$, $\boldsymbol{M}_2$ in $\mathbb{N}^{r\times(2q+r)}$ be the block matrices $\boldsymbol{M}_0 = [\boldsymbol{M}\,\boldsymbol{O}\,\boldsymbol{O}']$, $\boldsymbol{M}_1 = [\boldsymbol{O}\,\boldsymbol{M}\,\boldsymbol{O}']$ and $\boldsymbol{M}_2 = [\boldsymbol{O}''\,\boldsymbol{O}''\,\boldsymbol{I}]$, where $\boldsymbol{O}$ in $\mathbb{R}^{d\times q}$, $\boldsymbol{O}'$ in $\mathbb{R}^{d\times r}$, $\boldsymbol{O}''$ in $\mathbb{R}^{r\times q}$ are all-0 matrices, and $\boldsymbol{I}$ in $\mathbb{R}^{r\times r}$ is the identity matrix. For each i in $[r]$, the required $\boldsymbol{z}_i$ in $\mathbb{R}^r$ is the vector with all entries 0 except for the i-th position which is 1. Let $\boldsymbol{Z}_i$ be the corresponding matrix whose rows are copies of $\boldsymbol{z}_i$. We define $\boldsymbol{D}$ in $\mathbb{N}^{n\times(2q+r)}$ as:

$$\begin{aligned}\boldsymbol{D} &= \boldsymbol{F}\boldsymbol{M}_0 + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M}_1 + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{Z}_i\boldsymbol{M}_2 \\ &= \begin{bmatrix}\boldsymbol{F}\boldsymbol{M} & \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M} & \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{Z}_i\end{bmatrix}.\end{aligned}$$

The v-th row of $\boldsymbol{F}\boldsymbol{M}$ encodes the color of v, the v-th row of $\sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M}$ encodes the multiset of the colors of u, when we range over i in $[r]$ and u in $N_i(v)$, and the v-th row of $\sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{Z}_i$ contains the sizes of $N_i(v)$ for all i in $[r]$. Hence,

$$\hat{\Gamma}_G(\boldsymbol{F}) \equiv \boldsymbol{D}$$

if we view $\boldsymbol{D}$ as a coloring of G.

Let p be the number of distinct rows in $\boldsymbol{D}$ and let $\widetilde{\boldsymbol{D}}$ in $\mathbb{N}^{p\times(2q+r)}$ be the matrix whose rows are the distinct rows of $\boldsymbol{D}$ in an arbitrary but fixed order. We apply Lemma 12 to $\widetilde{\boldsymbol{D}}$ and obtain a matrix $\boldsymbol{X}$ in $\mathbb{R}^{(2q+r)\times p}$ such that $\mathsf{sign}(\widetilde{\boldsymbol{D}}\boldsymbol{X} - \boldsymbol{J})$ in $\mathbb{R}^{p\times p}$ is non-singular. In particular, $\mathsf{sign}(\boldsymbol{D}\boldsymbol{X} - \boldsymbol{J})$ is row-independent modulo equality and $\mathsf{sign}(\boldsymbol{D}\boldsymbol{X} - \boldsymbol{J}) \equiv \boldsymbol{D} \equiv \hat{\Gamma}_G(\boldsymbol{F})$. Let $\boldsymbol{W}_0 = \boldsymbol{M}_0\boldsymbol{X}$ in $\mathbb{R}^{d\times p}$, $\boldsymbol{X}_1 = \boldsymbol{M}_1\boldsymbol{X}$ in $\mathbb{R}^{d\times p}$, and $\boldsymbol{Y}_1 = \boldsymbol{M}_2\boldsymbol{X}$ in $\mathbb{R}^{r\times p}$. We have

$$\begin{aligned}\boldsymbol{F}' &= \mathsf{sign}(\boldsymbol{F}\boldsymbol{W}_0 + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{X}_1 + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{Z}_i\boldsymbol{Y}_1 - \boldsymbol{J}) \\ &= \mathsf{sign}(\boldsymbol{F}\boldsymbol{M}_0\boldsymbol{X} + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{F}\boldsymbol{M}_1\boldsymbol{X} + \sum_{i\in[r]} \boldsymbol{A}_i\boldsymbol{Z}_i\boldsymbol{M}_2\boldsymbol{X} - \boldsymbol{J}) \\ &= \mathsf{sign}(\boldsymbol{D}\boldsymbol{X} - \boldsymbol{J}).\end{aligned}$$

