A Novel Higher-order Weisfeiler-Lehman Graph Convolution

Clemens Damke CDAMKE@MAIL.UPB.DE
Vitalik Melnikov MELNIKOV@MAIL.UPB.DE
Eyke Hüllermeier EYKE@UPB.DE
Paderborn University
Heinz Nixdorf Institute and Department of Computer Science
Intelligent Systems and Machine Learning Group

Editors: Sinno Jialin Pan and Masashi Sugiyama

Abstract

Current *graph neural network* (GNN) architectures use a vertex neighborhood aggregation scheme, which limits their discriminative power to that of the 1-dimensional *Weisfeiler-Lehman* (WL) graph isomorphism test. Here, we propose a novel graph convolution operator that is based on the 2-dimensional WL test. We formally show that the resulting 2-WL-GNN architecture is more discriminative than existing GNN approaches. This theoretical result is complemented by experimental studies using synthetic and real data. On multiple common graph classification benchmarks, we demonstrate that the proposed model is competitive with state-of-the-art graph kernels and GNNs.

Keywords: graph neural networks, Weisfeiler-Lehman, cycle detection

1. Introduction

Graph-structured data has recently received increasing attention in machine learning, with applications ranging from the prediction of chemical properties, e.g., whether a molecule is toxic (Luechtefeld et al., 2018), to the analysis of social network structures (Fan et al., 2019) and source code (Richter et al., 2020). This paper focuses on the prediction of (global) graph properties, i.e., graph classification and regression. In order to predict a certain property of interest, for example to discriminate between graphs in a classification task, a learner must be able to *detect*, either explicitly or implicitly, characteristic features of a graph that are indicative of the sought property. To this end, suitable approaches have been developed in the fields of kernel-based machine learning and (deep) neural networks.

*Graph kernel*s (GKs) implicitly embed graphs in a kernel-induced feature space, thereby making the data — via the kernel trick — amenable to kernel-based learning methods such as *support vector machine*s (SVMs). The features correspond to different (local) properties of a graph. Examples of graph kernels include the multiscale Laplacian graph kernel (Kondor and Pan, 2016), the *Weisfeiler-Lehman subtree kernel* ($\mathrm{WL}_{\mathrm{ST}}$) (Shervashidze et al., 2011), and the *Weisfeiler-Lehman shortest-path kernel* ($\mathrm{WL}_{\mathrm{SP}}$). A comprehensive and up-to-date overview is provided by Kriege et al. (2020).

Unlike GKs, *graph neural network*s (GNNs) produce a prediction directly by applying so-called graph convolution layers, which iteratively aggregate vertex features. Nevertheless, they resemble GKs in so far as those graph convolutions typically use similar graph

features, e.g., the Laplacian spectrum (Bruna et al., 2013) or the *breadth-first-search* (BFS) subtrees that are also used by the WL_{ST} kernel. Two examples of approaches that utilize a BFS subtree characterization are the so-called GCN (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017) architectures. Recently, Xu et al. (2019) have shown that both approaches and, more generally, all GNNs that are expressible in terms of a vertex neighborhood aggregation scheme, cannot produce different predictions for graphs that are indistinguishable via the 1-dimensional *Weisfeiler-Lehman* (WL) graph isomorphism test.

Even though the 1-WL test is able to distinguish almost all pairs of non-isomorphic graphs (Babai et al., 1980), it fails at distinguishing any pair of d-regular graphs of the same size (Immerman and Lander, 1990, Cor. 1.8.5). This restriction alone is not necessarily an issue in the context of graph classification and regression problems, since in real-world domains such as social networks, graphs are rarely perfectly regular. A more relevant restriction of 1-WL, and therefore GNNs, is the fact that it is unable to detect cycles of length $m \geq 3$. In practice, this means that a 1-WL-bounded GNN is unable to make predictions based on important domain-specific graph properties, such as the clustering coefficient of a social network or the presence of aromatic rings in molecules.

Fürer (2017) shows that the cycle detection restriction of 1-WL does however not apply to higher dimensional generalizations of 1-WL. More specifically, he shows that the 2-dimensional WL test is already sufficient to count the number of m-cycles in any graph for all $m \leq 6$. Recently, Arvind et al. (2019) extended this proof to a maximum cycle length of $m \leq 7$. This advantage of 2-WL over 1-WL motivates the idea to define a 2-WL inspired graph convolution operator, which is able to utilize graph characteristics that are not detectable by existing GNNs. Such an operator will be proposed and formally analyzed in Section 4, which is the core of this paper. As a preparation, we start with a brief description of the WL test and the current 1-WL-bounded graph convolution operators in Section 2, and prove limitations of a related GNN extension in Section 3. In Section 5, our novel operator is evaluated on synthetic as well as real-world graph classification datasets, prior to concluding the paper with an outlook on future research in Section 6.

2. Preliminaries

In this section, we introduce some important definitions as well as terminology and notation that will be used throughout the paper.

Definition 2.1. A *graph* $G \coloneqq (\mathcal{V}_G, \mathcal{E}_G)$ consists of a finite set of vertices $v_i \in \mathcal{V}_G$ and a set of edges $e_{ij} = (v_i, v_j) \in \mathcal{E}_G \subseteq \mathcal{V}_G^2$. Continuous feature vectors $x_G[v_i] \in \mathcal{X}_\mathcal{V}$, $x_G[e_{ij}] \in \mathcal{X}_\mathcal{E}$ may be associated with all vertices $v_i \in \mathcal{V}_G$ and edges $e_{ij} \in \mathcal{E}_G$, respectively. In this paper, all graphs G are assumed to be undirected, i.e., $e_{ij} \in \mathcal{E}_G \leftrightarrow e_{ji} \in \mathcal{E}_G$ and $x_G[e_{ij}] = x_G[e_{ji}]$. We denote the set of all undirected graphs as $\mathcal{G}$ and the graph isomorphism relation as $G \simeq H$.

Definition 2.2. Let $\phi : A \to B$ be a function with arbitrary domain A and codomain B. We say that ϕ *distinguishes* $a, a' \in A$ ($a \not\simeq_\phi a'$) iff. $\phi(a) \neq \phi(a')$.

Definition 2.3. A function $\phi : \mathcal{G} \to B$ is *at least as discriminative as* $\psi : \mathcal{G} \to C$ (denoted as $\phi \succeq \psi$) iff. $\forall\, G, H \in \mathcal{G} : G \not\simeq_\psi H \to G \not\simeq_\phi H$. We say that ϕ has the *same discriminative power* (DP) as ψ (denoted as $\phi \equiv \psi$) iff. $\phi \succeq \psi \wedge \psi \succeq \phi$. Lastly, we say that ϕ is *more discriminative* than ψ (denoted as $\phi \succ \psi$) iff. $\phi \succeq \psi \wedge \phi \not\equiv \psi$.

2.1. The Weisfeiler-Lehman Graph Isomorphism Test

The Weisfeiler-Lehman test (Cai et al., 1992) is a popular method to distinguish graphs. There are multiple variations of this test in the literature; we will focus on the so-called Folklore WL test. For a given graph $G \in \mathcal{G}$, it assigns discrete labels $c \in \mathcal{C}$, called *colors*, to vertex k-tuples $(v_1, \dots, v_k) \in \mathcal{V}_G^k$, where $k \in \mathbb{N}$ is the so-called *WL-dimension* that can be chosen freely. A mapping $\chi_{G,k} : \mathcal{V}_G^k \to \mathcal{C}$ is called a *k-coloring* of G.

Definition 2.4. A coloring χ' *refines* χ ($\chi' \succeq \chi$) iff. $\forall a, b \in \mathcal{V}_G^k : a \not\simeq_\chi b \to a \not\simeq_{\chi'} b$, i.e. χ' distinguishes all tuples distinguished by χ. They are *equivalent* ($\chi \equiv \chi'$) iff. $\chi \preceq \chi' \wedge \chi \succeq \chi'$.

The k-dimensional WL test (k-WL) is computed by iteratively refining k-colorings $\chi_{G,k}^{(0)} \preceq \chi_{G,k}^{(1)} \preceq \dots$ of a given graph G until the convergence criterion $\chi_{G,k}^{(t)} \equiv \chi_{G,k}^{(t+1)}$ is satisfied. We denote the final, maximally refined k-WL coloring by $\hat{\chi}_{G,k}$.

In the 1-dimensional case this means that an initial color is assigned to each vertex of a graph G, e.g., the constant coloring $\forall v \in \mathcal{V}_G : \chi_{G,1}^{(0)}(v) = \mathbf{A}$ for some initial color $\mathbf{A} \in \mathcal{C}$. In each iteration of the 1-WL color refinement algorithm, the following neighborhood aggregation scheme is then used to compute a new color for each vertex.

Definition 2.5. $\chi_{G,1}^{(t+1)}(v) \coloneqq h\left(\chi_{G,1}^{(t)}(v), \{\!\{\chi_{G,1}^{(t)}(u) \mid u \in \Gamma_G(v)\}\!\}\right)$, with $\Gamma_G(v)$ the neighboring vertices of $v \in \mathcal{V}_G$, $\{\!\{ \cdot \}\!\}$ denoting a multiset, and $h : \mathcal{C}^* \to \mathcal{C}$ some freely-choosable injective hash function that assigns a unique color to each finite combination of colors.

Analogous to the 1-dimensional refinement step from Def. 2.5, the k-dimensional color refinement step is defined as follows.

Definition 2.6. $\chi_{G,k}^{(t+1)}(s) \coloneqq h\left(\chi_{G,k}^{(t)}(s), \{\!\{(\chi_{G,k}^{(t)}(s[u/1]), \dots, \chi_{G,k}^{(t)}(s[u/k])) \mid u \in \mathcal{V}_G\}\!\}\right)$

with $s = (v_1, \dots, v_k) \in \mathcal{V}_G^k, \quad s[u/j] \coloneqq (v_1, \dots, v_{j-1}, u, v_{j+1}, \dots, v_k)$.

In 1-WL, a vertex color is refined by combining the colors of neighboring vertices. In k-WL, the color of a k-tuple $s \in \mathcal{V}_G^k$ is refined by combining the colors of its neighborhood, which is defined as the set of all k-tuples in which at most one vertex differs from s. Note that each vertex k-tuple has one neighbor for each $u \in \mathcal{V}_G$, each of which is a k-tuple of vertex k-tuples. For $k = 2$, this means that each potential edge $(v, w) \in \mathcal{V}_G^2$ has all possible walks of length 2 from v to w as its neighbors. Even though k-WL refines k-tuple colors, lower-dimensional structures still get their own colors, since a tuple does not have to consist of distinct vertices: In k-WL, the color of a single vertex $v \in \mathcal{V}_G$ is described by $\hat{\chi}_{G,k}(s)$ for $s = (v, \dots, v) \in \mathcal{V}_G^k$; similarly, every possible edge $e_{ij} \in \mathcal{V}_G^2$ has at least one color that can encode the adjacency information, i.e., whether $e_{ij} \in \mathcal{E}_G$.

Definition 2.7. The color distribution $dist_{\chi_{G,k}} : \mathcal{C} \to \mathbb{N}_0$ of a k-coloring $\chi_{G,k}$ counts each color $c \in \mathcal{C}$ in the coloring, i.e., $dist_{\chi_{G,k}}(c) \coloneqq \left|\left\{v \in \mathcal{V}_G^k \mid \chi_{G,k}(v) = c\right\}\right|$.

The output of the k-WL algorithm is the color distribution $dist_{\hat{\chi}_{G,k}}$. Since the way in which WL colorings are refined is vertex order invariant, any difference in the final color distribution of two graphs always implies the non-isomorphism ($\not\simeq$) of the colored graphs, i.e., $G \not\simeq_{k\text{-WL}} H \implies G \not\simeq H$. Figure 1 shows that the opposite does however not necessarily hold. Additionally, it highlights the inability of 1-WL to detect cycles of varying lengths in graphs.

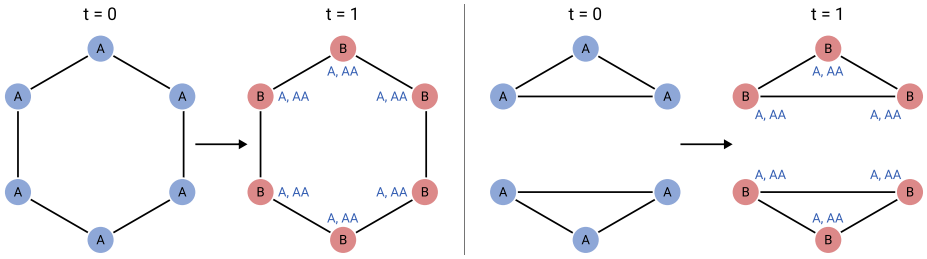


Figure 1: Two simple non-isomorphic graphs that are indistinguishable by 1-WL.

Definition 2.8. We say that k-WL *detects* m-cycles iff. k-WL $\succeq d_m$, where $d_m : \mathcal{G} \to \{0, 1\}$ is an indicator function, that determines whether a given graph contains at least one m-cycle.

Intuitively, Def. 2.8 describes cycle detection as the ability to solve the corresponding decision problem given only the color distributions $dist_{\hat{\chi}_{G,k}}$ for all $G \in \mathcal{G}$. As already mentioned in the introduction, to detect cycles of length $m \leq 7$, a WL dimension of $k \geq 2$ is required (Fürer, 2017; Arvind et al., 2019).

2.2. Graph Neural Networks

In this paper, we will focus on so-called spatial GNNs, which are expressible in terms of repeated vertex neighborhood aggregations. Such a GNN takes a graph $G \in \mathcal{G}$ with vertex feature vectors $x_G[v] \in \mathbb{R}^{d^{(0)}}$ as input; those features are typically represented as a matrix $Z_G^{(0)} \coloneqq \begin{pmatrix} x_G[v_1] \\ \vdots \\ x_G[v_n] \end{pmatrix} \in \mathbb{R}^{n \times d^{(0)}}$, where $n \coloneqq |\mathcal{V}_G|$. A GNN convolves this vertex feature matrix via a stack of graph convolution operators $S^{(t)} : \mathbb{R}^{n \times d^{(t-1)}} \to \mathbb{R}^{n \times d^{(t)}}$ s.t. $Z_G^{(t)} \coloneqq S^{(t)}(Z_G^{(t-1)})$. We use $Z_G^{(t)}[v] \in \mathbb{R}^{d^{(t)}}$ to denote the row vector of $Z_G^{(t)}$, which represents the convolved vertex features of $v \in \mathcal{V}_G$. After applying T convolutional layers, the convolved vertex features $Z_G^{(T)}[v]$ can be used directly for node classification problems, or they can be combined via a pooling layer, e.g., an element-wise mean, to obtain a global graph feature vector which in turn can be used to solve *graph classification and regression* (GC/GR) problems. In the rest of this paper, GNNs will be discussed in the context of GC/GR.

To get an intuition for how GNNs relate to the WL algorithm, one should think of the vertex feature vectors $Z_G^{(t)}$ as a continuous generalization of WL colors $\chi_{G,1}^{(t)}$. The graph convolution operators $S^{(t)}$ then directly correspond to WL color refinement steps. This intuition was recently formalized by Xu et al. (2019), who showed the following upper bound on the discriminative power of GNNs.

Proposition 2.9. *Any GNN that convolves vertex feature vectors via a convolution operator of the form* $Z_G^{(t)}[v] = h^{(t)}\left(Z_G^{(t-1)}[v], \{\!\{ Z_G^{(t-1)}[u] \mid u \in \Gamma_G(v) \}\!\}\right)$ *is at most as discriminative as 1-WL, where* $h^{(t)} : \left(\mathbb{R}^{d^{(t-1)}}\right)^* \to \mathbb{R}^{d^{(t)}}$ *is an arbitrary vertex neighborhood hashing function. Moreover, iff.* $h^{(t)}$ *is injective, the GNN has the same DP as 1-WL.*

Among others, this bound applies to the GCN (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017) architectures. Since those approaches use a non-injective hashing function $h^{(t)}$, their DP turns out to be strictly lower than that of 1-WL. On the other hand, the *graph isomorphism network* (GIN) (Xu et al., 2019) convolution operator achieves injectivity through the use of a *multilayer perceptron* (MLP), and therefore has the same DP as 1-WL (cf. Def. 2.5):

$$Z_G^{(t)}[v] \coloneqq \mathrm{MLP}^{(t)}\left((1+\varepsilon)Z_G^{(t-1)}[v] + \sum_{u \in \Gamma_G(v)} Z_G^{(t-1)}[u]\right) \quad \text{with some irrational } \varepsilon > 0. \tag{1}$$

3. Limitations of an Existing 2-GNN

The idea to extend GNNs along the lines of the higher-order WL algorithm, which we shall elaborate on in Section 4 below, is not entirely new. Morris et al. (2019) recently proposed the so-called k-GNN, which adapts the discrete k-WL refinement step (see Def. 2.6) to the continuous convolution setting. However, it turns out that k-GNNs do not preserve some of the desirable properties of k-WL. In particular, unlike 2-WL, 2-GNNs cannot count or even detect m-cycles in graphs. In this section, we give a proof of this statement.