Hence $\boldsymbol{F}'$ is row-independent modulo equality and $\boldsymbol{F}' = \mathsf{sign}(\boldsymbol{D}\boldsymbol{X} - \boldsymbol{J}) \equiv \hat{\Gamma}_G(\boldsymbol{F})$. □

The theorem follows directly by iteratively applying Lemma 17 starting with vertex features $(\boldsymbol{h}_v^{(0)})_{v\in V(G)}$ consistent with ℓ such that different features are linearly independent.

The case of CompGCN with point-wise summation/substraction follows from the fact that this architecture can simulate CompGCN with vector concatenation. Indeed, we can simulate one layer of CompGCN with vector concatenation using two layers of CompGCN with point-wise summation/substraction. Take a layer of the form

$$\boldsymbol{h}_v = \sigma\Big(\boldsymbol{g}_v\boldsymbol{W}_0 + \sum_{i\in[r]} \sum_{w\in N_i(v)} (\boldsymbol{g}_w, \boldsymbol{z}_i)\boldsymbol{W}_1 + \boldsymbol{b}\Big),$$

where $\boldsymbol{g}_u$ in $\mathbb{R}^d$, for u in $V(G)$, $\boldsymbol{W}_0 \in \mathbb{R}^{d\times e}$, $\boldsymbol{W}_1 \in \mathbb{R}^{(d+b)\times e}$ and $\boldsymbol{z}_i \in \mathbb{R}^b$. We first use a layer

$$\widetilde{\boldsymbol{h}}_v = \boldsymbol{g}_v \boldsymbol{B},$$

where $\boldsymbol{B} \in \mathbb{R}^{d\times(d+b)}$ is the $d \times d$ identity matrix with b additional all-0 columns. So $\widetilde{\boldsymbol{h}}_v = (\boldsymbol{g}_v, 0, \ldots, 0) \in \mathbb{R}^{d+b}$. Then we apply a layer

$$\boldsymbol{h}_v = \sigma\Big(\widetilde{\boldsymbol{h}}_v \boldsymbol{W}_0' + \sum_{i\in[r]} \sum_{w\in N_i(v)} (\widetilde{\boldsymbol{h}}_v + \boldsymbol{z}_i')\boldsymbol{W}_1 + \boldsymbol{b}\Big),$$

where $\boldsymbol{W}_0' \in \mathbb{R}^{(d+b)\times e}$ is the matrix $\boldsymbol{W}_0 \in \mathbb{R}^{d\times e}$ with b additional all-0 rows, while $\boldsymbol{z}_i' = (0, \ldots, 0, \boldsymbol{z}_i) \in \mathbb{R}^{d+b}$. □

Together with Proposition 15 and Theorem 11, this result states that CompGCN architectures based on vector summation or concatenation are provably weaker in terms of their capacity to distinguish nodes in graphs than the ones that use vector scaling.

B.2 A comparison between R-GCN and CompGCN architectures

We proved that R-GCN and CompGCN with point-wise multiplication have the same power discriminating nodes in (multi-relational) graphs. Here we show that these architectures actually define the same functions on multi-relational graphs.

Theorem 18. *The following statements hold:*

- *For any sequence of parameters $\mathbf{W}^{(t)}_{\textsf{CompGCN}}$ for CompGCN with point-wise multiplication, there is a sequence of parameters $\mathbf{W}^{(t)}_{\textsf{R-GCN}}$ for R-GCN such that for each labeled, multi-relational graph $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ and choice of initial vertex features, we have $\boldsymbol{h}^{(t)}_{v,\textsf{R-GCN}} = \boldsymbol{h}^{(t)}_{v,\textsf{CompGCN}}$, for each $v \in V(G)$.*
- *Conversely, for any sequence of parameters $\mathbf{W}^{(t)}_{\textsf{R-GCN}}$ for R-GCN, there exists a sequence of parameters $\mathbf{W}^{(2t)}_{\textsf{R-GCN}}$ for CompGCN with point-wise multiplication such that for each labeled, multi-relational graph $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ and choice of initial vertex features, we have $\boldsymbol{h}^{(2t)}_{v,\textsf{CompGCN}} = \boldsymbol{h}^{(t)}_{v,\textsf{R-GCN}}$, for each $v \in V(G)$.*