Similar to k-WL, a k-GNN iteratively refines/convolves the colors/features of combinations of k vertices. To reduce runtime complexity, k-GNNs assign feature vectors to vertex k-multisets instead of k-tuples. Additionally, only a "local" neighborhood of each multiset is considered in k-GNN convolutions, whereas in k-WL each tuple has a "global" neighborhood of size $n = |\mathcal{V}_G|$, one neighbor for each vertex $u \in \mathcal{V}_G$ (cf. Def. 2.6). As we will see next, the main difference between k-GNNs and k-WL lies in their respective notion of "neighborhood". More specifically, since 2-WL already has a significantly higher DP than 1-WL, we will analyze how the DP of 2-GNNs compares to that of 1-WL and 2-WL.

2-GNNs define the neighbors of an edge $e_{ij} = (v_i, v_j)$ to be the edges that are incident to either v_i or v_j. In 2-WL, on the other hand, the neighbors of e_{ij} are the edge pairs $\{(e_{il}, e_{lj})\}_{v_l \in \mathcal{V}_G}$, i.e., all possible walks of length two that start at v_i and end at v_j. This difference becomes clear when comparing the definition of convolution in 2-GNNs with that of color refinement in 2-WL:

$$\text{2-GNN}^1\colon\quad Z_G^{(t)}[e_{ij}] = \sigma\left(Z_G^{(t-1)}[e_{ij}]W^{(t)} + \left(\sum_{v_l \in \Gamma_G(v_j)} Z_G^{(t-1)}[e_{il}] + \sum_{i \neq j \,\wedge\, v_l \in \Gamma_G(v_i)} Z_G^{(t-1)}[e_{lj}]\right) W_\Gamma^{(t)}\right) \tag{2}$$

$$\text{2-WL:}\ \chi_{G,2}^{(t)}(e_{ij}) = h\left(\chi_{G,2}^{(t-1)}(e_{ij}),\quad \{\!\{(\chi_{G,2}^{(t-1)}(e_{il}), \chi_{G,2}^{(t-1)}(e_{lj})) \mid v_l \in \mathcal{V}_G\}\!\}\right)$$

In order to analyze what those different notions of neighborhood imply for the DP of 2-GNNs in comparison to 2-WL, we first show that the DP of 2-GNNs on all graphs $G \in \mathcal{G}$ is less than or equal to that of 1-WL on the so-called *edge neighborhood graphs* $G^{\mathcal{E}} \in \mathcal{G}^{\mathcal{E}}$.

1. Here, σ denotes some nonlinear activation function and $W^{(t)}, W_\Gamma^{(t)} \in \mathbb{R}^{d^{(t-1)} \times d^{(t)}}$ the weight matrices of the convolution operator. To highlight the relationship between 2-GNNs and 2-WL, a 2-GNN definition that uses two sums over $v_l \in \Gamma_G(v_j)$ resp. $v_l \in \Gamma_G(v_i)$ is shown; this is equivalent to a single sum over the features of the edges $\{(u, w) \in \mathcal{E}_G \mid u = v_i \vee w = v_j\}$ (see Morris et al., 2019).

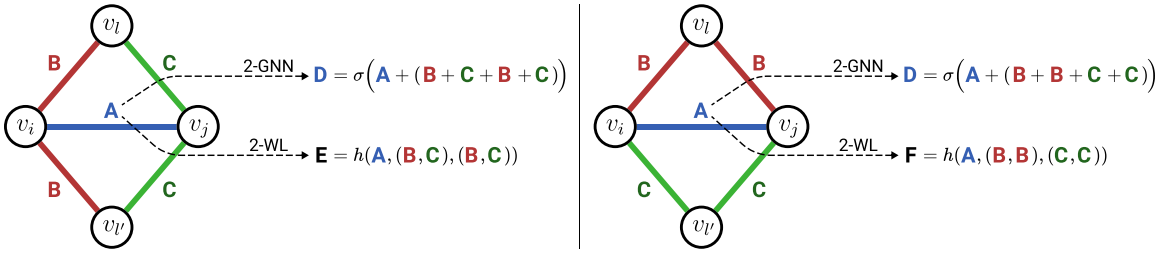


Figure 2: Two edge colorings on which 2-GNNs and 2-WL behave differently. A 2-GNN will refine the "color vector" of e_{ij} to **D** for both initial colorings. 2-WL on the other hand differentiates both colorings by preserving the color tuple information.

Definition 3.1. The *edge neighborhood graph* of a given graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ is defined as $G^{\mathcal{E}} \coloneqq (\mathcal{V}_{G^{\mathcal{E}}}, \mathcal{E}_{G^{\mathcal{E}}})$ with the vertices $\mathcal{V}_{G^{\mathcal{E}}} \coloneqq \{\{\!\{v, u\}\!\} \,|\, (v, u) \in \mathcal{E}_G \vee v = u\}$ and the edges $\mathcal{E}_{G^{\mathcal{E}}} \coloneqq \big\{(e, e') \in \mathcal{V}^2_{G^{\mathcal{E}}} \,\big|\, |e \cap e'| = 1\big\}$.

Proposition 3.2. *The DP of all 2-GNNs $h_2 : \mathcal{G} \to \mathcal{Y}$ is less than or equal to that of 1-WL on edge neighborhood graphs, i.e., $\forall\, G, H \in \mathcal{G} : G^{\mathcal{E}} \simeq_{\text{1-WL}} H^{\mathcal{E}} \to h_2(G) = h_2(H)$.*

Proof. By Def. 3.1, $\Gamma_{G^{\mathcal{E}}}(e_{ij}) = \{(u, w) \in \mathcal{E}_G \,|\, u = v_i \vee w = v_j\}$ for all $e_{ij} \in \mathcal{E}_G$. Therefore, the 2-GNN convolution in Eq. (2) can be rewritten as a vertex neighborhood convolution $Z_G^{(t)}[e] = \sigma\left(Z_G^{(t-1)}[e]W^{(t)} + \sum_{e' \in \Gamma_{G^{\mathcal{E}}}(e)} Z_G^{(t-1)}[e']W_\Gamma^{(t)}\right)$. Prop. 3.2 then follows from Prop. 2.9. □

Lemma 3.3. *1-WL cannot distinguish the edge neighborhood graphs $G^{\mathcal{E}}$ and $H^{\mathcal{E}}$ of any pair of d-regular graphs G and H with n vertices.*

Proof. Let G and H be two d-regular graphs of size n. Their corresponding edge neighborhood graphs $G^{\mathcal{E}}$ and $H^{\mathcal{E}}$ both have $n^{\mathcal{E}} = n + \frac{nd}{2}$ vertices, n of which correspond to the vertices of G and H respectively; we will refer to them as *loop vertices* L_G / L_H. The remaining $\frac{nd}{2}$ edge neighborhood vertices correspond to the edges of G and H; we will refer to them as *edge vertices* E_G / E_H.

W.l.o.g. we define the initial colors of the loop vertices as $\chi^{(0)}(v) = \mathbf{A}$ for all $v \in L_G \cup L_H$. The initial colors of the edge vertices are defined as $\chi^{(0)}(e) = \mathbf{B}$ for all $e \in E_G \cup E_H$. Note that each loop vertex $\{\!\{v_i, v_i\}\!\}$ with $v_i \in \mathcal{V}_G \cup \mathcal{V}_H$ has d neighbors, the edges incident to v_i. Similarly, each edge vertex $\{\!\{v_i, v_j\}\!\}$ has $2d$ neighbors, two of which are the loop vertices $\{\!\{v_i, v_i\}\!\}$ and $\{\!\{v_j, v_j\}\!\}$ with the remaining $2d - 2$ neighbors corresponding to the edges that are incident to e_{ij}.

After one color refinement step, we get $\chi^{(1)}(v) = h(\mathbf{A}, \{\!\{\underbrace{\mathbf{B}, \ldots, \mathbf{B}}_{d \text{ times}}\}\!\}) =: \mathbf{C}$ for all loop vertices $v \in L_G \cup L_H$ and $\chi^{(1)}(e) = h(\mathbf{B}, \{\!\{\mathbf{A}, \mathbf{A}, \underbrace{\mathbf{B}, \ldots, \mathbf{B}}_{2d-2 \text{ times}}\}\!\}) =: \mathbf{D}$. This means that $\chi^{(0)}$ and $\chi^{(1)}$ are identical up to the color substitutions $\mathbf{A} \to \mathbf{C}$ and $\mathbf{B} \to \mathbf{D}$, i.e. $\chi^{(0)} \equiv \chi^{(1)}$, which in turn implies that 1-WL terminates after one iteration. Lemma 3.3 then directly follows, since both $G^{\mathcal{E}}$ and $H^{\mathcal{E}}$ have n vertices with the final color **C** and $\frac{nd}{2}$ vertices with the final color **D**, i.e. $G^{\mathcal{E}} \simeq_{\text{1-WL}} H^{\mathcal{E}}$. □

Proposition 3.4. *A 2-GNN cannot distinguish regular graphs of the same size and therefore has a lower DP than 2-WL.*

Proof. The proposition directly follows from Prop. 3.2, Lem. 3.3 and the fact that 2-WL is able to distinguish most regular graphs (Immerman and Lander, 1990, Cor. 1.8.6). □

As previously mentioned, the DP of a model by itself is not necessarily relevant for real-world GC/GR problems. However, 2-WL is not only more discriminative than 1-WL, but is also able to detect and count the number of m-cycles in a given graph for all $m \leq 7$. We now show that 2-GNNs not only have a lower DP than 2-WL, but are also unable to detect cycles.

Proposition 3.5. *2-GNNs cannot detect m-cycles for $m \geq 3$.*

Proof. Let n be the lowest common multiple of 3 and some $m > 3$. We define $c_3 := \frac{n}{3}$ and $c_m := \frac{n}{m}$. Based on that, we define the following two graphs: Let G_3 be a graph consisting of c_3 disconnected cycles of length 3, analogously let G_m be a graph consisting of c_m disconnected cycles of length m. Since both G_3 and G_m are 2-regular and have the size n, any 2-GNN $h_2 : \mathcal{G} \to \mathcal{Y}$ must map both of them to the same $y \in \mathcal{Y}$ by Prop. 3.4, i.e., $G_3 \simeq_{h_2} G_m$.

Let us assume that h_2 is able to detect cycles of length 3, i.e. triangles. Following Def. 2.8, this would imply that h_2 is at least as discriminative as the triangle detection function $d_3 : \mathcal{G} \to \{0, 1\}$. It follows that $d_3(G_3) = 1 \wedge d_3(G_m) = 0 \Rightarrow G_3 \not\simeq_{d_3} G_m \xrightarrow{h_2 \succeq d_3} G_3 \not\simeq_{h_2} G_m$, which is a contradiction. Conversely, assuming that h_2 is able to detect cycles of length $m > 3$, the m-cycle detection function d_m also distinguishes G_3 and G_m, which again results in the contradiction $G_3 \simeq_{h_2} G_m \wedge G_3 \not\simeq_{h_2} G_m$. □

4. The 2-WL Graph Convolution Operator

In the previous section, we compared 2-GNNs with the 2-WL algorithm and found that the former have a significantly lower DP than the latter. Motivated by this limitation, we devote this section to a novel, more discriminative convolution operator, which is inspired by the higher-order WL algorithm and meant to overcome the limitations of 1-WL. Our operator is inspired by 2-WL but uses the following simplifications to reduce its computational cost.

Similar to k-GNNs, or more specifically, 2-GNNs, our novel operator refines/convolves the feature vectors of *2-multisets* $\{\!\{v_i, v_j\}\!\}$ instead of refining/convolving the feature vectors of 2-tuples (v_i, v_j). This simplification halves the number of feature vectors without affecting the DP because we assume that graphs are undirected, i.e. e_{ij} and e_{ji} have identical feature vectors $x[e_{ij}] = x[e_{ji}] \in \mathcal{X}_\mathcal{E}$ and the same 2-WL neighborhood. To simplify the notation, we assume that $e_{ij} = e_{ji} = \{\!\{v_i, v_j\}\!\}$ in the rest of the paper.

After applying the 2-multiset simplification, the 2-WL algorithm refines the color of all multisets $e_{ij} \in \mathcal{V}_G^2$ by hashing its current color and the colors of all neighbors $\{\{\!\{e_{il}, e_{lj}\}\!\}\}_{v_l \in \mathcal{V}_G}$. This means that the time complexity of a single refinement step is $\mathcal{O}(n^3)$ for $n := |\mathcal{V}_G|$, which quickly becomes infeasible for large graphs. To address this issue, we reduce both the number of colored edges as well as the number of neighbors of each edge. This is achieved by only considering the edges that are part of the so-called *r-th power of a graph G*, where $r \in \mathbb{N}$ is the freely choosable *neighborhood radius*.

Definition 4.1. The *r-th power of a graph* G is defined as

$$G^r \coloneqq \left(\mathcal{V}_G, \left\{e_{ij} \in \mathcal{V}_G^2 \,\middle|\, d_{\mathrm{SP},G}(v_i, v_j) \leq r\right\}\right) ,$$

where $d_{\mathrm{SP},G}(v_i, v_j)$ is the length of the shortest path between v_i and v_j in G. The distance of a vertex $v_i \in \mathcal{V}_G$ to itself is defined as $d_{\mathrm{SP},G}(v_i, v_i) \coloneqq 0$. Note that G^1 does not generally equal G because G^1 has self-loop edges $e_{ii} \in \mathcal{E}_{G^1}$ at all vertices.

For the neighborhood radius $r = 1$, only the self-loop edges $\{e_{ii}\}_{v_i \in \mathcal{V}_G}$ and the edges $\mathcal{E}_G$ are considered; for $r > 1$, edges between indirectly connected vertices are considered as well. Through the reduction of the considered edges, the neighbors of each $e_{ij} \in \mathcal{E}_{G^r}$ are in turn reduced to the common r-neighbors of v_i and v_j, i.e. $\{\{\!\{e_{il}, e_{lj}\}\!\} \mid v_l \in \Gamma_{G^r}(v_i) \cap \Gamma_{G^r}(v_j)\}$.

Let us now consider what the reduced number of used edges and the reduced number of edge neighbors implies for the runtime of a refinement step. If G is a sparse graph with the maximum vertex degree $d \coloneqq \max_{v \in \mathcal{V}_G} |\Gamma_G(v)|$, the number of considered edges is bounded by $\mathcal{O}(nd^r)$, where each edge has at most $\mathcal{O}(d^r)$ neighbors. Consequently, the time complexity of a refinement step becomes $\mathcal{O}(nd^{2r})$, which is a significant improvement over the $\mathcal{O}(n^3)$ bound of a full 2-WL refinement step (assuming $d \ll n$).

Based on the 2-multiset and the neighborhood localization simplifications, we now define the 2-WL convolution operator and the corresponding *2-WL-GNN*.