Proof. The first item follows since we can simulate one layer of CompGCN with point-wise multiplication using one layer of R-GCN. Indeed, take a layer of the form

$$\boldsymbol{h}_v = \sigma\Big(\boldsymbol{g}_v \boldsymbol{W}_0 + \sum_{i\in[r]} \sum_{w\in N_i(v)} (\boldsymbol{g}_w * \boldsymbol{z}_i)\boldsymbol{W}_1\Big),$$

where $\boldsymbol{g}_u, \boldsymbol{z}_i \in \mathbb{R}^d$. This is equivalent to

$$\boldsymbol{h}_v = \sigma\Big(\boldsymbol{g}_v \boldsymbol{W}_0 + \sum_{i\in[r]} \sum_{w\in N_i(v)} \boldsymbol{g}_w \boldsymbol{W}_i\Big),$$

where $\boldsymbol{W}_i = \boldsymbol{\Lambda}_i \boldsymbol{W}_1$, where $\boldsymbol{\Lambda}_i \in \mathbb{R}^{d\times d}$ is the diagonal matrix whose diagonal is precisely $\boldsymbol{z}_i$.

For the second item, we can simulate one layer of R-GCN with two layers of CompGCN with point-wise multiplication. Take a layer

$$\boldsymbol{h}_v = \sigma\Big(\boldsymbol{g}_v \boldsymbol{W}_0 + \sum_{i\in[r]} \sum_{w\in N_i(v)} \boldsymbol{g}_w \boldsymbol{W}_i\Big),$$

where $\boldsymbol{g}_u \in \mathbb{R}^d$, $\boldsymbol{W}_0 \in \mathbb{R}^{d\times e}$, $\boldsymbol{W}_i \in \mathbb{R}^{d\times e}$. We first apply a layer

$$\widetilde{\boldsymbol{h}}_v = \boldsymbol{g}_v \boldsymbol{B}$$

where $\boldsymbol{B} \in \mathbb{R}^{d\times dr}$ is the concatenation of r copies of the $d \times d$ identity matrix. In particular, $\widetilde{\boldsymbol{h}}_v \in \mathbb{R}^{dr}$ is the vector $\boldsymbol{g}_v$ repeated r times. Then we use the layer

$$\boldsymbol{h}_v = \sigma\Big(\widetilde{\boldsymbol{h}}_v \boldsymbol{W}_0' + \sum_{i\in[r]} \sum_{w\in N_i(v)} (\widetilde{\boldsymbol{h}}_w * \boldsymbol{z}_i)\boldsymbol{W}_1'\Big),$$

where $\boldsymbol{W}'_0 \in \mathbb{R}^{dr \times e}$ is the matrix $\boldsymbol{W}_0 \in \mathbb{R}^{d \times e}$ with $d(r-1)$ additional all-0 rows, $\boldsymbol{W}'_1 \in \mathbb{R}^{dr \times e}$ is the (vertical) concatenation of the matrices $\boldsymbol{W}_i$ for $i \in [r]$, and $\boldsymbol{z}_i \in \mathbb{R}^{dr}$ is the vector with all entries 0 except for the d positions $(i-1)d+1, \ldots, (i-1)d+d$ which contain the value 1. □

Remark 19. A similar result holds for the case of CompGCN with point-wise summation/subtraction and CompGCN with vector concatenation. The simulations between these two architectures are implicitly given in the proof of Theorem 16.

Remark 20. Note that, as a consequence of Theorem 11, Proposition 15 and the first item of Theorem 16, there are functions defined by R-GCN or CompGCN with point-wise multiplication that cannot be expressed by CompGCN with point-wise summation/subtraction or vector concatenation. This even holds in the non-uniform sense, that is, if we focus on a single labeled multi-relational graph (the one from Proposition 15).

C Missing proofs in Section 4

Proposition 21 (Proposition 5 in the main text)**.** *For all $r \geq 1$, there exists a pair of non-isomorphic graphs $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ and $H = (V(H), R_1(H), \ldots, R_r(H), \ell)$ that cannot be distinguished by R-GCN or CompGCN.*

Proof. We explicitly construct the graphs G and H for $r \geq 2$. To do so, we take a pair of graphs A and B, non-distinguishable by 1-WL, and transform them into the multi-relational graphs G and H. Let A be a cycle on six vertices and B be the disjoint union of two three cycles. Clearly, the 1-WL cannot distinguish the two graphs. Now let $V(G) \coloneqq V(A)$ and $V(H) \coloneqq V(B)$. Further, let $R_i(G) \coloneqq E(A)$ and $R_i(H) \coloneqq E(B)$ for i in $[r]$. Observe that the multi-relational 1-WL will reach the stable coloring after one iteration and it will not distinguish the multi-relational graphs G and H. Hence, by Theorem 10, the result follows. □

Proposition 22 (Proposition 6 in the main text)**.** *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$, $r > 0$, $k \geq 1$, and all choices of $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, and all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)$,*

$$C^{(t)}_{k,r}(\mathbf{v}) = C^{(t)}_{k,r}(\mathbf{w}) \implies \boldsymbol{h}^{(t)}_{\mathbf{v},k} = \boldsymbol{h}^{(t)}_{\mathbf{w},k}.$$

Proof sketch. The proof is analogous to the proof of Morris et al. [17, Proposition 3]. □

Proposition 23 (Proposition 7 in the main text)**.** *Let $G = (V(G), R_1(G), \ldots, R_r(G), \ell)$ be a labeled, multi-relational graph. Then for all $t \geq 0$ and $k \geq 1$, there exists $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, such that for all $\mathbf{v}$ and $\mathbf{w}$ in $V(G)$,*

$$C^{(t)}_{k,r}(\mathbf{v}) = C^{(t)}_{k,r}(\mathbf{w}) \iff \boldsymbol{h}^{(t)}_{\mathbf{v},k} = \boldsymbol{h}^{(t)}_{\mathbf{w},k}.$$

Proof. To prove the results, we need to ensure that there exists instantiations of $\mathsf{UPD}^{(t)}$ and $\mathsf{AGG}^{(t)}$ that are injective. To show the existence of injective instantiations of $\mathsf{AGG}^{(t)}$ for $t > 0$, we write $\mathsf{AGG}^{(t)}$ as

$$\mathsf{AGG}^{(t)}_{\text{out}}\Big(\mathsf{AGG}^{(t)}_{\text{in},1}\big(\{\!\!\{\phi(\boldsymbol{h}^{(t-1)}_{\theta_1(\mathbf{v},w),k}, \boldsymbol{z}^{(t)}_i) \mid w \in N_i(v_1) \text{ and } i \in [r]\}\!\!\}\big), \ldots,$$
$$\mathsf{AGG}^{(t)}_{\text{in},k}\big(\{\!\!\{\phi(\boldsymbol{h}^{(t-1)}_{\theta_k(\mathbf{v},w),k}, \boldsymbol{z}^{(t)}_i) \mid w \in N_i(v_k) \text{ and } i \in [r]\}\!\!\}\big)\Big),$$

where $\mathsf{AGG}^{(t)}_{\text{out}}$ and $\mathsf{AGG}^{(t)}_{\text{in},j}$ for j in $[k]$ may be a differentiable parameterized functions, e.g., neural networks. Observe that we can represent $\mathsf{AGG}^{(t)}_{\text{in},j}$ as

$$\sum_{i \in [r]} \sum_{w \in N_i(v_j)} \phi\big(\boldsymbol{h}^{(t-1)}_{\theta_j(\mathbf{v},w)}, \boldsymbol{z}^{(t)}_i\big) \cdot \boldsymbol{W}^{(t)}_1,$$

for j in $[k]$, resembling the aggregation of Equation (3), by Theorem 11, the injectiveness of the above aggregation function follows. A similar argument can be made for $\mathsf{AGG}^{(t)}_{\text{out}}$ and $\mathsf{UPD}^{(t)}$, implying the result. □

Moreover, the following result implies that increasing k leads to a strict boost in terms of expressivity of the k-RLWL and k-RNs architectures in terms of distinguishing non-isomorphic multi-relational graphs.