Definition 4.2. The *initial feature matrix* $Z_G^{(0)}$ of the 2-WL convolution operator with the neighborhood radius $r \in \mathbb{N}$ contains both the vertex features $x[v_i] \in \mathcal{X}_\mathcal{V} = \mathbb{R}^{d_\mathcal{V}}$ as well as the edge features $x[e_{ij}] \in \mathcal{X}_\mathcal{E} = \mathbb{R}^{d_\mathcal{E}}$ of a given graph G. More specifically, $Z_G^{(0)} \in \mathbb{R}^{|\mathcal{E}_{G^r}| \times (d_\mathcal{V} + d_\mathcal{E})}$ assigns a row vector $Z_G^{(0)}[e_{ij}]$ to all edges $e_{ij} \in \mathcal{E}_{G^r}$. Those initial edge feature vectors are defined by the following vector concatenation ($\oplus$):

$$Z_G^{(0)}[e_{ij}] \coloneqq \begin{pmatrix} \begin{cases} x[v_i] & \text{if } i = j \\ \mathbf{0} & \text{else} \end{cases} \end{pmatrix} \oplus \begin{pmatrix} \begin{cases} x[e_{ij}] & \text{if } e_{ij} \in \mathcal{E}_G \\ \mathbf{0} & \text{else} \end{cases} \end{pmatrix}$$

Definition 4.3. We define the *2-WL graph convolution operator* as

$$Z_G^{(t)}[e_{ij}] \coloneqq \sigma\left(Z_G^{(t-1)}[e_{ij}] W_{\mathrm{L}}^{(t)} + \sum_{v_l \in \Gamma_{G^r}(v_i) \cap \Gamma_{G^r}(v_j)} \kappa^{(t)}\left(Z_G^{(t-1)}[e_{ij}], \{\!\{Z_G^{(t-1)}[e_{il}], Z_G^{(t-1)}[e_{lj}]\}\!\}\right)\right) ,$$
$$\text{with } \kappa^{(t)}(z_{ij}, \{\!\{z_{il}, z_{lj}\}\!\}) \coloneqq \left(z_{ij} W_{\mathrm{F}}^{(t)}\right) \odot \sigma_\Gamma\left(\left(z_{il} + z_{lj}\right) W_\Gamma^{(t)}\right) .$$

This operator is parameterized by the three matrices $W_{\mathrm{L}}^{(t)}, W_{\mathrm{F}}^{(t)}, W_\Gamma^{(t)} \in \mathbb{R}^{d^{(t-1)} \times d^{(t)}}$ and uses two freely choosable activation functions σ and σ_Γ. We use $\odot$ to denote element-wise multiplication. In the following, we will analyze the DP of GNNs using the 2-WL convolution operator. Our goal is to show that such 2-WL-GNNs have a strictly higher DP than 1-WL. We begin by proving that 2-WL-GNNs are at least as discriminative as 1-WL.

Definition 4.4. A GNN $h_1 : \mathcal{G} \to \mathcal{Y}$ uses *weighted vertex neighborhood sums* if its convolutional layers can be described by

$$Z_G^{(t)}[v_i] = \mathrm{MLP}^{(t)}\left(w_{ii} Z_G^{(t-1)}[v_i] + \sum_{v_j \in \Gamma_G(v_i)} w_{ij} Z_G^{(t-1)}[v_l]\right) .$$

Definition 4.4 includes GCNs (Kipf and Welling, 2017), where the MLP only consists of a single layer with weights $w_{ij} = (|\Gamma_G(v_i)| + 1)^{-\frac{1}{2}}(|\Gamma_G(v_j)| + 1)^{-\frac{1}{2}}$. GINs also trivially satisfy the definition (see Eq. (1)).

Theorem 4.5. *For each GNN h_1 using weighted vertex neighborhood sums, there is a 2-WL-GNN h_2 that simulates h_1, i.e., such that $\forall G \in \mathcal{G} : h_1(G) = h_2(G)$.*

Proof. We prove by construction. Let $G \in \mathcal{G}$ be an arbitrary input graph with $n \coloneqq |\mathcal{V}_G|$ and $m \coloneqq n + |\mathcal{E}_G|$. By definition, h_1 is a GNN of the form $Pool_1(Conv_1(G))$, where $Conv_1$ is a stack of T weighted vertex neighborhood sum convolutions $\left\{S^{(t)} : \mathbb{R}^{n \times d^{(t-1)}} \to \mathbb{R}^{n \times d^{(t)}}\right\}_{t=1}^{T}$ with each corresponding $\text{MLP}^{(t)}$ having K layers. $Pool_1$ combines the vertex feature vectors produced by $Conv_1$. Let h_2 be a GNN of the form $Pool_2(Conv_2(G))$, where $Conv_2$ is a stack of $(2+K)T$ 2-WL convolution layers $\left\{S^{(t,k)} : \mathbb{R}^{m \times d^{(t,k-1)}} \to \mathbb{R}^{m \times d^{(t,k)}}\right\}_{(t,k) \in [T] \times [2+K]}$ (see Fig. 3) with the neighborhood radius $r \coloneqq 1$. The layers $\left\{S^{(t,2+K)}\right\}_{t=1}^{T}$ produce the feature matrices $Z^{(t,2+K)} = Z^{(t+1,0)}$ which are fed as input into the successor layer $S^{(t+1,1)}$.

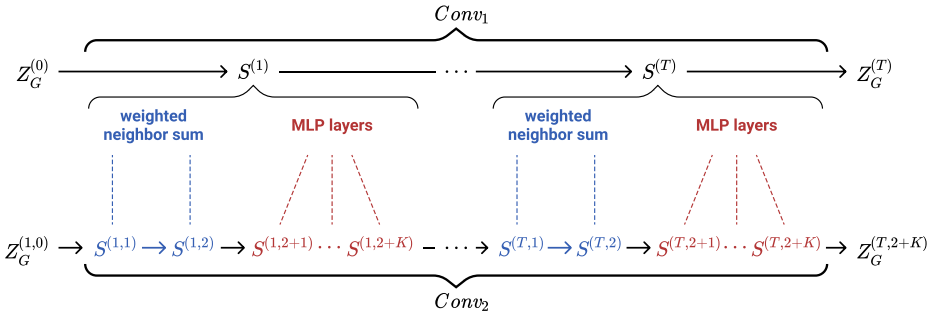


Figure 3: Illustration of the correspondence between $Conv_1$ and $Conv_2$.

Let $\varphi : \mathbb{R}^{d^{(T,2+K)}} \to \mathbb{R}^{d^{(T)}} \cup \{\texttt{nil}\}$ be a function that maps the final 2-WL feature vectors produced by $Conv_2$ to the output space of $Conv_1$ or the constant `nil`. Let $Pool_2\left(Z_G^{(T,2+K)}\right) \coloneqq Pool_1(\{\!\{z_{ij} \,|\, z_{ij} = \varphi\left(Z_G^{(T,2+K)}[e_{ij}]\right) \wedge e_{ij} \in \mathcal{E}_{G^1} \wedge z_{ij} \neq \texttt{nil}\}\!\})$. Theorem 4.5 then follows if there is a function φ s.t. $\forall v_i \in \mathcal{V}_G : Conv_1(G)[v_i] = \varphi(Conv_2(G)[e_{ii}])$ and $\forall e_{ij} \in \mathcal{E}_G : \varphi(Conv_2(G)[e_{ij}]) = \texttt{nil}$. To guarantee that there is such a function φ, we now inductively prove the following three invariants, which have to hold for all $t \in \{0, \ldots, T\}$:

(P1) $Z_G^{(t,2+K)}[e_{ij}]_1 = \mathbb{1}[i = j]$, i.e., the first component of each 2-WL feature vector allows φ to decide whether that vector should be mapped to `nil`.

(P2) $Z_G^{(t,2+K)}[e_{ii}]_{2,\ldots,(d^{(t)}+1)} = Z_G^{(t)}[v_i]$, i.e., the second to $(1 + d^{(t)})$-th components of each self-loop feature vector in h_2 contain the corresponding convolved vertex feature vector at layer t in h_1.

(P3) $Z_G^{(t,2+K)}[e_{ij}]_{d^{(t)}+2} = w_{ij}$, i.e., the weights for the vertex neighborhood sums are encoded in the edge and self-loop feature vectors.