Proposition 24. *For $k \geq 2$ and $r \geq 1$, there exists a pair of non-isomorphic multi-relational graphs $G_r = (V(G_r), R_1(G_r), \ldots, R_r(G_r), \ell)$ and $H_r = (V(H_r), R_1(H_r), \ldots, R_r(H_r), \ell)$ that can be distinguished by the $(k+1)$-MLWL but not by the k-MLWL.*

Proof. See Appendix C.1. □

Corollary 25 (Proposition 8 in the main text)**.** *For $k \geq 2$ and $r \geq 1$, there exists a pair of non-isomorphic multi-relational graphs $G_r = (V(G_r), R_1(G_r), \ldots, R_r(G_r), \ell)$ and $H = (V(H_r), R_1(H_r), \ldots, R_r(H_r), \ell)$ such that:*

- *For all choices of $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, for $t > 0$, and $\mathsf{READOUT}$ the k-RN architecture will not distinguish the graphs G_r and H_r.*
- *There exists $\mathsf{UPD}^{(t)}$, $\mathsf{AGG}^{(t)}$, for $t > 0$, and $\mathsf{READOUT}$ such that the $(k+1)$-RN will distinguish them.*

Proof. Follows from Proposition 23 and Proposition 24. □

Corollary 26. There exists a 2-RN architecture that is strictly more expressive than the CompGCN and the R-GCN architecture in terms of distinguishing non-isomorphic graphs.

Proof. This follows from Corollary 25 and the fact that a 2-RN is capable to distinguish the graphs constructed in the proof of Proposition 21, which follows from the fact that the δ-2-LWL can distinguish the graphs A and B; see, e.g., the proof of Lemma 13 in [44]. □

k-RNs for node-level prediction. As defined in Equations (6) and (7), an k-RN architecture either computes k-tuple- or graph-level features. However, it is straightforward to compute a vertex-level features, see, e.g., Morris et al. [44, Section 4.1].

Scalability. Although the k-RN is provably expressive, see Proposition 8, it suffer some high memory requirement. Similar to the k-WL, it's memory complexity can only be lower bounded in $\Omega(n^k)$, making it not applicable for large knowledge graphs. However, recent progress in making higher-order architectures more scalable, e.g., [44, 61, 62], can be straightforwardly lifted to the multi-relational case.

C.1 Proof of Proposition 24

In the following, we outline the proof of Proposition 24. We modify the construction employed in [23], Appendix C.1.1., where they provide an infinite family of graphs $(G_k, H_k)_{k \in \mathbb{N}}$ such that the k-WL does not distinguish G_k and H_k, although the δ-k-LWL distinguishes G_k and H_k. We recall some relevant definitions from their paper.

Construction of G_k and H_k. Let K denote the complete graph on $k+1$ vertices (without any self-loops). The vertices of K are indexed from 0 to k. Let $E(v)$ denote the set of edges incident to v in K: clearly, $|E(v)| = k$ for all v in $V(K)$. We call the elements of $V(K)$ *base vertices*, and the elements of $E(K)$ *base edges*. Define the graph G_k as follows:

1. For the vertex set $V(G_k)$, we add
   (a) (v, S) for each v in $V(K)$ and for each *even* subset S of $E(v)$,
   (b) two vertices e^1, e^0 for each edge e in $E(K)$.
2. For the edge set $E(G_k)$, we add
   (a) an edge $\{e^0, e^1\}$ for each e in $E(K)$,
   (b) an edge between (v, S) and e^1 if v in e and e in S,
   (c) an edge between (v, S) and e^0 if v in e and e not in S,

Define a companion graph H_k, in a similar manner to G_k, with the following exception: in Step 1(a), for the vertex 0 in $V(K)$, we choose all *odd* subsets of $E(0)$.

Distance-two-cliques. A set S of vertices is said to form a *distance-two-clique* if the distance between any two vertices in S is exactly 2. The following results were shown in [23].

Lemma 27 ([23])**.** The following holds for the graphs G_k and H_k defined above.

- There exists a distance-two-clique of size $(k+1)$ inside G_k.
- There does not exist a distance-two-clique of size $(k+1)$ inside H_k.

Hence, G_k and H_k are non-isomorphic.