For $t = 0$, all invariants apply to the initial feature matrix $Z_G^{(0,2+K)} = Z_G^{(1,0)}$ by Def. 4.2:

$$\forall v_i \in \mathcal{V}_G : Z_G^{(1,0)}[e_{ii}] \coloneqq (1) \oplus x[v_i] \oplus (w_{ii}) \text{ and } \forall e_{ij} \in \mathcal{E}_G : Z_G^{(1,0)}[e_{ij}] \coloneqq (0) \oplus \mathbf{0} \oplus (w_{ij}).$$

Assuming the invariants hold for $t-1$, we now show that they also hold for t. The layers $S^{(t,1)}$ and $S^{(t,2)}$ are used to compute the weighted vertex neighborhood sums

$$Z^{(t,2)}[e_{ii}]_{2,\dots,(1+d^{(t-1)})} = w_{ii} Z^{(t,0)}[e_{ii}]_{2,\dots,(1+d^{(t-1)})} + \sum_{v_j \in \Gamma_G(v_i)} w_{ij} Z^{(t,0)}[e_{jj}]_{2,\dots,(1+d^{(t-1)})}.$$

We now explicitly define parameter matrices for $S^{(t,1)}$ and $S^{(t,2)}$ s.t. this weighted sum is produced. Note that the weighted vertex neighborhood sum only requires scalar multiplication and vector addition, i.e., the $d^{(t-1)}$ vertex feature dimensions are mutually independent. W.l.o.g. this allows us to simplify notation by treating the vertex feature vectors as if they were scalars in the following definitions, i.e., we can assume $d^{(t-1)} = 1$ and $Z^{(t,0)}[e_{ii}] = (1, Z^{(t-1)}[v_i], w_{ii}) \in \mathbb{R}^3$. Using this simplification, the layer $S^{(t,1)}$ is defined by

$$\begin{aligned}
Z^{(t,1)}[e_{ij}] &= Z^{(t,0)}[e_{ij}] W_{\mathrm{L}}^{(t,1)} + \sum_{v_l \in \Gamma_{G^1}(v_i) \cap \Gamma_{G^1}(v_j)} \left(Z^{(t,0)}[e_{ij}] W_{\mathrm{F}}^{(t,1)} \right) \odot \left(\left(Z^{(t,0)}[e_{il}] + Z^{(t,0)}[e_{lj}] \right) W_{\Gamma}^{(t,1)} \right) \\
&= \begin{cases} (1,0,w_{ii},0) + \left(0,0,0,2w_{ii} Z^{(t-1)}[v_i]\right) + \sum_{v_l \in \Gamma_{G^1}(v_i)} \left(0, \frac{2}{2} Z^{(t-1)}[v_i] w_{il}, 0, 0\right) & \text{if } i = j \\ (0,0,w_{ij},0) + \left(0,0,0,w_{ij}(Z^{(t-1)}[v_i] + 0)\right) + \left(0,0,0,w_{ij}(0 + Z^{(t-1)}[v_j])\right) & \text{else} \end{cases} \\
&= \begin{cases} \left(1, \sum_{v_l \in \Gamma_{G^1}(v_i)} w_{il} Z^{(t-1)}[v_i], w_{ii}, 2w_{ii} Z^{(t-1)}[v_i]\right) & \text{if } i = j \\ \left(0,0,w_{ij}, w_{ij}(Z^{(t-1)}[v_i] + Z^{(t-1)}[v_j])\right) & \text{else} \end{cases},
\end{aligned}$$

$$\text{with } W_{\mathrm{L}}^{(t,1)} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, W_{\mathrm{F}}^{(t,1)} \coloneqq \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, W_{\Gamma}^{(t,1)} \coloneqq \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The vertex neighborhood summation is completed via $S^{(t,2)}$, which is defined by

$$\begin{aligned}
Z^{(t,2)}[e_{ij}] &= \begin{cases} \left(1, -\sum_{v_l \in \Gamma_{G^1}(v_i)} w_{il} Z^{(t-1)}[v_i], w_{ii}\right) + \sum_{v_l \in \Gamma_{G^1}(v_i)} \left(0, w_{il} \left(Z^{(t-1)}[v_i] + Z^{(t-1)}[v_l]\right), 0\right) & \text{if } i = j \\ (0,0,w_{ij}) & \text{else} \end{cases} \\
&= \begin{cases} \left(1, w_{ii} Z^{(t-1)}[v_i] + \sum_{v_l \in \Gamma_G(v_i)} w_{il} Z^{(t-1)}[v_l], w_{ii}\right) & \text{if } i = j \\ (0,0,w_{ij}) & \text{else} \end{cases},
\end{aligned}$$

$$\text{with } W_{\mathrm{L}}^{(t,2)} \coloneqq \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad W_{\mathrm{F}}^{(t,2)} \coloneqq \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad W_{\Gamma}^{(t,2)} \coloneqq \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

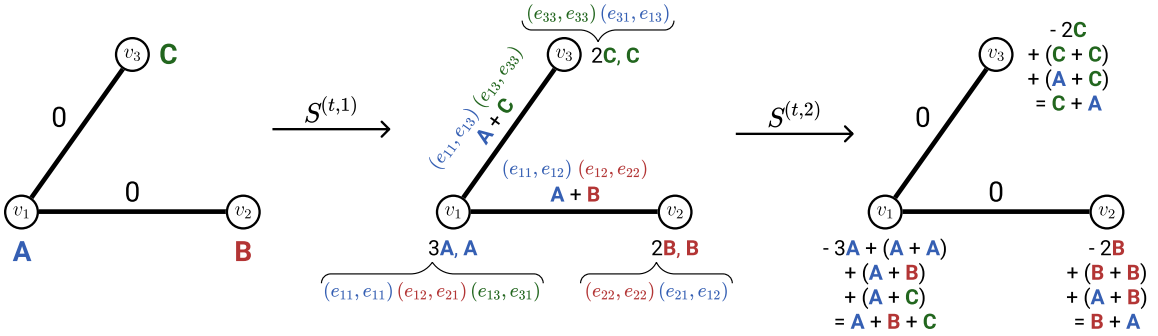


Figure 4: Intuition for how $S^{(t,1)}$ and $S^{(t,2)}$ compute vertex neighborhood sums in two steps. For simplicity, weights are ignored, i.e. all $w_{ij} = 1$. For each self-loop/edge $e_{ij} \in \mathcal{E}_{G^1}$, the set of localized 2-WL neighbors $\Gamma_{G^1}(v_i) \cap \Gamma_{G^1}(v_j)$ is shown in the middle, after the first step. The colors in this illustration are unrelated to the colored parts in the previous equations.

Using the two layers $S^{(t,1)}$ and $S^{(t,2)}$ that we just defined, the weighted vertex neighborhood sum for all $v_i \in \mathcal{V}_G$ is contained in $Z^{(t,2)}[e_{ii}]$. Additionally, for all $e_{ij} \in \mathcal{E}_{G^1}$, the indicators $Z^{(t,2)}[e_{ij}]_1 = \mathbb{1}[i = j]$ and the weights $Z^{(t,2)}[e_{ij}]_{d^{(t)}+2} = w_{ij}$ are preserved. This means that invariants (P1) and (P3) are satisfied after $S^{(t,2)}$.

To complete the induction step, it now remains to show that all three invariants hold after applying the layers $S^{(t,2+1)}, \ldots, S^{(t,2+K)}$. Note that a 2-WL convolution layer is reduced to a fully connected layer if $W_{\mathrm{F}}^{(t)} = \mathbf{0}$. Via the universal approximation theorem (Hornik, 1991), we can therefore use $S^{(t,2+1)}, \ldots, S^{(t,2+K)}$ to simulate the K layers of $\mathrm{MLP}^{(t)}$ without changing the first and last dimension of each feature vector to preserve invariants (P1) and (P3). The resulting feature matrix $Z^{(t,2+K)}$ then satisfies all three invariants, which completes the induction.

Using invariants (P1) and (P2) for $t = T$, we can therefore set

$$\varphi\left(Z_G^{(T,2+K)}[e_{ij}]\right) \coloneqq \begin{cases} Z_G^{(T,2+K)}[e_{ij}]_{2,\ldots,(d^T+1)} & \text{if } Z_G^{(T,2+K)}[e_{ij}]_1 = 1 \\ \texttt{nil} & \text{else} \end{cases}.$$

By our previous definition of $Pool_2$, this in turn implies that $Pool_2(Z_G^{(T,2+K)}) = Pool_1(Z_G^{(T)}) \iff h_2(G) = h_1(G)$, which concludes the proof. □

Corollary 4.6. *2-WL-GNNs have at least the same DP as 1-WL.*

Proof. The corollary directly follows from the fact that 2-WL-GNNs can simulate GINs by Thm. 4.5 and the fact that GINs have the same DP as 1-WL by Prop. 2.9, because they use injective vertex neighborhood hashing functions. □

To complete our analysis of the DP of the 2-WL-GNN, we now show that it is not just as discriminative as 1-WL but is in fact more discriminative than 1-WL.