Lemma 28 ([23])**.** The δ-k-LWL distinguishes G_k and H_k, while the k-WL does not distinguish G_k and H_k.

Moreover, we need the following result showing that the δ-k-LWL forms a hierarchy.

Lemma 29. For $k \geq 2$, the δ-k-LWL distinguishes G_k and H_k, while the δ-$(k-1)$-LWL does not distinguish G_k and H_k.

Proof. The fact that δ-k-LWL distinguishes the graphs G_k and H_k follows from Lemma 28. We know argue that the δ-$(k-1)$-LWL does not distinguish the two graphs. First, the (oblivious) k-WL has the same expressive power in distinguishing non-isomorphic graphs as the non-oblivious or folklore $(k-1)$-WL; see [46] for details. The non-oblivious $(k-1)$-WL [46] uses the following aggregation function

$$M^{(t)}((v_1,\ldots,v_{k-1})) \coloneqq \{\!\{(C_k^{(t)}(\theta_1(\mathbf{v},w)),\ldots,C_k^{(t)}(\theta_{k-1}(\mathbf{v},w))) \mid w \in V(G)\}\!\},$$

instead of Equation (8). Notice that from $(C_k^{(t)}(\theta_1(\mathbf{v},w)),\ldots,C_k^{(t)}(\theta_{k-1}(\mathbf{v},w)))$ we can recover if there is an edge between the vertex w and a vertex v_j for j in $[k-1]$ in the underlying graph. Hence, the non-oblivious $(k-1)$-WL is at least as powerful as the δ-$(k-1)$-LWL, implying that the δ-$(k-1)$-LWL is weaker than the δ-k-LWL. □

We now construct non-isomorphic multi-relational graphs $G_r = (V(G_r), R_1(G_r),\ldots,R_r(G_r),\ell)$ and $H_r = (V(H_r), R_1(H_r),\ldots,R_r(H_r),\ell)$ that can be distinguished by the $(k+1)$-RLWL but not by the k-RLWL.

Let $V(G_r) \coloneqq V(G_k)$ and $V(H_r) \coloneqq V(H_k)$. Further, let $R_i(G_r) \coloneqq E(G_k)$ and $R_i(H_r) \coloneqq E(H_k)$ for i in $[r]$. By a straightforward inductive argument it follows that $M_\delta^{(t)}(\mathbf{v}) = M_\delta^{(t)}(\mathbf{w})$ implies $M_r^{(t)}(\mathbf{v}) = M_r^{(t)}(\mathbf{w})$ for all k-tuples $\mathbf{v}$ and $\mathbf{w}$ in $V(G_k)^k$ or $V(H_k)^k$. This finishes the proof.

D R-GCN

Additionally, we probe a modification of the R-GCN model with an MLP transformation (denoted as R-GCN+MLP) to facilitate parameter sharing between different relation-specific message propagations:

$$\boldsymbol{h}_{v,\text{R-GCN}}^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_{v,\text{R-GCN}}^{(t-1)} \cdot \boldsymbol{W}_0^{(t)} + \sum_{i\in[r]} \text{MLP}\big(\sum_{w\in N_i(v)} \boldsymbol{h}_{w,\text{R-GCN}}^{(t-1)} \cdot \boldsymbol{W}_i^{(t)}\big)\Big) \in \mathbb{R}^d. \tag{9}$$

This modification has a slightly higher count of learnable parameters.

E CompGCN

The original CompGCN architecture proposed in Vashishth et al. [32] considers directed graphs with self-loops, and uses an additional sum to differentiate between in-going, out-going, and self-loop edges, a degree-based normalization, and different weight matrices for these three cases, i.e.,

$$\boldsymbol{h}_{v,\text{CompGCN}}^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_{v,\text{CompGCN}}^{(t-1)}\boldsymbol{W}_0^{(t)} + \sum_{i\in[r]}\sum_{d\in D}\frac{1}{c_{v,w}}\sum_{w\in N_i^d(v)} \phi\big(\boldsymbol{h}_{w,\text{CompGCN}}^{(t-1)}, \boldsymbol{z}_i^{(t)}\big)\boldsymbol{W}_{1,d}^{(t)}\Big) \in \mathbb{R}^e,$$

where $D \coloneqq \{\text{in}, \text{out}\}$, representing in-going, out-going edges, respectively. Here, $N_i^d(v)$ is the restriction of $N_i^d(v)$ of $N_i(v)$ to in-going, out-going, and self-loop edges incident to the vertex v. Further, $c_{v,w} \coloneqq \sqrt{|N_i^d(v)| \cdot |N_i^d(w)|}$. The update of the previous node state is performed via the self-loop direction which we separate into the term $\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)} \boldsymbol{W}_0^{(t)}$ for the sake of a unified notation.

In the ablation studies, we probe the following modifications and combinations of those.

- CompGCN without normalization (*-norm*):

$$\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)} \boldsymbol{W}_0^{(t)} + \sum_{i \in [r]} \sum_{d \in D} \sum_{w \in N_i^d(v)} \phi\big(\boldsymbol{h}_{w,\mathsf{CompGCN}}^{(t-1)}, \boldsymbol{z}_i^{(t)}\big) \boldsymbol{W}_{1,d}^{(t)}\Big) \in \mathbb{R}^e,$$

- CompGCN without direction-specific weights (*-dir*): q

$$\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t)} \coloneqq \sigma\Big(\boldsymbol{h}_{v,\mathsf{CompGCN}}^{(t-1)} \boldsymbol{W}_0^{(t)} + \sum_{i \in [r]} \frac{1}{c_{v,w}} \sum_{w \in N_i(v)} \phi\big(\boldsymbol{h}_{w,\mathsf{CompGCN}}^{(t-1)}, \boldsymbol{z}_i^{(t)}\big) \boldsymbol{W}_1^{(t)}\Big) \in \mathbb{R}^e, \tag{10}$$

- CompGCN without relations update: $\mathbf{z}_i^{t+1} = \mathbf{z}_i^t$ (*-rp*).

As a composition function $\phi(h_w, \mathbf{z}_i)$ we probe several element-wise functions and an MLP:

- `add`: $\phi(\mathbf{h}_w, \mathbf{z}_i) = \mathbf{h}_w + \mathbf{z}_i$ – element-wise addition
- `mult`: $\phi(\mathbf{h}_w, \mathbf{z}_i) = \mathbf{h}_w * \mathbf{z}_i$ – element-wise multiplication (Hadamard product)
- `rotate` [63]: $\phi(\mathbf{h}_w, \mathbf{z}_i) = \mathbf{h}_w \odot \mathbf{z}_i$ – rotation in complex space
- `MLP`: $\phi(\mathbf{h}_w, \mathbf{z}_i) = \text{MLP}([\mathbf{h}_w, \mathbf{z}_i])$ where $[\cdot]$ is column-wise concatenation

F Datasets and Hyperparameters

Statistics about the datasets are presented in Table 1. As neither of the datasets contain an explicit validation set, we retain a random 15% sample of train nodes for validation and use it to optimize hyperparameters.

Table 1: Vertex classification datasets statistics.

Dataset	Vertices	Edges	Relations	Train nodes	Test nodes	Classes
AIFB	8,285	29,043	45	140	36	4
AM	1,666,764	5,988,321	133	802	198	11

Final hyperparameters are listed in Table 2, the total parameter count for all trained models is presented in Table 3. Due to the size of the AM graph and identified stability of the initial node feature dimension, we only train models with dimension $d = 4$ on AM.

Table 2: Hyperparameters

	AIFB			AM		
	R-GCN	R-GCN + MLP	CompGCN	R-GCN	R-GCN + MLP	CompGCN
# Layers	2	2	2	3	3	2
LR	0.001	0.001	0.001	0.03	0.03	0.03
# epochs	8,000	8,000	8,000	100	400	800
Dropout		0.0			0.0	
Optimizer			Adam			
Weight decay			0.0005			

Table 3: Parameters count

	AIFB			AM		
dim	R-GCN	R-GCN + MLP	CompGCN	R-GCN	R-GCN + MLP	CompGCN
2	1,092	1,144	262			
4	2,912	2,992	576	20,311	20,655	1,292
8	8,736	8,920	1,168			
16	29,120	29,704	3,636			
32	104,832	106,984	10,852			
64	396,032	404,392	36,036			
128	1,537,576	1,570,600	129,412			