Proposition 4.7. *There are d-regular graphs G and H of size n, which can be distinguished by 2-WL-GNNs.*

Proof. The proposition follows if we choose the six-cycle graph for G and the two three-cycles graph for H (see Fig. 1). Let $h_2 = Pool \circ S$ be a 2-WL-GNN with the neighborhood radius $r = 1$, which consists of a single 2-WL convolution layer $S : \mathbb{R}^{*\times 2} \to \mathbb{R}^{*\times 1}$ and the pooling layer $Pool = \min$. In accordance with Def. 4.2, we set the initial feature vectors of the vertices v_i of G and H to $Z^{(0)}[e_{ii}] \coloneqq (1, 0)$ and the initial feature vectors of their edges e_{ij} to $Z^{(0)}[e_{ij}] \coloneqq (0, 1)$. Let the weight matrices of S be $W_{\mathrm{L}} \coloneqq \mathbf{0}$ and $W_{\mathrm{F}} = W_{\Gamma} \coloneqq \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. For simplicity, we choose the identity activation functions $\sigma = \sigma_{\Gamma} = \mathrm{id}$. By Def. 4.3, all self-loops e_{ii} of G^1 and H^1 have the three neighbors $\{\{\!\{e_{ii}, e_{ii}\}\!\}, \{\!\{e_{ij}, e_{ji}\}\!\}, \{\!\{e_{il}, e_{li}\}\!\}\}$, i.e., the length-two walk along e_{ii} itself and the length-two walks to and from the two neighboring vertices $\Gamma(v_i) = \{v_j, v_l\}$. Therefore, the convolved feature vector of all self-loops are $Z^{(1)}[e_{ii}] = (1) \odot ((1 + 1) + (1 + 1) + (1 + 1)) = 6$. However, for the non-self-loops of G^1 and H^1, i.e., the edges of G and H, we get differing convolved feature vectors. The 2-WL neighbors of $e_{ij} \in \mathcal{E}_G$ are $\{\{\!\{e_{ii}, e_{ij}\}\!\}, \{\!\{e_{ij}, e_{jj}\}\!\}\}$. The 2-WL neighbors of $e'_{ij} \in \mathcal{E}_H$ are $\{\{\!\{e'_{ii}, e'_{ij}\}\!\}, \{\!\{e'_{ij}, e'_{jj}\}\!\}, \{\!\{e'_{il}, e'_{lj}\}\!\}\}$, where $v'_l \in \mathcal{V}_H$ is the common neighbor of v'_i and v'_j. The different neighborhood sizes of the edges of G and H imply that $\forall e_{ij} \in \mathcal{E}_G : Z^{(1)}[e_{ij}] = 4$, while $\forall e'_{ij} \in \mathcal{E}_H : Z^{(1)}[e_{ij}] = 6$. Thus $h_2(G) = \min\{4, 6\} \neq \min\{6, 6\} = h_2(H)$, which concludes the proof. □

Corollary 4.8. *The DP of 2-WL-GNNs is strictly higher than that of the 1-WL algorithm.*

Proof. The corollary directly follows from Cor. 4.6 and Prop. 4.7, since 1-WL cannot distinguish regular graphs (Immerman and Lander, 1990, Cor. 1.8.5). □

This concludes our analysis of the discriminative power of 2-WL-GNNs. The key insight in this section is that 2-WL-GNNs are more discriminative than all vertex neighborhood aggregation GNNs, because the DP of the latter is at most that of 1-WL. Additionally, we can conclude that 2-WL-GNNs are able to distinguish graphs that are indistinguishable by 2-GNNs due to Prop. 3.4 and 4.7.

Note that no statement regarding the DP of 2-WL-GNNs compared to 2-WL was made. It is easy to see that 2-WL-GNNs *generally* cannot have the same power as 2-WL due to the neighborhood localization simplification: For a small neighborhood radius of $r = 1$, nonexistent edges $e_{ij} \notin \mathcal{E}_G$ do not have a feature vector; those missing feature vectors are however required by the proof of 2-WL's m-cycle counting ability for $m \geq 4$ (see the proof by Fürer, 2017, Lem. 1 and Thm. 2). We leave a thorough discussion of the relation between 2-WL-GNNs and 2-WL for future work.

5. Evaluation

In our experimental evaluation, we compare the proposed 2-WL convolution layer with other state-of-the-art approaches. We focus on two types of learners: SVMs using graph kernels and GNNs. We evaluate those learners by comparing their test accuracies on multiple binary classification problems. To obtain those accuracies, we use the graph classification benchmarking framework recently proposed by Errica et al. (2020): We use 10-fold stratified training/test splits; for each split the hyperparameters are tuned via a second 90%/10% validation holdout split of the training data. Experiments were run three times to smooth

out differences caused by random weight initialization. To train models that require gradient-based optimization, we use the well-known Adam optimizer and the standard binary cross-entropy loss.

Using this assessment strategy, we evaluate SVMs with the following graph kernels: $\mathrm{WL}_{\mathrm{ST}}$, $\mathrm{WL}_{\mathrm{SP}}$, and the so-called 2-LWL and 2-GWL kernels (Morris et al., 2017); the last two are essentially 2-WL variants of the $\mathrm{WL}_{\mathrm{ST}}$ kernel. We additionally evaluate the following GNNs: 2-WL-GNN *(our method)*, GIN, 2-GNN, and a structure-unaware baseline that applies an MLP to each vertex feature vector $x_G[v]$, sums up the resulting vectors and applies another MLP to the sum (see Errica et al., 2020).

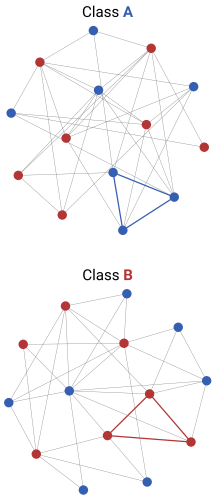
5.1. Synthetic Data

We begin with an evaluation on a synthetic binary classification dataset, which demonstrates the potential advantages of a higher dimensional WL method. To determine the classes of the graphs in this dataset, a learner has to solve the following *unicolored triangle detection* problem: Given a graph G with vertices that are colored as either $l_G[v] = \mathsf{A}$ or as $l_G[v] = \mathsf{B}$, the learner has to find the unique triangle (v_i, v_j, v_k) in G for which $l_G[v_i] = l_G[v_j] = l_G[v_k]$. The class of G is then determined by the color of the vertices (v_i, v_j, v_k).

Based on this problem, we generated a synthetic triangle detection dataset. It contains randomly generated graphs with varying vertex counts and vertex color proportions. We use this dataset to evaluate whether a learner is able to ignore varying amounts of noisy random structure and focus on relevant local substructures, in this case unicolored triangles. For the evaluation of 2-WL-GNNs the neighborhood radius $r = 2$ is used.

Table 1: Mean accuracies and standard deviations on the triangle detection dataset.

	Model (Iterations/Pooling)	Train	Test
Kernel	**$\mathrm{WL}_{\mathrm{ST}}$** ($T = 1$)	88.3 ± 6.9	64.8 ± 13.2
	$\mathrm{WL}_{\mathrm{ST}}$ ($T = 3$)	98.0 ± 1.7	56.9 ± 11.1
	$\mathrm{WL}_{\mathrm{ST}}$ ($T = 5$)	$\mathbf{100.0} \pm 0.0$	62.6 ± 11.2
	$\mathrm{WL}_{\mathrm{SP}}$ ($T = 3$)	96.9 ± 8.4	$\mathbf{68.0} \pm 10.7$
	2-LWL ($T = 3$)	97.3 ± 3.3	56.5 ± 6.2
	2-GWL ($T = 3$)	99.9 ± 0.2	61.8 ± 8.8
GNN	**Baseline** (sum)	48.8 ± 1.6	44.6 ± 8.1
	GIN (sum)	84.2 ± 10.6	70.0 ± 7.4
	2-GNN (mean)	93.2 ± 3.1	76.8 ± 10.7
	2-GNN (weighted mean)	97.1 ± 2.9	81.8 ± 7.6
	2-WL-GNN (mean)	98.3 ± 2.6	92.9 ± 8.4
	2-WL-GNN (weighted mean)	$\mathbf{99.8} \pm 0.4$	$\mathbf{99.4} \pm 1.3$



Looking at the results in Tbl. 1, it can be seen that the structure-unaware baseline method is completely unable to detect triangles, as expected. The structure-aware learners on the other hand all perform better than random guessing and are in fact mostly able to fit the training data perfectly. This shows that all generated graphs are 1-WL distinguishable; the WL subtree kernel SVM, for example, can simply "memorize" the training graphs via their unique 1-WL color distribution after $T = 5$ refinement steps.

However, the ability to distinguish training graphs is not sufficient to also classify previously unseen graphs correctly. Since 1-WL cannot detect triangles, all 1-WL bounded approaches (WL_{ST}, WL_{SP}, Baseline, GIN) are therefore unable to generalize, as suggested by their test accuracies. Performance better than random guessing can be explained by availability of the following proxy indicator: The presence of an **A**-colored triangle in a graph G implies that there is a local region with a slightly higher density of **A**-colored vertices than in a **B**-colored graph H with the same vertex color proportions. This local difference in color density is already detectable in the depth-1 BFS subtrees used by 1-WL after a single refinement step, which explains why WL_{ST} performs similarly for $T = 1$ and $T = 5$.

As for the 2-WL inspired kernels, 2-LWL and 2-GWL, it is interesting to see that both kernels do not appear to generalize better than the 1-WL bounded methods. We explain this by the fairly small size of the triangle detection dataset (228 graphs); even though both kernels embed graphs into a space which contains dimensions that indicate the presence of a unicolored triangle, i.e., their DP is sufficiently high to solve the problem, there are so many of those triangle-indicating embedding dimensions that the relevant indicator dimensions found in a given training split might not overlap with those in the test split.

Looking at the 2-WL inspired GNNs (2-GNN, 2-WL-GNN), we find that the 2-WL-GNN significantly outperforms all other methods, which is in line with our results from Sections 3 and 4.

5.2. Evaluation on Real-World Data

We evaluate the approaches on five common binary graph classification benchmark datasets, namely the NCI1 (Shervashidze et al., 2011), PROTEINS (Borgwardt et al., 2005), and D&D (Dobson and Doig, 2003) datasets from the domain of bioinformatics, and the REDDIT-B and IMDB-B datasets (Yanardag and Vishwanathan, 2015) from social network analysis. Tbl. 2 shows our evaluation results. For the evaluation of 2-WL-GNNs, different neighborhood radii were used for each dataset. In the order of the columns in the table, the results were obtained with the radii $r = 8$, 5, 2, 1 and 4, respectively.

Table 2: Mean test accuracies and standard deviations on real-world data.

	Model (Iter./Pooling)	NCI1	PROTEINS	D&D	REDDIT-B	IMDB-B
Kernel	**WL_{ST}** ($T = 1$)	73.9 ± 2.6	72.8 ± 3.3	$\mathbf{78.9} \pm 4.2$	76.3 ± 2.5	71.0 ± 2.2
	WL_{ST} ($T = 3$)	$\mathbf{84.8} \pm 1.6$	73.0 ± 2.4	78.8 ± 4.3	$\mathbf{78.0} \pm 2.7$	72.9 ± 2.5
	WL_{SP} ($T = 3$)	OOM	73.1 ± 3.5	OOM	OOM	$\mathbf{74.4} \pm 3.5$
	2-LWL ($T = 3$)	76.7 ± 2.2	69.4 ± 4.6	76.6 ± 3.5	75.8 ± 2.9	72.2 ± 3.3
	2-GWL ($T = 3$)	71.6 ± 2.1	$\mathbf{73.1} \pm 3.6$	76.3 ± 3.9	75.4 ± 3.2	70.4 ± 3.2
GNN	**Baseline** (sum)	67.7 ± 3.1	74.0 ± 4.9	$\mathbf{75.7} \pm 2.5$	72.1 ± 7.8	50.7 ± 2.4
	GIN (sum)	77.4 ± 2.9	71.8 ± 3.1	75.2 ± 3.4	87.0 ± 4.4	66.8 ± 3.9
	2-GNN (mean)	75.9 ± 2.0	74.8 ± 3.4	72.9 ± 4.1	OOM	71.4 ± 3.6
	2-GNN (w. mean)	$\mathbf{78.3} \pm 1.8$	73.8 ± 3.5	69.6 ± 3.9	OOM	70.9 ± 3.2
	2-WL-GNN (mean)	72.4 ± 2.9	$\mathbf{76.5} \pm 2.7$	75.4 ± 3.3	83.7 ± 5.2	71.2 ± 4.0
	2-WL-GNN (w. mean)	73.5 ± 2.9	76.2 ± 3.3	74.7 ± 3.1	$\mathbf{89.4} \pm 2.6$	$\mathbf{72.2} \pm 3.1$

Compared with the triangle detection dataset, the advantage of 2-WL-GNNs over the other approaches is clearly less pronounced. This indicates that the theoretical advantages of 2-WL over 1-WL are not as relevant for the five real-world domains as they are for the synthetic

problem. Nonetheless, the test performance of 2-WL-GNNs is generally comparable to that of the other state-of-the-art learners, in the sense that the performance of the evaluated 2-WL-GNN models is within the 2σ confidence interval of the best evaluated model.

If we look at the enzyme detection problem (PROTEINS and D&D), we observe that all evaluated approaches appear to be unable to leverage structural information for a significant improvement over the baseline learner. On the social network datasets (REDDIT-B and IMDB-B), on the other hand, the structure aware methods clearly outperform the baseline. This confirms similar results by Errica et al. (2020).

6. Conclusion

We proposed the novel 2-WL-GNN and showed it to be strictly more discriminative than 1-WL bounded GNNs. This theoretical advantage was clearly confirmed experimentally on synthetic data, while results competitive to state-of-the-art graph kernels and GNNs could be achieved on real-world data. We envision two main directions for future research: First, a more thorough theoretical analysis of the relation between 2-WL-GNNs and 2-WL is required to answer questions such as how the neighborhood radius r relates to the discriminative power of a 2-WL-GNN. Second, evaluations on a broader range of domains and other problem types, such as vertex labeling, link prediction, or graph regression, will help to determine in which contexts the theoretical advantages of 2-WL-GNNs also lead to practical improvements.

Acknowledgments

This work was supported by German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901/3 project no. 160364472).

References

V. Arvind, F. Fuhlbrück, J. Köbler, and O. Verbitsky. On Weisfeiler-Leman Invariance: Subgraph Counts and Related Graph Properties. In *Fundamentals of Computation Theory*, pages 111–125. Springer International Publishing, 2019.

L. Babai, P. Erdős, and S. M. Selkow. Random graph isomorphism. *SIaM Journal on computing*, 9(3):628–635, 1980.

K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21:i47–i56, 2005.

J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral Networks and Locally Connected Networks on Graphs, 2013.

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

P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. In *Int. Conf. on Learn. Rep.*, ICLR, 2020.

W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin. Graph Neural Networks for Social Recommendation. In *The Web Conf.*, WWW'19, 2019.

M. Fürer. On the Combinatorial Power of the Weisfeiler-Lehman Algorithm. In *Lecture Notes in Computer Science*, pages 260–271. Springer International Publishing, 2017.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st Int. Conf. on Neural Inf. Proc. Sys.*, NIPS'17, page 1025–1035, 2017.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity theory retrospective*, pages 59–81. Springer, 1990.

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

R. Kondor and H. Pan. The Multiscale Laplacian Graph Kernel. In *Proceedings of the 30th Int. Conf. on Neural Inf. Proc. Sys.*, NIPS'16, page 2990–2998, 2016.

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

T. Luechtefeld, D. Marsh, C. Rowlands, and T. Hartung. Machine learning of toxicological big data enables read-across structure activity relationships (RASAR) outperforming animal test reproducibility. *Toxicological Sciences*, 165(1):198–212, 2018.

C. Morris, K. Kersting, and P. Mutzel. Glocalized Weisfeiler-Lehman Graph Kernels: Global-Local Feature Maps of Graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on AI*, 33:4602–4609, 2019.

C. Richter, E. Hüllermeier, M. Jakobs, and H. Wehrheim. Algorithm selection for software validation based on graph kernels. *Automated Software Engineering*, 27:153–186, 2020.

N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *JMLR*, 12:2539–2561, 2011.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, ICLR, 2019.

P. Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.